Calculating preimages via de Bruijn networks in cellular automata

Juan Carlos Seck Tuoh Mora, Manuel González Hernández,

Genaro Juárez Martínez and Harold V. McIntosh

Universidad Autónoma del Estado de Hidalgo, México

University of the West of England, United Kingdom

Universidad Autónoma del Estado de Puebla, México

Abstract

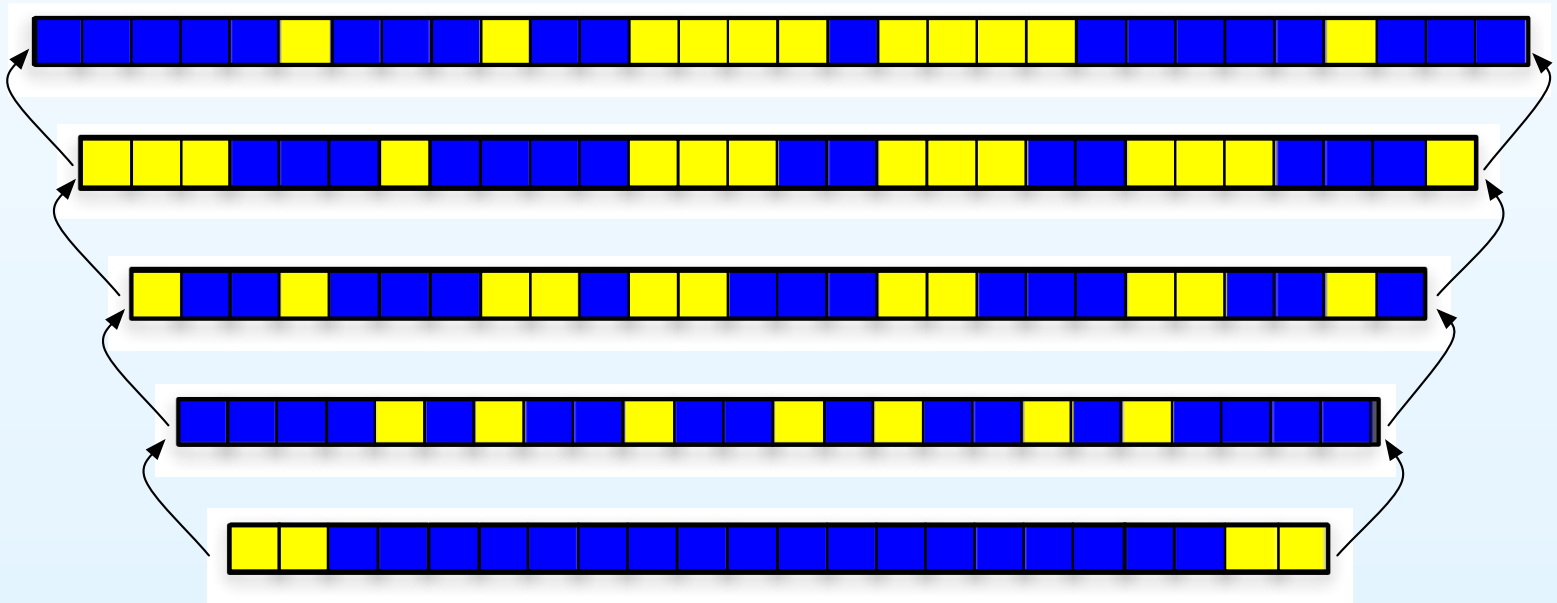
This work describes some advances in calculating preimages for several generations in one-dimensional cellular automata. Based on results obtained by Jeras and Dobnika, de Bruijn networks are used for this task. Examples using Rule 110 ($k = 2$; $r = 1$) are presented.

Index

- Problem description
- De Bruijn networks
- Computational implementation
- Results in Rule 110
- Discussion

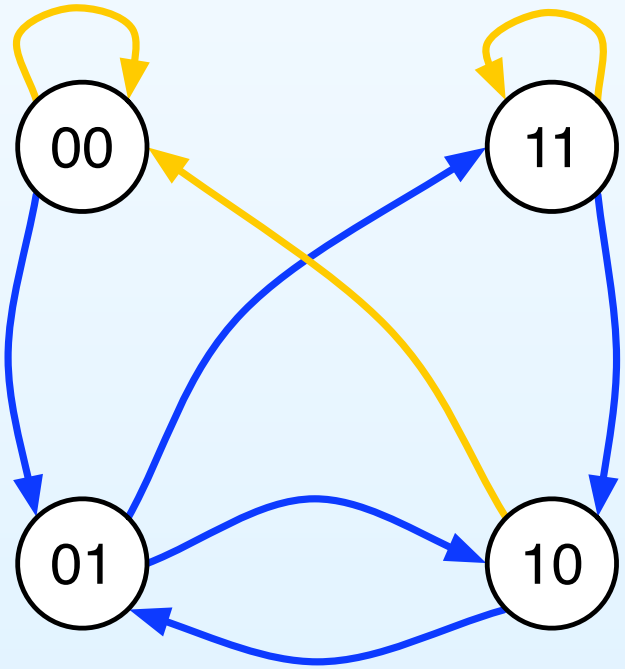
Statement of the problem

Given a one-dimensional cellular automaton, calculate the preimages of a finite configuration in several steps.



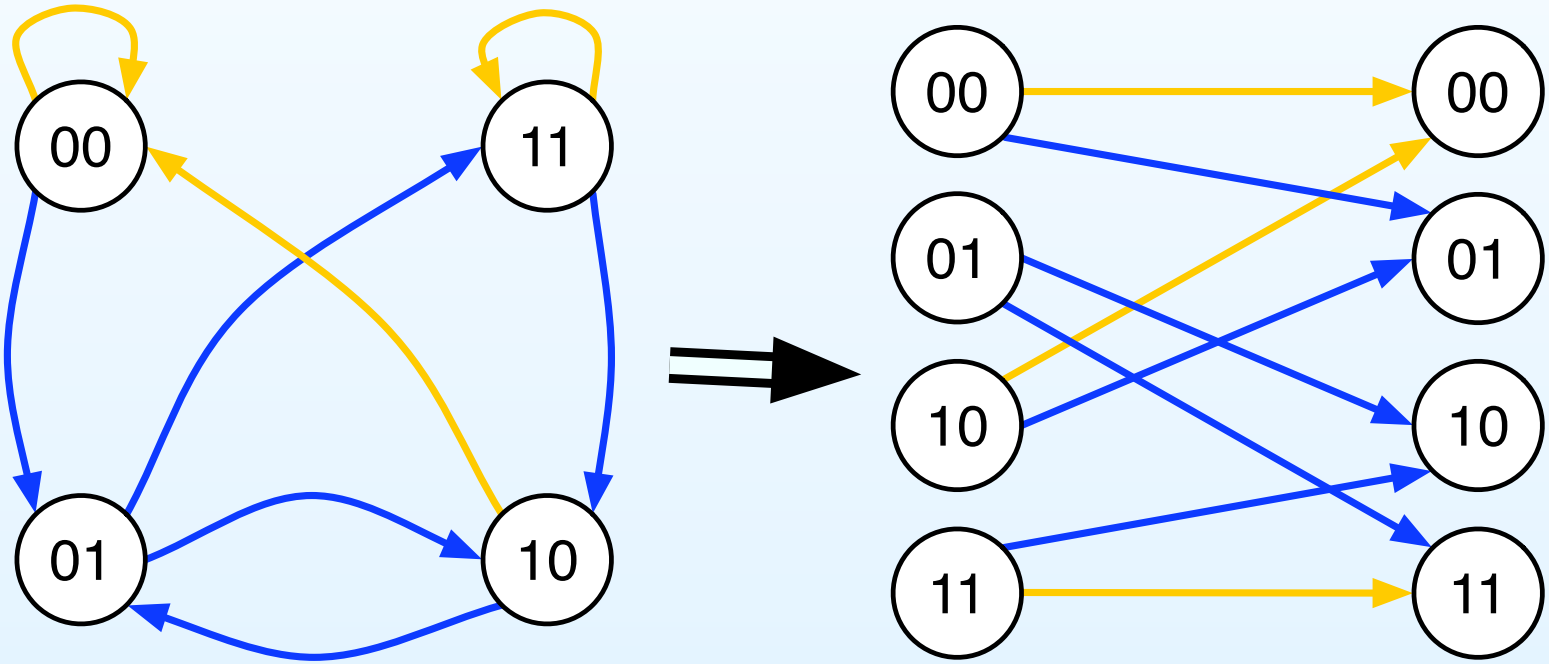
Statement of the problem

- Graphical and matrix approaches have been used for treating the problem (Jen, Wolfram, McIntosh, Voorhees, Wuensche, Jeras, Gómez-Soto).
- Most of the efforts have used the properties and extensions of de Bruijn graphs.



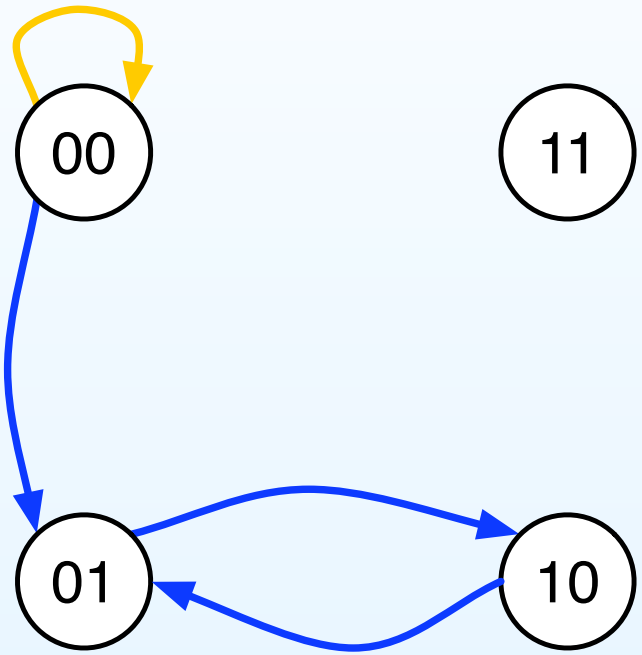
De Bruijn networks

Iztok Jeras and Andrej Dobnikar (**Algorithms for computing preimages of cellular automata configurations**, http://www.rattus.info/al/al.html).



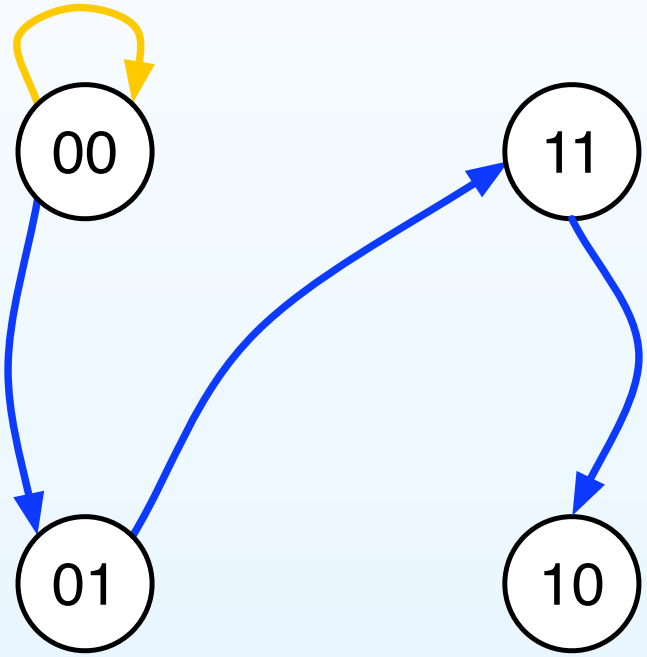
De Bruijn networks

Preimages of 0111



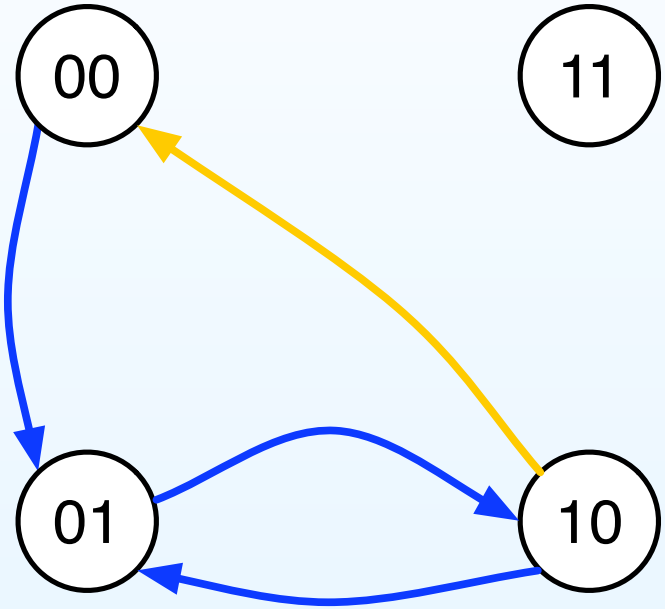
De Bruijn networks

Preimages of 0111



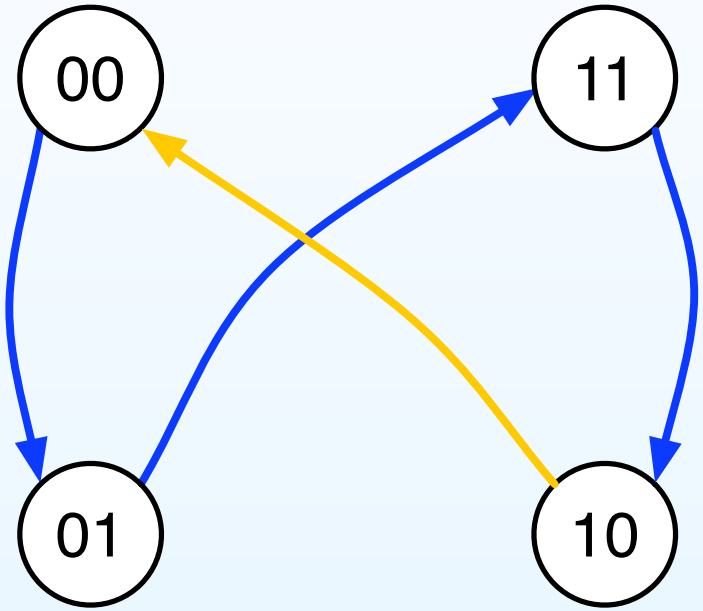
De Bruijn networks

Preimages of 0111



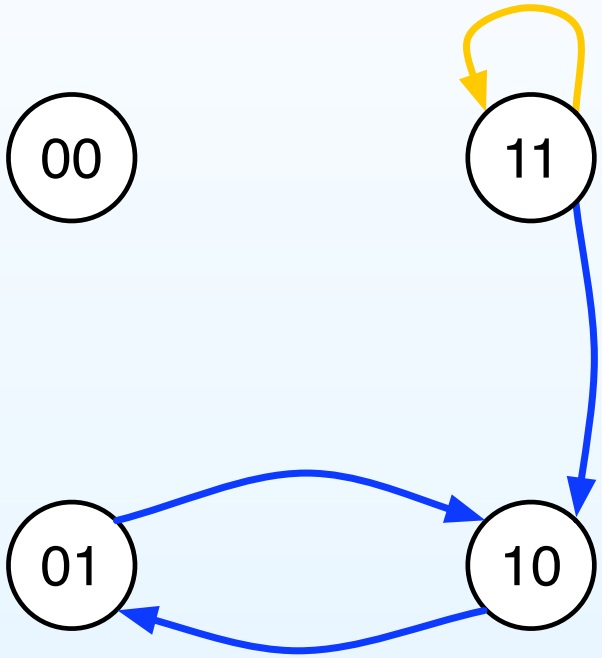
De Bruijn networks

Preimages of 0111



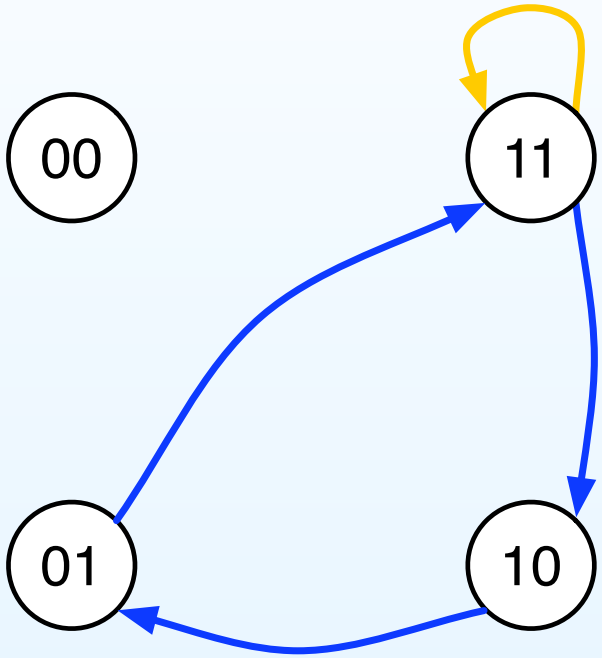
De Bruijn networks

Preimages of 0111



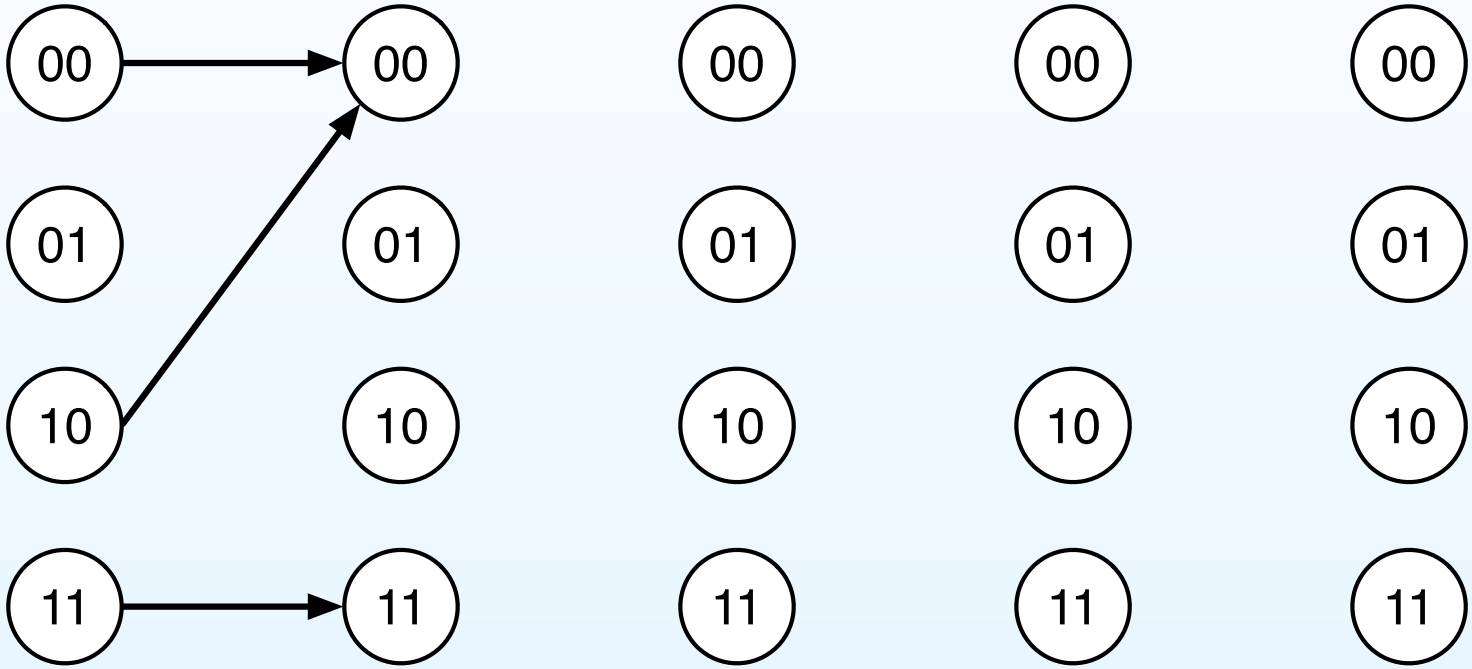
De Bruijn networks

Preimages of 0111



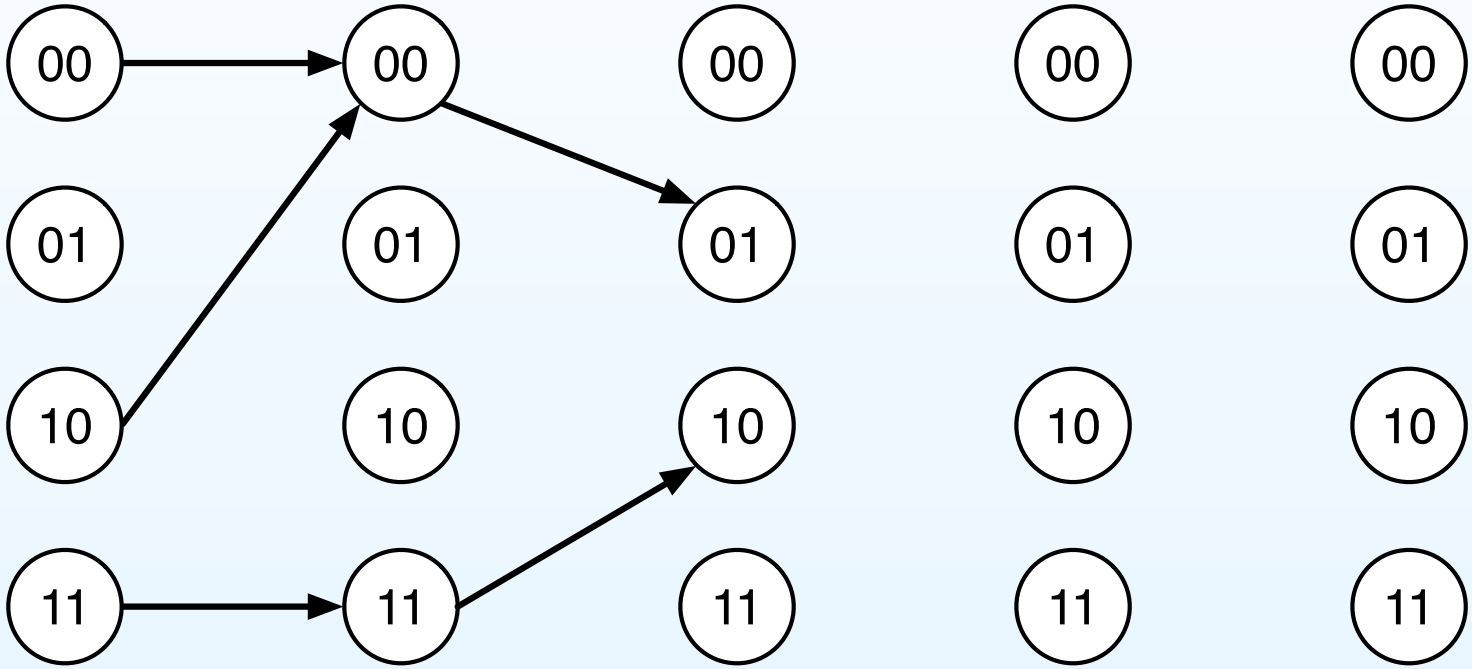
De Bruijn networks

Preimages of 0111



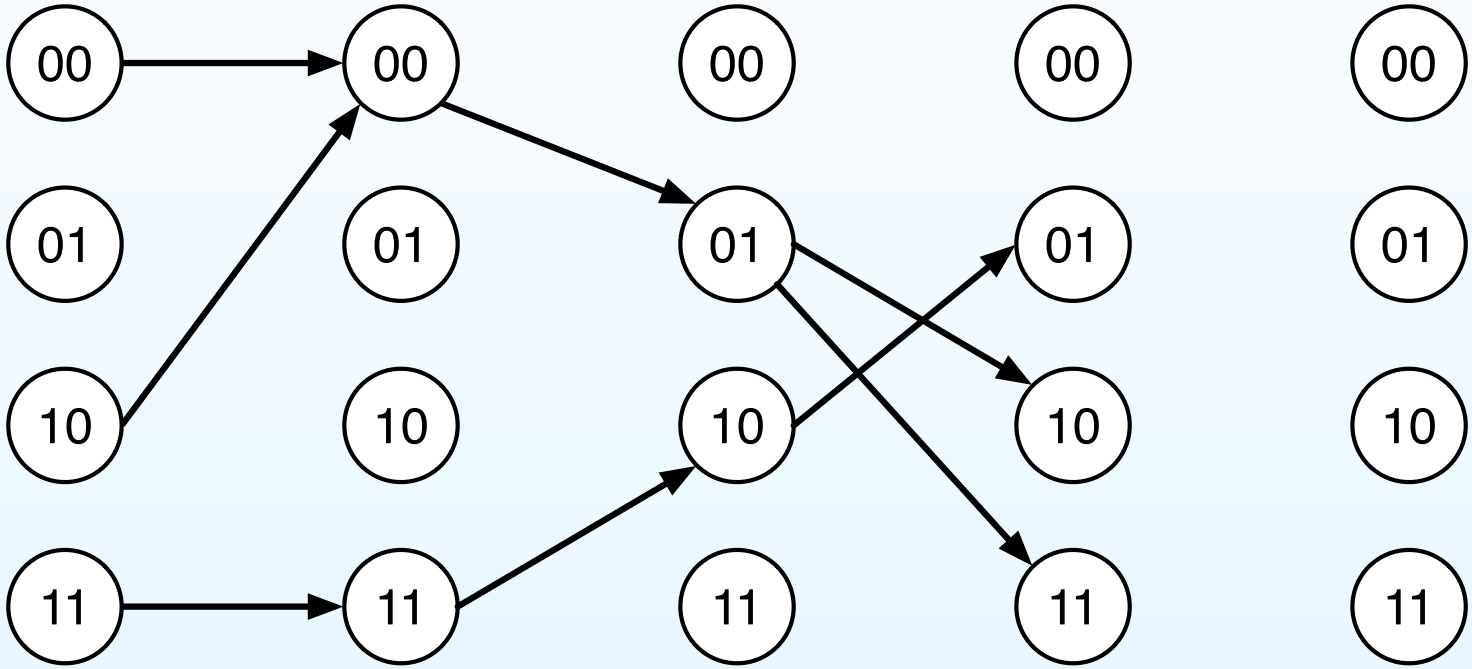
De Bruijn networks

Preimages of 0111



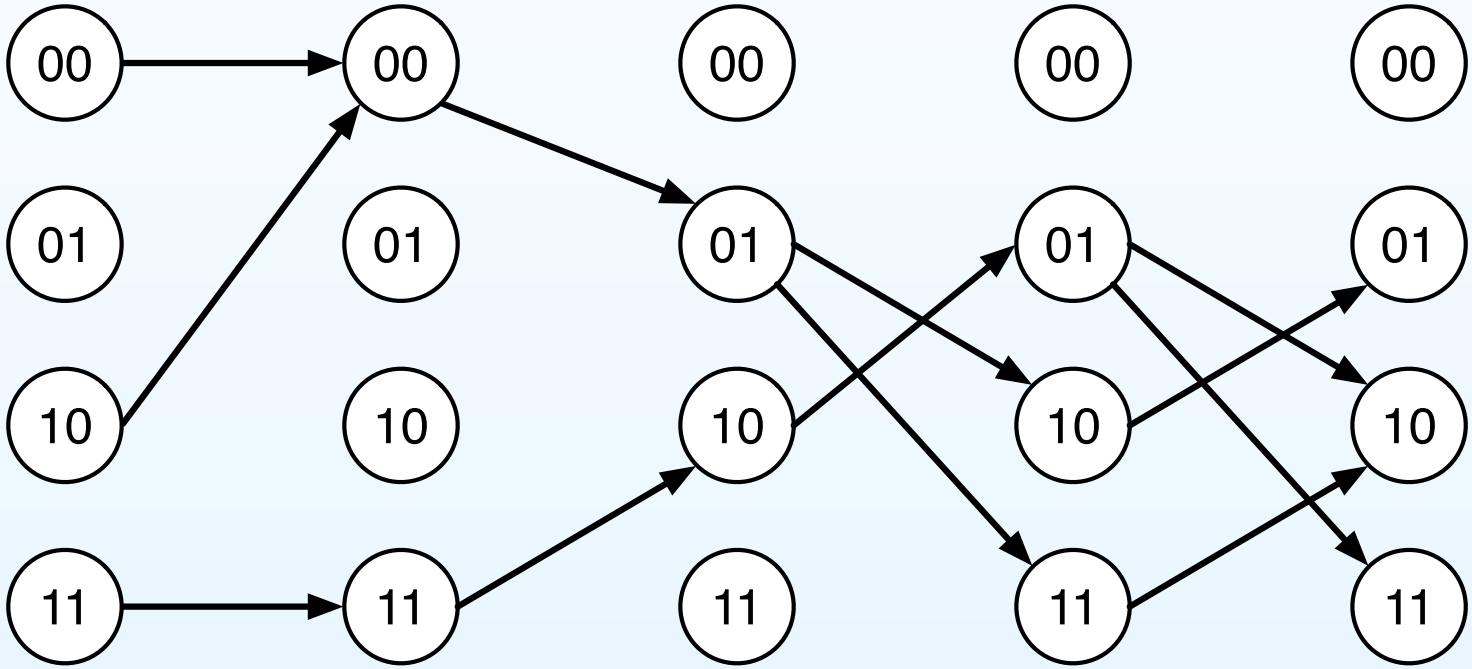
De Bruijn networks

Preimages of 0111



De Bruijn networks

Preimages of 0111



De Bruijn networks

Using a neighborhood size 2

$\varphi : K^3 \to K$

De Bruijn networks

Using a neighborhood size 2

$$\varphi : K^3 \to K \quad \Rightarrow \quad \varphi : K^4 \to K^2$$

De Bruijn networks

Using a neighborhood size 2

$$\varphi : K^3 \to K \quad \Rightarrow \quad \varphi : K^4 \to K^2 \quad \Rightarrow \quad \varphi : (K^2 \times K^2) \to K^2$$

De Bruijn networks

Using a neighborhood size 2

$$\varphi : K^3 \to K \quad \Rightarrow \quad \varphi : K^4 \to K^2 \quad \Rightarrow \quad \varphi : (K^2 \times K^2) \to K^2$$

$$\Downarrow$$

$$\tau : (S^2) \to S \text{ where } |S| = k^2$$

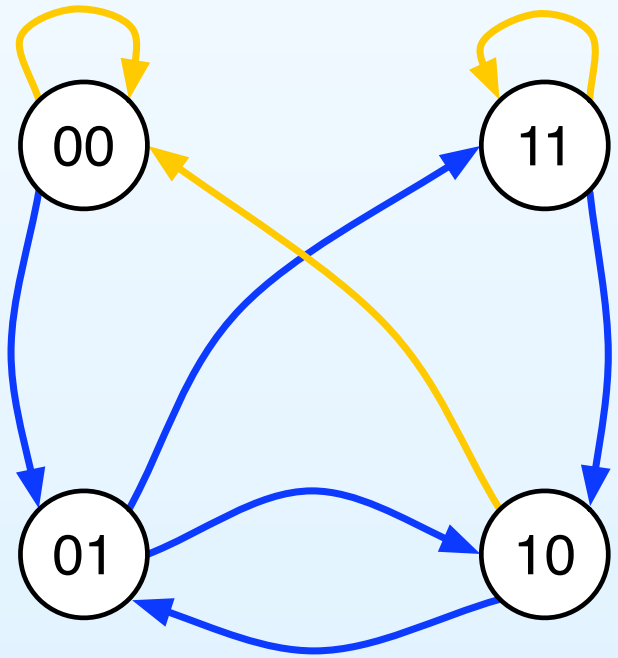
De Bruijn networks

Using a neighborhood size 2

$$\varphi : K^3 \to K \quad \Rightarrow \quad \varphi : K^4 \to K^2 \quad \Rightarrow \quad \varphi : (K^2 \times K^2) \to K^2$$

$$\Downarrow$$

$$\tau : (S^2) \to S \text{ where } |S| = k^2$$



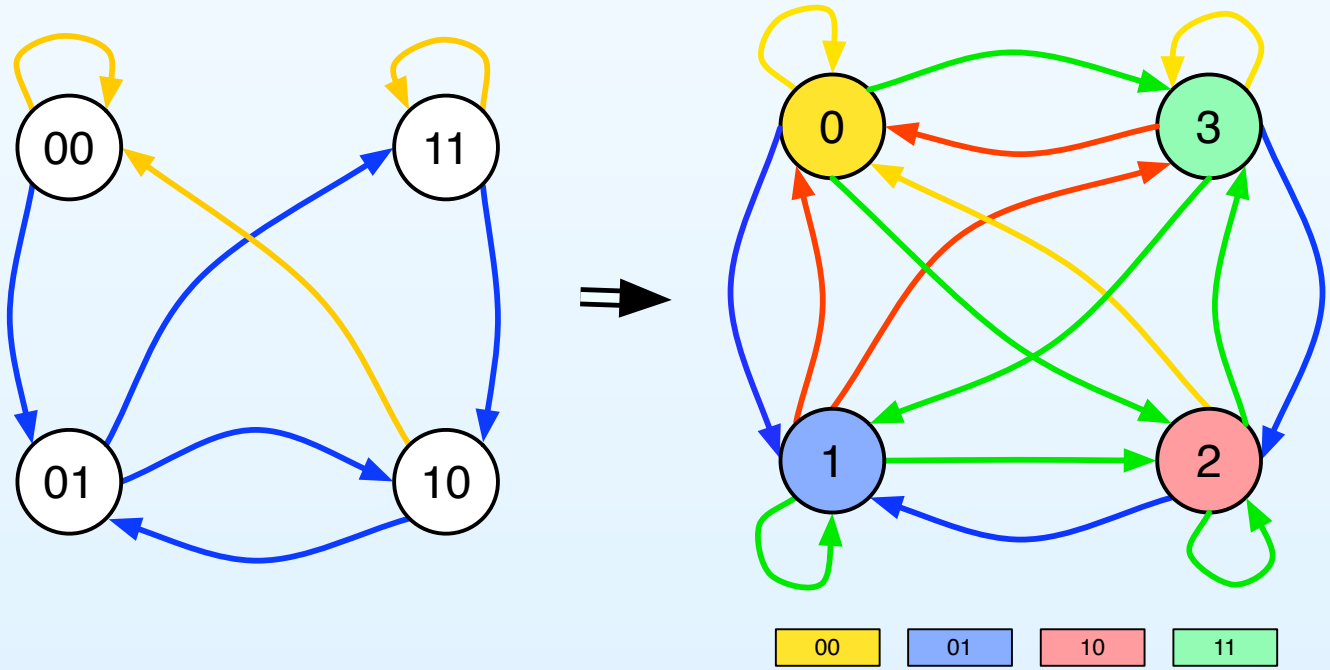
De Bruijn networks

Using a neighborhood size 2

$$\varphi : K^3 \to K \quad \Rightarrow \quad \varphi : K^4 \to K^2 \quad \Rightarrow \quad \varphi : (K^2 \times K^2) \to K^2$$

$$\Downarrow$$

$$\tau : (S^2) \to S \text{ where } |S| = k^2$$



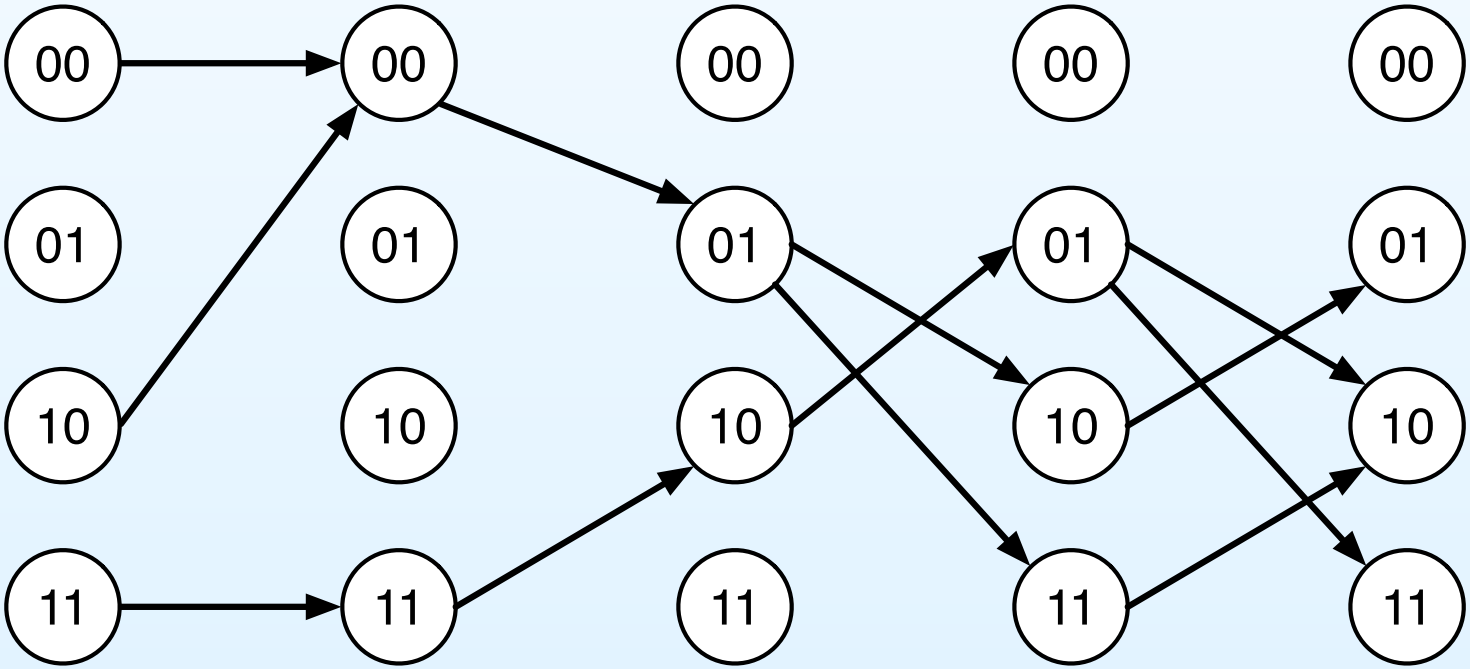
De Bruijn networks

Using a neighborhood size 2

$$\varphi : K^3 \to K \quad \Rightarrow \quad \varphi : K^4 \to K^2 \quad \Rightarrow \quad \varphi : (K^2 \times K^2) \to K^2$$

$$\Downarrow$$

$$\tau : (S^2) \to S \text{ where } |S| = k^2$$



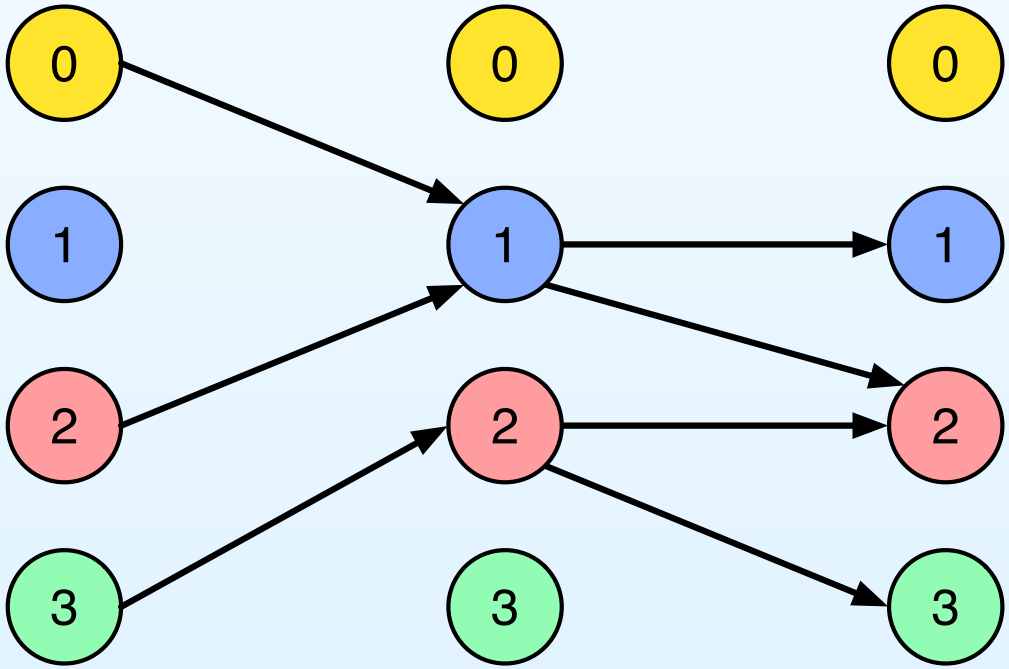
De Bruijn networks

Using a neighborhood size 2

$$\varphi : K^3 \to K \quad \Rightarrow \quad \varphi : K^4 \to K^2 \quad \Rightarrow \quad \varphi : (K^2 \times K^2) \to K^2$$

$$\Downarrow$$

$$\tau : (S^2) \to S \text{ where } |S| = k^2$$



De Bruijn networks for several steps

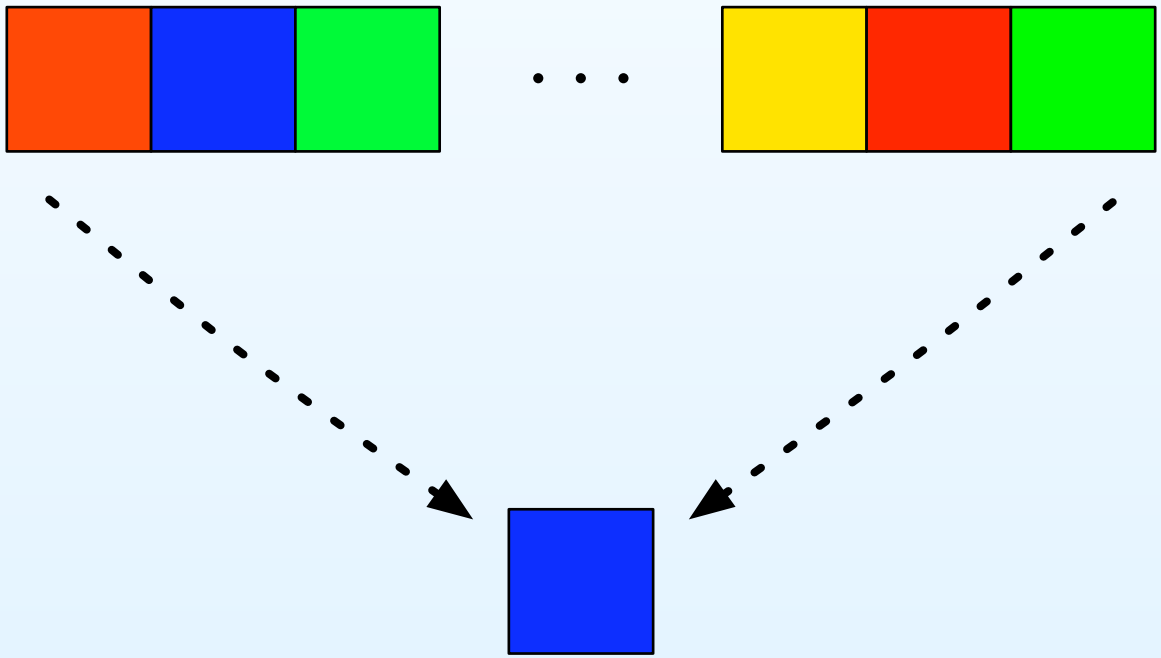
Problem: Obtaining the preimages of $w \in K^*$ for n steps.

De Bruijn networks for several steps

Problem: Obtaining the preimages of $w \in K^*$ for n steps.

Proposed algorithm:

1) Take every $v \in K^{n+1}$ and calculate its evolution in n steps.

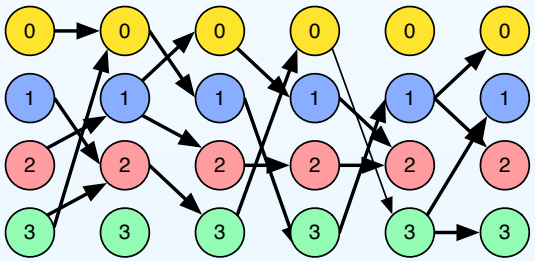


De Bruijn networks for several steps

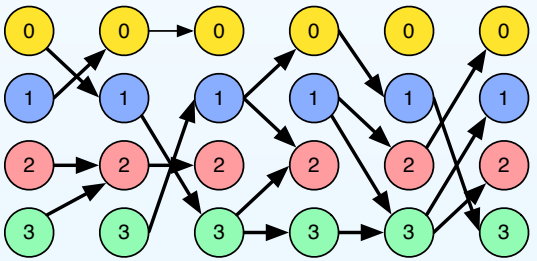
Problem: Obtaining the preimages of $w \in K^*$ for n steps.

Proposed algorithm:

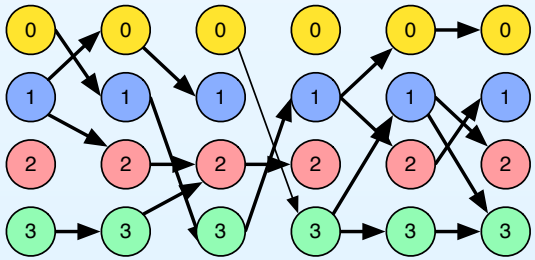
2) Arrange all sequences in $|K| = k$ networks, one for each state.



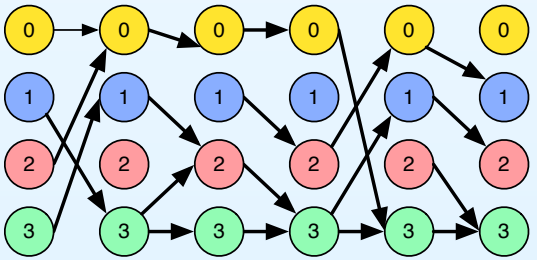
State 0



State 1



State 2



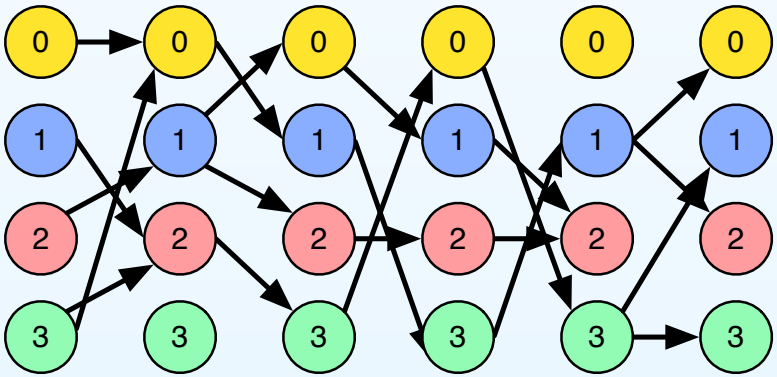
State 3

De Bruijn networks for several steps

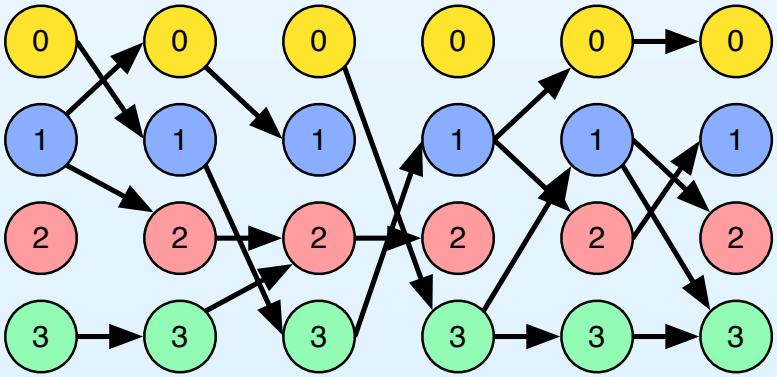
Problem: Obtaining the preimages of $w \in K^*$ for n steps.

Proposed algorithm:

3) Overlap networks to obtain the preimages.



State 0



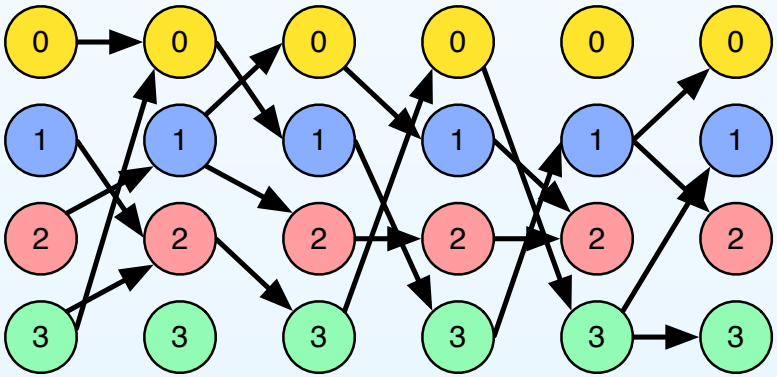
State 2

De Bruijn networks for several steps

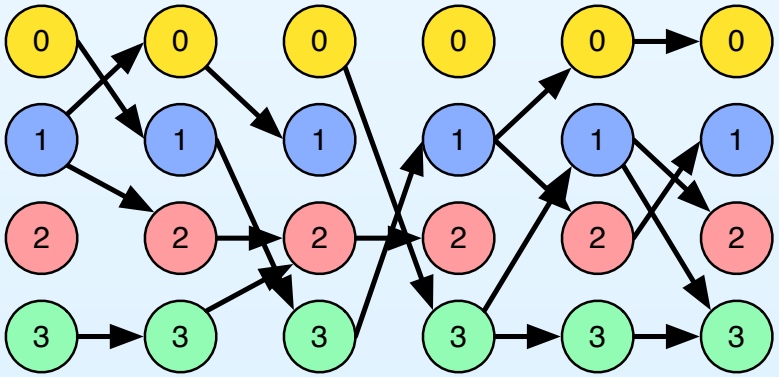
Problem: Obtaining the preimages of $w \in K^*$ for n steps.

Proposed algorithm:

3) Overlap networks to obtain the preimages.



State 0



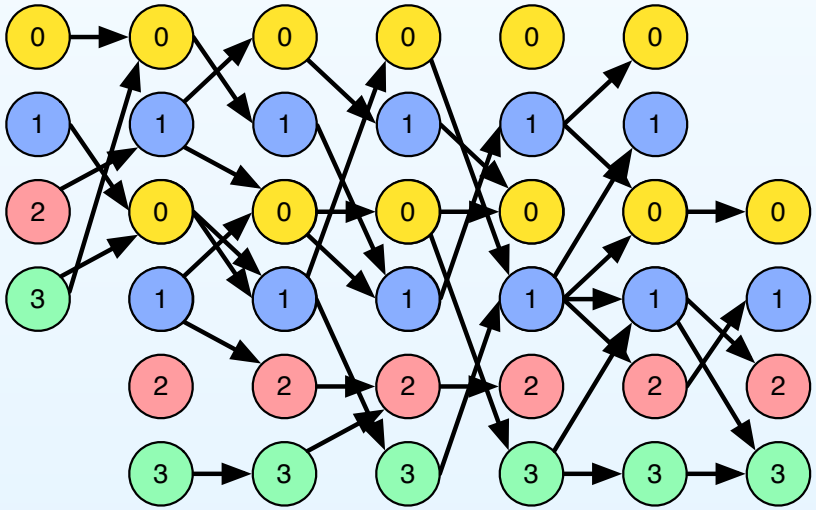
State 2

De Bruijn networks for several steps

Problem: Obtaining the preimages of $w \in K^*$ for n steps.

Proposed algorithm:

3) Overlap networks to obtain the preimages.



State 0

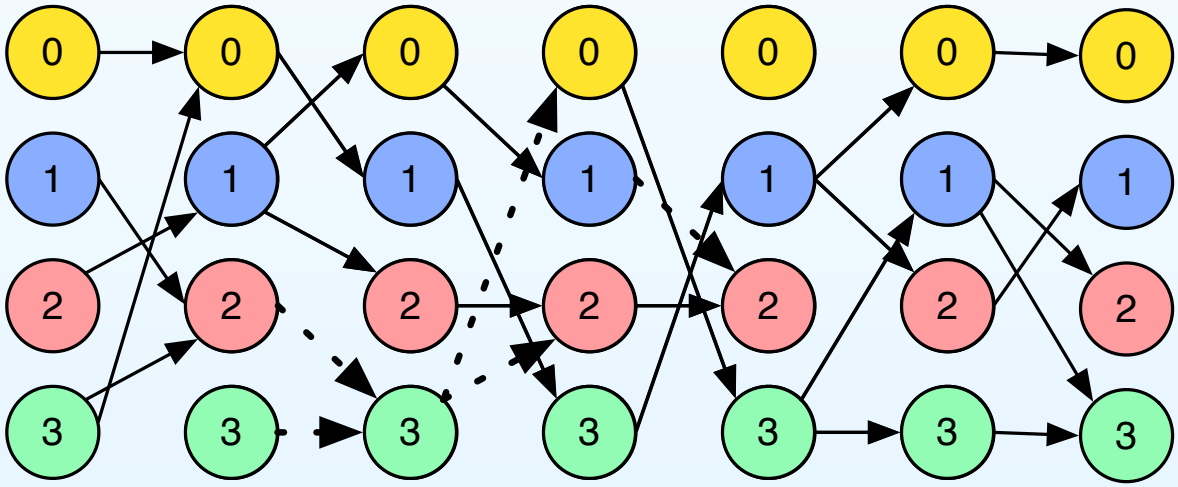
State 2

De Bruijn networks for several steps

Problem: Obtaining the preimages of $w \in K^*$ for n steps.

Proposed algorithm:

3) Overlap networks to obtain the preimages.



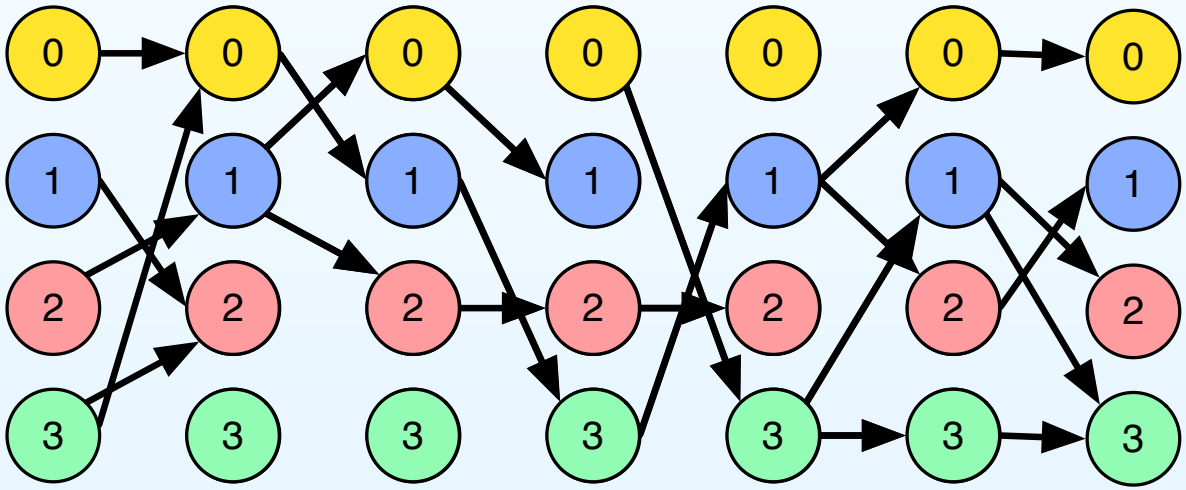
Sequence 02

De Bruijn networks for several steps

Problem: Obtaining the preimages of $w \in K^*$ for n steps.

Proposed algorithm:

4) Delete non-overlapping edges.



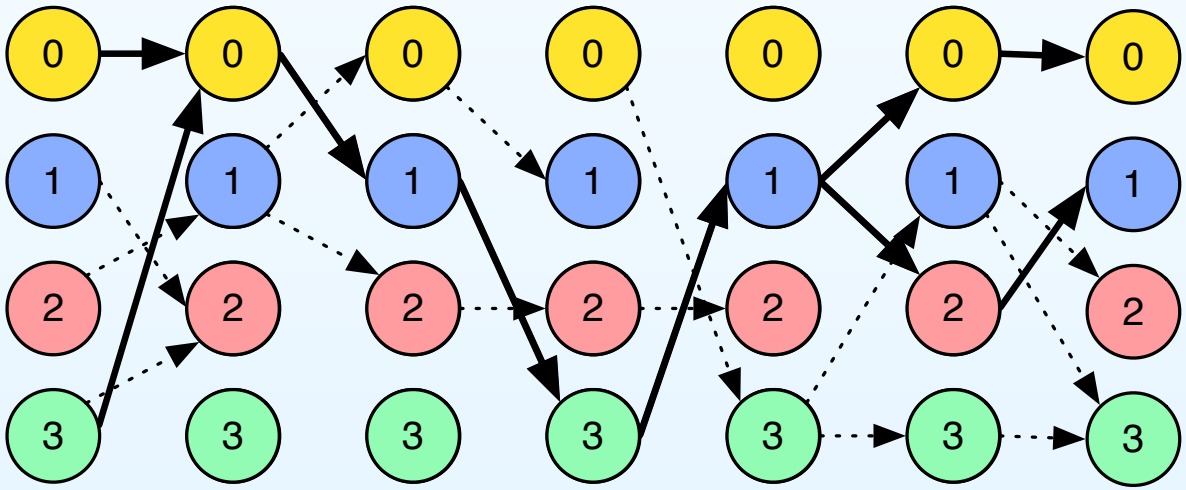
Sequence 02

De Bruijn networks for several steps

Problem: Obtaining the preimages of $w \in K^*$ for n steps.

Proposed algorithm:

5) Delete incomplete paths.



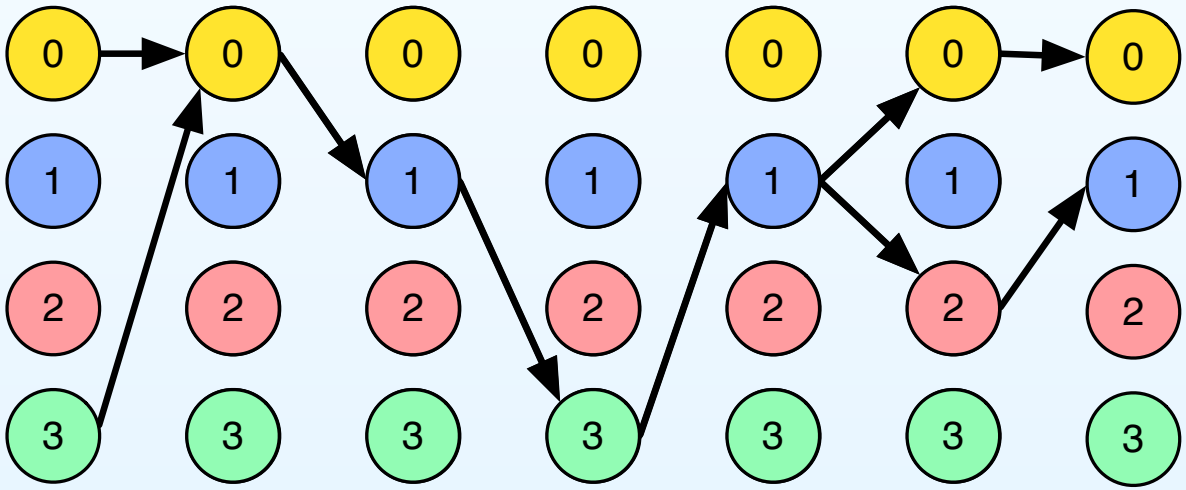
Sequence 02

De Bruijn networks for several steps

Problem: Obtaining the preimages of $w \in K^*$ for n steps.

Proposed algorithm:

5) Delete incomplete paths.



Sequence 02

Relevant considerations for basic networks

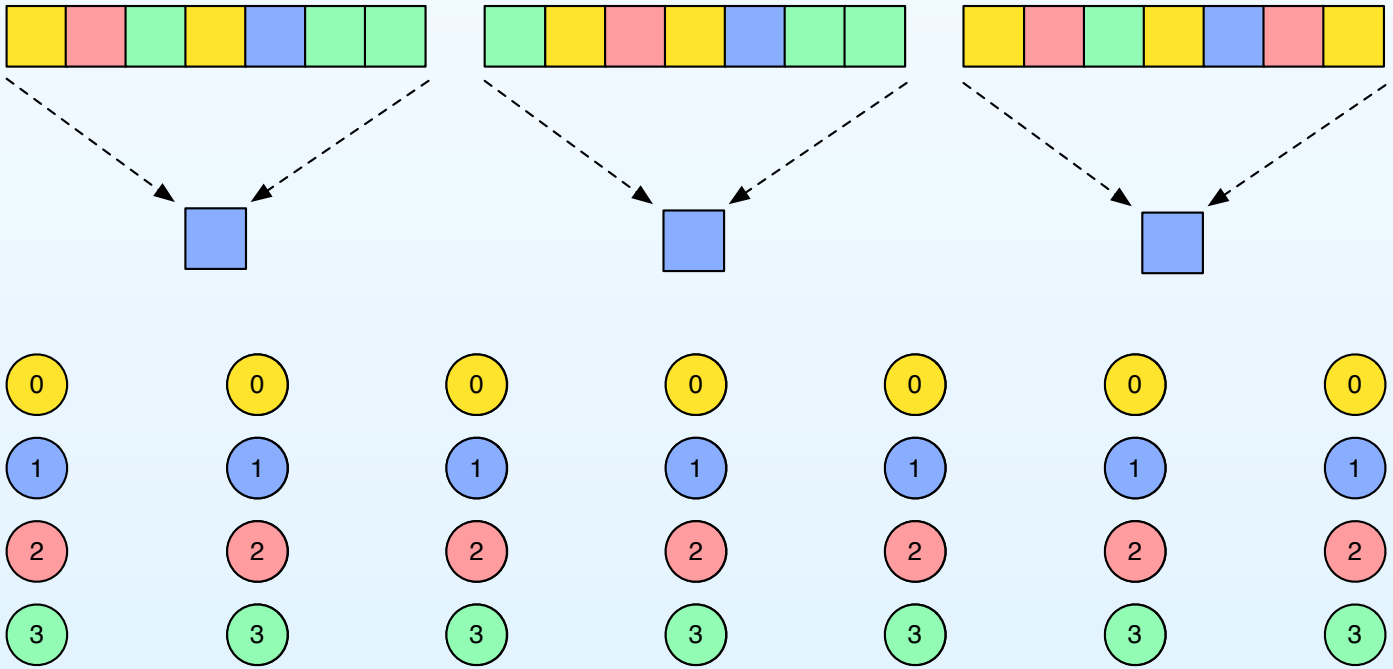
1) For n steps the original de Bruijn diagram has k^n vertices and all de Bruijn networks has $k^2(n+1)$.

Relevant considerations for basic networks

1) For n steps the original de Bruijn diagram has k^n vertices and all de Bruijn networks has $k^2(n+1)$.
2) Paths in a basic de Bruijn network must be carefully aggregated.

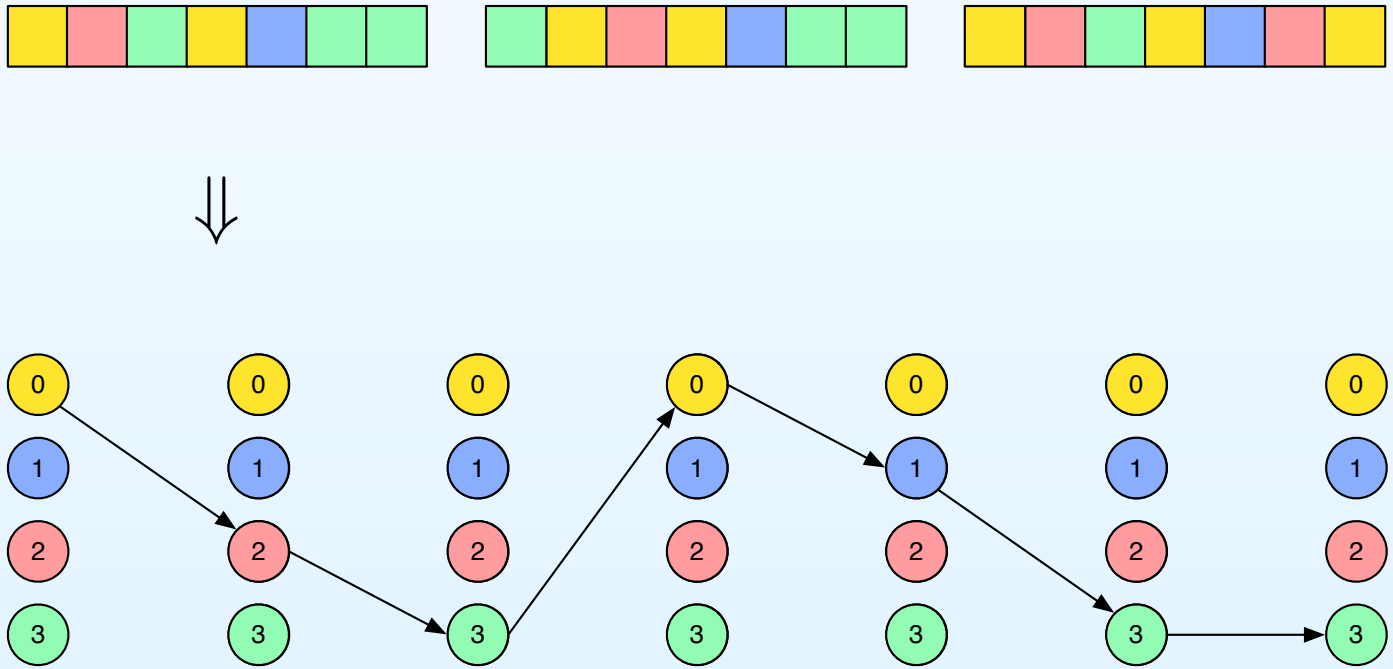
Relevant considerations for basic networks

1) For n steps the original de Bruijn diagram has k^n vertices and all de Bruijn networks has $k^2(n+1)$.
2) Paths in a basic de Bruijn network must be carefully aggregated.



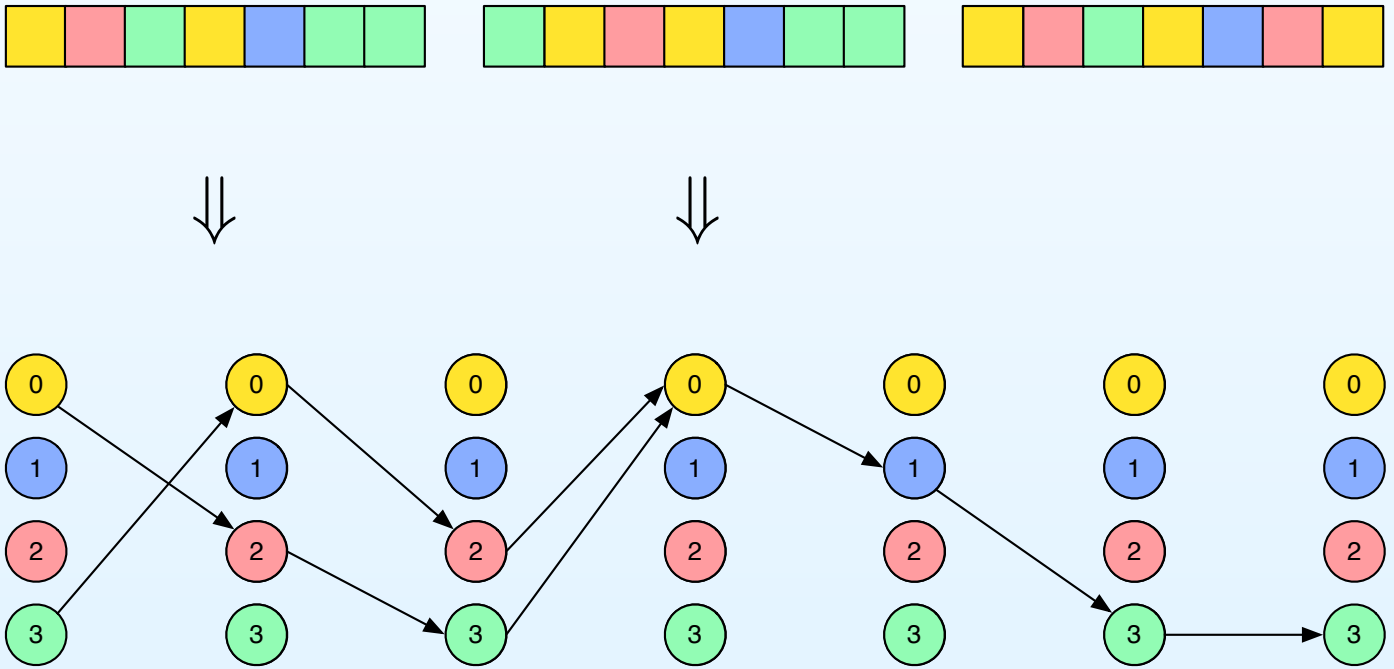
Relevant considerations for basic networks

1) For n steps the original de Bruijn diagram has k^n vertices and all de Bruijn networks has $k^2(n+1)$.
2) Paths in a basic de Bruijn network must be carefully aggregated.



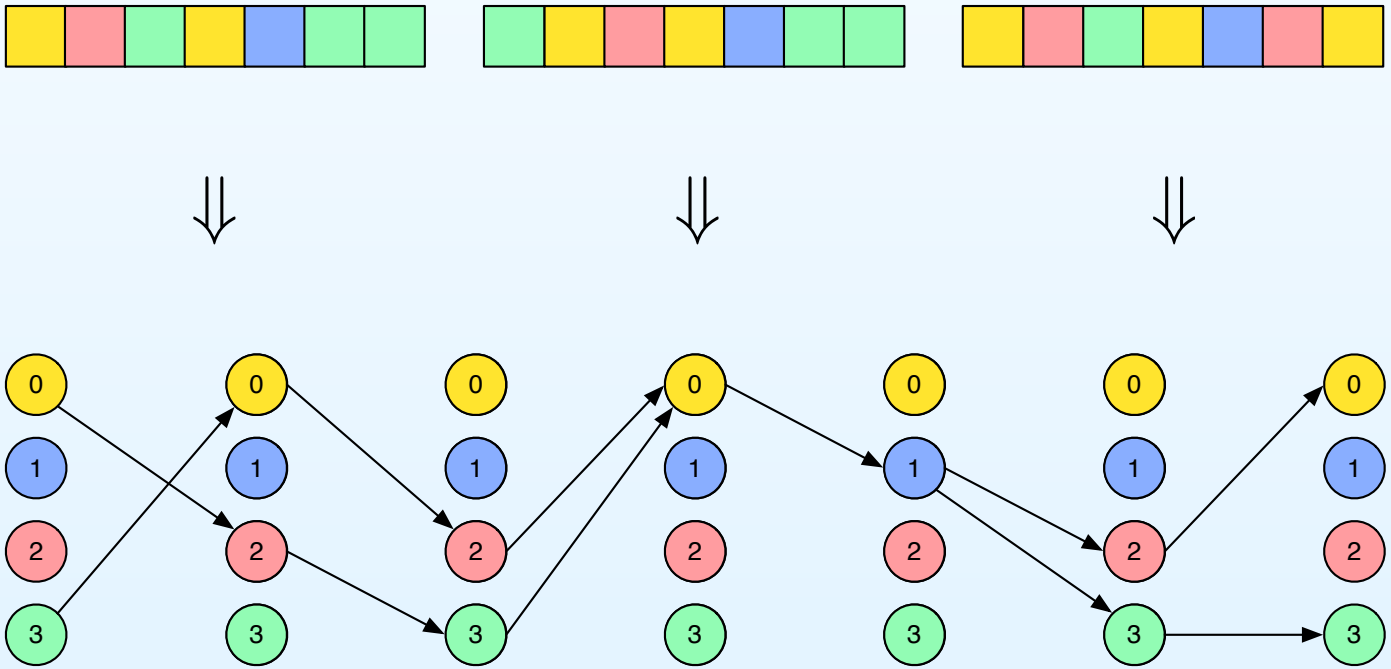
Relevant considerations for basic networks

1) For n steps the original de Bruijn diagram has k^n vertices and all de Bruijn networks has $k^2(n+1)$.
2) Paths in a basic de Bruijn network must be carefully aggregated.



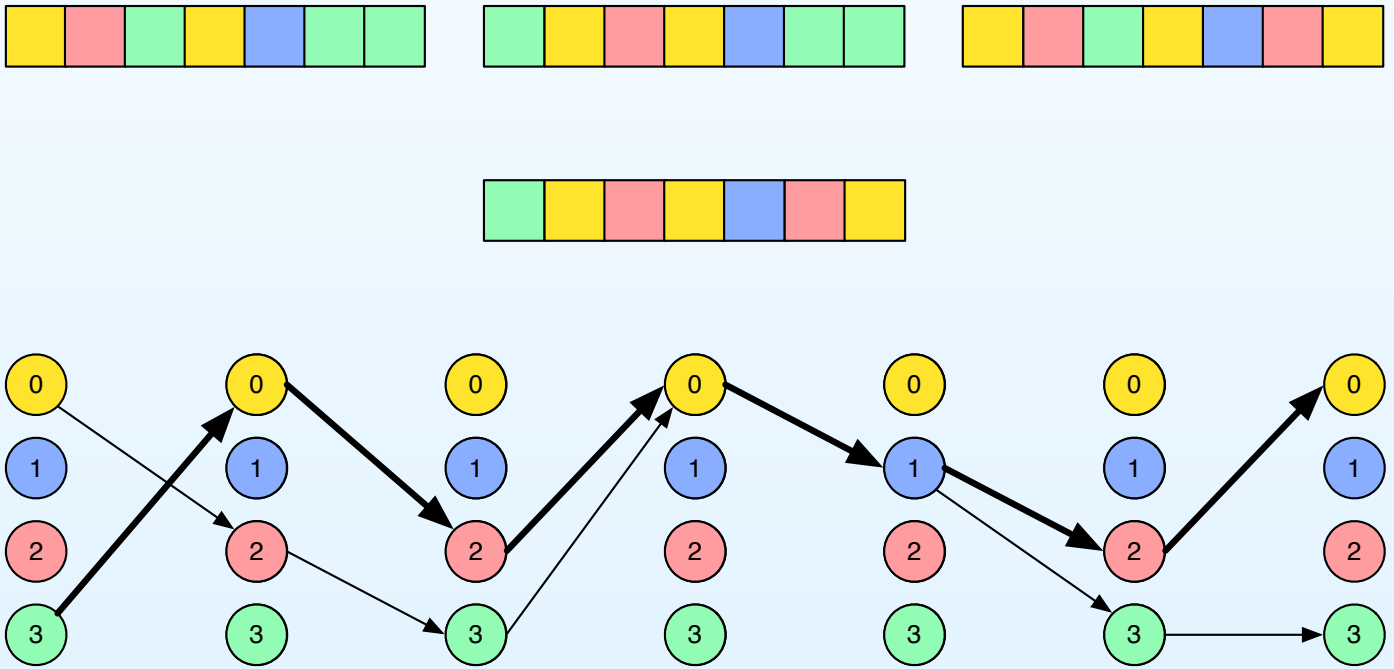
Relevant considerations for basic networks

1) For n steps the original de Bruijn diagram has k^n vertices and all de Bruijn networks has $k^2(n+1)$.
2) Paths in a basic de Bruijn network must be carefully aggregated.



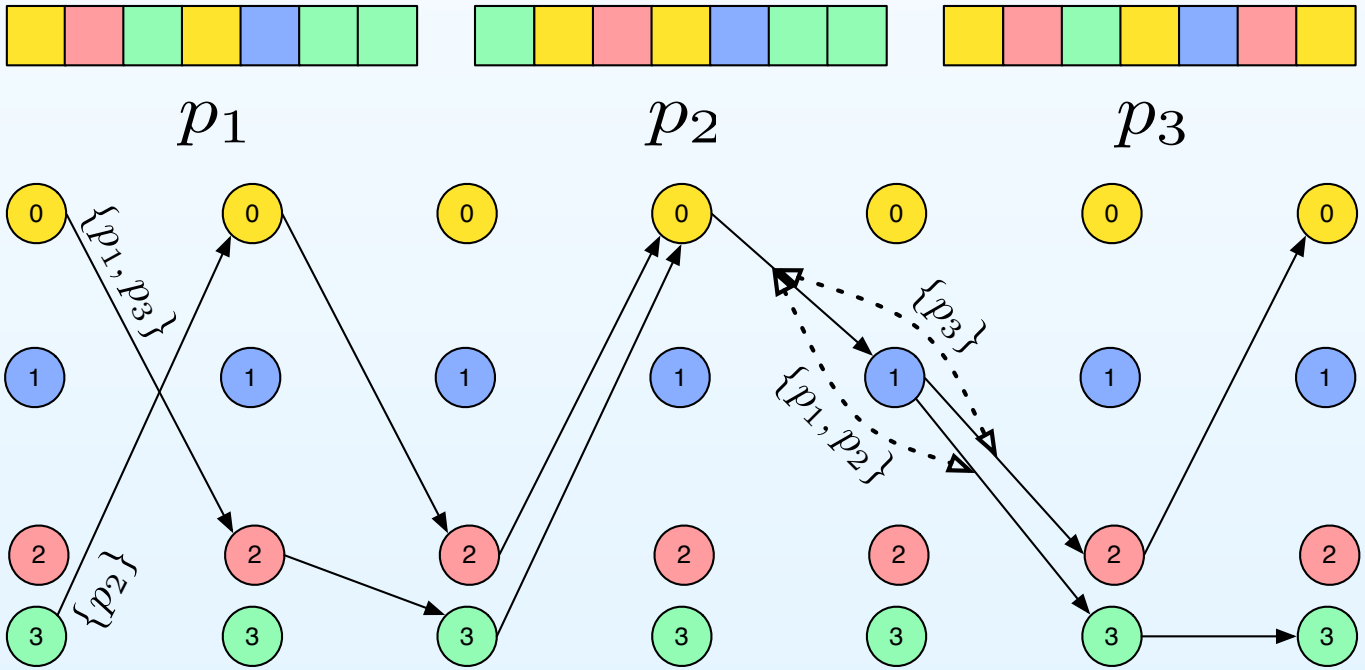
Relevant considerations for basic networks

1) For n steps the original de Bruijn diagram has k^n vertices and all de Bruijn networks has $k^2(n+1)$.
2) Paths in a basic de Bruijn network must be carefully aggregated.



Computational implementation

1) Every preimage is enumerated and a list of preimages is kept in the initial edges of a basic de Bruijn network.



Computational implementation

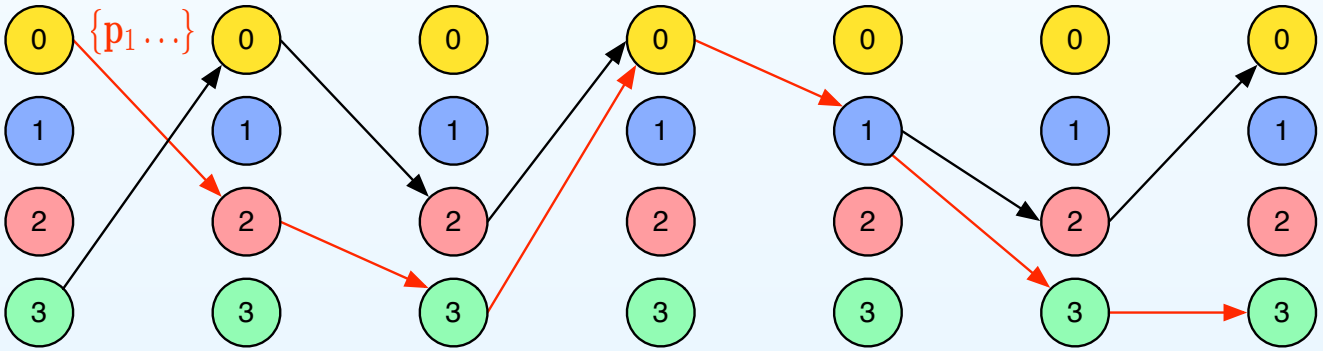
2) A general list of preimages is composed indicating for each the corresponding de Bruijn network.

Path	Network
p_1	0
p_2	0
$\ldots$	$\ldots$
$p_{(k^n-1)}$	3

Computational implementation

3) To overlap networks of states a, b:

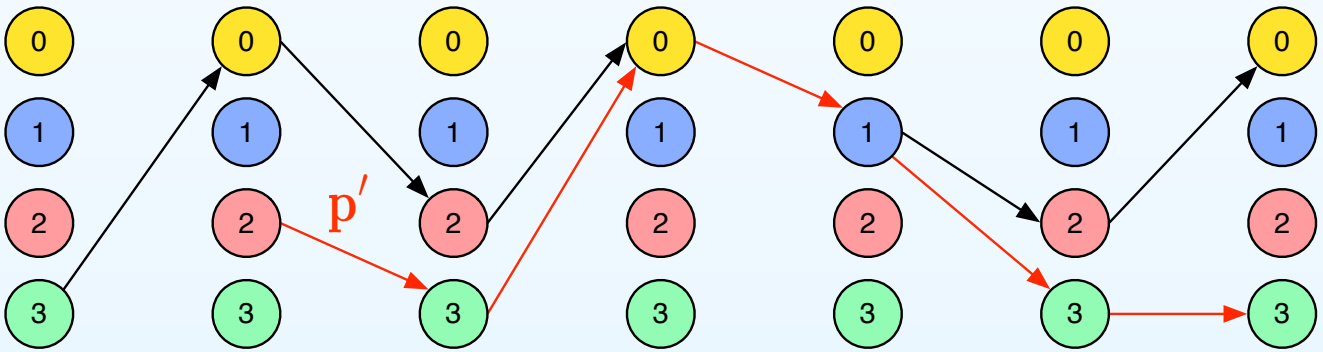
- Take network a:



Computational implementation

3) To overlap networks of states a, b:

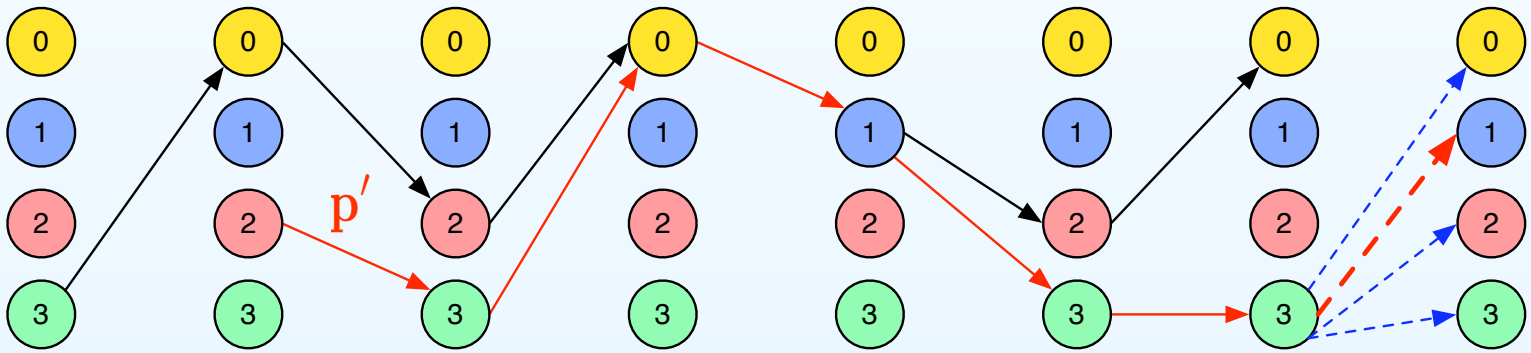
- Take $p' = p_1/k$:



Computational implementation

3) To overlap networks of states a, b:

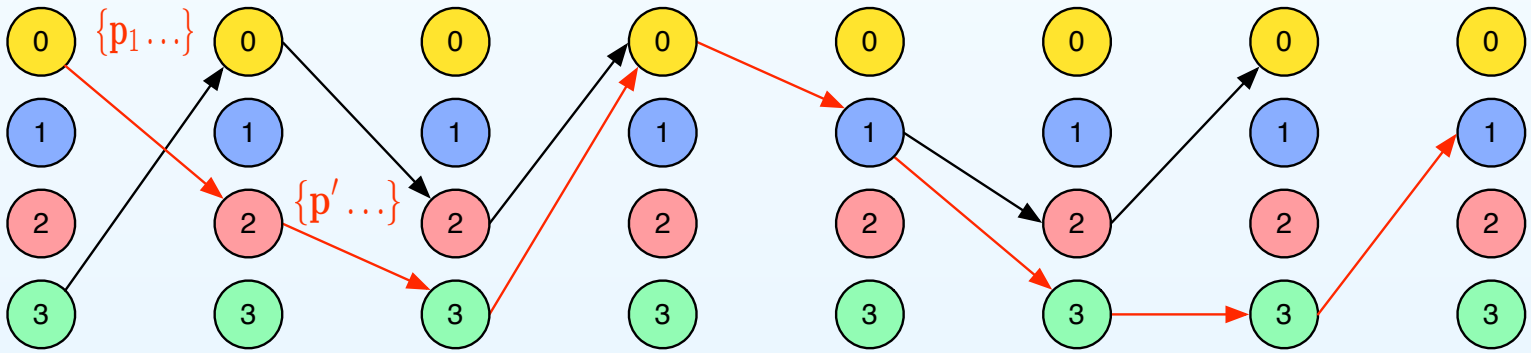
- For $0 \leq i \leq k-1$, take $p' = p' + i$:



Computational implementation

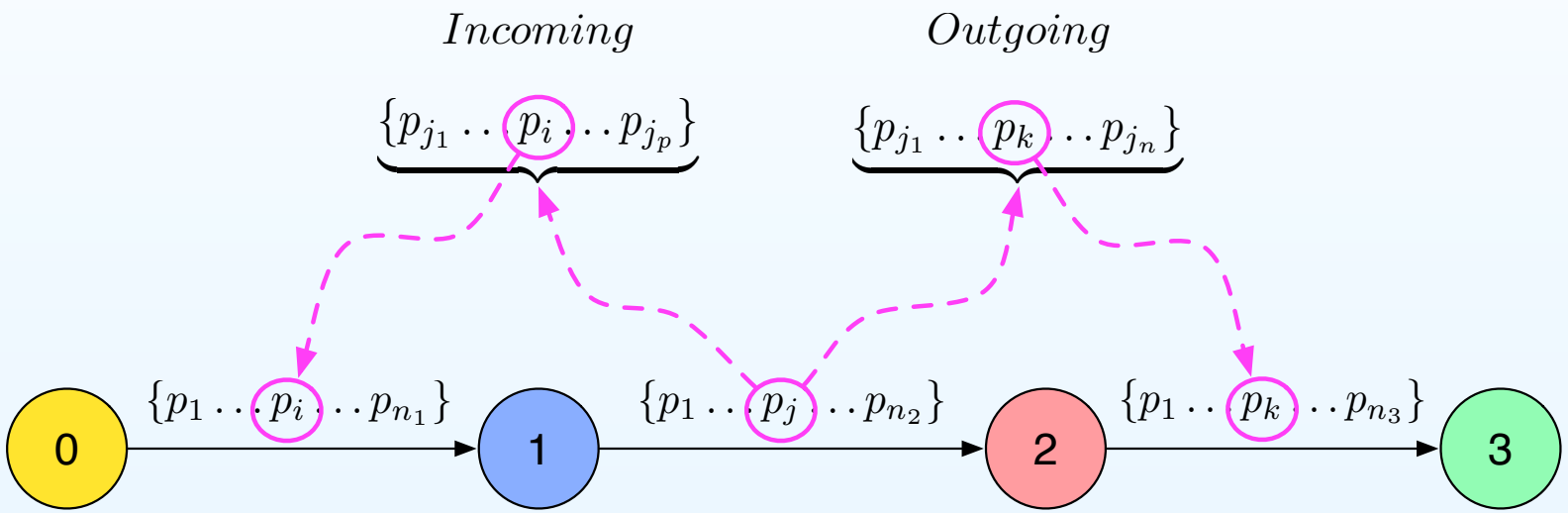
3) To overlap networks for states a, b:

- If $p'.Network = b$:



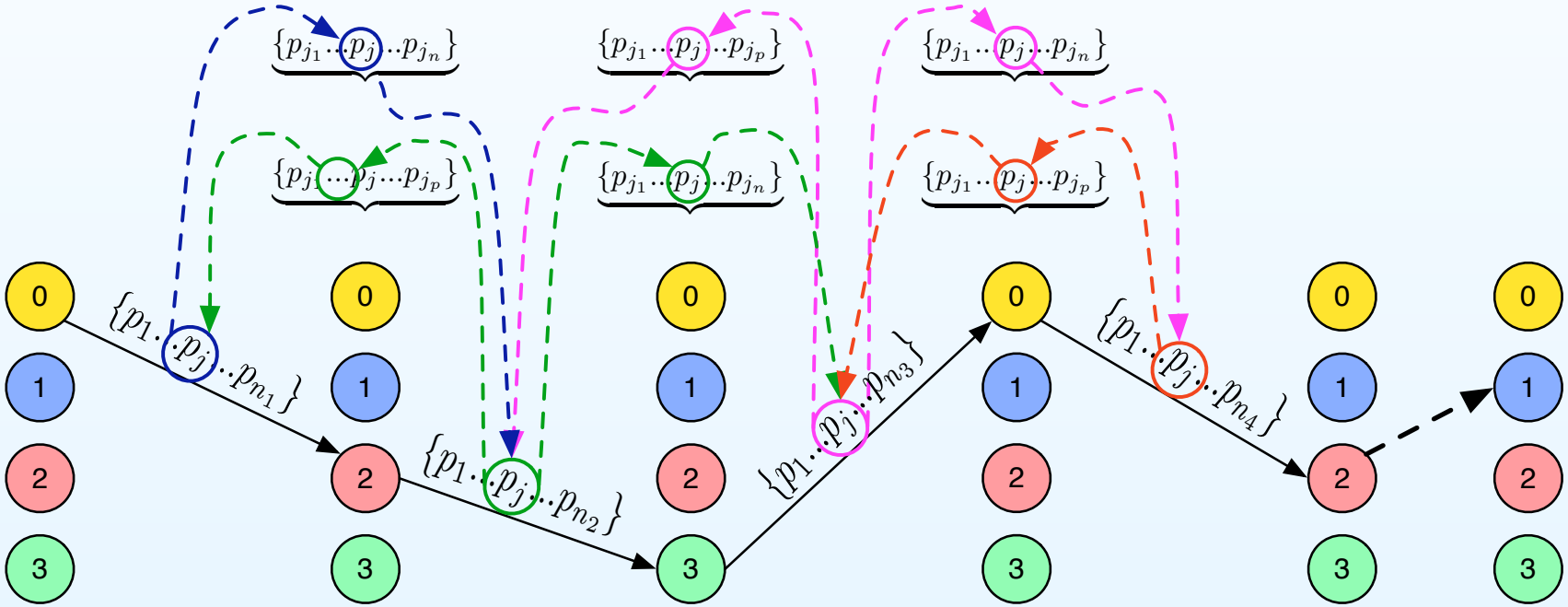
Computational implementation

4) Listing preimages:



Computational implementation

4) Listing preimages:

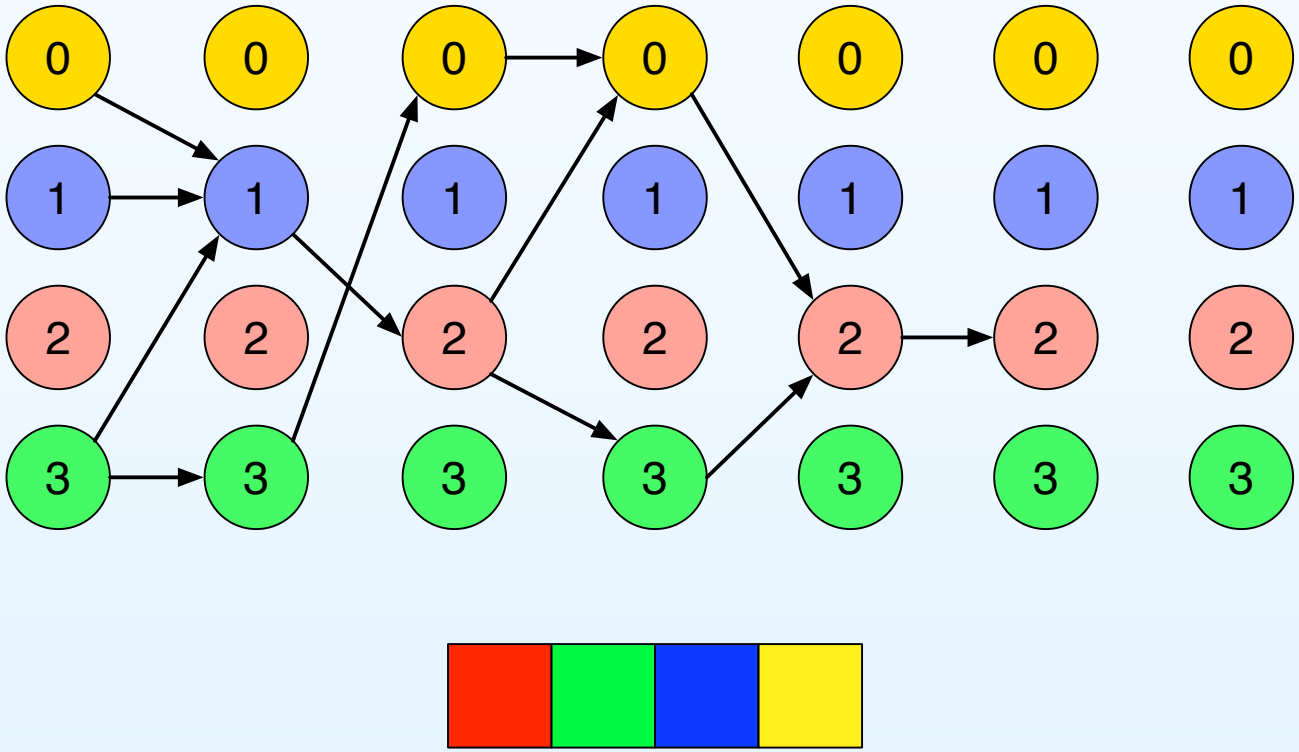


Improving the process

1) Calculate networks for sequences of n_1 states.

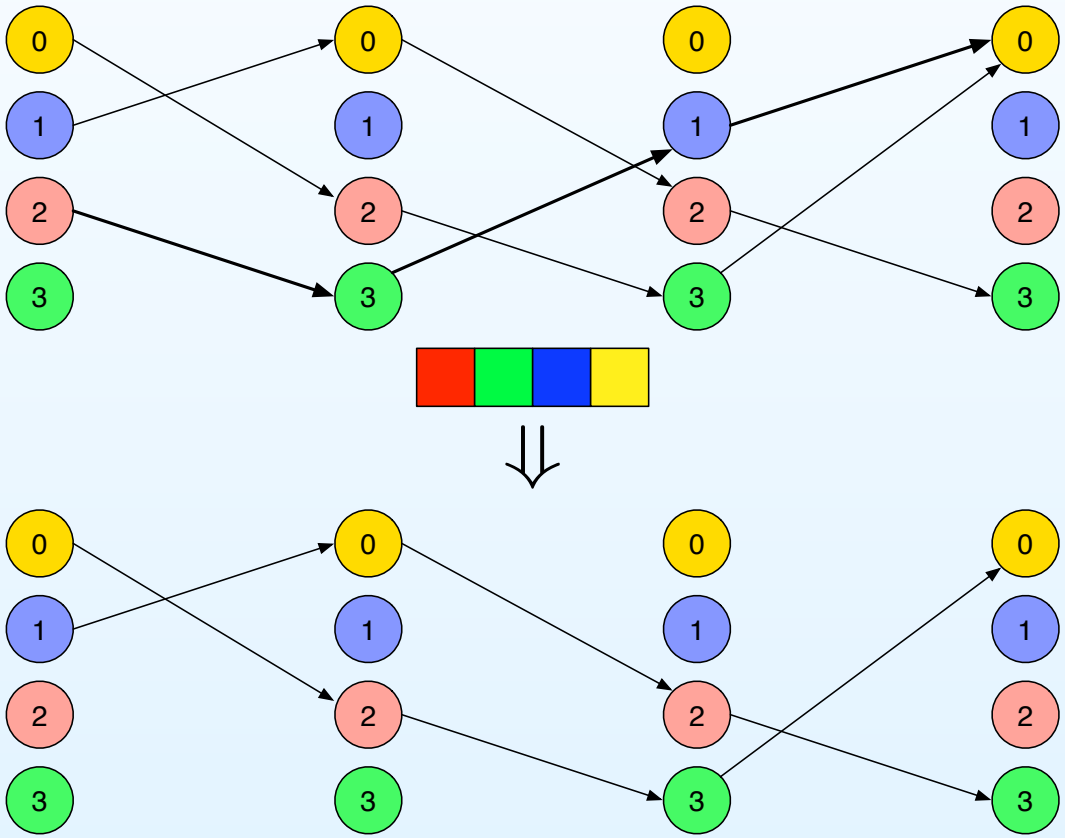
Improving the process

2) Take $n_3 \geq n_1$ and calculate Garden-of-Eden sequences of n_2 states for $n_1 \leq n_2 \leq n_3$.



Improving the process

3) Simplify the de Bruijn networks deleting all Garden-of-Eden paths with length n_1.



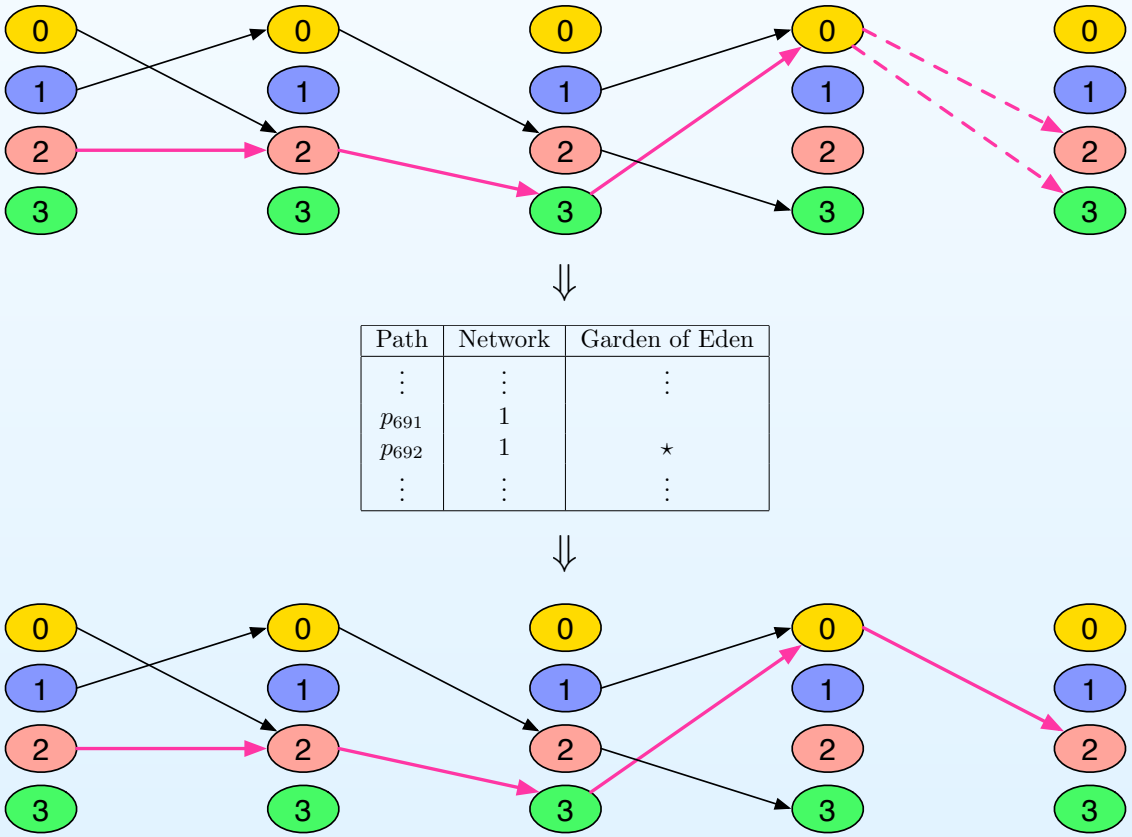
Improving the process

3) Simplify the de Bruijn networks deleting all Garden-of-Eden paths with length n_1.

Path	Network	Garden of Eden
p_1	0	$\star$
p_2	0	
$\ldots$	$\ldots$	$\ldots$
p_{181}	0	$\star$
$\ldots$	$\ldots$	$\ldots$
$p_{(k^n-1)}$	3	

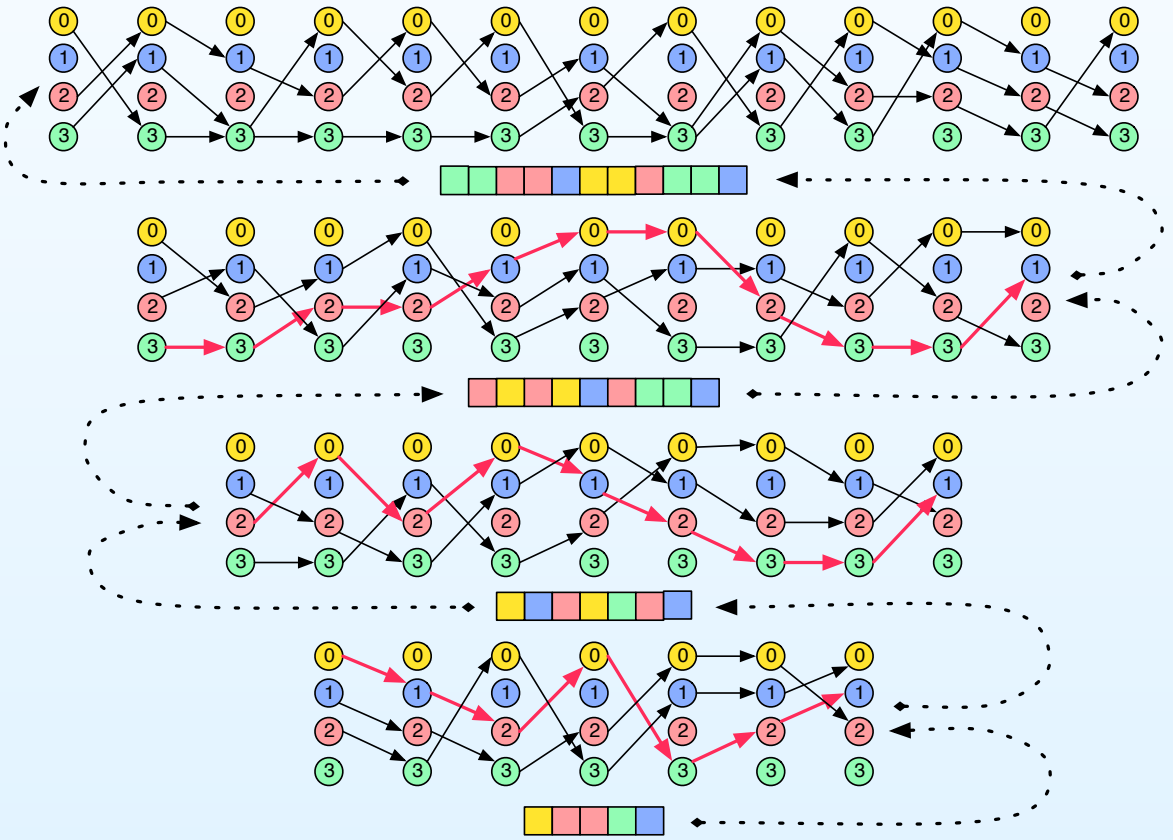
Improving the process

4) For a desired $w \in K^*$, calculate the de Bruijn network of preimages by overlapping basic networks; checking that no Garden-of-Eden paths are added.



Improving the process

5) Take a path in the de Bruijn network previously calculated and repeat step 4. Repeat the process n_4 times for obtaining preimages of $(n_1 - 1) * n_4$ generations.



Results in Rule 110

Parameters to find preimages for $T_{18}, T_{20}, T_{22}, T_{26}$ [a].

- Sequences of length 7 for generating the de Bruijn networks.

[a] Running in an iMac Intel Core 2 Duo, 2.16 GHz, 1 GB RAM

Results in Rule 110

Parameters to find preimages for $T_{18}, T_{20}, T_{22}, T_{26}$ [a].

- Sequences of length 7 for generating the de Bruijn networks.
- Sequences from length 7 to 10 to avoid Garden-of-Eden subsequences.

[a] Running in an iMac Intel Core 2 Duo, 2.16 GHz, 1 GB RAM

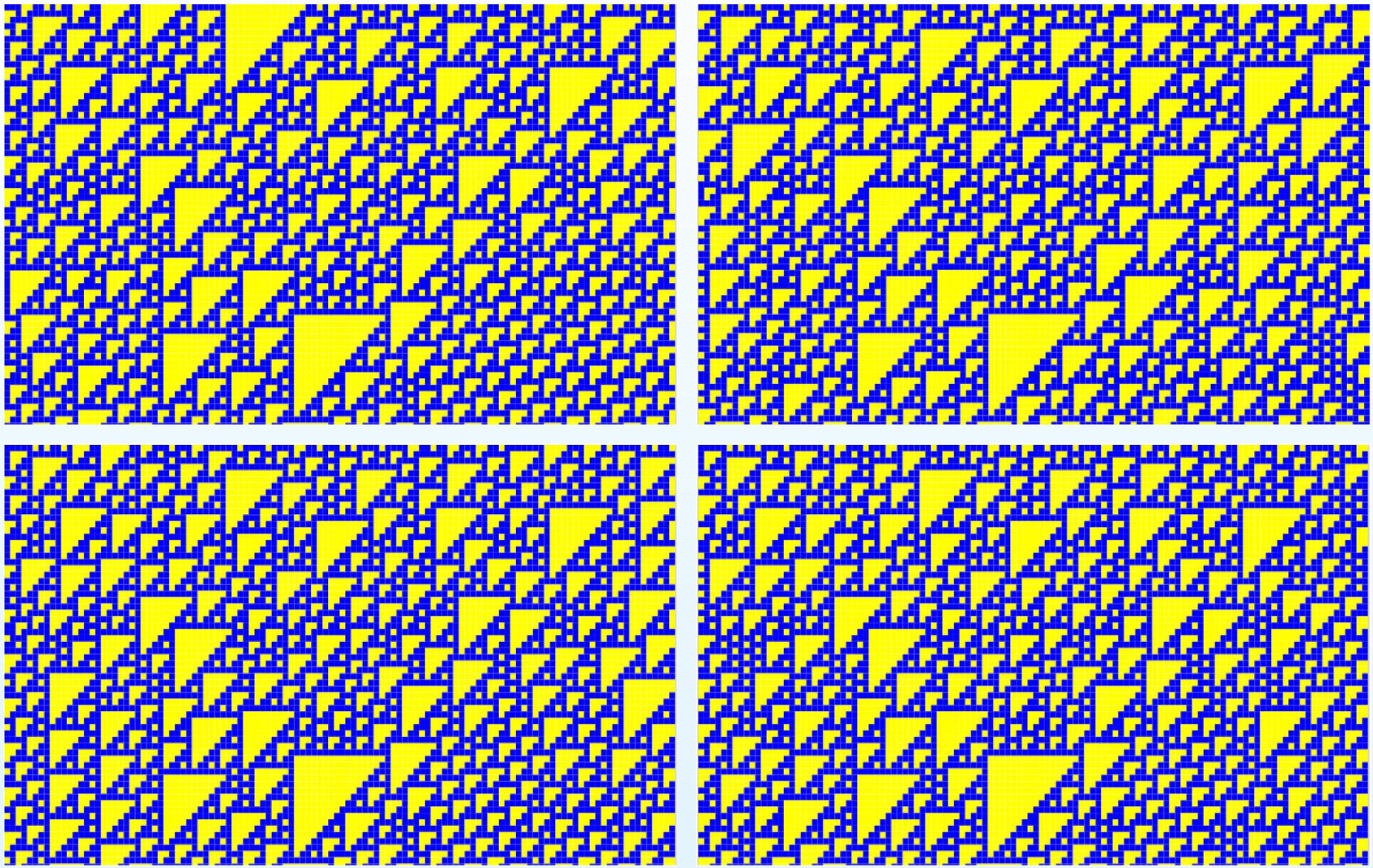
Results in Rule 110

Parameters to find preimages for T_{18}, T_{20}, T_{22}, T_{26} [a].

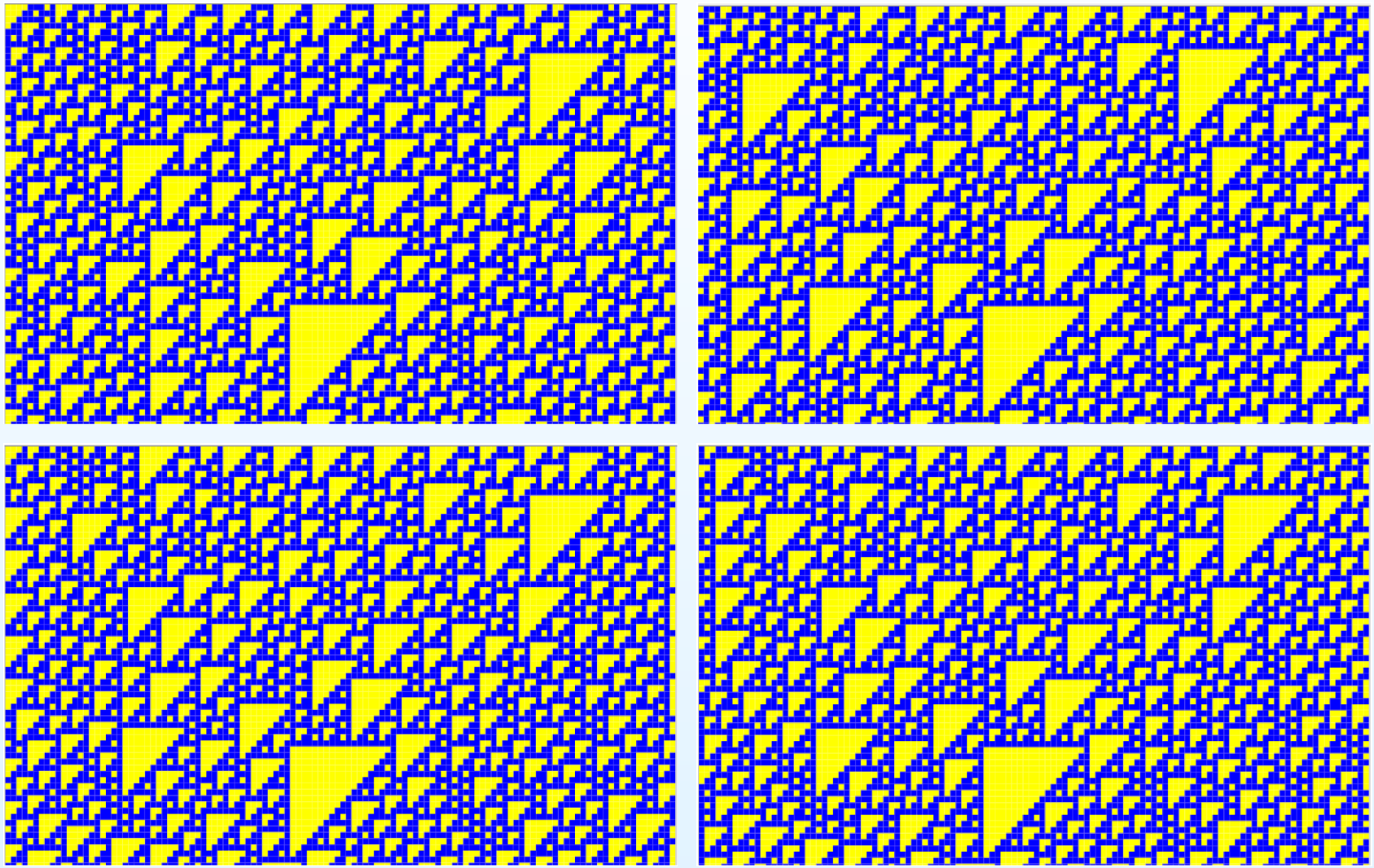
- Sequences of length 7 for generating the de Bruijn networks.
- Sequences from length 7 to 10 to avoid Garden-of-Eden subsequences.
- 8 iterations to return 48 generations.

[a] Running in an iMac Intel Core 2 Duo, 2.16 GHz, 1 GB RAM

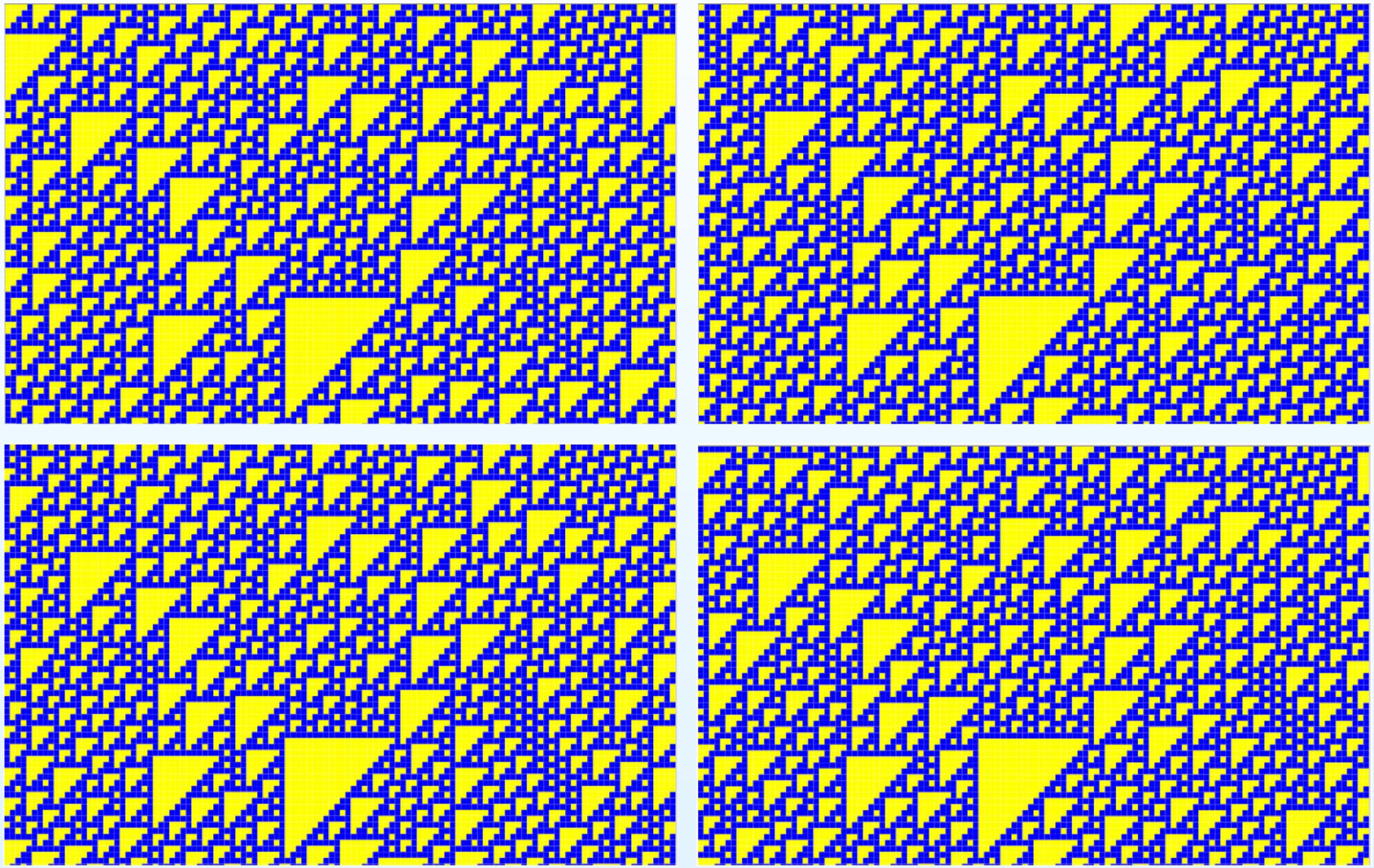
Results in Rule 110, T_{18}



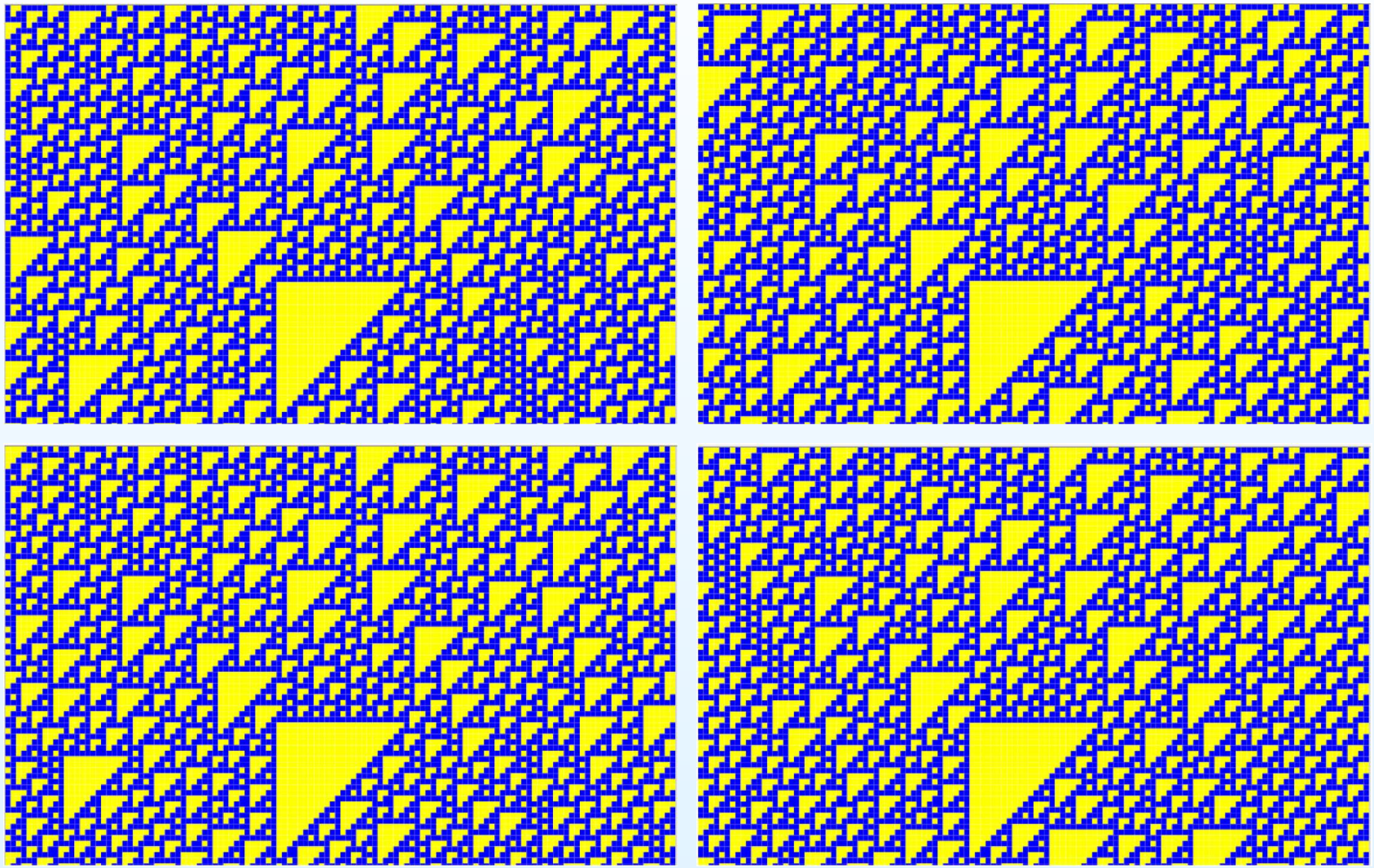
Results in Rule 110, T_{20}



Results in Rule 110, T_{22}



Results in Rule 110, T_{26}



Discussion

- De Bruijn networks may reduce time and space in order to calculate preimages in several generations.

Discussion

- De Bruijn networks may reduce time and space in order to calculate preimages in several generations.
- This procedure is practical up to a few tens of steps backwards.

Discussion

- De Bruijn networks may reduce time and space in order to calculate preimages in several generations.
- This procedure is practical up to a few tens of steps backwards.
- More efforts are necessary for the study and applications of de Bruijn diagrams.

Internet references

- http://en.wikibooks.org/wiki/Cellular_Automata/Listing_Preimages
- http://www.rattus.info/al/al.html
- http://cellular.ci.ulsa.mx/
- http://uncomp.uwe.ac.uk/genaro/index.html
- http://www.ci.ulsa.mx/~jmgomez/