# De Bruijn -graphs

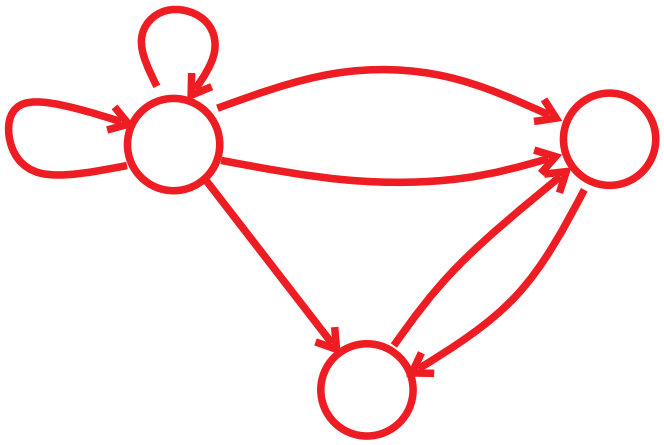
A finite graph representation of a **one-dimensional** CA from which one can see whether the CA is injective or surjective.

A **directed (multi)graph** consists of

- a finite **vertex** set $V$,
- a finite **edge** set $E$,
- functions $h : E \longrightarrow V$ and $t : E \longrightarrow V$ that give for each edge its **head** and **tail** vertex.

We say that an edge $e \in E$ is from $t(e)$ to $h(e)$. If $t(e) = h(e)$ then the edge $e$ is a **loop**.

The graph is conveniently drawn as a diagram with nodes $V$ and each edge $e$ drawn as an arrow from node $t(e)$ to node $h(e)$:



The **outdegree** of a vertex $v$ is the number of **outgoing** edges from $v$ (=edges whose tail is $v$).

The **indegree** of a vertex $v$ is the number of **incoming** edges from $v$ (=edges whose head is $v$).

A finite **path** (of length $k$) is a sequence

$$e_1, e_2, \ldots, e_k$$

of edges where for all $i$

$$h(e_i) = t(e_{i+1}).$$

(Paths "follow the arrows" in the diagram.)

A **two-way infinite path** is a sequence

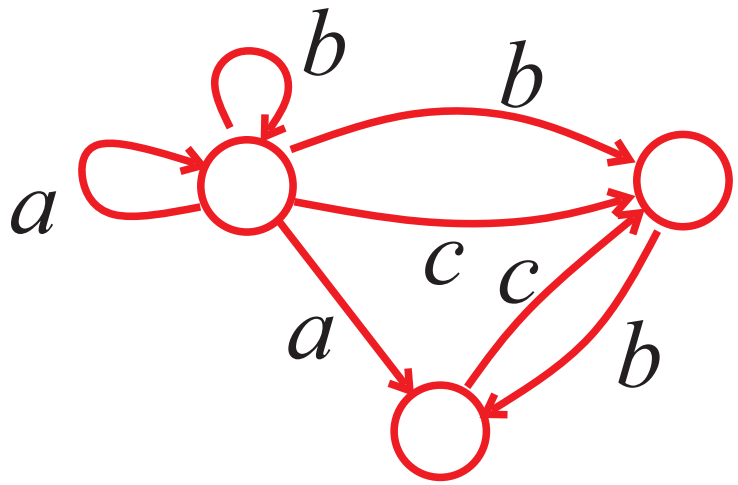
$$p : \mathbb{Z} \longrightarrow E$$

such that for every $i \in \mathbb{Z}$ we have $h(p(i)) = t(p(i+1))$.

An (edge) **labeled directed graph** is a directed graph together with a labeling function

$$\ell : E \longrightarrow \Sigma$$

which assigns each edge a symbol from a finite set $\Sigma$ of labels.



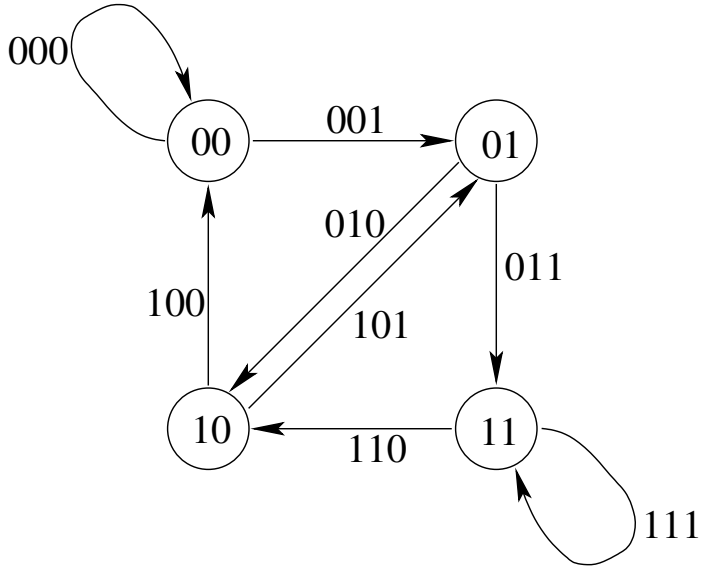

The label of a finite or infinite path is the sequence of elements of $\Sigma$ obtained by reading the labels of its edges:

$$\ldots \mathbf{aaabcbcbcbc} \ldots$$

Let $m$ be a positive integer and let $S$ be a finite set. The **de Bruijn graph** of width $m$ over alphabet $S$ is the directed graph with

$$\begin{aligned} V &= S^{m-1}, \\ E &= S^m, \\ t(s_1 s_2 \ldots s_m) &= s_1 s_2 \ldots s_{m-1}, \text{ and} \\ h(s_1 s_2 \ldots s_m) &= s_2 s_3 \ldots s_m. \end{aligned}$$

(There is an edge $sut$ from node $su$ to node $ut$ for all $s, t \in S$ and $u \in S^{m-2}$.)

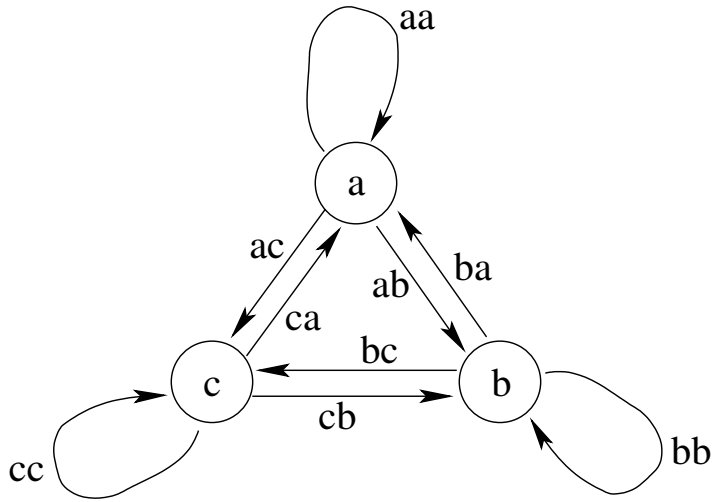



$S = \{0, 1\}, m = 3$

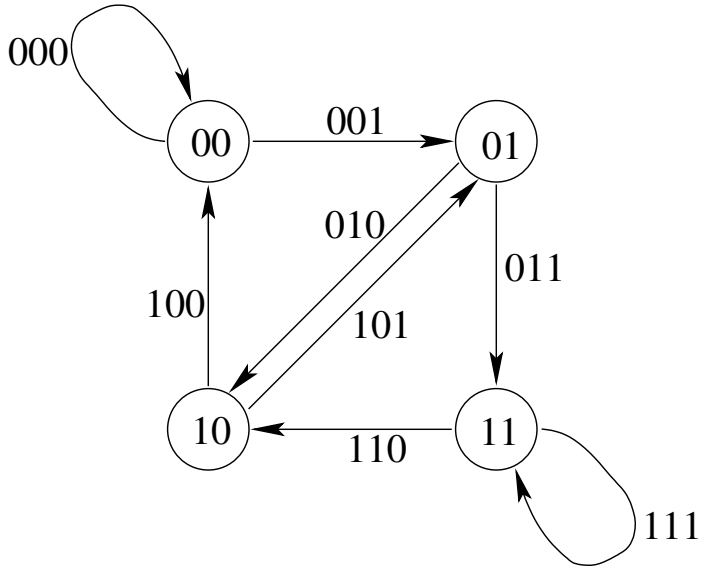
Let $m$ be a positive integer and let $S$ be a finite set. The **de Bruijn graph** of width $m$ over alphabet $S$ is the directed graph with

$$\begin{aligned} V &= S^{m-1}, \\ E &= S^m, \\ t(s_1 s_2 \ldots s_m) &= s_1 s_2 \ldots s_{m-1}, \text{ and} \\ h(s_1 s_2 \ldots s_m) &= s_2 s_3 \ldots s_m. \end{aligned}$$

(There is an edge $sut$ from node $su$ to node $ut$ for all $s, t \in S$ and $u \in S^{m-2}$.)



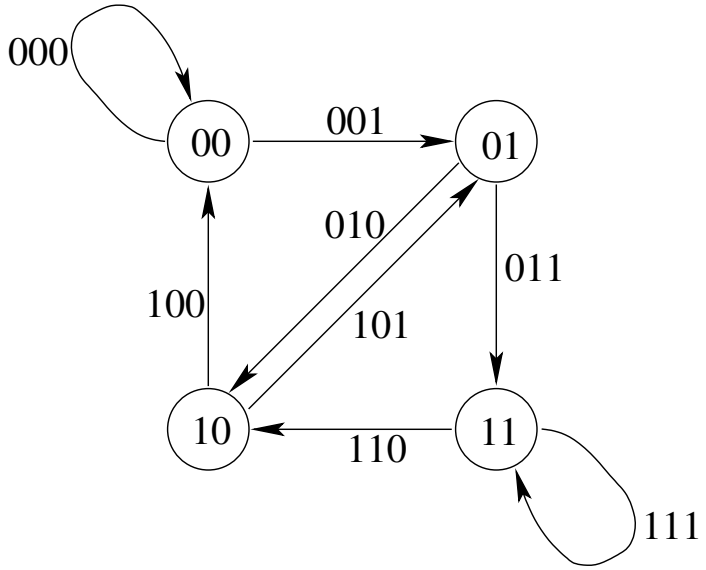

$S = \{a, b, c\}, m = 2$




For every $c \in S^{\mathbb{Z}}$ there is a corresponding two-way infinite path $p \in E^{\mathbb{Z}}$ with

$$p(i) = c(i)c(i+1)\ldots c(i+m-1) \text{ for all } i \in \mathbb{Z}.$$

Path $p$ is obtained by sliding a window of width $m$ over $c$. The views through the sliding window are the edges on $p$.

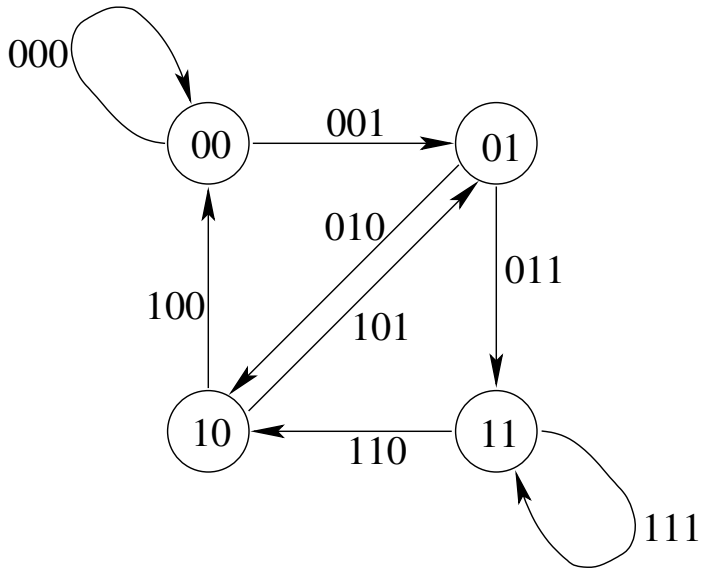



For every $c \in S^{\mathbb{Z}}$ there is a corresponding two-way infinite path $p \in E^{\mathbb{Z}}$ with

$$p(i) = c(i)c(i+1)\ldots c(i+m-1) \text{ for all } i \in \mathbb{Z}.$$

Path $p$ is obtained by sliding a window of width $m$ over $c$. The views through the sliding window are the edges on $p$.

$$\mathbf{c} = \ldots\ \mathbf{0\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ 1}\ \ldots$$
$$\mathbf{p} = \ldots \mathbf{001, 010, 101, 011, 110, 100, 010, 101, 011, 111}, \ldots$$




For every $c \in S^{\mathbb{Z}}$ there is a corresponding two-way infinite path $p \in E^{\mathbb{Z}}$ with

$$p(i) = c(i)c(i+1)\ldots c(i+m-1) \text{ for all } i \in \mathbb{Z}.$$

Path $p$ is obtained by sliding a window of width $m$ over $c$. The views through the sliding window are the edges on $p$.

$$\mathbf{c} = \ldots\ \mathbf{0\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ 1}\ \ldots$$
$$\mathbf{p} = \ldots \mathbf{001, 010, 101, 011, 110, 100, 010, 101, 011, 111}, \ldots$$

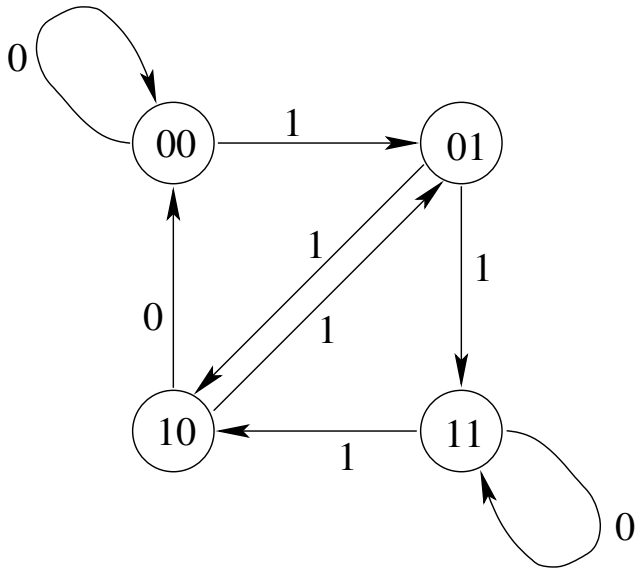
The correspondence $c \leftrightarrow p$ is **bijective**: $c$ is obtained from path $p$ by reading the first letters of the edges. Let us denote the configuration $c$ that corresponds to path $p$ by $c_p$.

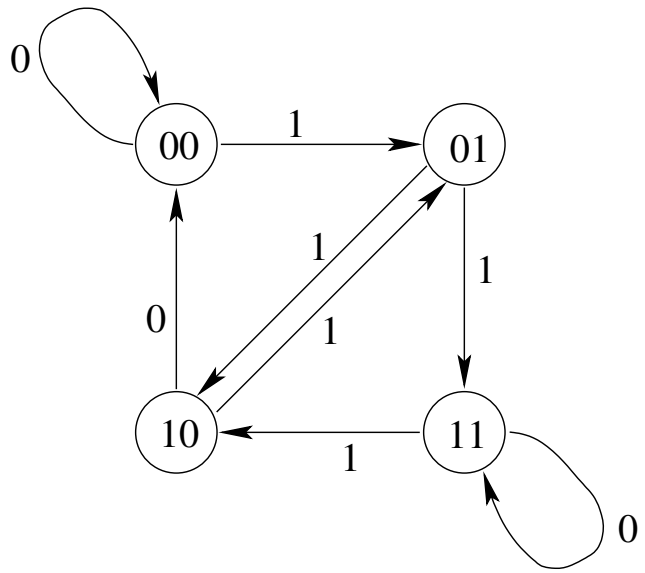
Let $G$ be a one-dimensional CA with

- state set $S$,
- neighborhood range $m$, and
- local rule $f : S^m \longrightarrow S$.

The **de Bruijn representation** of the CA is the labeled de Bruijn graph of width $m$ over the alphabet $S$ where edge $e \in S^m$ is labeled by $f(e) \in S$.

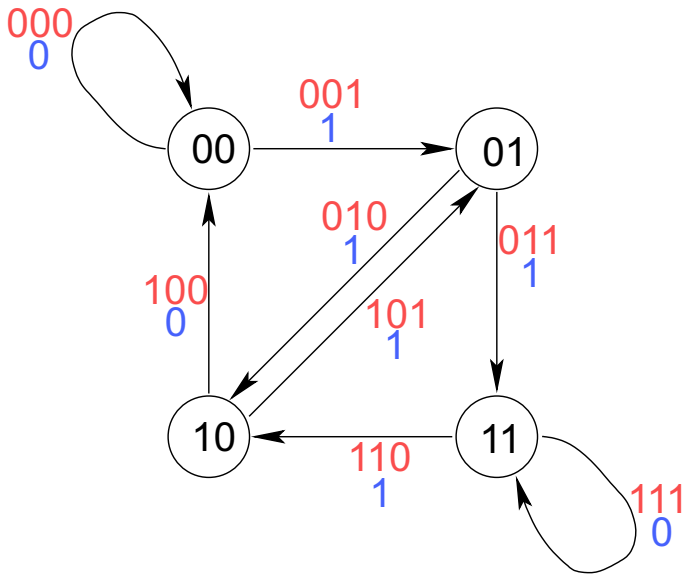
Here's de Bruijn representation of rule 110:






**Remark:** Since the position of the CA neighborhood is not given, the CA functions $G \circ \sigma^k$ are represented by the same de Bruijn graph for all $k \in \mathbb{Z}$.

Fortunately this is not a problem since we use the de Bruijn representations to study properties (injectivity and surjectivity) that are not affected by translations.



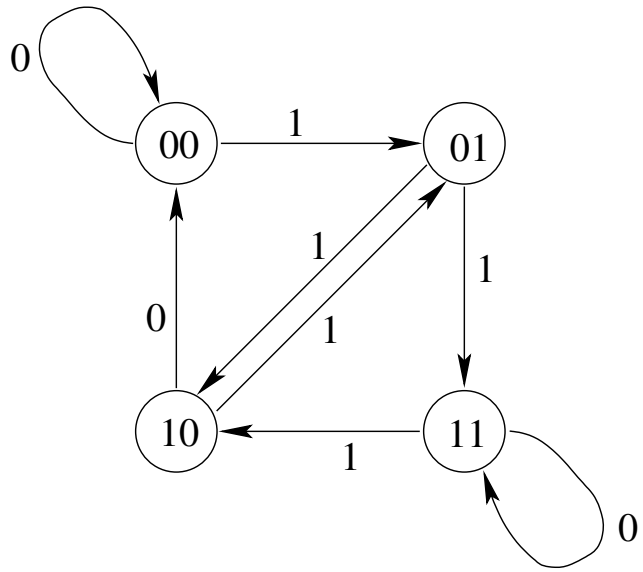

Infinite paths $p$ in the de Bruijn representations provide **two** configurations:

- $c_p$, the sequence obtained by reading the first symbol of the names of the edges along $p$,
- $f_p$, the sequence obtained by reading the labels of the edges.

Because the labels are the outputs of the local update rule of the CA, $f_p$ is (possibly translated) $G(c_p)$:

$$\sigma^k(f_p) = G(c_p) \text{ for some } k \in \mathbb{Z}.$$

(Number $k$ depends on the positioning of the neighborhood in the CA.)




- The CA is **injective** if and only if different two-way infinite paths have always different labels.
- The CA is **surjective** if and only if for every $c \in S^{\mathbb{Z}}$ there is a path labeled by $c$.
- The CA is **pre-injective**if and only if the graph does not have a **diamond**: two different finite paths with identical labels that begin in the same vertex and end in the same vertex.
- An **orphan** is a word over alphabet $S$ that is not a label of any path.

Recall some **finite automata** terminology:

- A labeled digraph is a finite **semiautomaton**. Vertices are the **states** of the automaton, labels are **input letters**, edges are **transitions**.

Recall some **finite automata** terminology:

- A labeled digraph is a finite **semiautomaton**. Vertices are the **states** of the automaton, labels are **input letters**, edges are **transitions**.
- Specifying two subsets $I \subseteq V$ and $F \subseteq V$ of vertices turns the graph into a **finite automaton**. Elements of $I$ and $F$ are the **initial** and the **final** states.

Recall some **finite automata** terminology:

- A labeled digraph is a finite **semiautomaton**. Vertices are the **states** of the automaton, labels are **input letters**, edges are **transitions**.
- Specifying two subsets $I \subseteq V$ and $F \subseteq V$ of vertices turns the graph into a **finite automaton**. Elements of $I$ and $F$ are the **initial** and the **final** states.
- Finite automaton **accepts** a word $w$ if there exists a path labeled by $w$ from an initial state to a final state.

Recall some **finite automata** terminology:

- A labeled digraph is a finite **semiautomaton**. Vertices are the **states** of the automaton, labels are **input letters**, edges are **transitions**.
- Specifying two subsets $I \subseteq V$ and $F \subseteq V$ of vertices turns the graph into a **finite automaton**. Elements of $I$ and $F$ are the **initial** and the **final** states.
- Finite automaton **accepts** a word $w$ if there exists a path labeled by $w$ from an initial state to a final state.
- The **language recognized** by a finite automaton is the set of all words that it accepts.

Recall some **finite automata** terminology:

- A labeled digraph is a finite **semiautomaton**. Vertices are the **states** of the automaton, labels are **input letters**, edges are **transitions**.
- Specifying two subsets $I \subseteq V$ and $F \subseteq V$ of vertices turns the graph into a **finite automaton**. Elements of $I$ and $F$ are the **initial** and the **final** states.
- Finite automaton **accepts** a word $w$ if there exists a path labeled by $w$ from an initial state to a final state.
- The **language recognized** by a finite automaton is the set of all words that it accepts.
- Languages that are recognized by finite automata are **regular**.

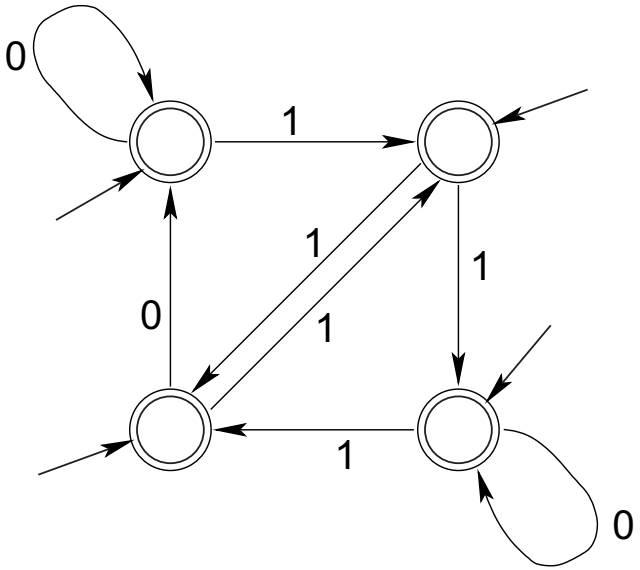
Recall some **finite automata** terminology:

- A labeled digraph is a finite **semiautomaton**. Vertices are the **states** of the automaton, labels are **input letters**, edges are **transitions**.
- Specifying two subsets $I \subseteq V$ and $F \subseteq V$ of vertices turns the graph into a **finite automaton**. Elements of $I$ and $F$ are the **initial** and the **final** states.
- Finite automaton **accepts** a word $w$ if there exists a path labeled by $w$ from an initial state to a final state.
- The **language recognized** by a finite automaton is the set of all words that it accepts.
- Languages that are recognized by finite automata are **regular**.
- Two finite automata are **equivalent** if they recognize the same language.

**Recall:** A word is **not an orphan** if and only if it labels some finite path in the de Bruijn representation of the CA.

Make every state initial and final

$\Longrightarrow$ a finite automaton that recognizes exactly non-orphan words.



To recognize orphans we need a finite automaton for the **complement language**. And for complementation we need determinism.

A finite automaton is **deterministic** (DFA) if

- there is only one initial state, and
- for each state $v \in V$ and letter $a \in \Sigma$ there is at most one transition with label $a$ from state $v$.

$\Longrightarrow$ at most one path from the initial state for every input word $w \in \Sigma^*$. Word $w$ is accepted iff the path ends at a final state:

A DFA is **complete** if there is a (unique) transition from every state with every input letter.

It is easy to make any DFA into an equivalent complete DFA by adding a new state (which is not final) and making all missing transitions into this **sink** state.

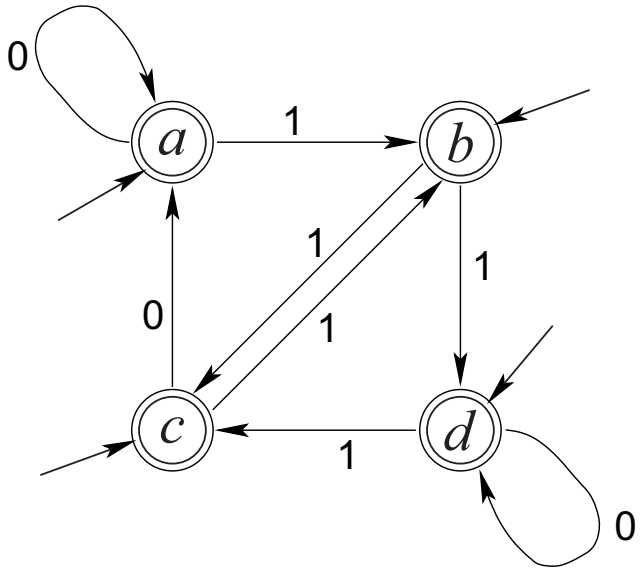
**Well known:** Any finite automaton can be effectively converted into an equivalent complete DFA using the **powerset construction**.

- The states of the powerset automaton are subsets of the original state set $V$.
- For any $X \subseteq V$ and $a \in \Sigma$ we add the transition

  $X \xrightarrow{a} \{v \in V |$ for some $x \in X$ there is an edge $x \xrightarrow{a} v$ with label $a\}$.

- The set $I \subseteq V$ of initial states is made the unique initial state.
- A set $X \subseteq V$ is made final if and only if $X$ contains at least one original final state.

**Example.** Let us construct a complete DFA that is equivalent to the de Bruijn representation of rule 110:




Note that the construction can be started with the initial state set $I$, and only the sets reached from $I$ need to be included in the powerset automaton.

A complete DFA is easily modified to recognize the complement language: simply make final states non-final and vice versa:

$$\text{final} \longleftrightarrow \text{non-final}$$

In the case of the powerset automaton of the de Bruijn representation of CA this means that there will be a unique final state: the empty set $\emptyset$.

A complete DFA is easily modified to recognize the complement language: simply make final states non-final and vice versa:

$$\text{final} \longleftrightarrow \text{non-final}$$

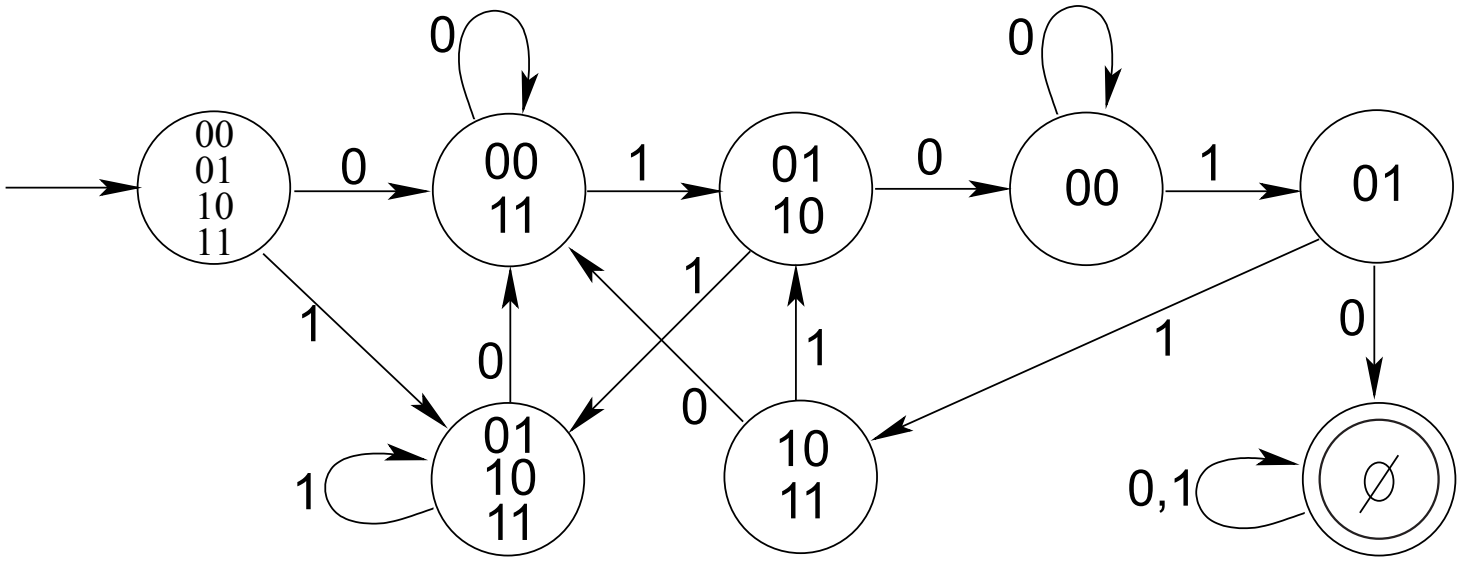
In the case of the powerset automaton of the de Bruijn representation of CA this means that there will be a unique final state: the empty set $\emptyset$.

The resulting complete DFA exactly recognizes all words that are orphans. From the automaton we can determine

- what are the shortest orphans,
- are there orphans at all (i.e., is the automaton surjective or not).

**Proposition.** The orphan words of a one-dimensional CA form a regular language.

**Example.** A complete DFA for the orphans of rule 110:




The shortest orphan is 01010, given by the shortest path from the initial state to the final state.

The CA is surjective if and only if there is no path in the complemented powerset automaton from the initial state ($= V$) to the final state ($= \emptyset$).

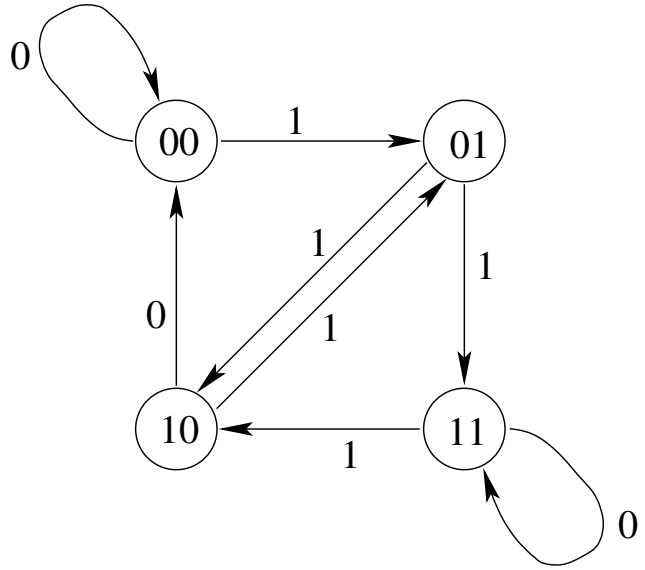
This **algorithm** to decide surjectivity is however very **inefficient**: the size of the powerset grows exponentially with the number of states in the CA.

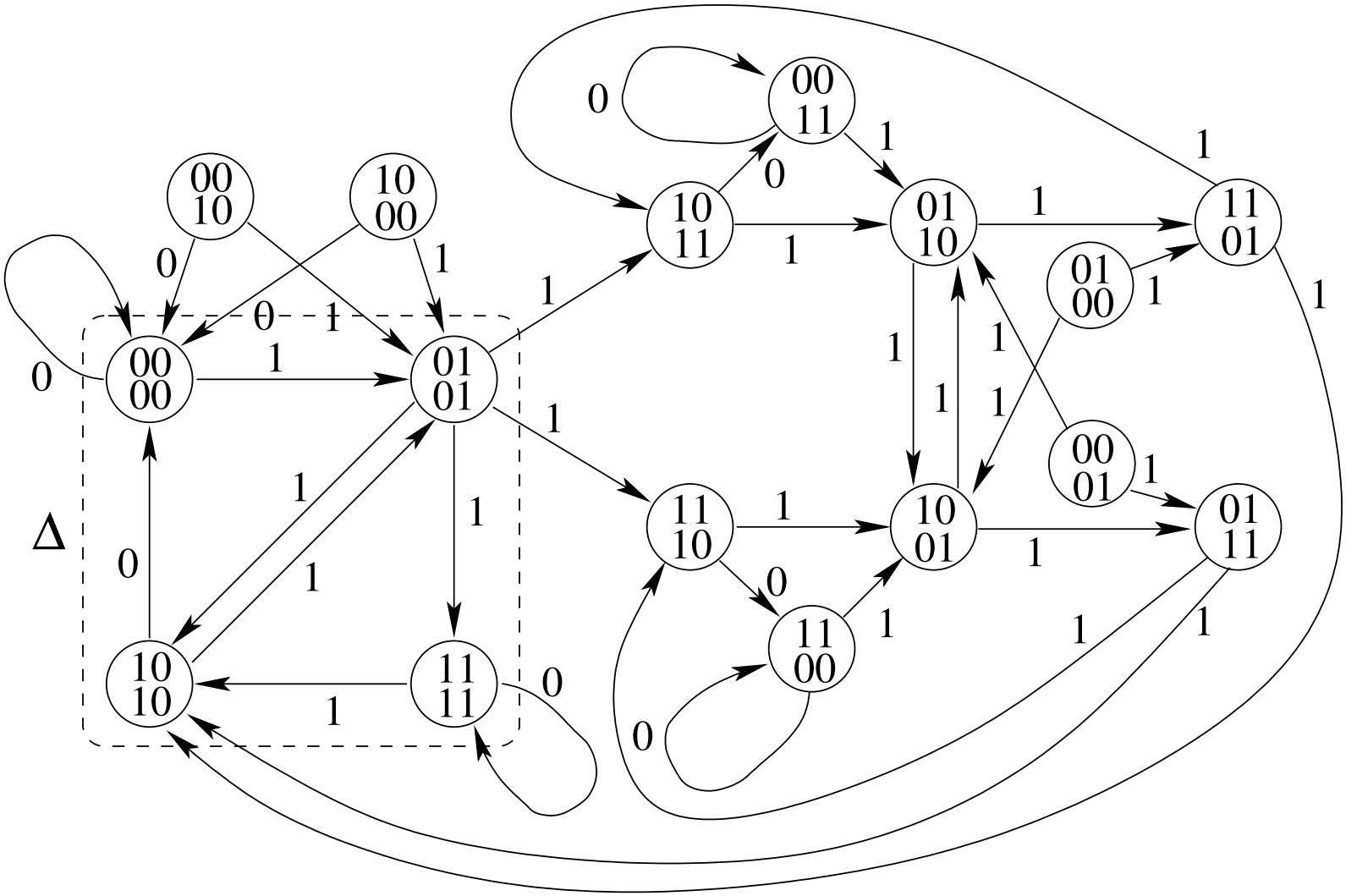
The **pair automaton** provides an efficient way of deciding surjectivity and injectivity. Let $B$ be the de Bruijn representation of the CA. In the pair automaton

- the state set is $V \times V$ where $V = S^{m-1}$ is the vertex set of $B$,
- there is an edge
$$(u_1, u_2) \xrightarrow{a} (v_1, v_2)$$
with label $a$ if and only if in $B$ there are both edges
$$u_1 \xrightarrow{a} v_1 \text{ and } u_2 \xrightarrow{a} v_2.$$

No initial or final states need to be fixed. (So semiautomaton.)

**Example.** Let us construct the pair automaton for rule 110 from its de Bruijn representation




00
10
10
00
00
00
01
01
10
10
11
11
Δ
00
11
10
11
01
10
11
01
01
00
00
01
11
10
01
11
00
0
1

Any two-way infinite **path $p$ in the pair graph** corresponds to a pair of paths in the original de Bruijn automaton with identical labels. These represent a **pair of configurations with the same image** under the CA.

Conversely, and pair $c_p^{(1)}$ and $c_p^{(2)}$ of configurations such that $G(c_p^{(1)}) = G(c_p^{(2)})$ provides a path $p$ in the pair automaton.

The correspondence

$$p \leftrightarrow (c_p^{(1)}, c_p^{(2)})$$

is a bijection between paths $p$ in the pair automaton and pairs of configurations that satisfy

$$G(c_p^1) = G(c_p^2).$$

Let

$$\Delta = \{(u, u) \mid u \in V\}$$

the set of **diagonal vertices** in the pair automaton.

The induced subgraph with vertex set $\Delta$ is an isomorphic copy of the de Bruijn automaton. In particular, there is a path between any two vertices in $\Delta$.

Any two-way infinite path $p$ that only uses diagonal vertices has $c_p^{(1)} = c_p^{(2)}$, so it does not provide two different configurations with the same image. Only paths that contain a vertex outside of $\Delta$ provide such configurations.

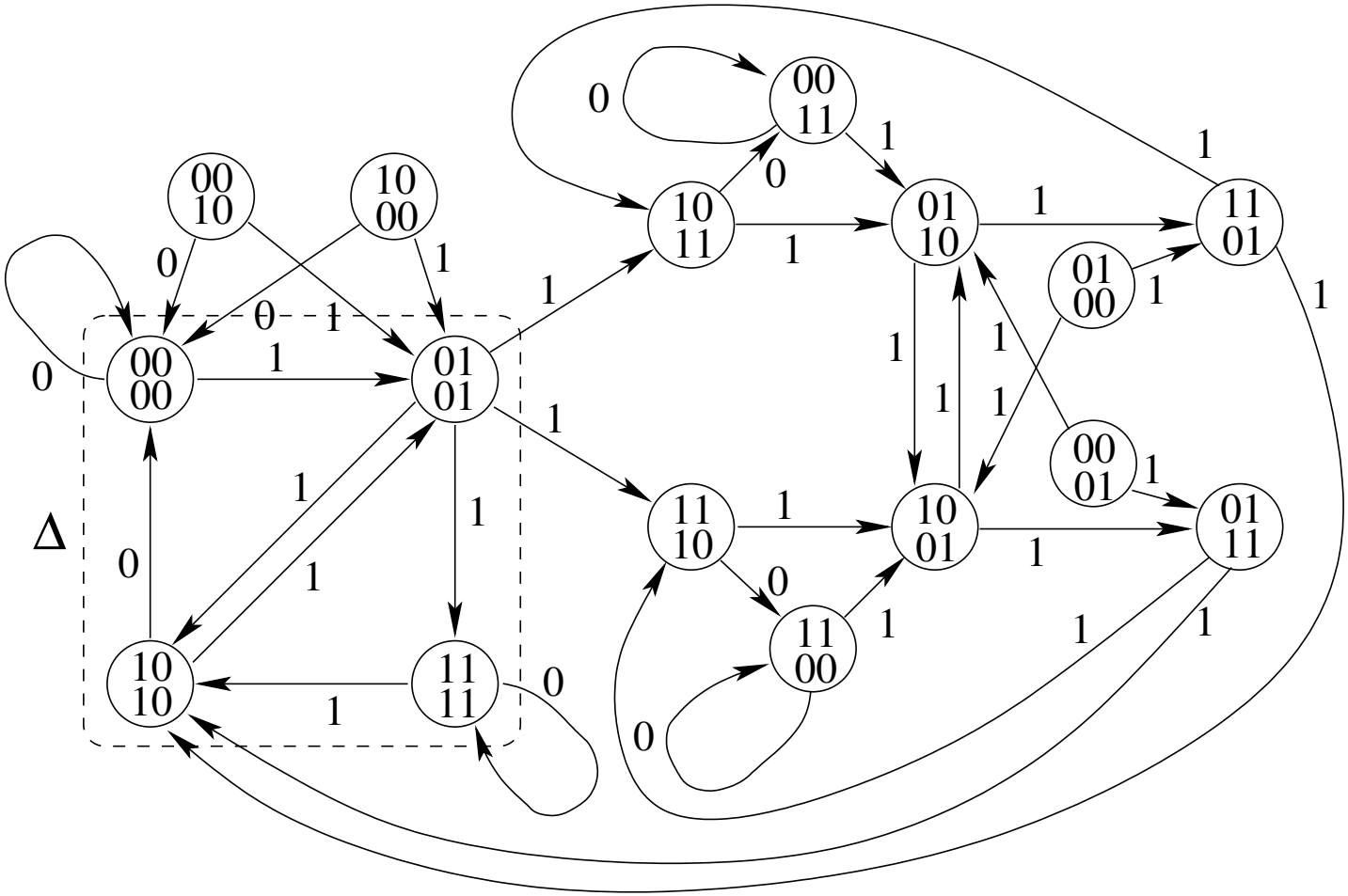
Basic connections between the pair graph and the CA:

**Proposition.** A one-dimensional CA $A$ is

(i) not injective if and only if its pair graph has a cycle that contains a node outside of $\Delta$,

(ii) not surjective if and only the pair graph has a cycle that contains a node of $\Delta$ and a node outside of $\Delta$.

**Proof.**




**Example.** One sees from the cycles that rule 110 is not injective or surjective.

The shortest path that begins and ends in node $(00, 00)$ and is not inside $\Delta$ has length 6. The corresponding patterns 00110100 and 00101100 are the shortest pair of distinct patterns that begin and end in 00, and have the same image.

One gets directly from the pair graph some of our **previous results**:

- If $G$ is surjective then any distinct pair $c, e$ of configurations with the same image $G(c) = G(e)$ has three possibilities regarding asymptoticity (inside $\Delta$) and $(m-1)$-separation (outside $\Delta$). Namely, path going

$$\text{inside } \Delta \longrightarrow \text{outside } \Delta \longrightarrow \text{inside } \Delta$$

  is not possible.

- $G$ is non-injective ($\exists$ an infinite path containing a node outside of $\Delta$) if and only if $G_P$ is non-injective ($\exists$ a cycle containing a node outside of $\Delta$).

The pair graph is **symmetric** w.r.t. swapping the two components of the vertices.

The **reduced pair graph** is obtained by

- Merging all pairs $(u, v)$ and $(v, u)$, and
- Merging all vertices of $\Delta$. Add no loop at the merged $\Delta$.

The size of the graph gets reduced by a factor $\approx 2$.

The CA is

- **injective** if and only if there is no cycle in the reduced pair graph,
- **surjective** if and only if there is no cycle through node $\Delta$.

**Example.** The reduced pair graph of **rule 110**