On the flora of asynchronous locally non-monotonic Boolean automata networks

A study of xor-networks

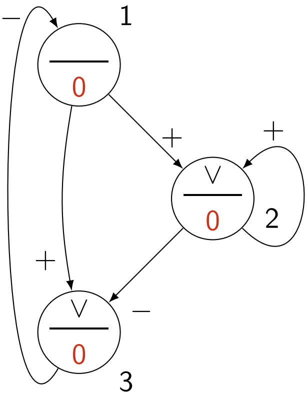
Aurore Alcolei, Kévin Perrot and Sylvain Sené

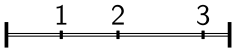
CANA team, LIF, Aix-Marseille University

SASB - 8th September 2015

Asynchronous Boolean automata networks (BANs)

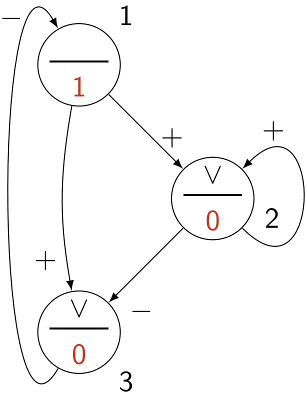
- $\mathcal{N} = \{f_i : \mathbb{B}^n \to \mathbb{B}\}_{i=1}^n$ (size n)
- configuration:
  $x = (x_1, \ldots, x_n) \in \mathbb{B}^n$
- eg: $f_1(x) = \neg x_3$, $f_2(x) = x_1 \vee x_2$,
  $f_3(x) = x_1 \vee \neg x_2$
- model for regulation systems
  (gene or neural networks)
- Asynchronous dynamics:
  - non deterministic and complete,
  - one automaton is updated at a time

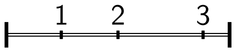




Asynchronous Boolean automata networks (BANs)

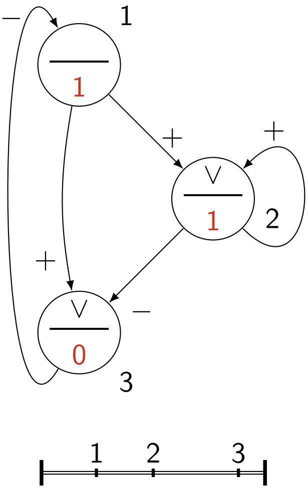
- $\mathcal{N} = \{f_i : \mathbb{B}^n \to \mathbb{B}\}_{i=1}^n$ (size n)
- configuration: $x = (x_1, \ldots, x_n) \in \mathbb{B}^n$
- eg: $f_1(x) = \neg x_3$, $f_2(x) = x_1 \vee x_2$, $f_3(x) = x_1 \vee \neg x_2$
- model for regulation systems (gene or neural networks)
- Asynchronous dynamics:
  - non deterministic and complete,
  - one automaton is updated at a time





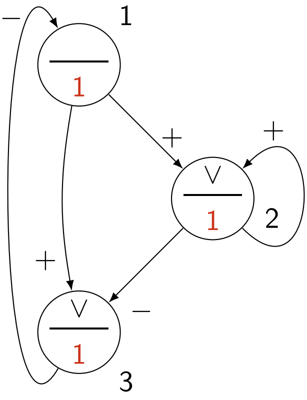
Asynchronous Boolean automata networks (BANs)

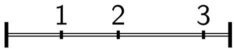
- $\mathcal{N} = \{f_i : \mathbb{B}^n \to \mathbb{B}\}_{i=1}^n$ (size n)
- configuration:
  $x = (x_1, \dots, x_n) \in \mathbb{B}^n$
- eg: $f_1(x) = \neg x_3$, $f_2(x) = x_1 \vee x_2$,
  $f_3(x) = x_1 \vee \neg x_2$
- model for regulation systems
  (gene or neural networks)
- Asynchronous dynamics:
  - non deterministic and complete,
  - one automaton is updated at a time



Asynchronous Boolean automata networks (BANs)

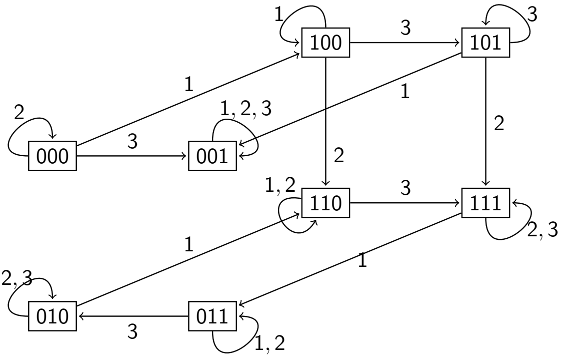
- $\mathcal{N} = \{f_i : \mathbb{B}^n \to \mathbb{B}\}_{i=1}^n$ (size n)
- configuration:
  $x = (x_1, \ldots, x_n) \in \mathbb{B}^n$
- eg: $f_1(x) = \neg x_3$, $f_2(x) = x_1 \vee x_2$,
  $f_3(x) = x_1 \vee \neg x_2$
- model for regulation systems
  (gene or neural networks)
- Asynchronous dynamics:
  - non deterministic and complete,
  - one automaton is updated at a time





Asynchronous transition graph

- nodes = configurations
  arrows = transitions

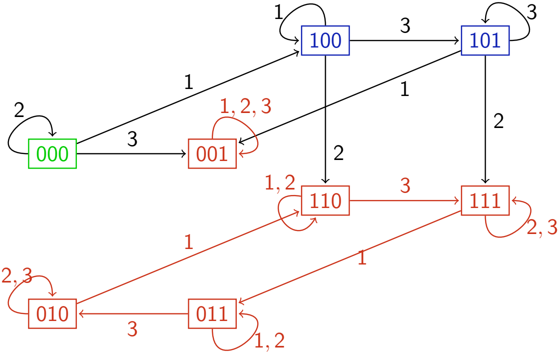


Asynchronous transition graph

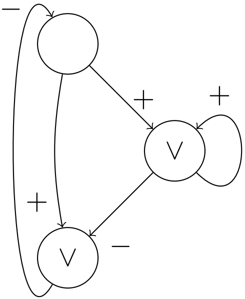
- nodes = configurations
  arrows = transitions
- configurations:
  - recurrent:
  fixed point and
  stable oscillations.

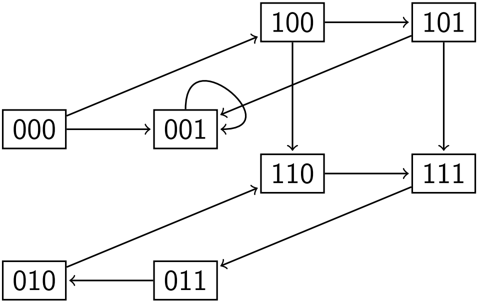
  - unreachable.

  - transient
  (reversible or
  irreversible).



Asynchronous dynamics of Boolean automata networks

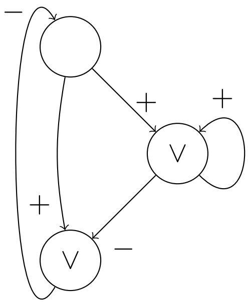


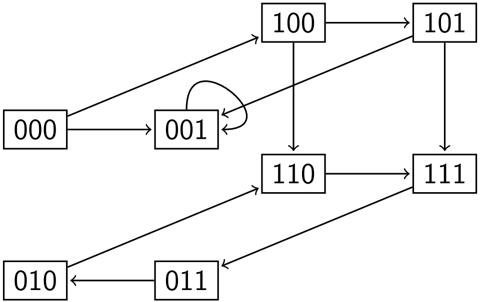


Questions:

What can we say about the dynamics (transition graph) of a BAN when only looking at its static definition? How do they relate?

Asynchronous dynamics of Boolean automata networks



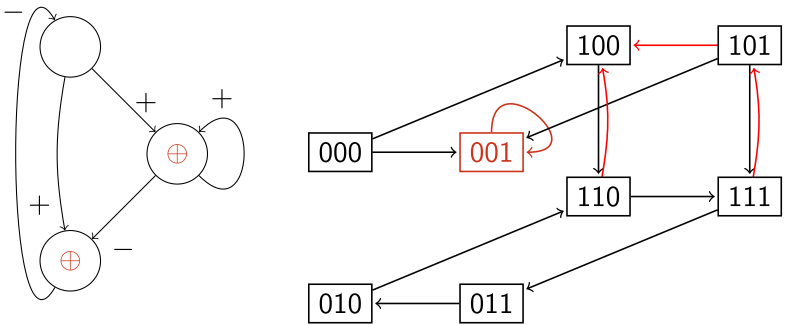


Questions:

- Usually locally monotonic.

$\rightarrow$ A questionable restriction for the expressiveness of the model.

Asynchronous dynamics of xor-Boolean automata networks



Questions:

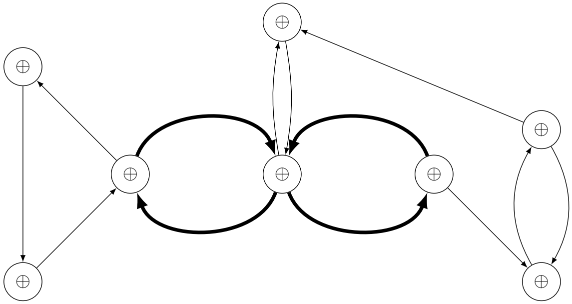
→ What is the impact of non local monotony on the dynamics of BANs?

→ Study of ⊕-networks.

A general result

Theorem [Strong connectivity/High expressiveness]:

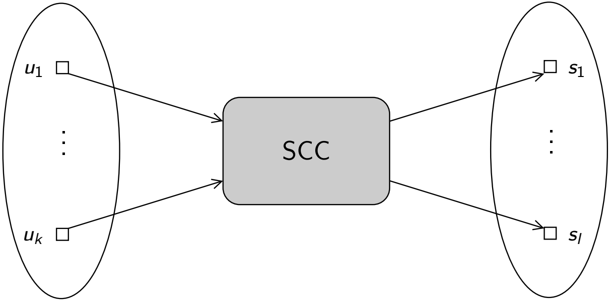
In a strongly connected ⊕-BAN with an **induced double cycle** of size greater than 3, one can go from any unstable configuration to any reachable configuration in a quadratic number of updates.



A general result

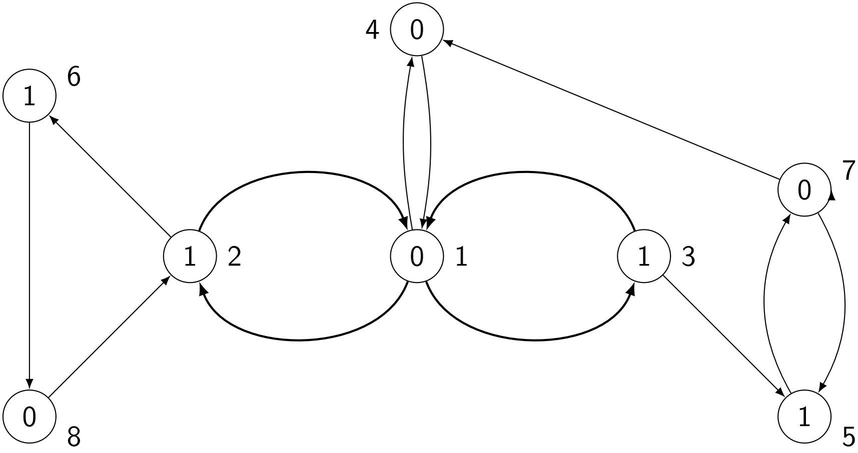
Theorem [Strong connectivity/High expressiveness]:

In a strongly connected ⊕-BAN with an **induced double cycle** of size greater than 3, one can go from any unstable configuration to any reachable configuration in a quadratic number of updates.



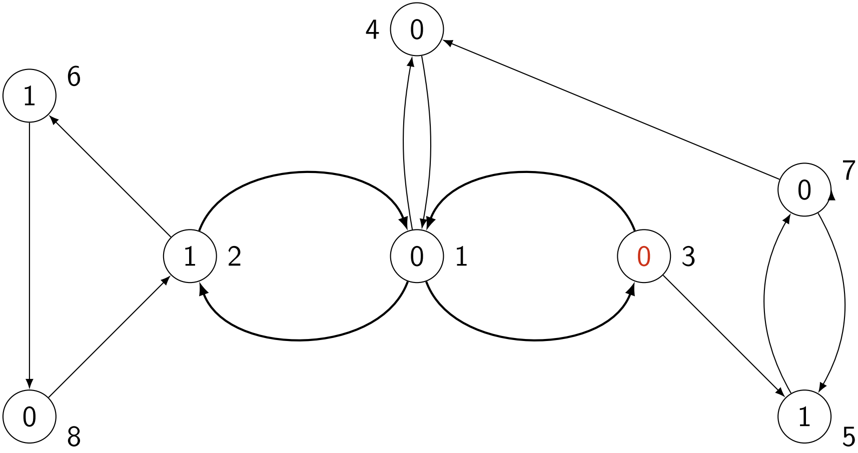
An algorithmic proof (sketch)

Idea: Using the induced double cycle as a state generator.
Eg: 01101100 $\to$ 01010011



An algorithmic proof (sketch)

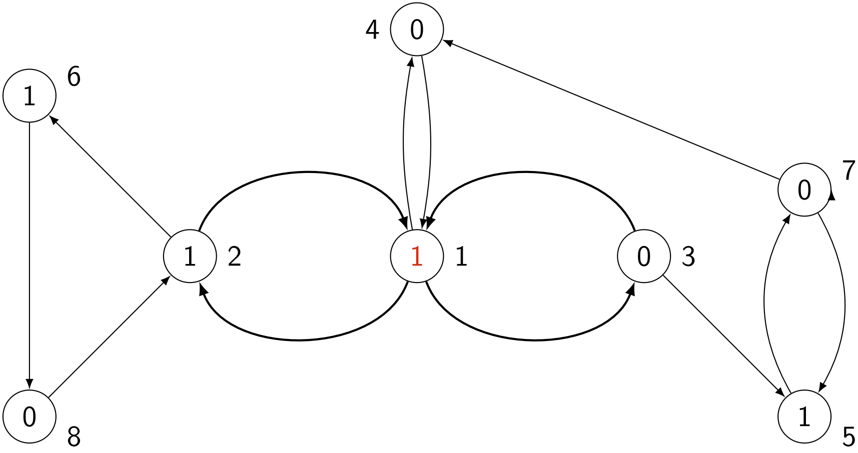
Idea: Using the induced double cycle as a state generator.
Eg: 01101100 → 01010011



An algorithmic proof (sketch)

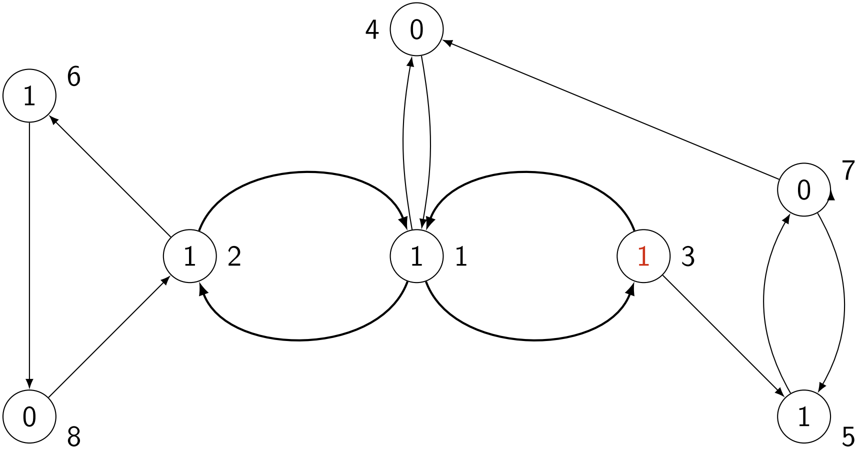
Idea: Using the induced double cycle as a state generator.

Eg: $01101100 \rightarrow 01010011$



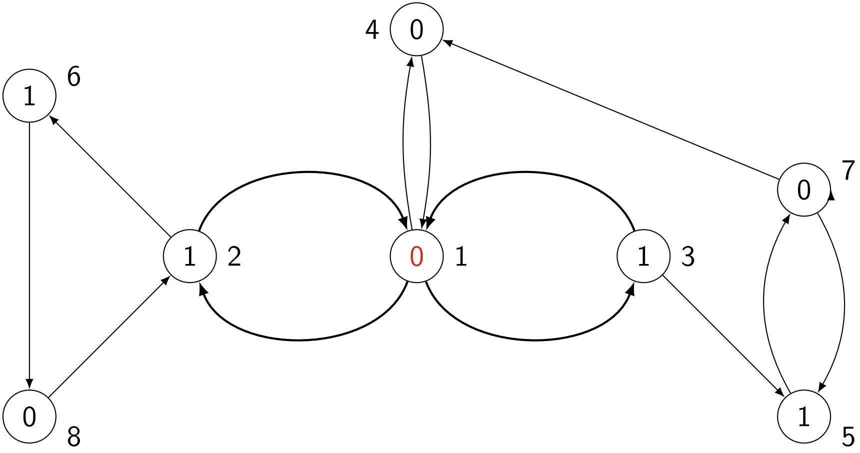
An algorithmic proof (sketch)

Idea: Using the induced double cycle as a state generator.
Eg: 01101100 → 01010011



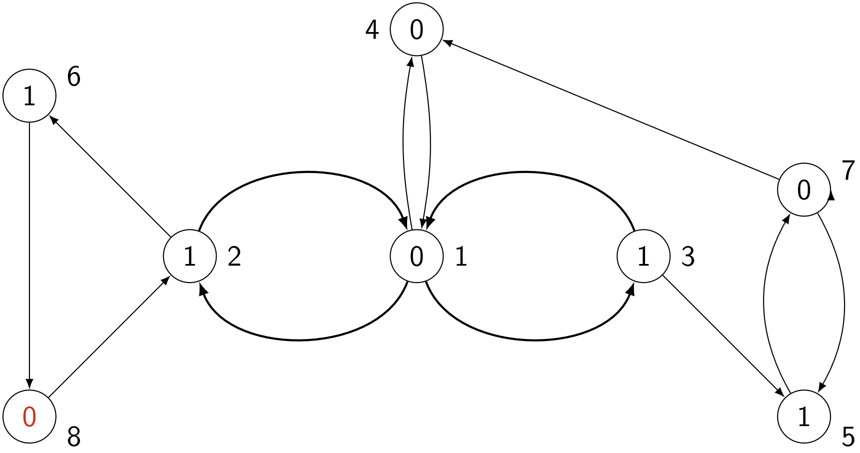
An algorithmic proof (sketch)

Idea: Using the induced double cycle as a state generator.
Eg: 01101100 → 01010011



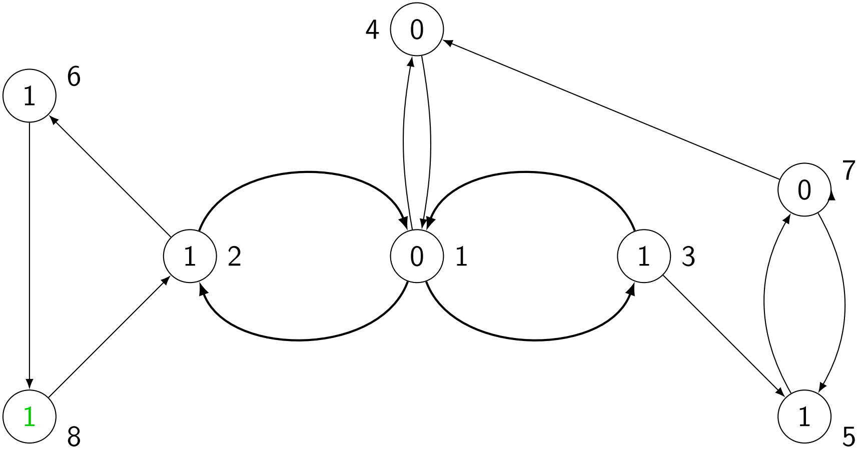
An algorithmic proof (sketch)

Idea: Using the induced double cycle as a state generator.
Eg: 01101100 → 01010011



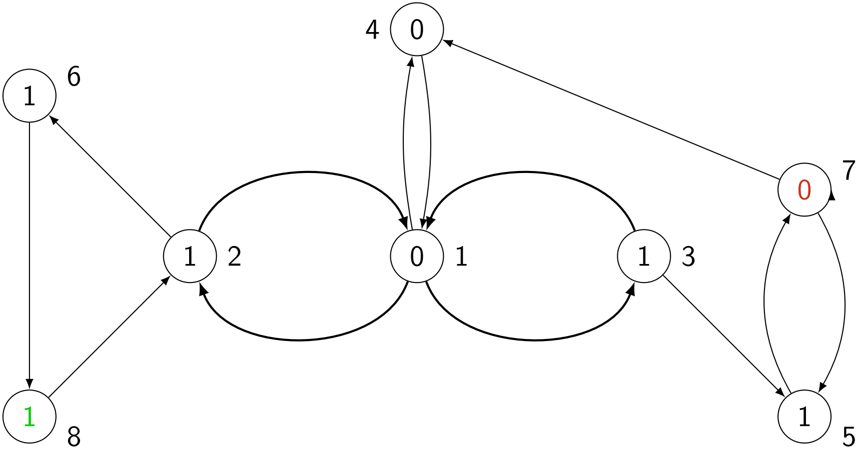
An algorithmic proof (sketch)

Idea: Using the induced double cycle as a state generator.
Eg: 01101100 → 01010011



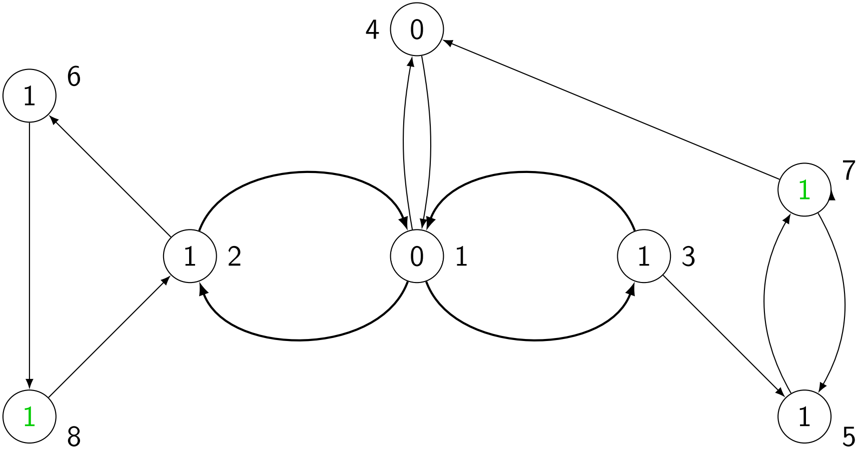
An algorithmic proof (sketch)

Idea: Using the induced double cycle as a state generator.
Eg: 01101100 $\rightarrow$ 01010011



An algorithmic proof (sketch)

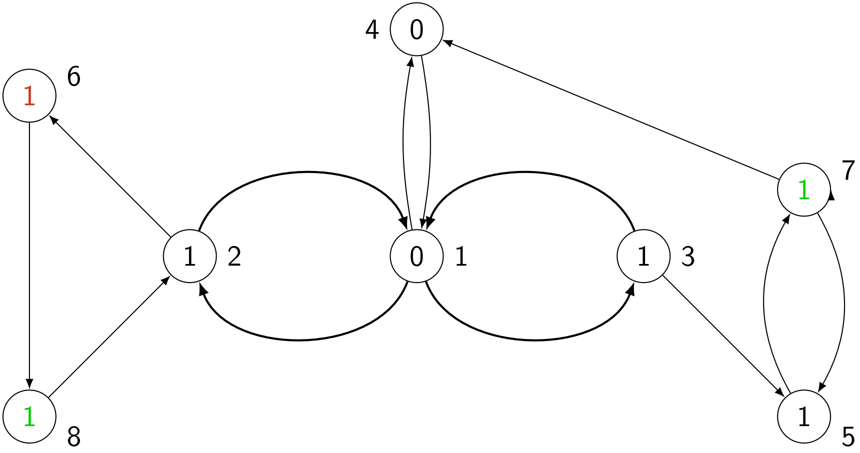
Idea: Using the induced double cycle as a state generator.
Eg: 01101100 → 01010011



An algorithmic proof (sketch)

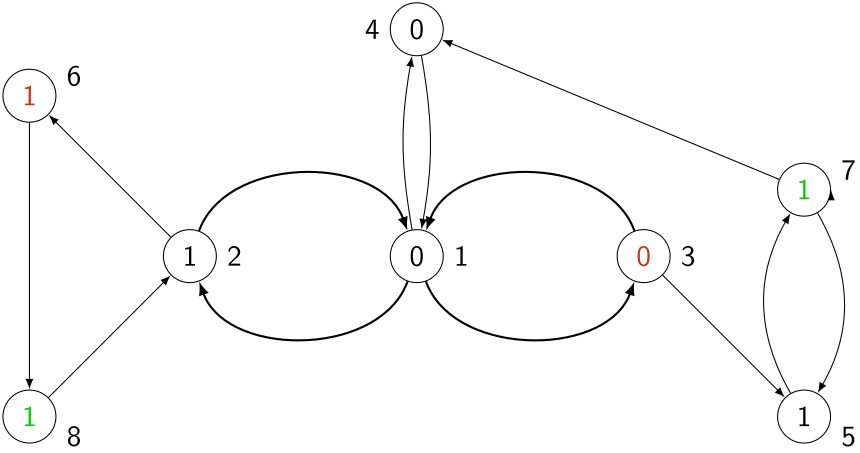
Idea: Using the induced double cycle as a state generator.

Eg: 01101100 → 01010011



An algorithmic proof (sketch)

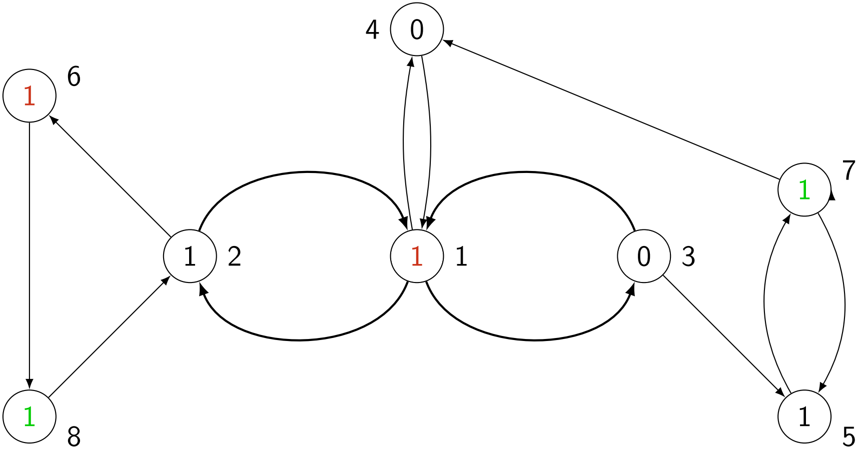
Idea: Using the induced double cycle as a state generator.
Eg: 01101100 → 01010011



An algorithmic proof (sketch)

Idea: Using the induced double cycle as a state generator.

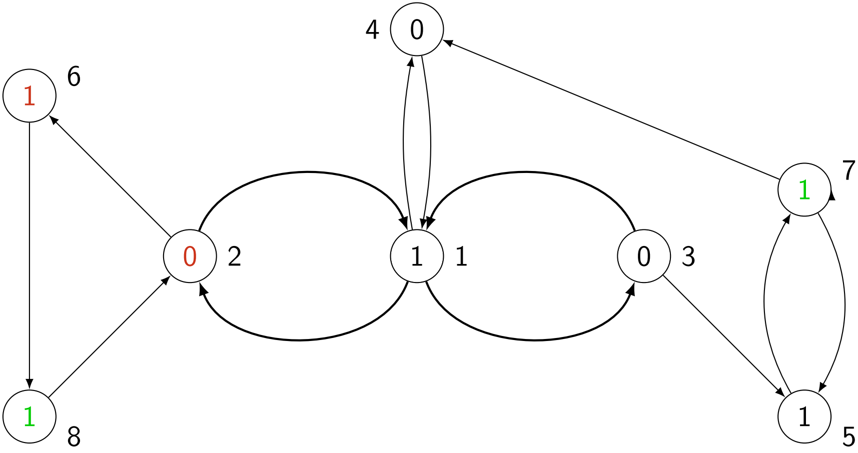
Eg: 01101100 → 01010011



An algorithmic proof (sketch)

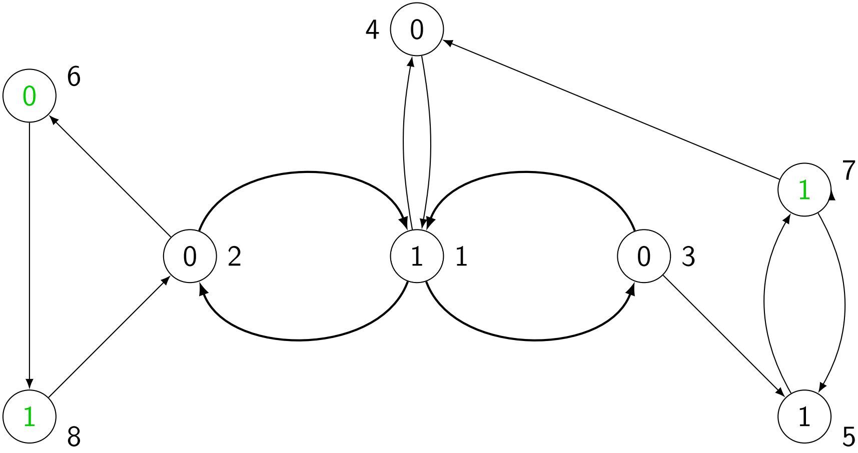
Idea: Using the induced double cycle as a state generator.

Eg: 01101100 → 01010011



An algorithmic proof (sketch)

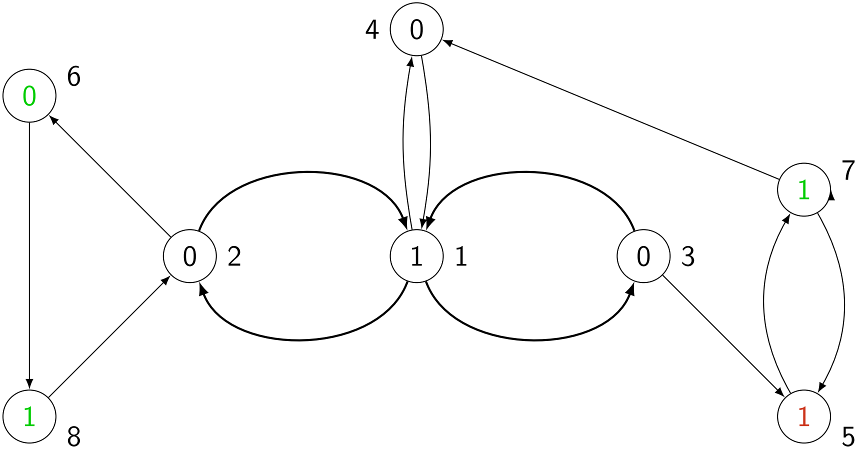
Idea: Using the induced double cycle as a state generator.
Eg: 01101100 → 01010011



An algorithmic proof (sketch)

Idea: Using the induced double cycle as a state generator.

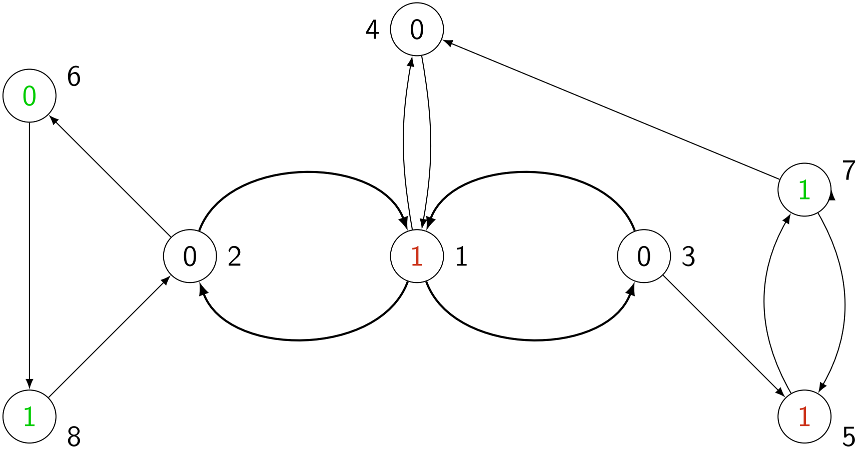
Eg: 01101100 → 01010011



An algorithmic proof (sketch)

Idea: Using the induced double cycle as a state generator.

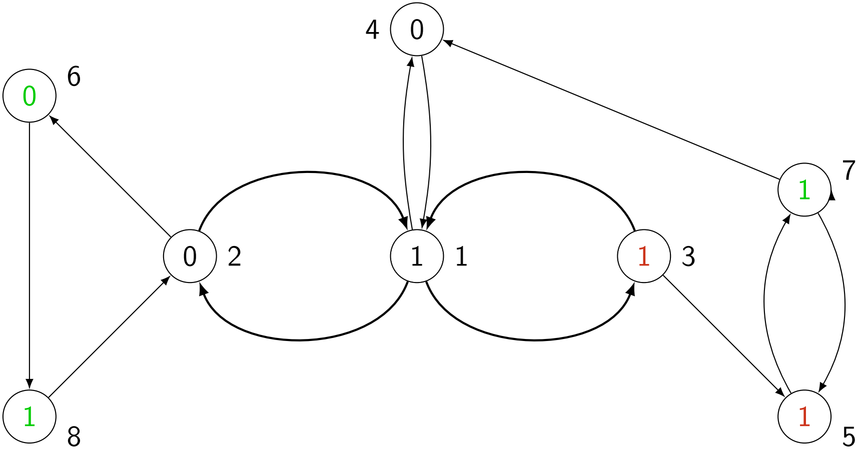
Eg: 01101100 → 01010011



An algorithmic proof (sketch)

Idea: Using the induced double cycle as a state generator.

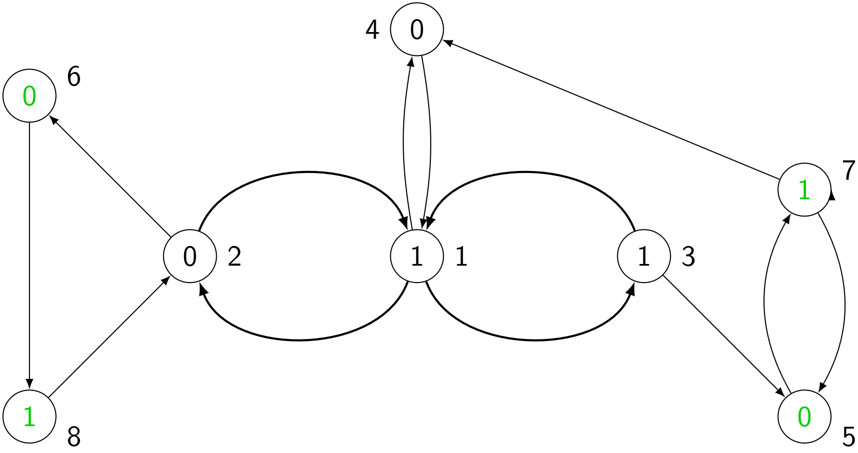
Eg: 01101100 → 01010011



An algorithmic proof (sketch)

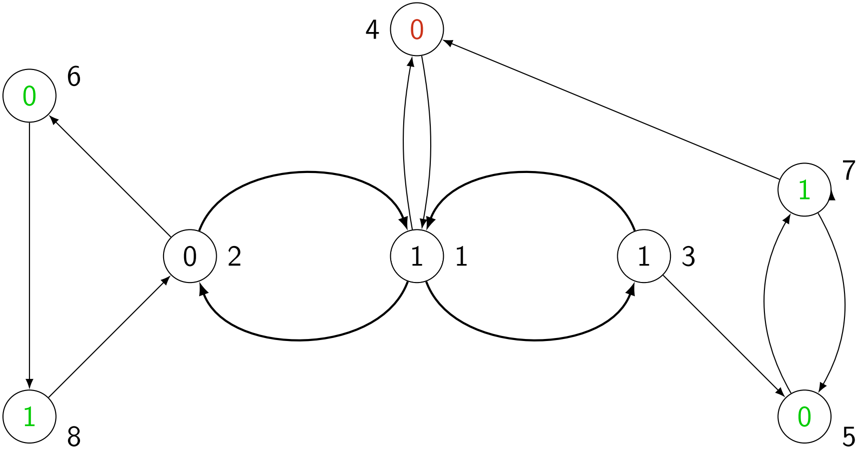
Idea: Using the induced double cycle as a state generator.

Eg: 01101100 → 01010011



An algorithmic proof (sketch)

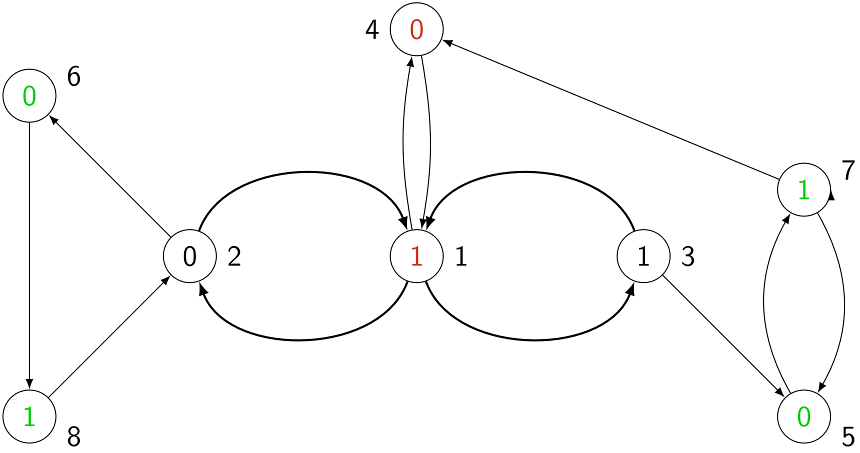
Idea: Using the induced double cycle as a state generator.
Eg: 01101100 → 01010011



An algorithmic proof (sketch)

Idea: Using the induced double cycle as a state generator.

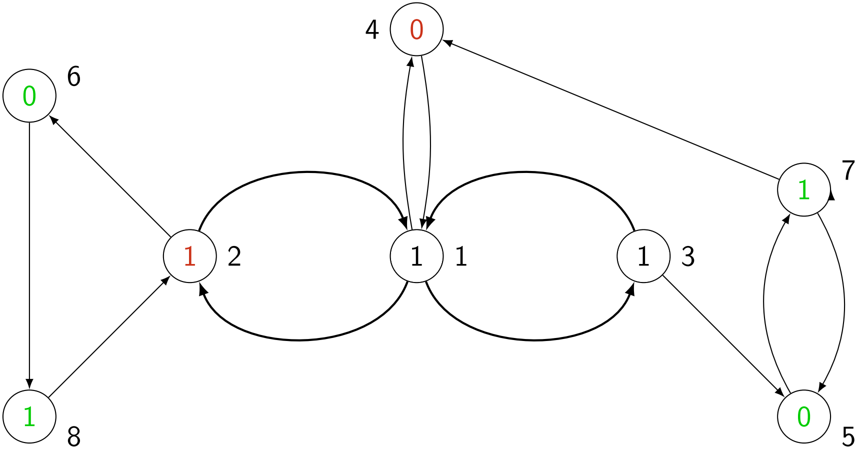
Eg: 01101100 → 01010011



An algorithmic proof (sketch)

Idea: Using the induced double cycle as a state generator.

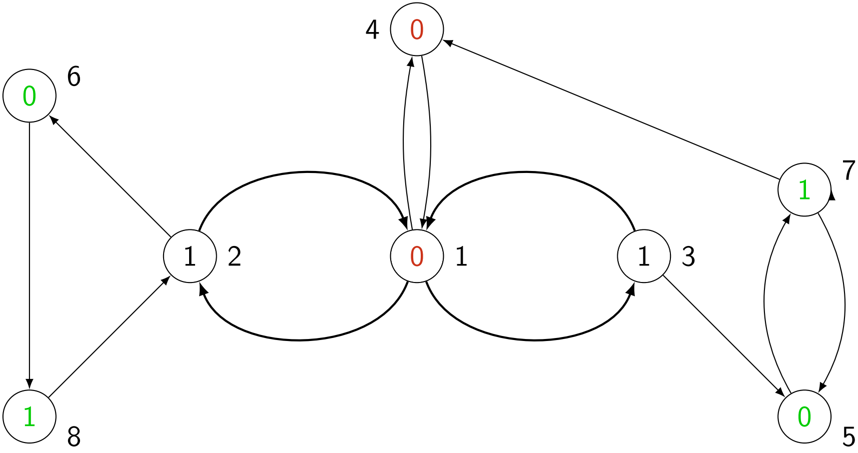
Eg: 01101100 → 01010011



An algorithmic proof (sketch)

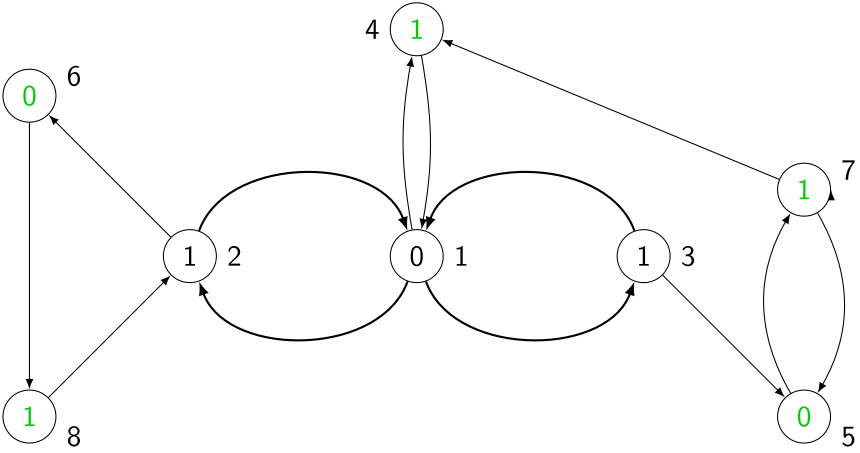
Idea: Using the induced double cycle as a state generator.

Eg: 01101100 → 01010011



An algorithmic proof (sketch)

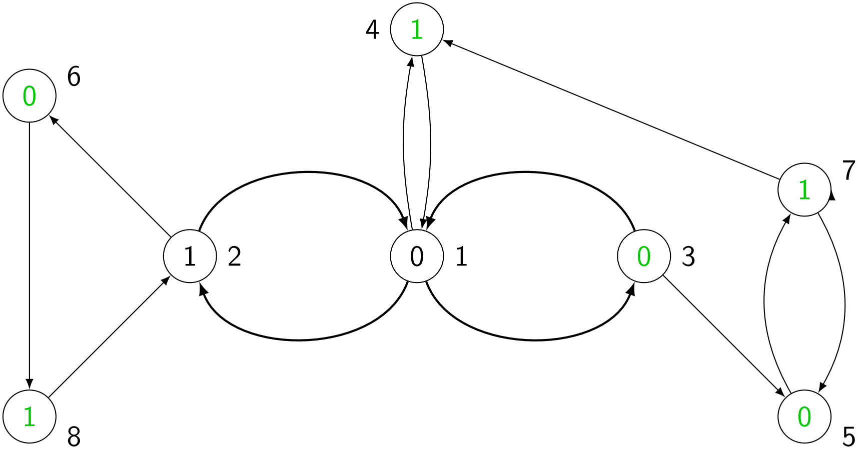
Idea: Using the induced double cycle as a state generator.
Eg: 01101100 → 01010011



An algorithmic proof (sketch)

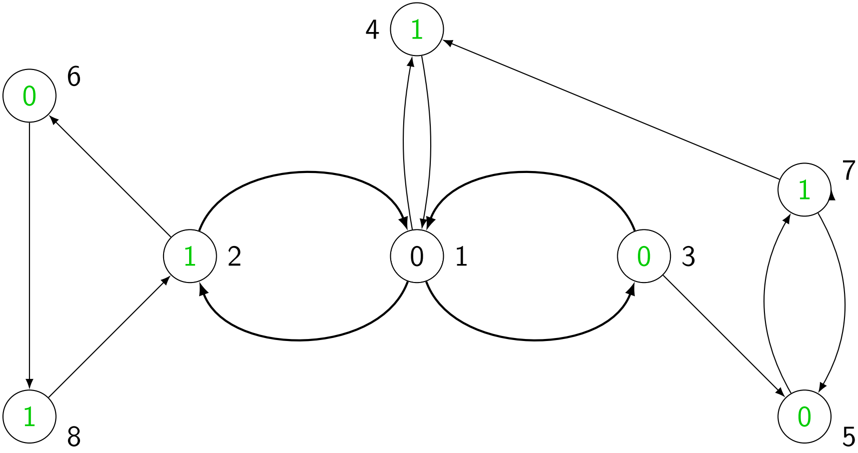
Idea: Using the induced double cycle as a state generator.

Eg: 01101100 → 01010011



An algorithmic proof (sketch)

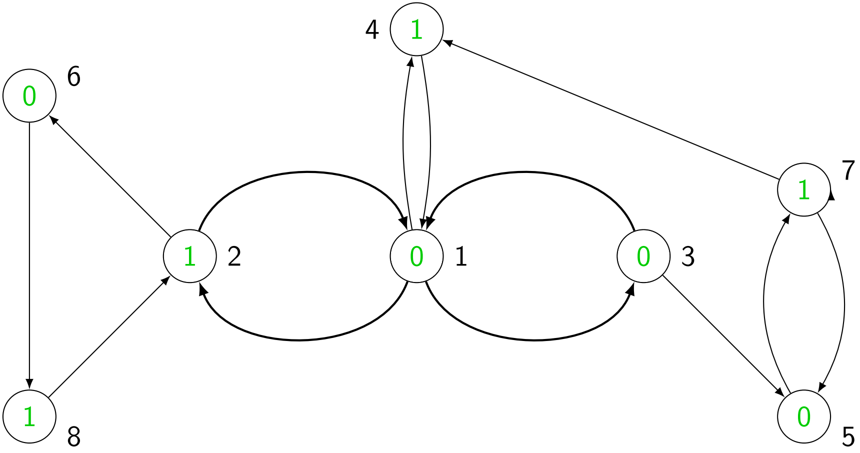
Idea: Using the induced double cycle as a state generator.
Eg: 01101100 → 01010011



An algorithmic proof (sketch)

Idea: Using the induced double cycle as a state generator.

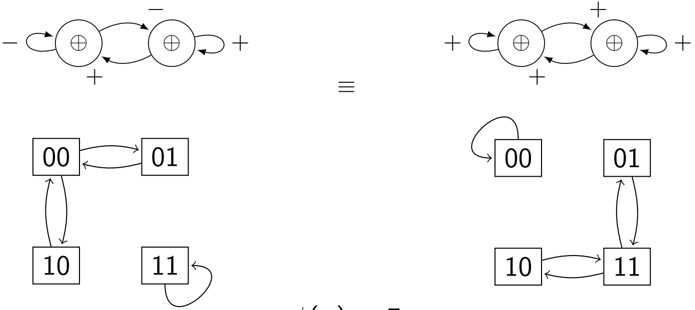
Eg: 01101100 → 01010011



Isomorphism definition

Isomorphism relation:

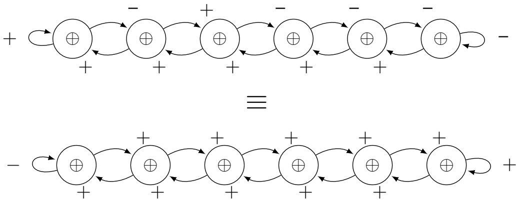
Two BANs are isomorphic if their transition graphs are isomorphic.



$$\phi(x) = \bar{x}$$

Isomorphism relation and rewritings

Property: Any cycle chain structure induces at most two classes of isomorphic $\oplus$-networks.

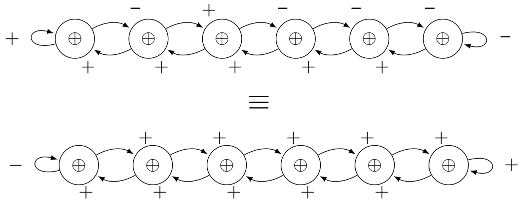


Proof sketch:

1. a set of rewriting rules that preserve the isomorphism relation.
2. rewrites that converge to canonical networks.

Isomorphism relation and rewritings

Property: Any cycle chain structure induces at most two classes of isomorphic ⊕-networks.



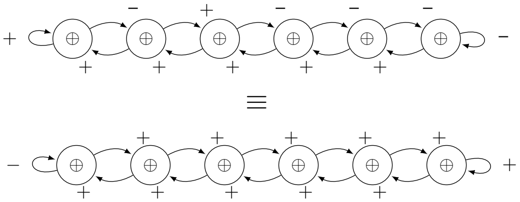
Proof sketch:

1 a set of rewriting rules that preserve the isomorphism relation.

2 rewrites that converge to canonical networks.

Isomorphism relation and rewritings

Property: Any cycle chain structure induces at most two classes of isomorphic ⊕-networks.



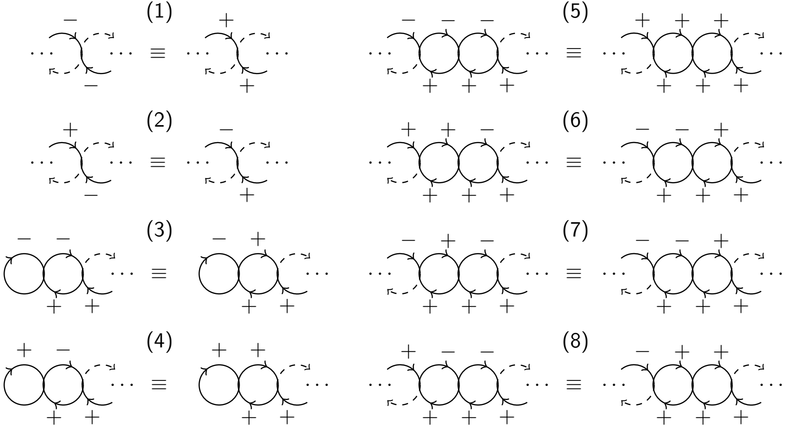
Proof sketch:

1 a set of rewriting rules that preserve the isomorphism relation.

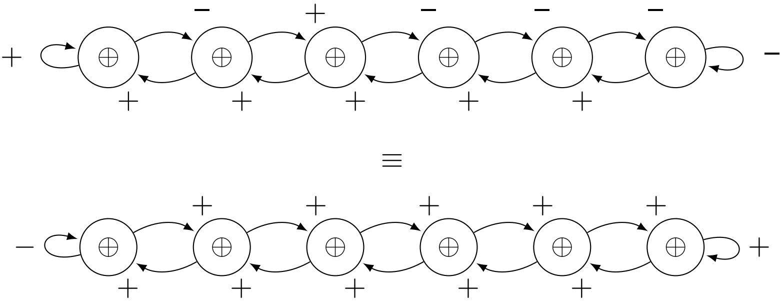
2 rewrites that converge to canonical networks.

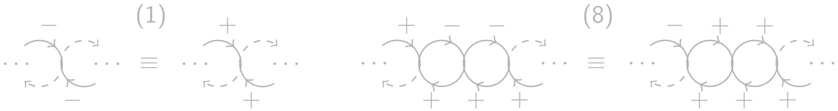
Rewriting rules

Idea: removing or pushing the rightmost − sign to the left.
Meaning: The networks induced by the right and left patterns are isomorphic.

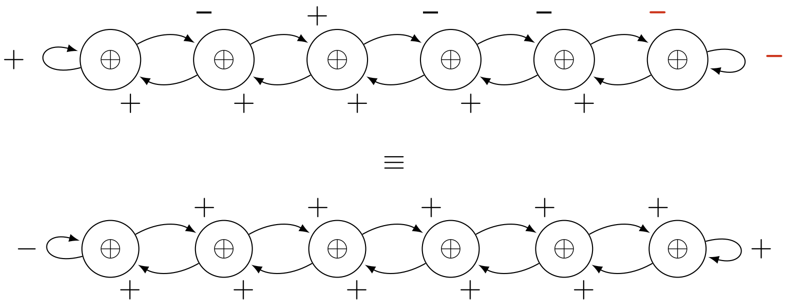


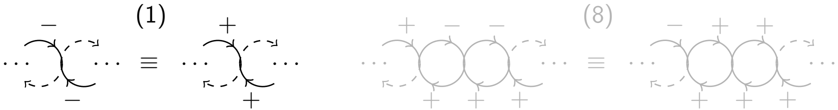
An example



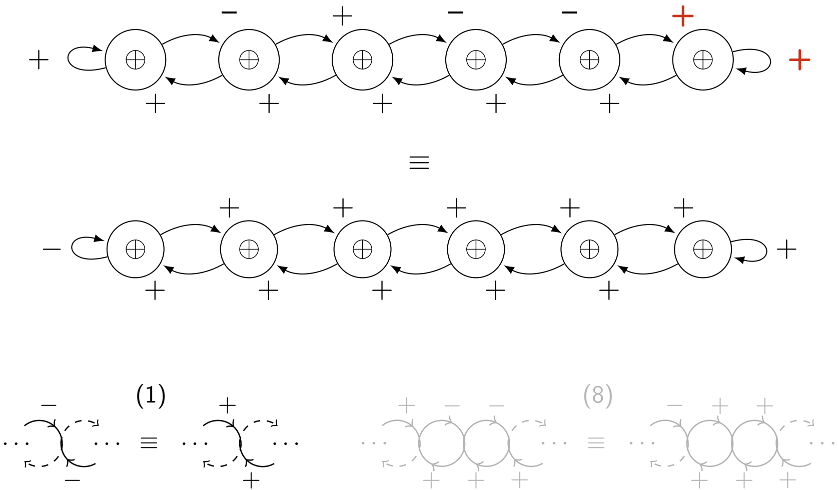


An example

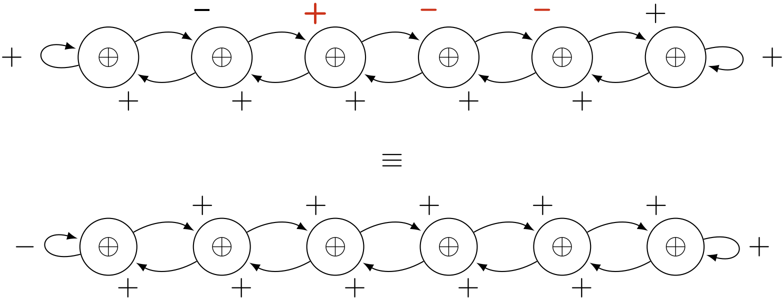


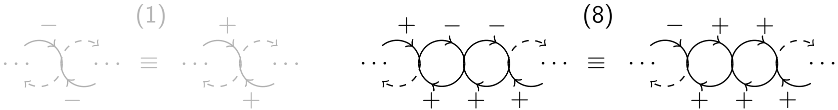


An example

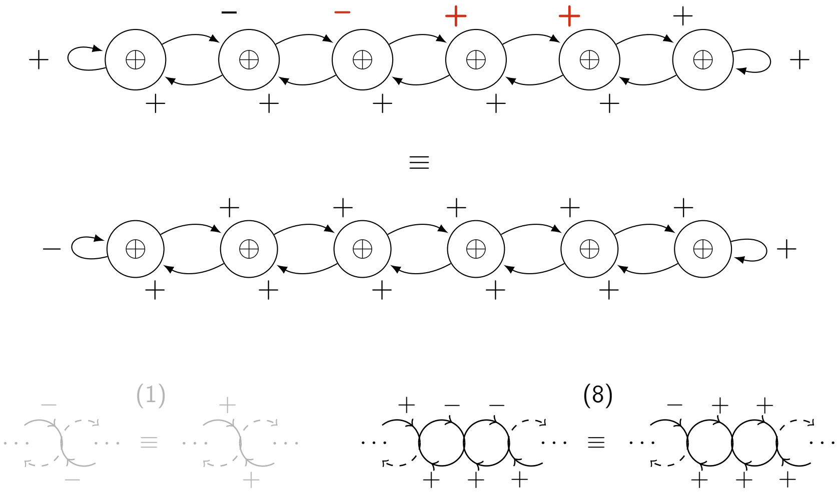


An example

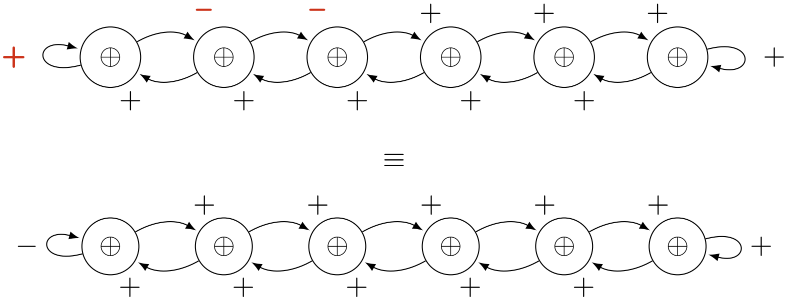


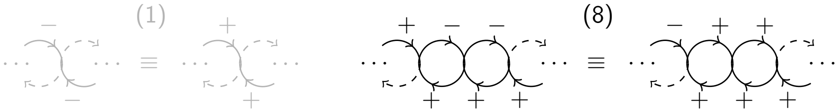


An example

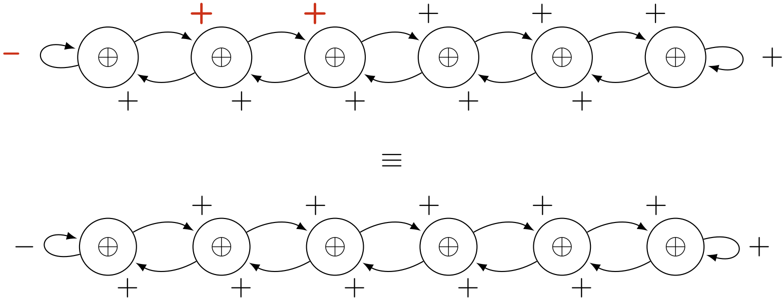


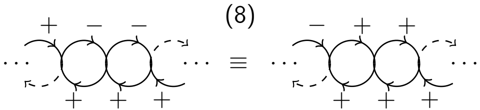
An example





An example





Explaining Equivalence 8

Goals:

1- To remove the rightmost $-$ sign without changing anything on the right,

2- and by changing the least number of automata on the left.

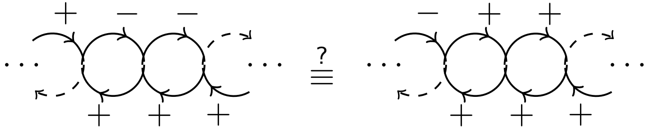
Idea: Finding a morphism that can be expressed locally, that is, $\phi(x) = (\phi_1(x_1), \ldots, \phi_n(x_n))$ such that for all i:

- ϕ_i is invertible
- $\phi_i(f_i(x)) = f_i'(\phi(x))$

$\phi_3 = \ ?$

$\phi_2 = \ ?$

$\phi_1 = \ ?$



Explaining Equivalence 8

Goals:

1- To remove the rightmost $-$ sign without changing anything on the right,

2- and by changing the least number of automata on the left.

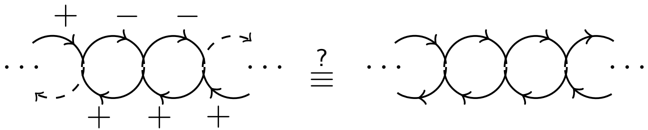
Idea: Finding a morphism that can be expressed locally, that is, $\phi(x) = (\phi_1(x_1), \ldots, \phi_n(x_n))$ such that for all i:

- ϕ_i is invertible
- $\phi_i(f_i(x)) = f_i'(\phi(x))$

$\phi_3 =$?

$\phi_2 =$?

$\phi_1 =$?



Explaining Equivalence 8

Goals:

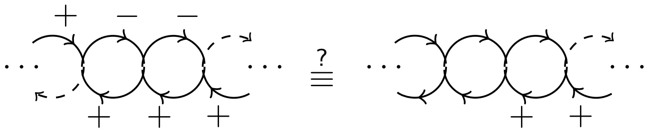
1- To remove the rightmost − sign without changing anything on the right,

2- and by changing the least number of automata on the left.

Idea: Finding a morphism that can be expressed locally, that is, $\phi(x) = (\phi_1(x_1), \ldots, \phi_n(x_n))$ such that for all i:

- ϕ_i is invertible
- $\phi_i(f_i(x)) = f_i'(\phi(x))$

$\phi_3 = \text{id}$
$\phi_2 = \ ?$
$\phi_1 = \ ?$



Explaining Equivalence 8

Goals:

1- To remove the rightmost $-$ sign without changing anything on the right,

2- and by changing the least number of automata on the left.

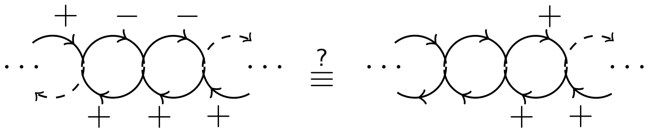
Idea: Finding a morphism that can be expressed locally, that is,
$\phi(x) = (\phi_1(x_1), \ldots, \phi_n(x_n))$ such that for all i:

- ϕ_i is invertible
- $\phi_i(f_i(x)) = f_i'(\phi(x))$

$\phi_3 =$ id

$\phi_2 =$ neg

$\phi_1 =$?



Explaining Equivalence 8

Goals:

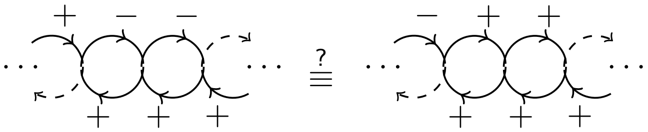
1- To remove the rightmost − sign without changing anything on the right,

2- and by changing the least number of automata on the left.

Idea: Finding a morphism that can be expressed locally, that is, $\phi(x) = (\phi_1(x_1), \ldots, \phi_n(x_n))$ such that for all i:

- ϕ_i is invertible
- $\phi_i(f_i(x)) = f'_i(\phi(x))$

$\phi_3 =$ id
$\phi_2 =$ neg
$\phi_1 =$ id ?



Conclusion

- Analysis of the asynchronous dynamics of a large number of strongly connected $\oplus$-networks.

$\rightarrow$ enrichment with larger class of BANs (non $\oplus$, non strongly connected...)

$\rightarrow$ different possible interpretations:

· propagation of contradictory information (entropy generator),

· ability to recover from "bad choices" (convergence to fixed points rather than stable oscillations).

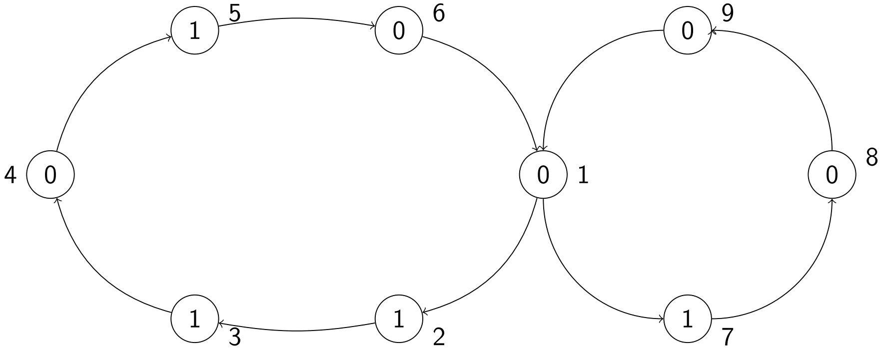
- Two useful tools:

$\rightarrow$ algorithmic formalism,

$\rightarrow$ equivalence relations to classify BANs (eg: isomorphism relation).

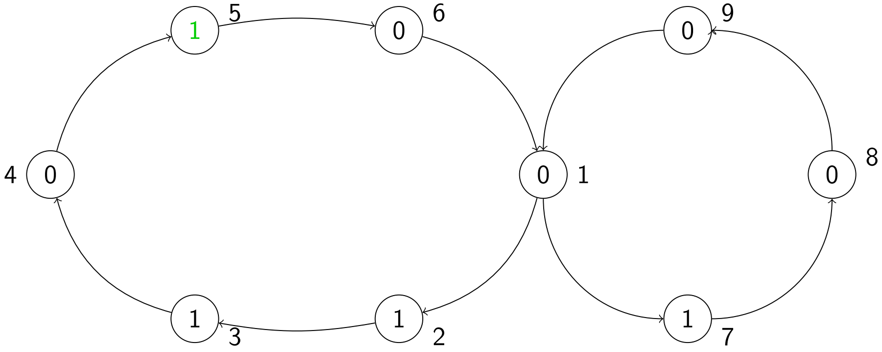
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



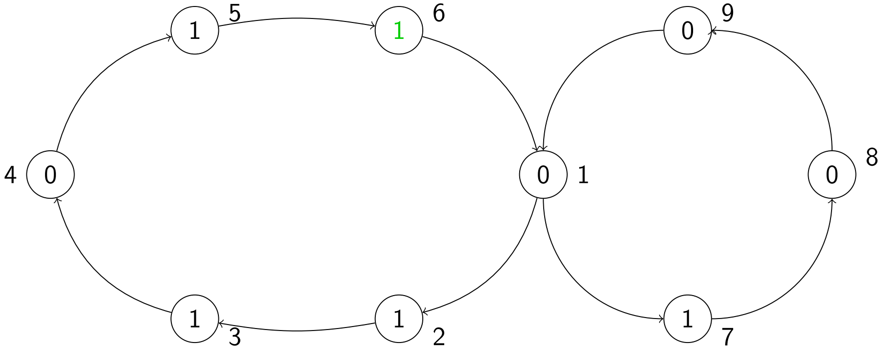
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



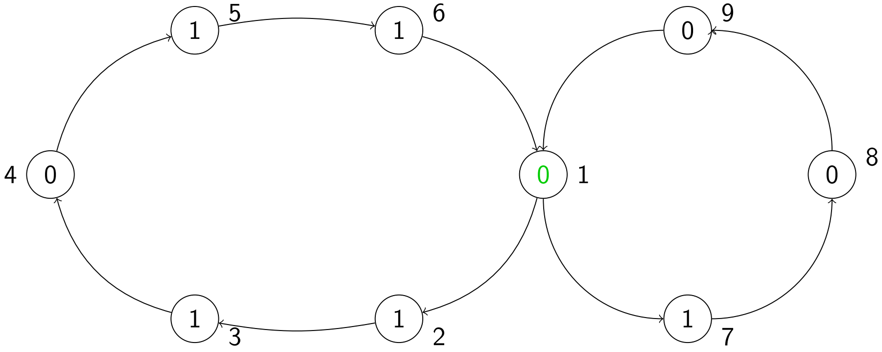
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



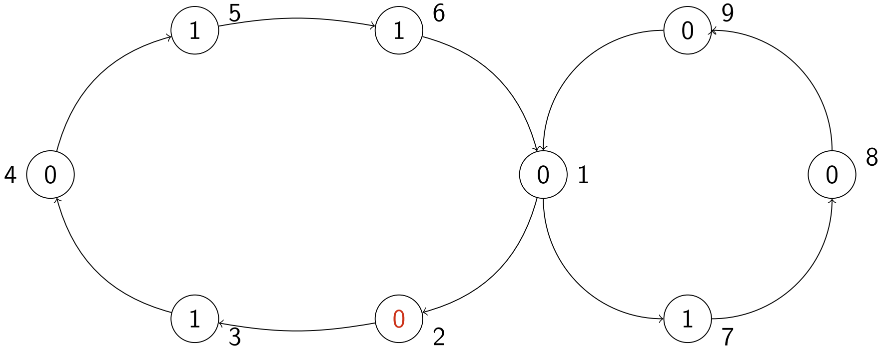
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



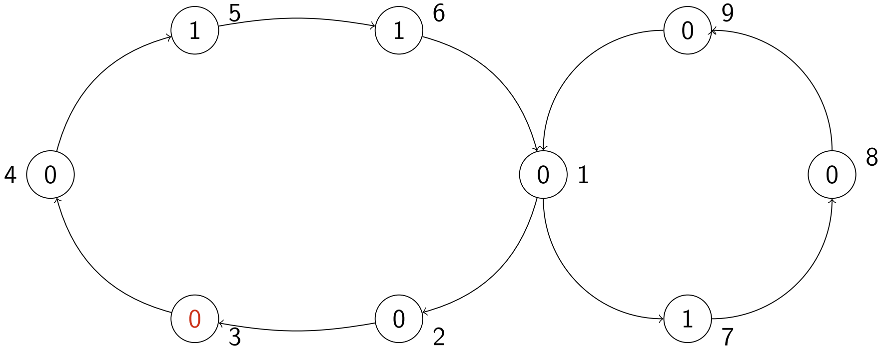
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



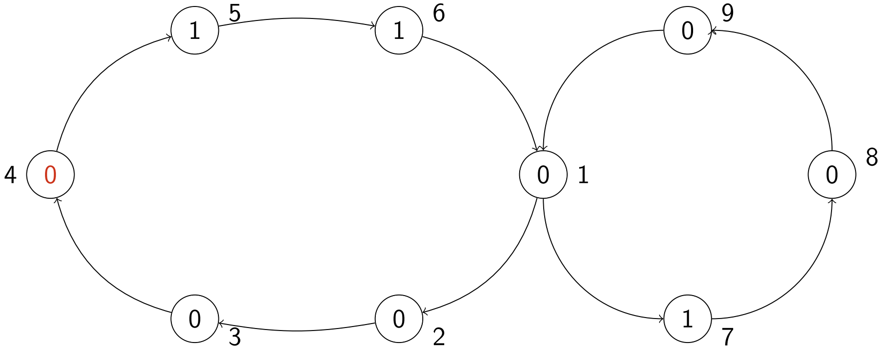
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



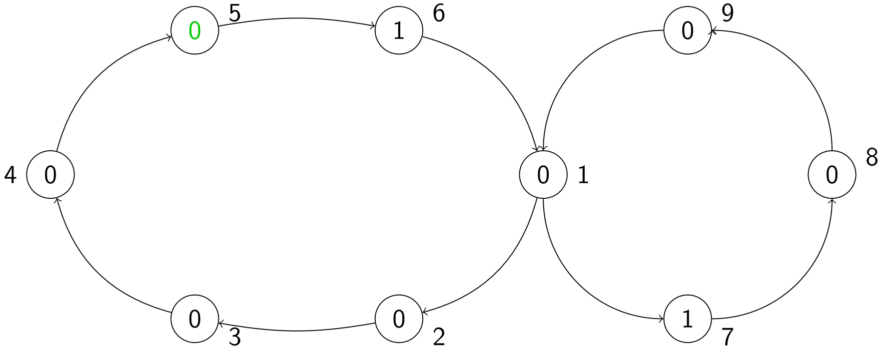
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



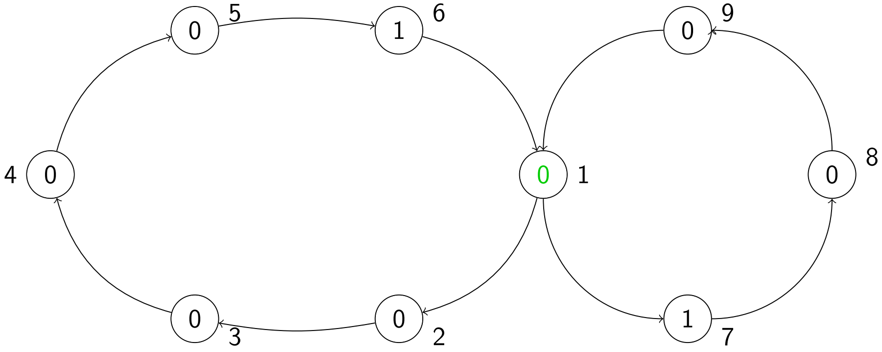
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



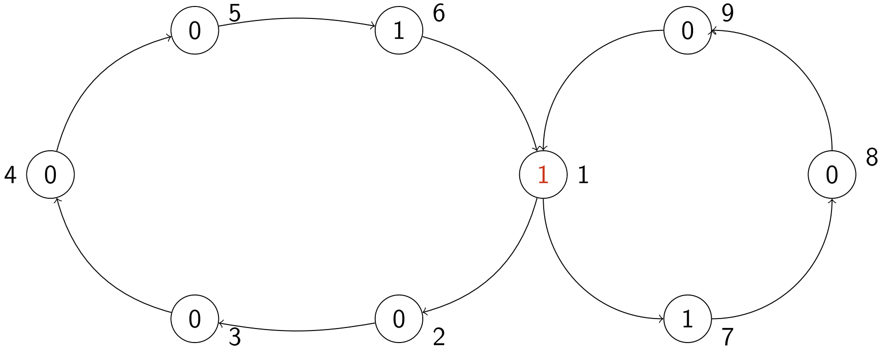
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



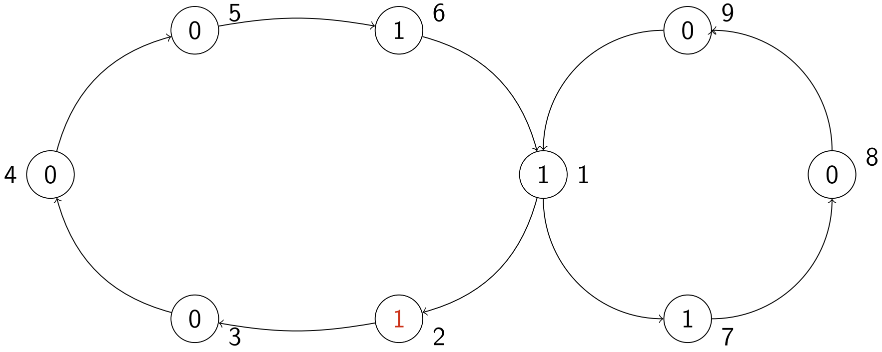
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



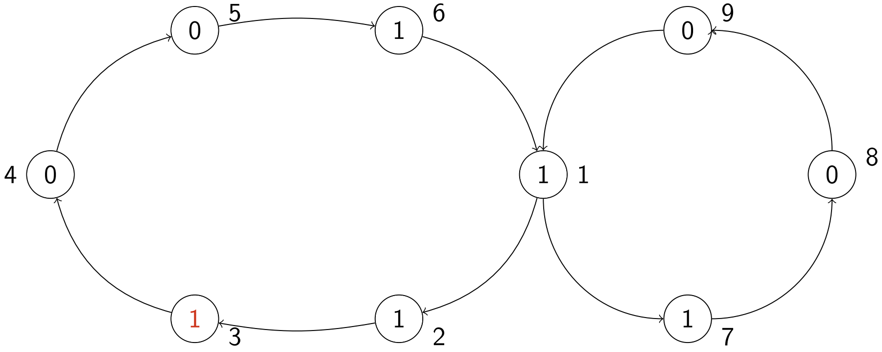
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



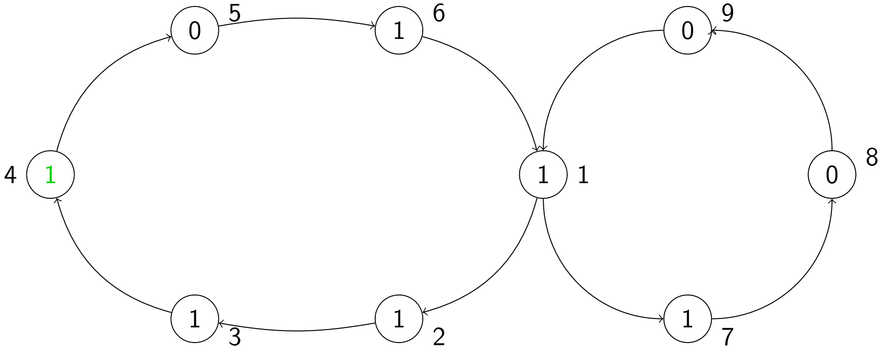
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



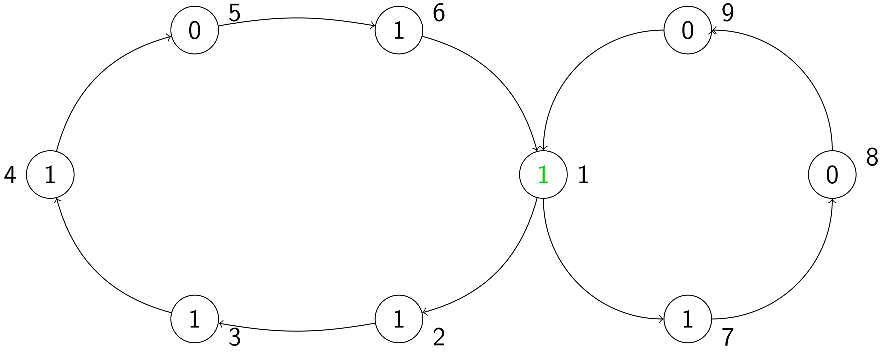
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



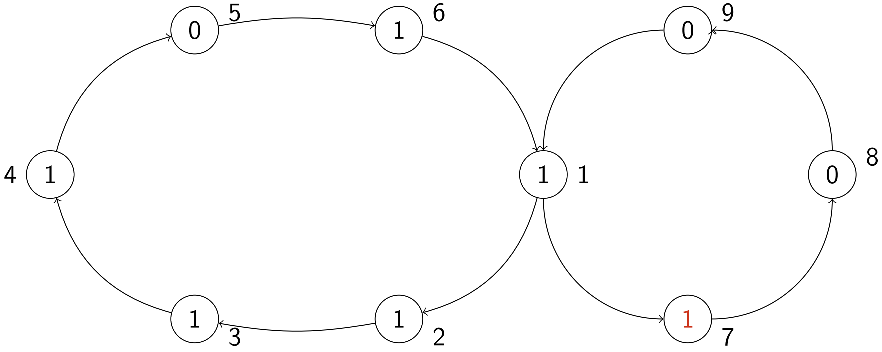
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



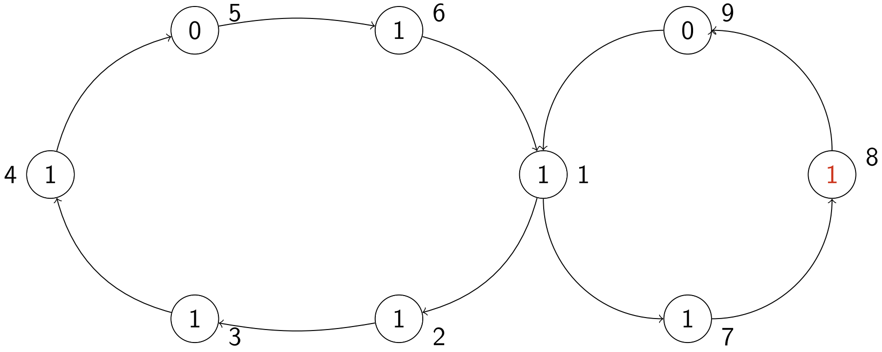
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



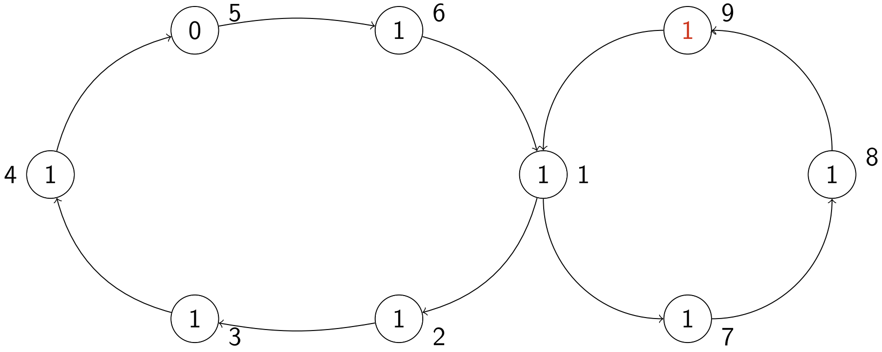
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



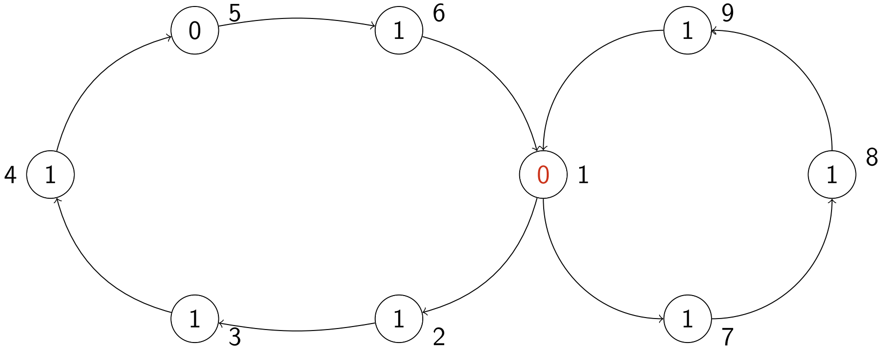
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



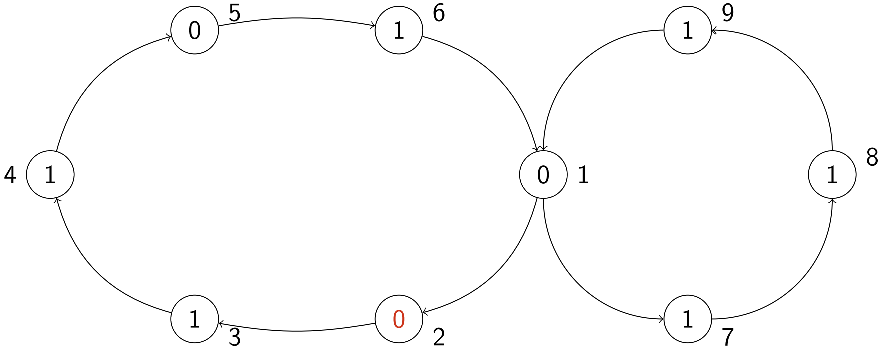
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



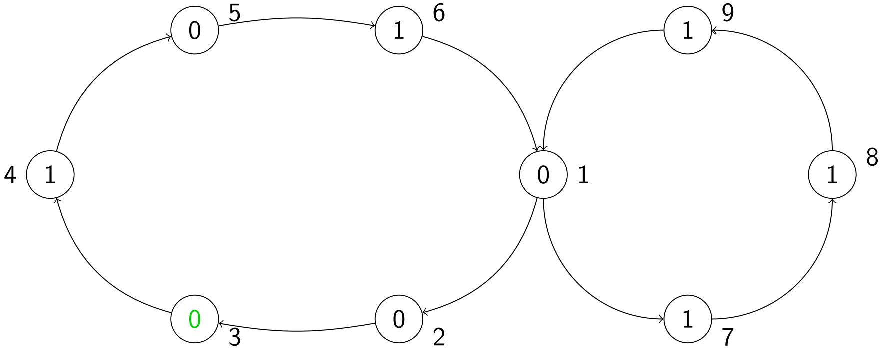
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



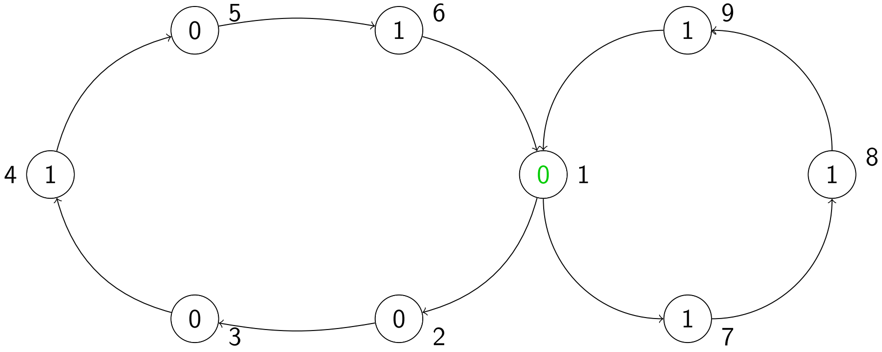
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



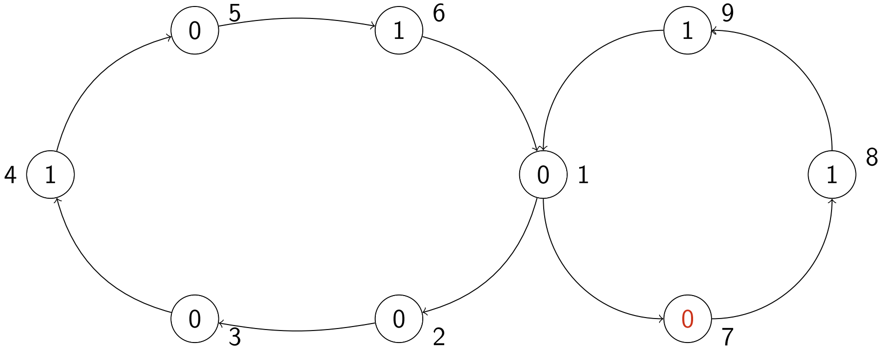
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



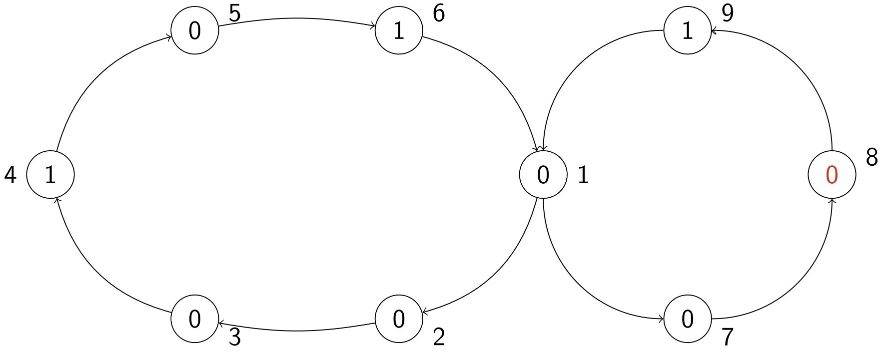
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



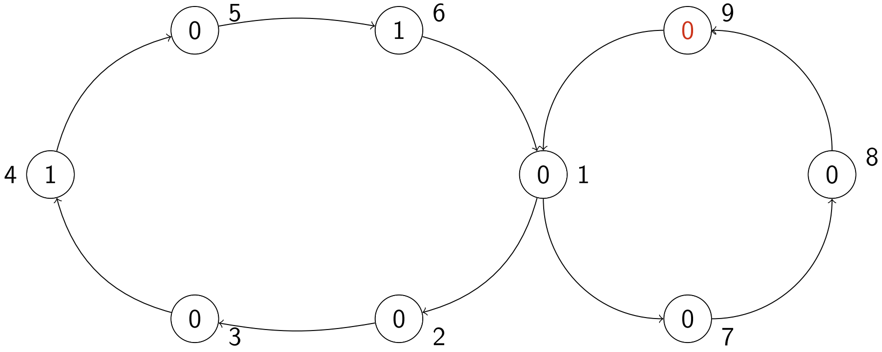
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



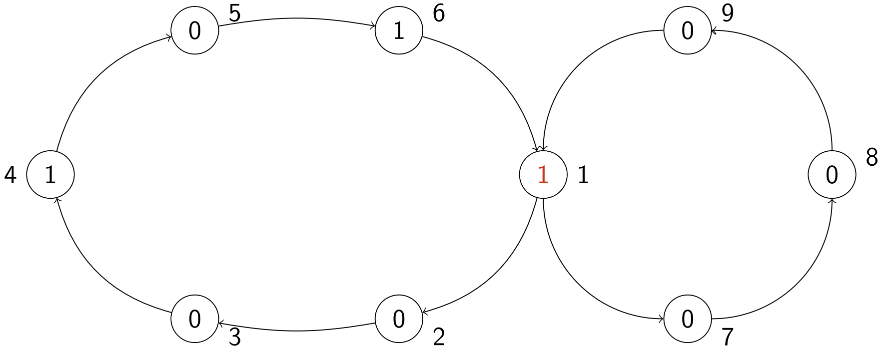
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



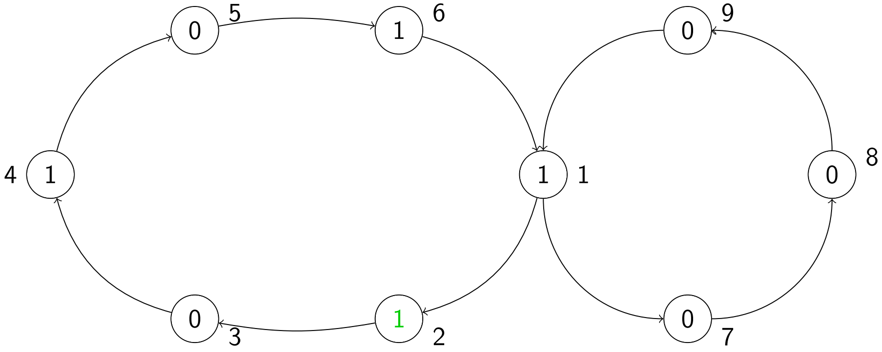
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



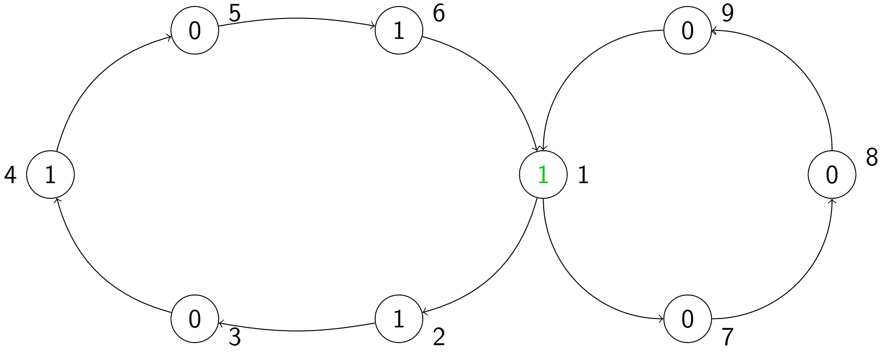
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



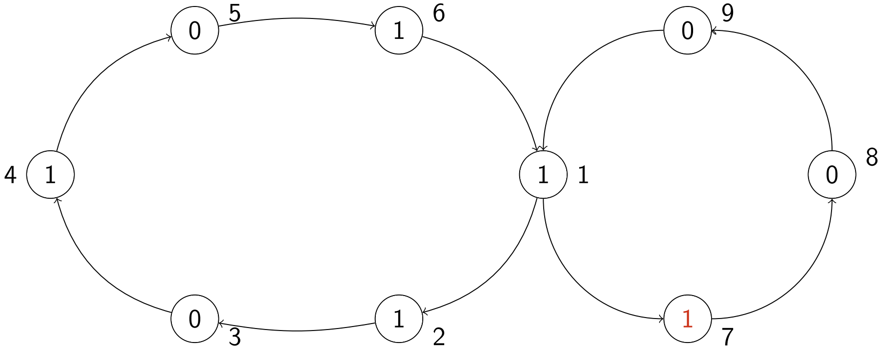
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



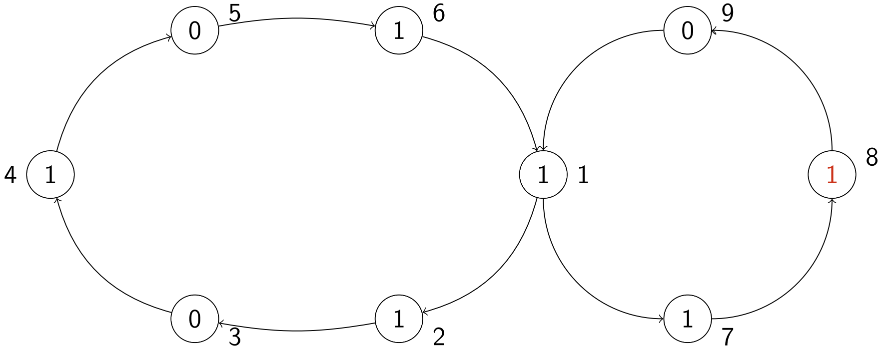
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



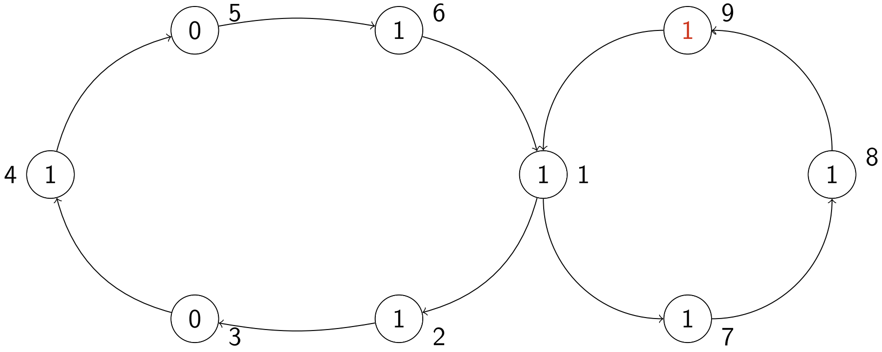
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



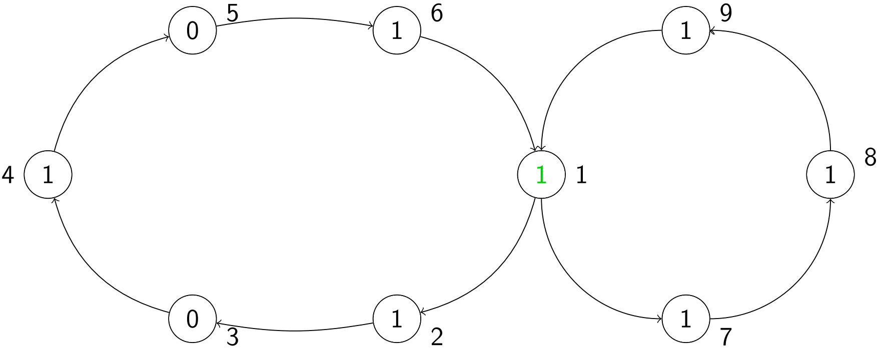
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



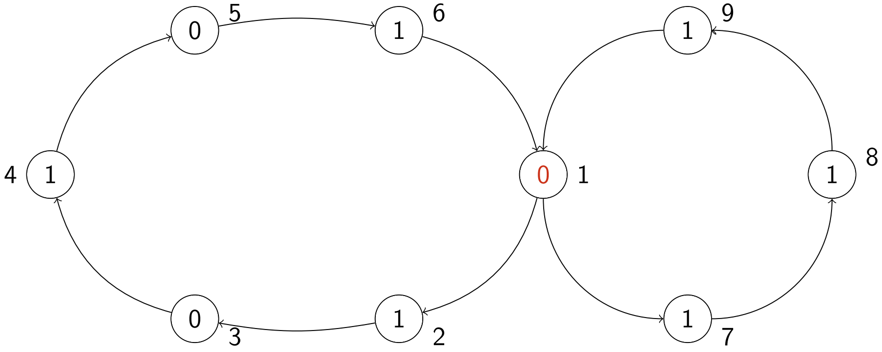
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



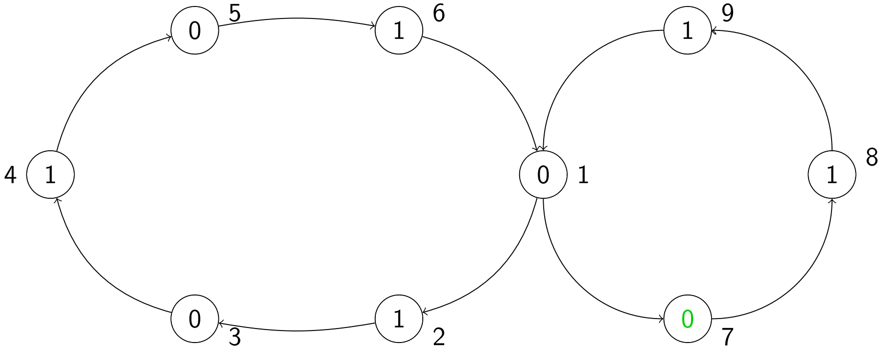
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



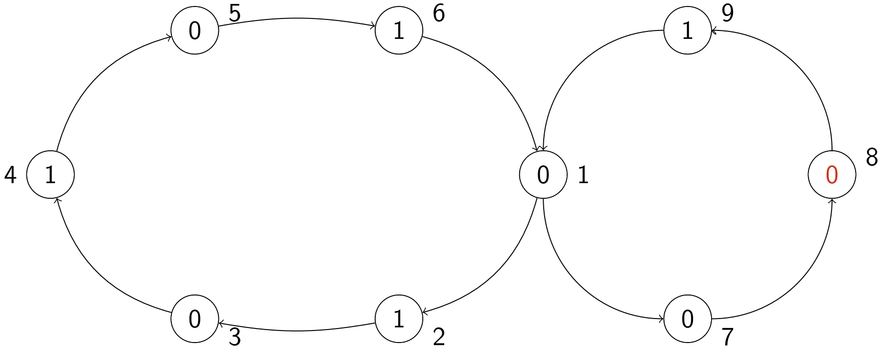
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



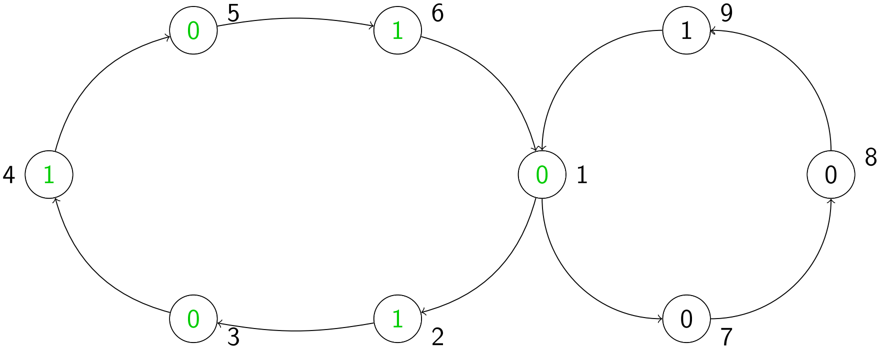
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



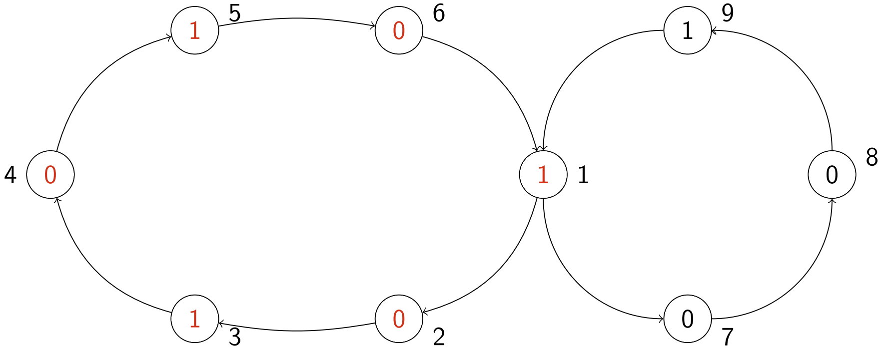
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



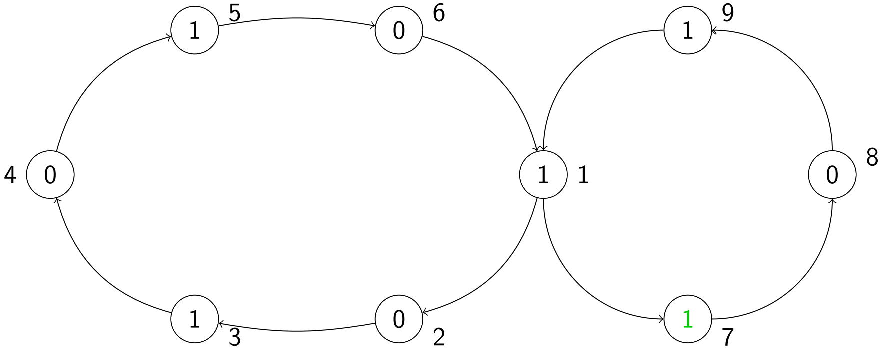
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



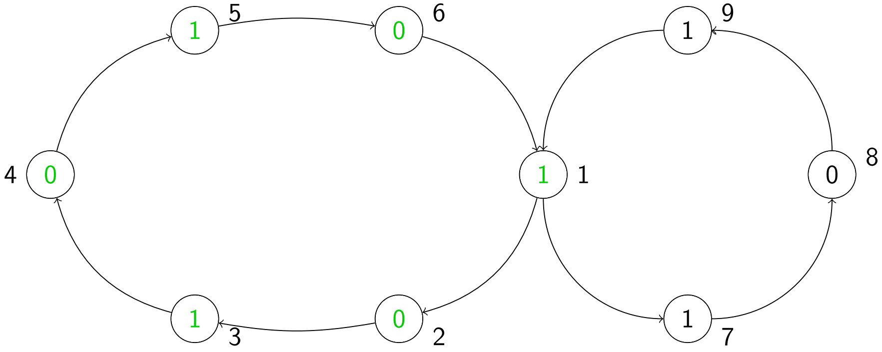
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



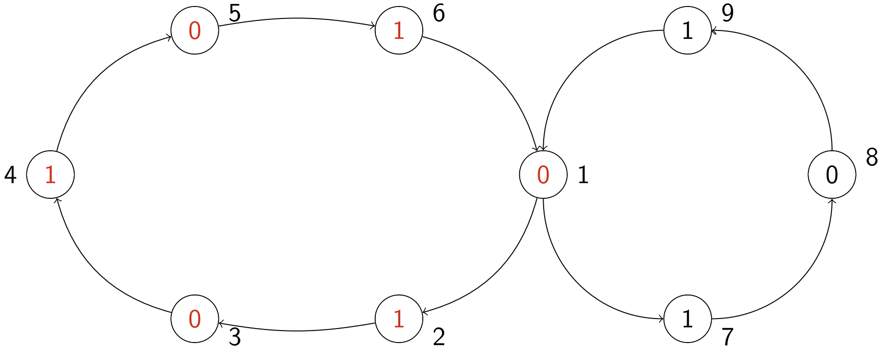
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



An algorithmic proof (sketch 2)

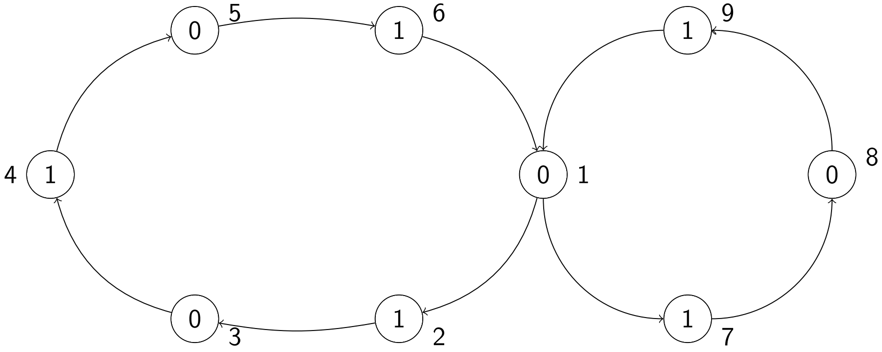
Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



An algorithmic proof (sketch 2)

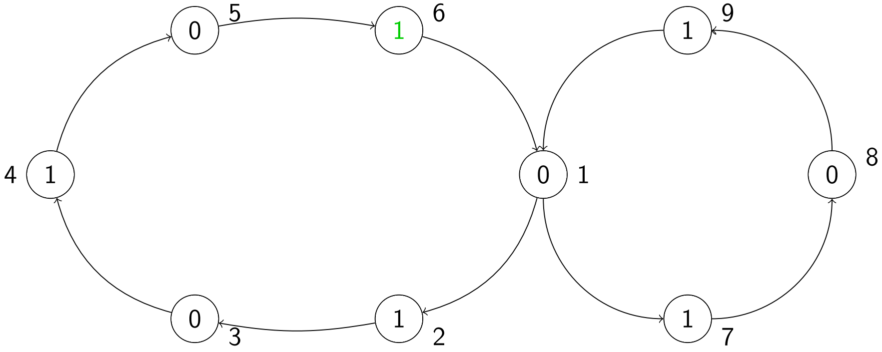
Idea: Reaching a highly expressive/unstable configuration.

$x' = 101001011$



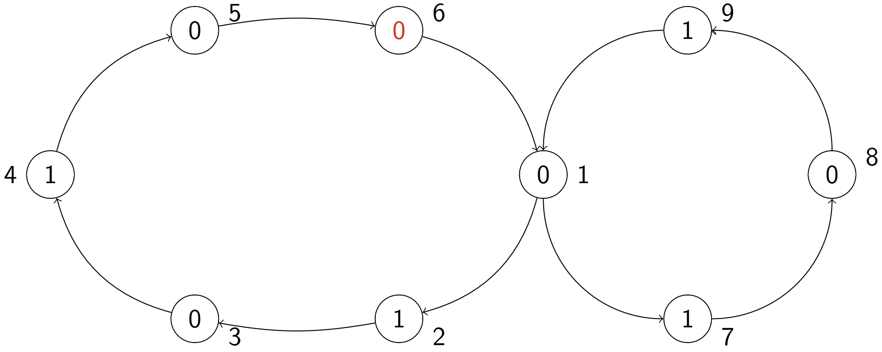
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



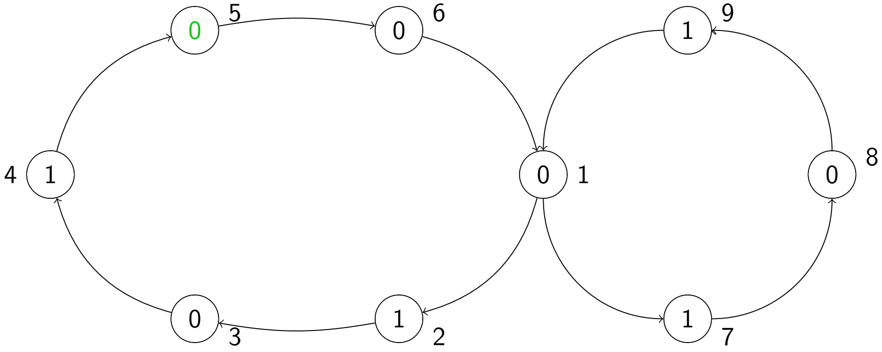
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



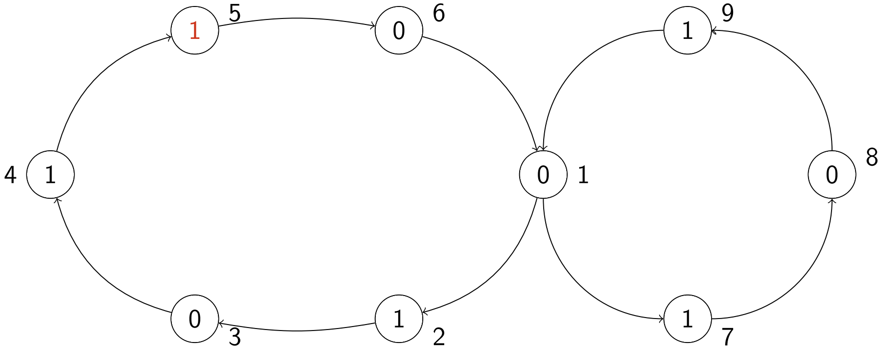
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



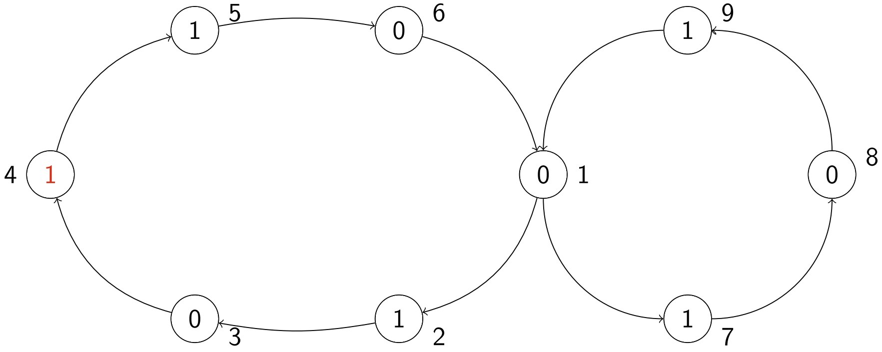
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



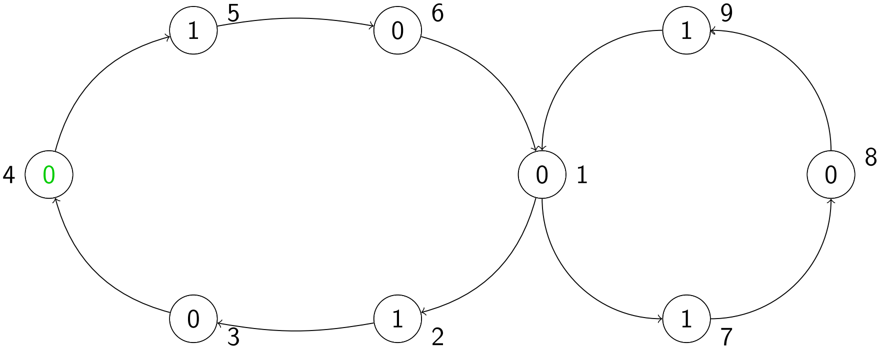
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



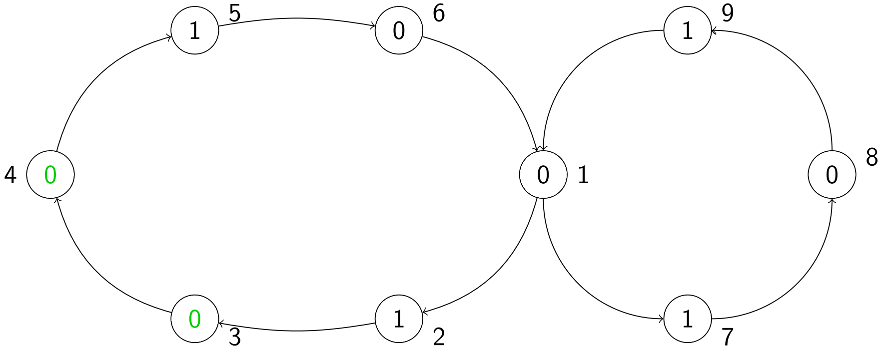
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



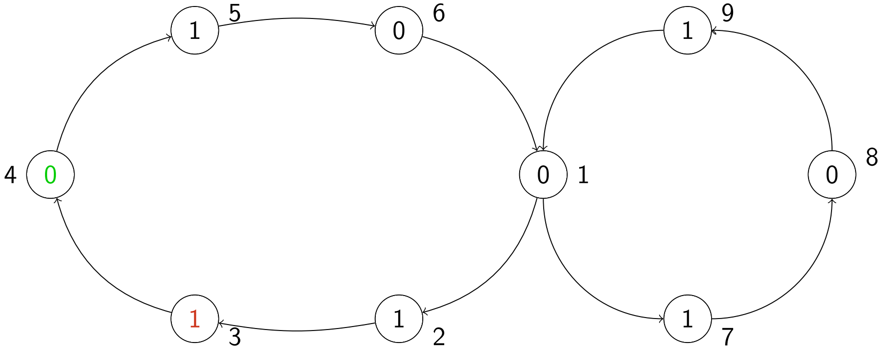
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



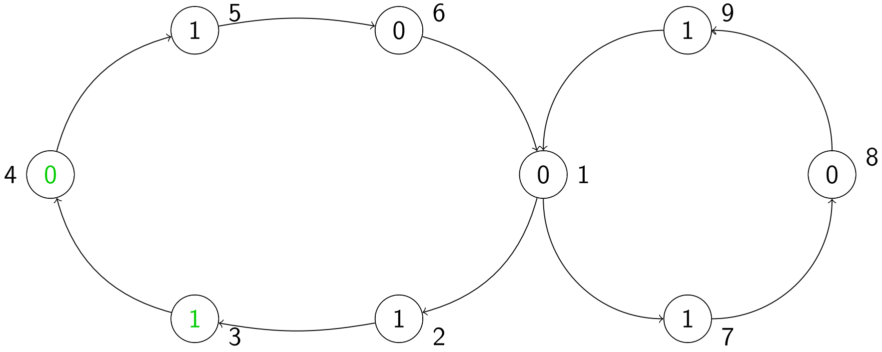
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



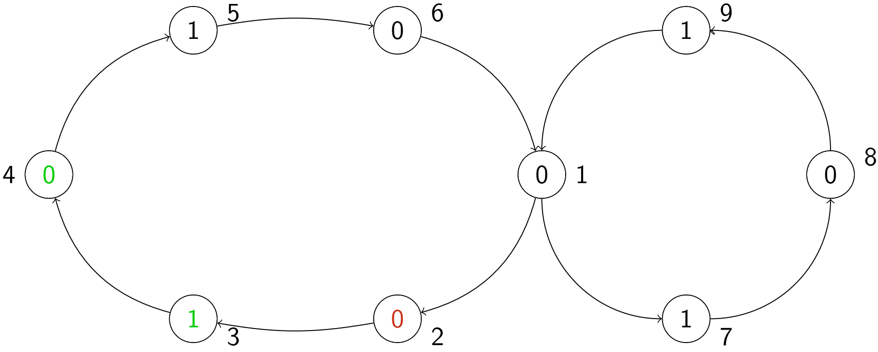
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



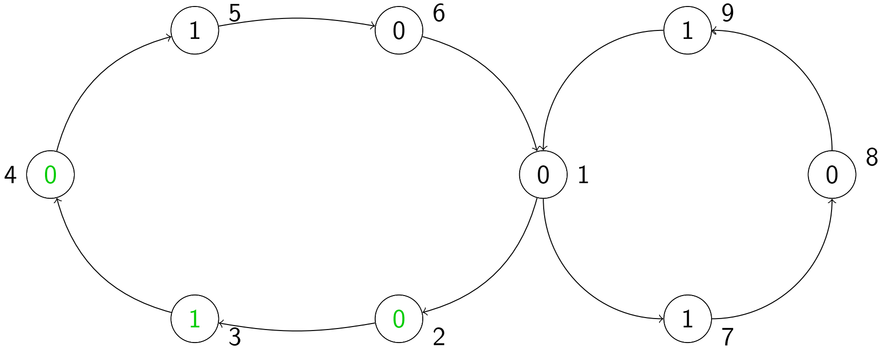
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



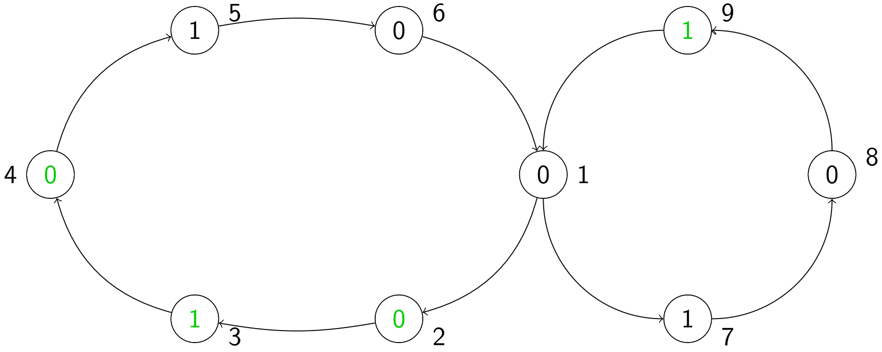
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



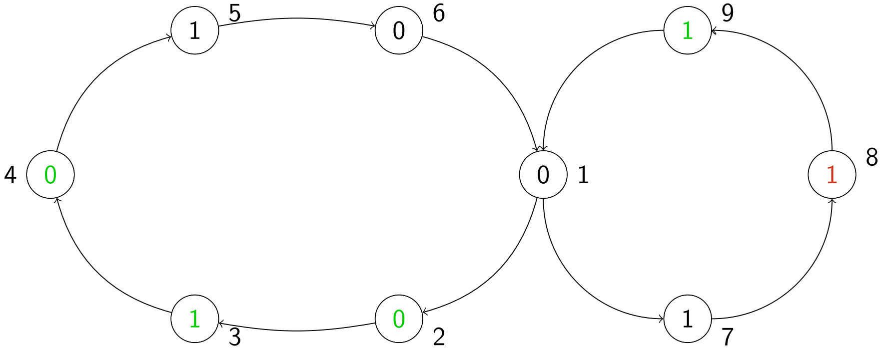
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



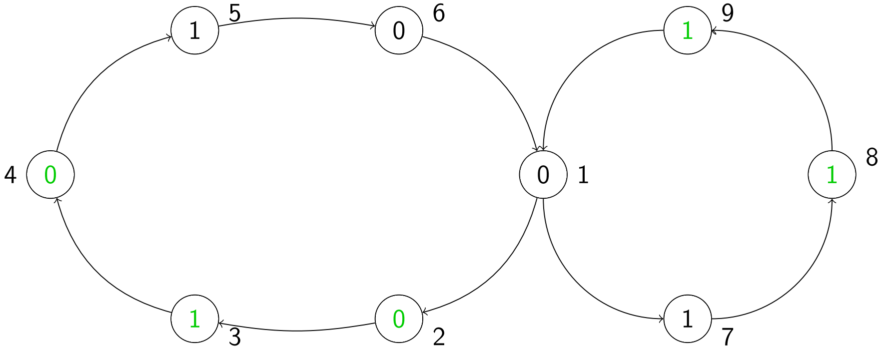
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



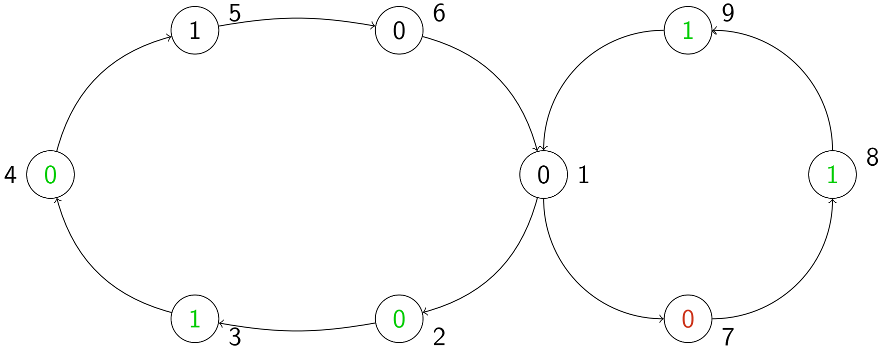
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



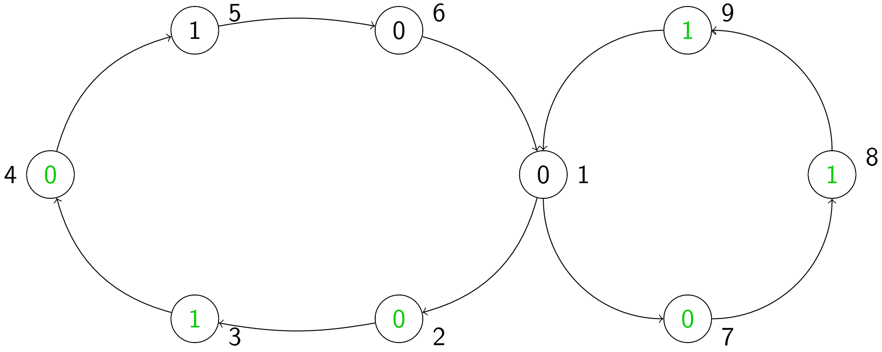
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



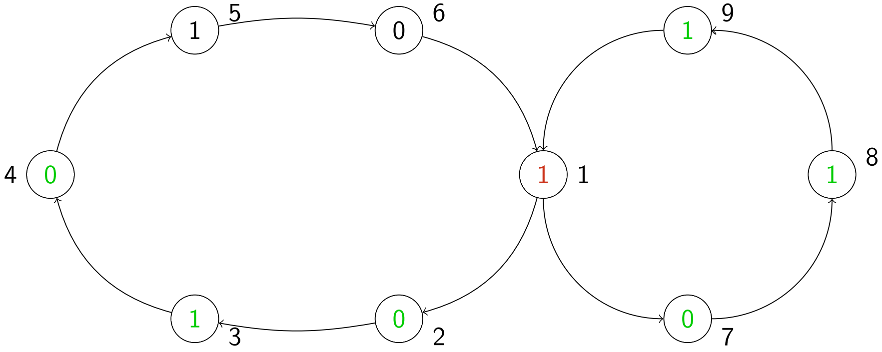
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$

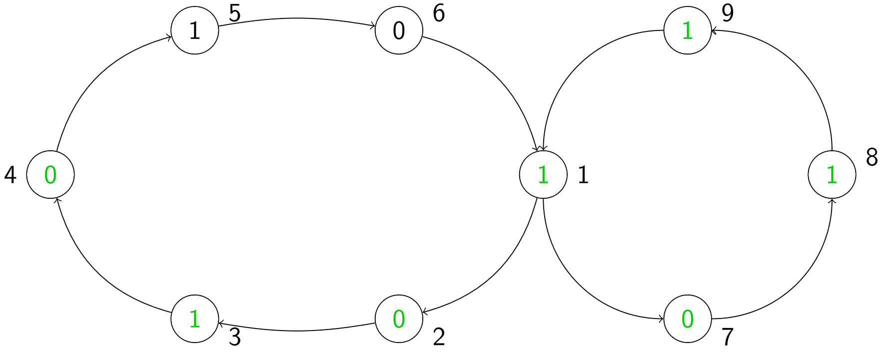


An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$

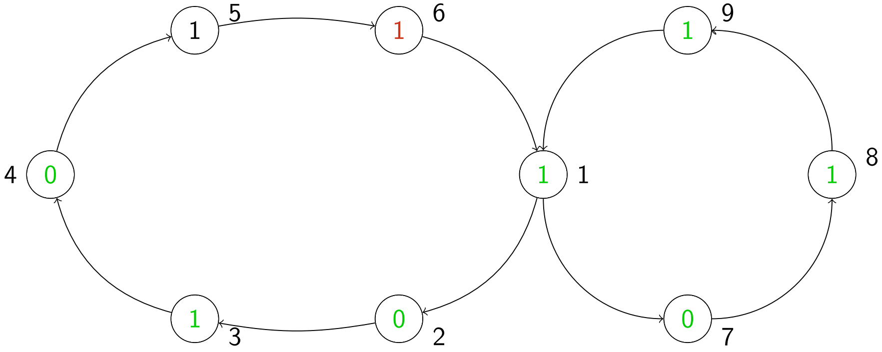


Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



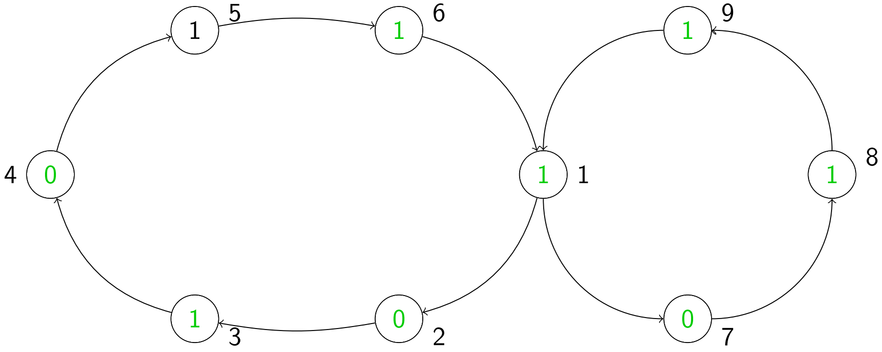
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



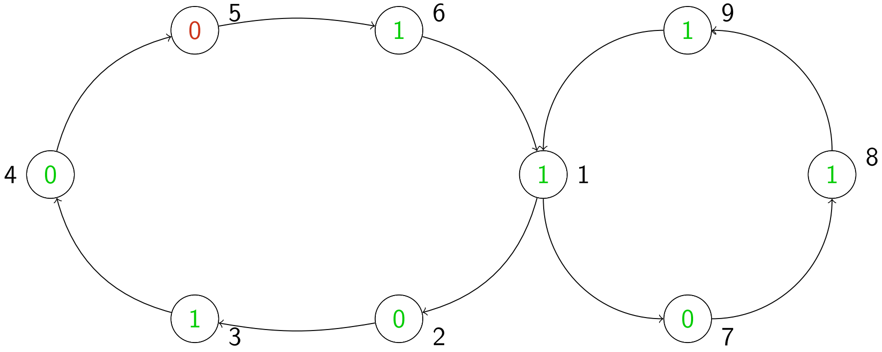
An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$



An algorithmic proof (sketch 2)

Idea: Reaching a highly expressive/unstable configuration.
$x' = 101001011$