Cellular Automata Surviving k Steps

Zeke Medley & Panagiotis Manolios
Northeastern University

May 2025

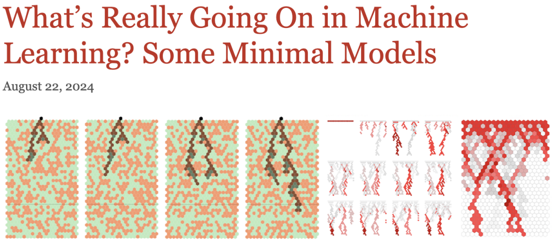
Surviving k Steps

1. Problem posed by Wolfram in 2024.

[1] writings.stephenwolfram.com posts dated May 3, 2024; August 22, 2024; December 5, 2024; February 3, 2025.

Surviving k Steps

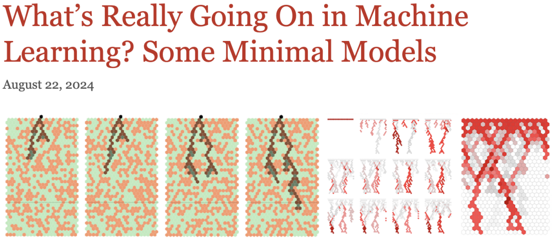
1. Problem posed by Wolfram in 2024.
2. Explored in several essays. [1]



[1] writings.stephenwolfram.com posts dated May 3, 2024; August 22, 2024; December 5, 2024; February 3, 2025.

Surviving k Steps

1. Problem posed by Wolfram in 2024.
2. Explored in several essays. [1]



3. Wolfram takes an empirical view, we take a mathematical one.

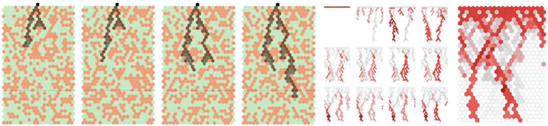
[1] writings.stephenwolfram.com posts dated May 3, 2024; August 22, 2024; December 5, 2024; February 3, 2025.

Surviving k Steps

1. Problem posed by Wolfram in 2024.
2. Explored in several essays. [1]

What's Really Going On in Machine Learning? Some Minimal Models

August 22, 2024



3. Wolfram takes an empirical view, we take a mathematical one.
4. Our work: $> 99\%$ solved.

[1] writings.stephenwolfram.com posts dated May 3, 2024; August 22, 2024; December 5, 2024; February 3, 2025.

Elementary Cellular Automata

1. Functions from $\{\blacksquare, \square\}^3 \to \{\blacksquare, \square\}$.

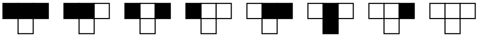
Elementary Cellular Automata

1. Functions from $\{\blacksquare, \square\}^3 \to \{\blacksquare, \square\}$.
2. Let λ_i be the ith one ("Wolfram Code" enumeration).

Elementary Cellular Automata

1. Functions from $\{\blacksquare, \square\}^3 \to \{\blacksquare, \square\}$.
2. Let λ_i be the ith one ("Wolfram Code" enumeration).
3. For example, λ_4's truth table

λ_4

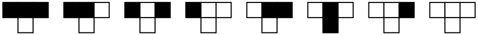


$$\lambda_4(\blacksquare, \blacksquare, \blacksquare) = \square, \quad \lambda_4(\square, \blacksquare, \square) = \blacksquare, \quad \ldots$$

Growing λ_4 (visually)

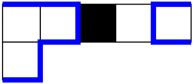
$x^{(j)} =$ 

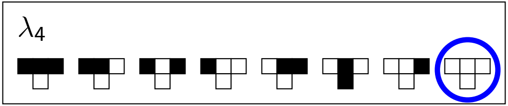
λ_4



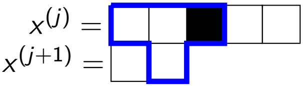
Growing λ_4 (visually)

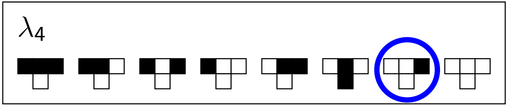
$x^{(j)} =$
$x^{(j+1)} =$





Growing λ_4 (visually)

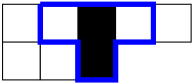


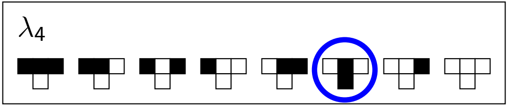


Growing λ_4 (visually)

$x^{(j)} =$

$x^{(j+1)} =$

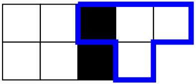


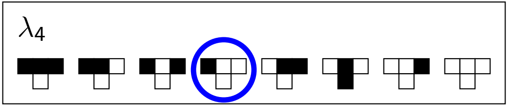


Growing λ_4 (visually)

$x^{(j)} =$

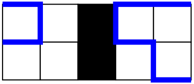
$x^{(j+1)} =$

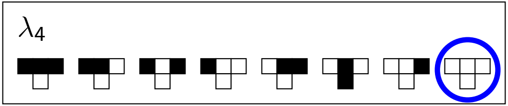




Growing λ_4 (visually)

$x^{(j)} =$
$x^{(j+1)} =$

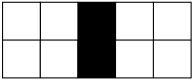


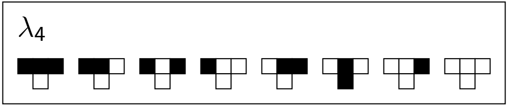


Growing λ_4 (visually)

$x^{(j)} =$

$x^{(j+1)} =$





Growing λ_4

jth state $x^{(j)} \in \{\blacksquare, \square\}^w$, next state $x^{(j+1)}$,

$$x_i^{(j+1)} = \lambda_4(x_{i-1 \bmod w}^{(j)}, x_i^{(j)}, x_{i+1 \bmod w}^{(j)})$$

Inhomogeneous Cellular Automata

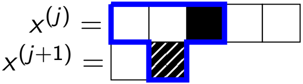
1. Choose what λ_i to use where, instead of using the same λ_i everywhere.

Inhomogeneous Cellular Automata

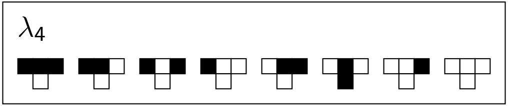
1. Choose what λ_i to use where, instead of using the same λ_i everywhere.
2. Formally, use a *rule array* $a : \mathbb{N}^2 \to \{\lambda_{i_1}, \ldots, \lambda_{i_t}\}$,

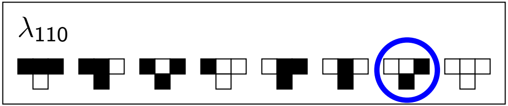
$$x_i^{(j+1)} = a(i,j)(x_{i-1 \bmod w}^{(j)}, x_i^{(j)}, x_{i+1 \bmod w}^{(j)})$$

Growing Inhomogeneous Cellular Automata (visually)



$$a(1, j) = \lambda_{110}$$





Inhomogeneous Cellular Automata

1. Choose what λ_i to use where, instead of using the same λ_i everywhere.
2. Formally, use a *rule array* $a : \mathbb{N}^2 \to \{\lambda_{i_1}, \ldots, \lambda_{i_t}\}$,

$$x_i^{(j+1)} = a(i,j)(x_{i-1 \bmod w}^{(j)}, x_i^{(j)}, x_{i+1 \bmod w}^{(j)})$$

3. **What can we synthesize with appropriate choice of a?**

Figure: An inhomogeneous cellular automata using λ_{126} and λ_{225}.

Inhomogeneous Cellular Automata

1. Inhomogeneous Cellular Automata are very expressive[2].

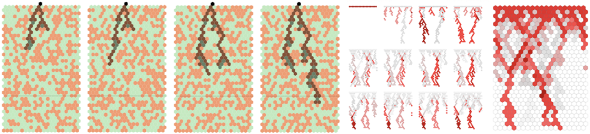
[2] *A survey of cellular automata: types, dynamics, non-uniformity and applications*, Kamalika Bhattacharjee *et. al.*

Inhomogeneous Cellular Automata

1. Inhomogeneous Cellular Automata are very expressive[2].
2. Our work looks at the Survive-k-Steps problem posed by Wolfram in 2024.

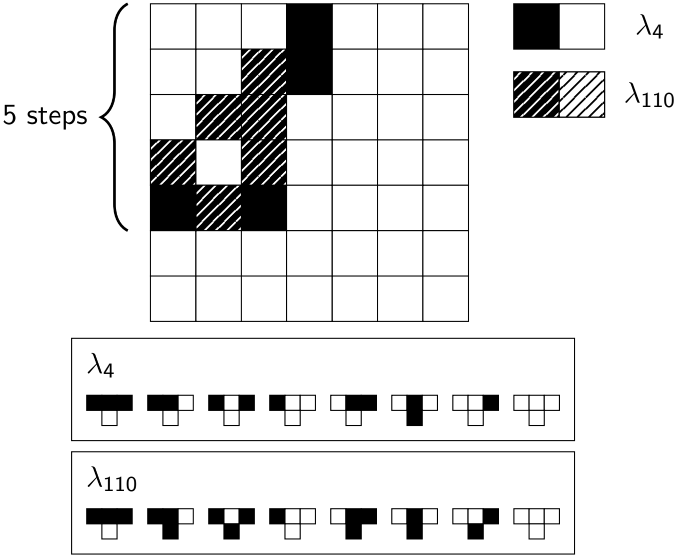
What's Really Going On in Machine Learning? Some Minimal Models

August 22, 2024

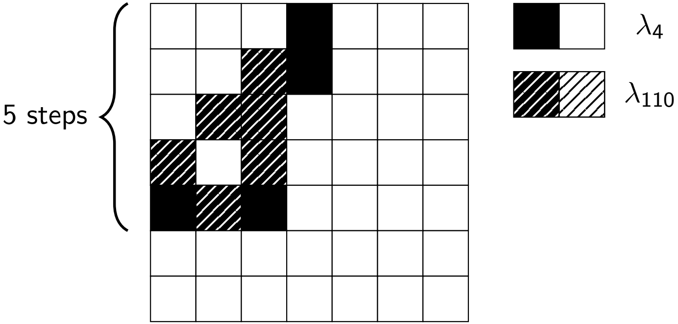


[2] *A survey of cellular automata: types, dynamics, non-uniformity and applications*, Kamalika Bhattacharjee *et. al.*

Survive-5-Steps (visually)



Survive-5-Steps (visually)



This proves λ_4, λ_{110} can Survive-5-Steps, but which k are solvable in general?

Survive-k-Steps

If there exists a rule array using only λ_i, λ_j such that,

Survive-k-Steps

If there exists a rule array using only λ_i, λ_j such that,

1. $x^{(0)}$ has exactly one ■,

Survive-k-Steps

If there exists a rule array using only λ_i, λ_j such that,

1. $x^{(0)}$ has exactly one $\blacksquare$,
2. $\forall i < k, \blacksquare \in x^{(i)}$,

Survive-k-Steps

If there exists a rule array using only λ_i, λ_j such that,

1. $x^{(0)}$ has exactly one $\blacksquare$,
2. $\forall i < k, \blacksquare \in x^{(i)}$,
3. $\forall i \geq k, \blacksquare \notin x^{(i)}$

Survive-k-Steps

If there exists a rule array using only λ_i, λ_j such that,

1. $x^{(0)}$ has exactly one $\blacksquare$,
2. $\forall i < k, \blacksquare \in x^{(i)}$,
3. $\forall i \geq k, \blacksquare \notin x^{(i)}$

then, λ_i and λ_j can solve Survive-k-Steps .

Research Question

For any i, j, is the set

$$\{k \mid \lambda_i, \lambda_j \text{ solve Survive-}k\text{-Steps}\}$$

finite?

Evidence of Structure

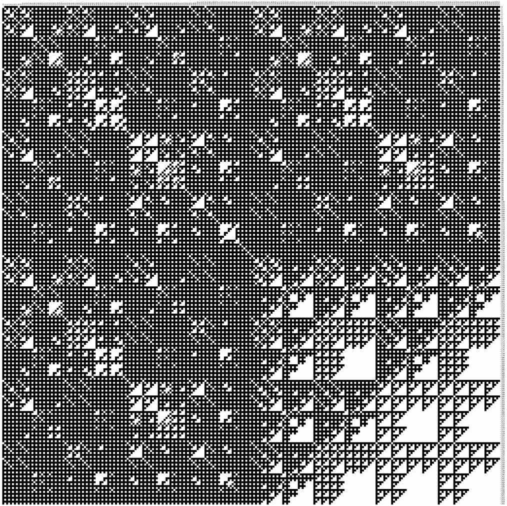
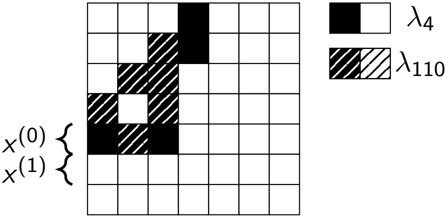


Figure: ■ at position (i, j) means Survive-50-Steps on a width 31 board is possible with λ_i and λ_j. Top-left is $(0, 0)$, bottom right is $(255, 255)$.

The ■-to-□ Transition

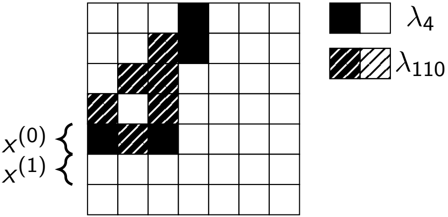
To Survive-k-Steps λ_i,λ_j need to “die”.



“die”, i.e. $\exists x^{(0)}, \blacksquare \in x \wedge \blacksquare \notin x^{(1)}$,

The ■-to-□ Transition

To Survive-k-Steps λ_i,λ_j need to "die".



"die", i.e. $\exists x^{(0)}$, $\blacksquare \in x \wedge \blacksquare \notin x^{(1)}$,i.e. the ■-to-□ Transition.

When is ■-to-□ impossible?

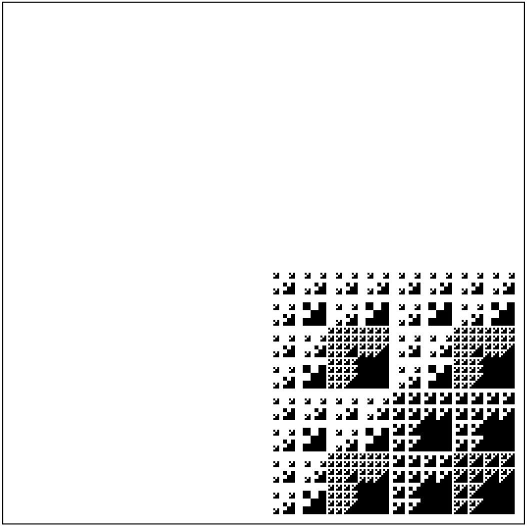


Figure: ■ at position (i, j) means λ_i, λ_j cannot do the ■-to-□ transition on a width 31 board. Top-left is $(0, 0)$, bottom right is $(255, 255)$.

When is ■-to-□ impossible?

Let's prove it.

When is ■-to-□ impossible?

Let's prove it.

1. $\blacksquare \in x^{(0)}$ if x contains a *pattern* with a ■. For example,

$$x = \square\blacksquare\boxed{\blacksquare\square\blacksquare}.$$

   contains the pattern ■□■.

When is ■-to-□ impossible?

Let's prove it.

1. ■ $\in x^{(0)}$ if x contains a *pattern* with a ■. For example,

$$x = \square\blacksquare\boxed{\blacksquare\square\blacksquare}.$$

   contains the pattern ■□■.

2. ■-to-□ is possible if λ_i or λ_j output □ on that pattern.

When is $\blacksquare$-to-$\square$ impossible?

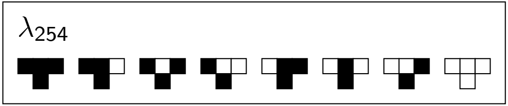
Let's prove it.

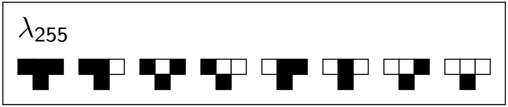
1. $\blacksquare \in x^{(0)}$ if x contains a *pattern* with a $\blacksquare$. For example,

$$x = \square\blacksquare\boxed{\blacksquare\square\blacksquare}.$$

   contains the pattern $\blacksquare\square\blacksquare$.

2. $\blacksquare$-to-$\square$ is possible if λ_i or λ_j output $\square$ on that pattern.
3. For example,





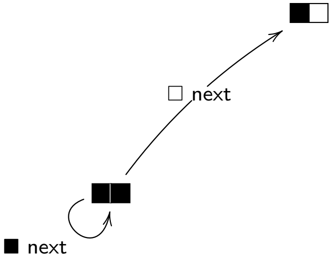
   cannot transition from $\blacksquare$-to-$\square$.

Reasoning about patterns

1. We need a way to reason about patterns in $x \in \{\blacksquare, \square\}^w$.

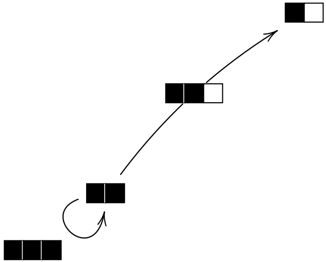
Reasoning about patterns

1. We need a way to reason about patterns in $x \in \{\blacksquare, \square\}^w$.
2. Consider a **two-cell sliding window** starting at ■■.



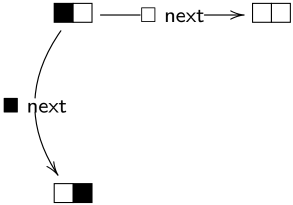
Reasoning about patterns

1. We need a way to reason about patterns in $x \in \{\blacksquare, \square\}^w$.
2. Consider a **two-cell sliding window** starting at $\blacksquare\blacksquare$.



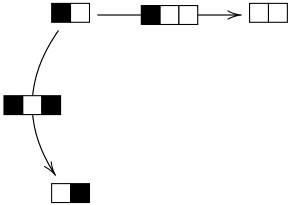
Reasoning about patterns

1. We need a way to reason about patterns in $x \in \{■, □\}^w$.
2. Consider a **two-cell sliding window** starting at ■□.



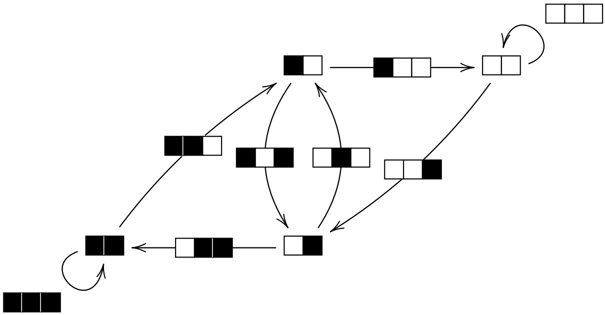
Reasoning about patterns

1. We need a way to reason about patterns in $x \in \{■, □\}^w$.
2. Consider a **two-cell sliding window** starting at ■□.



Reasoning about patterns

1. We need a way to reason about patterns in $x \in \{\blacksquare, \square\}^w$.
2. Consider a **two-cell sliding window**.



This is a De Bruijn graph.

Cycle $\leftrightarrow$ State

There is a bijection between $\{\blacksquare, \square\}^w$ (with wrapping) and cycles in this graph.

Cycle $\leftrightarrow$ State

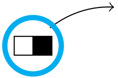
There is a bijection between $\{\blacksquare, \square\}^w$ (with wrapping) and cycles in this graph. For example,

$$x = \boxed{\square\blacksquare}\blacksquare\square\blacksquare.$$

Cycle $\leftrightarrow$ State

There is a bijection between $\{\blacksquare, \square\}^w$ (with wrapping) and cycles in this graph. For example,

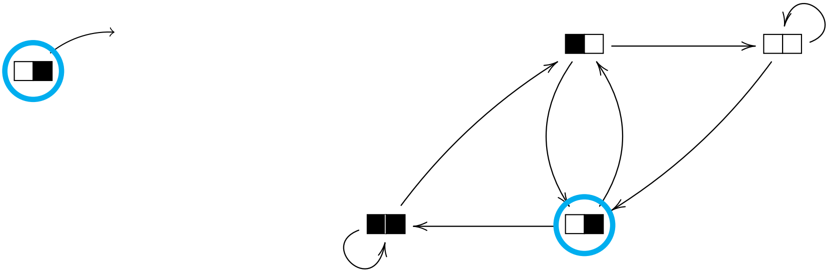
$$x = \boxed{\square\blacksquare}\blacksquare\square\blacksquare.$$



Cycle $\leftrightarrow$ State

There is a bijection between $\{\blacksquare, \square\}^w$ (with wrapping) and cycles in this graph. For example,

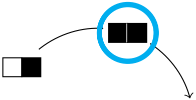
$$x = \square\blacksquare\blacksquare\square\blacksquare.$$

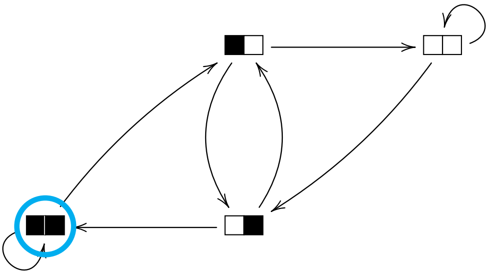


Cycle $\leftrightarrow$ State

There is a bijection between $\{■, □\}^w$ (with wrapping) and cycles in this graph. For example,

$x =$ □■■□■.

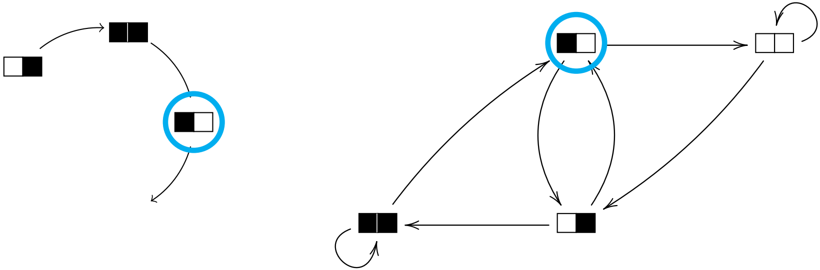




Cycle $\leftrightarrow$ State

There is a bijection between $\{\blacksquare, \square\}^w$ (with wrapping) and cycles in this graph. For example,

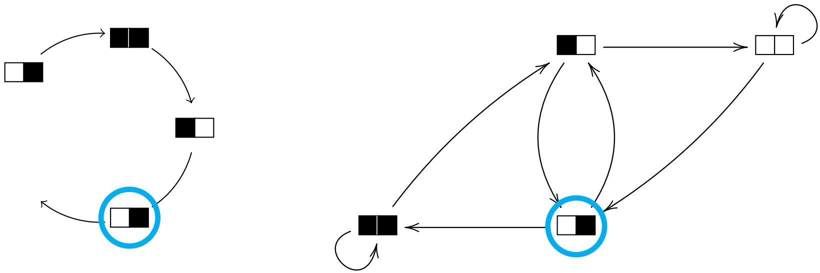
$$x = \square\blacksquare\boxed{\blacksquare\square}\blacksquare.$$



Cycle $\leftrightarrow$ State

There is a bijection between $\{\blacksquare, \square\}^w$ (with wrapping) and cycles in this graph. For example,

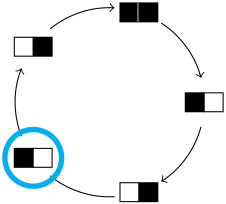
$$x = \square\blacksquare\blacksquare\boxed{\square\blacksquare}.$$

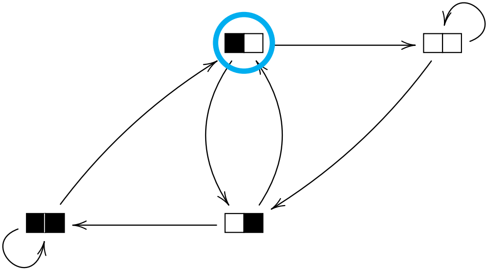


Cycle $\leftrightarrow$ State

There is a bijection between $\{\blacksquare, \square\}^w$ (with wrapping) and cycles in this graph. For example,

$$x = \square\blacksquare\blacksquare\square\blacksquare.$$

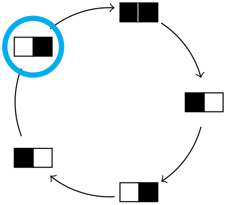


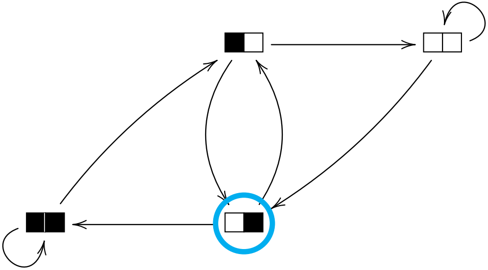


Cycle $\leftrightarrow$ State

There is a bijection between $\{\blacksquare, \square\}^w$ (with wrapping) and cycles in this graph. For example,

$x =$ .





Edge $\leftrightarrow$ Input

The key insight: The labels of the edges traversed are the inputs an inhomogeneous cellular automata will receive when run on x.

Edge ↔ Input

The key insight: The labels of the edges traversed are the inputs an inhomogeneous cellular automata will receive when run on x.

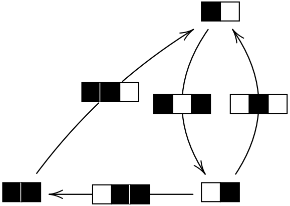
$$x = \square\blacksquare\blacksquare\square\blacksquare.$$

Edge ↔ Input

The key insight: The labels of the edges traversed are the inputs an inhomogeneous cellular automata will receive when run on x.

$x =$ .

Corresponds to

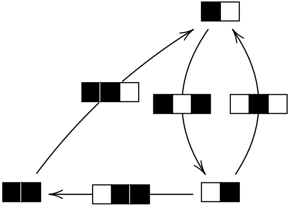


Edge $\leftrightarrow$ Input

The key insight: The labels of the edges traversed are the inputs an inhomogeneous cellular automata will receive when run on x.

$$x = □■■□■.$$

Corresponds to



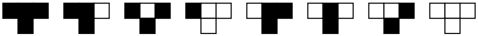
Corresponds to

■□■, □■■, ■■□, and □■□.

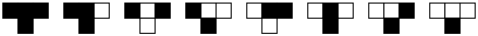
■-to-□ with λ_{238}, λ_{215}

Let's connect our graph with ability to transition.

λ_{238}



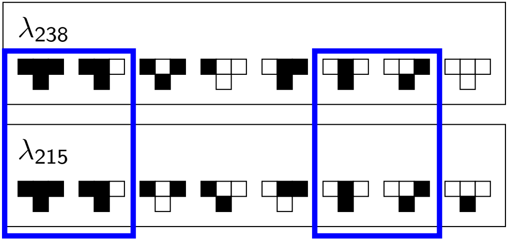
λ_{215}



Can these automata do the ■-to-□ transition?

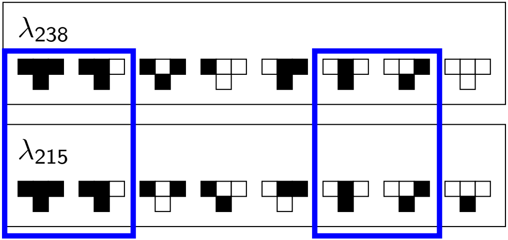
■-to-□ with λ_{238}, λ_{215}

Both output ■ on ■■■, ■■□, □■□, □□■.



■-to-□ with λ_{238}, λ_{215}

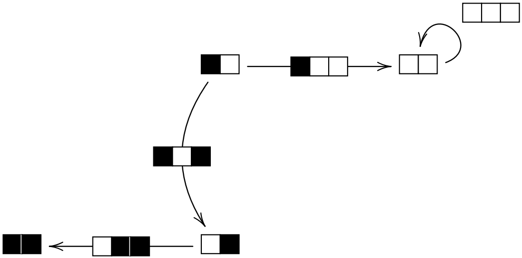
Both output ■ on ■■■, ■■□, □■□, □□■.



Let's remove those edges and see if we can still make an $x^{(0)}$.

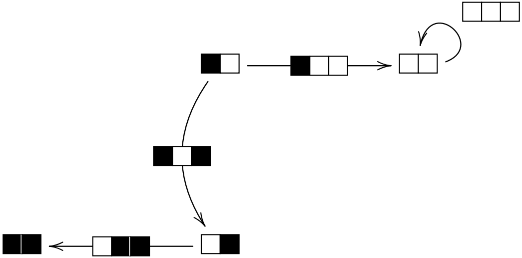
■-to-□ with λ_{238}, λ_{215}

λ_{238}, λ_{215}'s graph:



■-to-□ with λ_{238}, λ_{215}

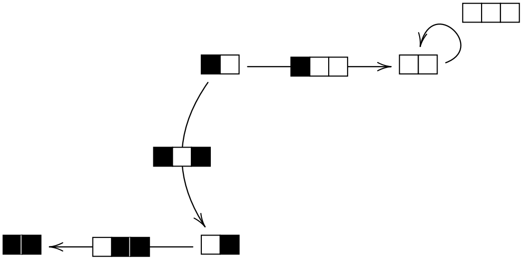
λ_{238}, λ_{215}'s graph:



There are no cycles with a ■.

■-to-□ with $\lambda_{238}, \lambda_{215}$

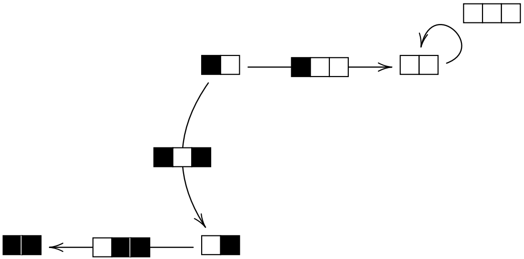
$\lambda_{238}, \lambda_{215}$'s graph:



So there are no states containing a ■ that $\lambda_{238}, \lambda_{215}$ aren't forced to output a ■ on.

■-to-□ with λ_{238}, λ_{215}

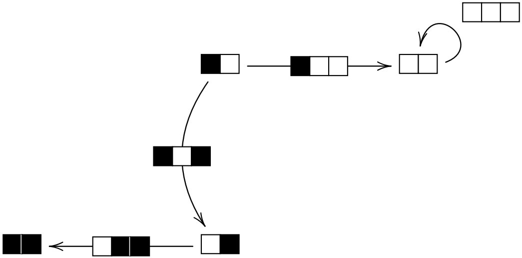
λ_{238}, λ_{215}'s graph:



So using λ_{238}, λ_{215} the ■-to-□ transition is impossible.

■-to-□ with λ_{238}, λ_{215}

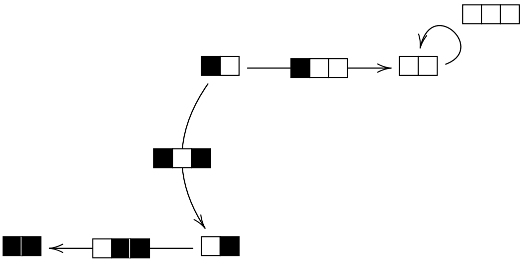
λ_{238}, λ_{215}'s graph:



So λ_{238}, λ_{215} cannot "die".

■-to-□ with λ_{238}, λ_{215}

λ_{238}, λ_{215}'s graph:



So λ_{238}, λ_{215} cannot Survive k Steps for any k.

When is ■-to-□ impossible?

Putting this together.

If we prune all edges from our graph labeled with a pattern both λ_i, λ_j output ■ on

When is ■-to-□ impossible?

Putting this together.

If we prune all edges from our graph labeled with a pattern both λ_i, λ_j output ■ on and there is cycle passing through a node with a ■,

When is ■-to-□ impossible?

Putting this together.
If we prune all edges from our graph labeled with a pattern both λ_i, λ_j output ■ on and there is cycle passing through a node with a ■, that cycle corresponds to $x^{(0)}$ s.t. $\blacksquare \in x^{(0)}$ and λ_i,λ_j can be composed to make $x^{(1)}$ all-□.

When is $\blacksquare$-to-$\square$ impossible?

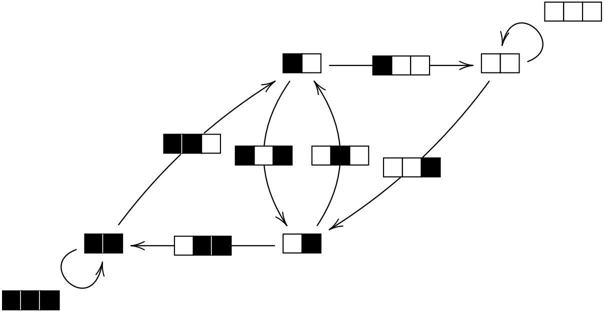
Putting this together.
If we prune all edges from our graph labeled with a pattern both λ_i, λ_j output $\blacksquare$ on and there is cycle passing through a node with a $\blacksquare$, that cycle corresponds to $x^{(0)}$ s.t. $\blacksquare \in x^{(0)}$ and λ_i,λ_j can be composed to make $x^{(1)}$ all-$\square$. I.e the. $\blacksquare$-to-$\square$ transition is possible.

When is ■-to-□ impossible?

Putting this together.
If we prune all edges from our graph labeled with a pattern both λ_i, λ_j output ■ on and there is cycle passing through a node with a ■, that cycle corresponds to $x^{(0)}$ s.t. $\blacksquare \in x^{(0)}$ and λ_i,λ_j can be composed to make $x^{(1)}$ all-□. I.e the. ■-to-□ transition is possible. **Q.E.D.**

Automating This

In the general case, checking for length-w cycles can be done efficiently.



Hint: Every cycle is a linear combination of simple cycles.

Conclusion

1. Our workshop paper:

Conclusion

1. Our workshop paper:
   1.1 Can't "die" $\rightarrow$ can't survive k steps (`technique2`).

Conclusion

1. Our workshop paper:
   1.1 Can't "die" $\rightarrow$ can't survive k steps (`technique2`).
   1.2 i,j are odd $\rightarrow$ can't survive k steps (`technique1`).

Conclusion

1. Our workshop paper:
   1.1 Can't "die" $\rightarrow$ can't survive k steps (`technique2`).
   1.2 i,j are odd $\rightarrow$ can't survive k steps (`technique1`).
   1.3 Classify 67% of automata tuples.

Conclusion

1. Our workshop paper:
   1.1 Can't "die" $\rightarrow$ can't survive k steps (`technique2`).
   1.2 i,j are odd $\rightarrow$ can't survive k steps (`technique1`).
   1.3 Classify 67% of automata tuples.
2. Forthcoming work:

Conclusion

1. Our workshop paper:
   1.1 Can't "die" $\rightarrow$ can't survive k steps (`technique2`).
   1.2 i,j are odd $\rightarrow$ can't survive k steps (`technique1`).
   1.3 Classify 67% of automata tuples.
2. Forthcoming work:
   2.1 Classifies $> 99\%$ of automata tuples.

Conclusion

1. Our workshop paper:
   1.1 Can't "die" $\rightarrow$ can't survive k steps (`technique2`).
   1.2 i,j are odd $\rightarrow$ can't survive k steps (`technique1`).
   1.3 Classify 67% of automata tuples.
2. Forthcoming work:
   2.1 Classifies $>$ 99% of automata tuples.
   2.2 Efficient automata synthesis with SAT solver.

Conclusion

1. Our workshop paper:
   1.1 Can't "die" $\rightarrow$ can't survive k steps (`technique2`).
   1.2 i,j are odd $\rightarrow$ can't survive k steps (`technique1`).
   1.3 Classify 67% of automata tuples.
2. Forthcoming work:
   2.1 Classifies $>$ 99% of automata tuples.
   2.2 Efficient automata synthesis with SAT solver.
   2.3 Precisely what SAT features explain performance.