Solving temporal equations by folding de Bruijn graphs

Sam van Gool, Johannes Marti and Michelle Sweering

IRIF Université Paris Cité, University of Zürich, and CWI Amsterdam

ATLAS, Rennes, 25 April 2024

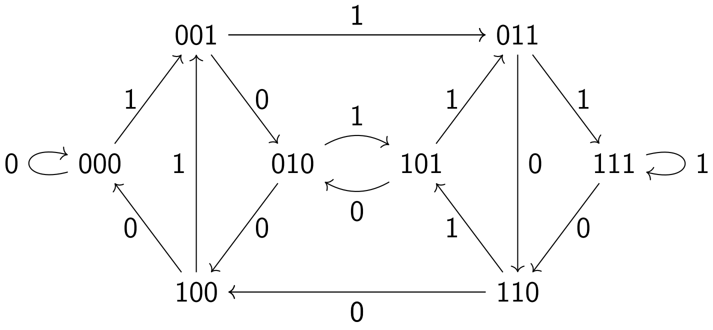
de Bruijn graphs

The de Bruijn graph $B_d(\Sigma)$ of dimension d over alphabet Σ is the deterministic automaton that 'remembers the last d letters'.

de Bruijn graphs

The de Bruijn graph $B_d(\Sigma)$ of dimension d over alphabet Σ is the deterministic automaton that 'remembers the last d letters'.

For example, when $d = 3$ and $\Sigma = \{0, 1\}$:



The de Bruijn graph $B_3(\{0, 1\})$

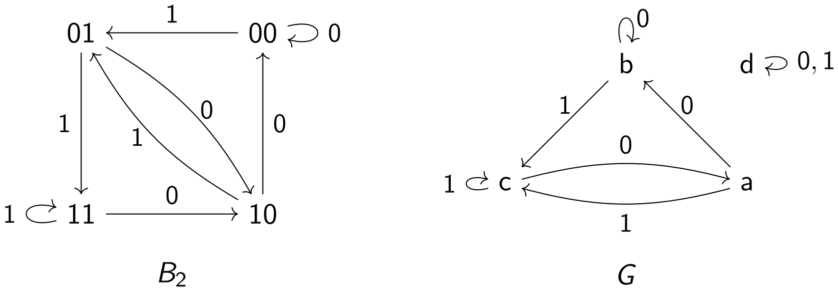
Homomorphisms of de Bruijn graphs

A homomorphism is a vertex function that preserves labeled edges.

Homomorphisms of de Bruijn graphs

A homomorphism is a vertex function that preserves labeled edges.

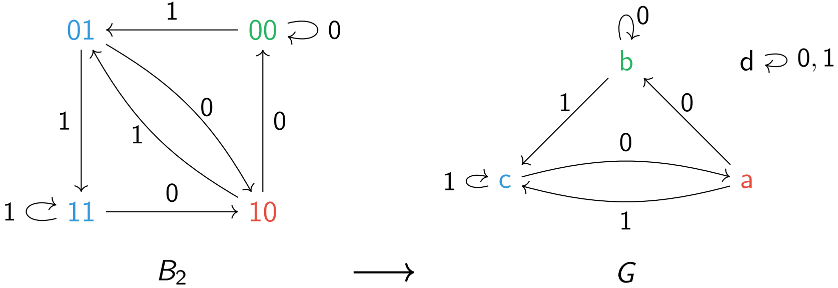
For example, a homomorphism from $B_2(\Sigma)$ to a graph G:



Homomorphisms of de Bruijn graphs

A homomorphism is a vertex function that preserves labeled edges.

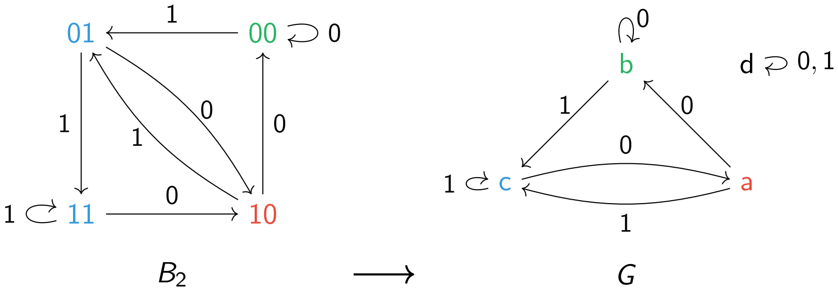
For example, a homomorphism from $B_2(\Sigma)$ to a graph G:



Homomorphisms of de Bruijn graphs

A homomorphism is a vertex function that preserves labeled edges.

For example, a homomorphism from $B_2(\Sigma)$ to a graph G:



Note:

- The codomain graph may fail to be deterministic.
- There may be more than one homomorphism.

A decision problem

Problem (de Bruijn graph mapping problem)

Given a finite edge-labeled graph $G = (V_G, E_G)$, *do there exist* $d \geq 1$ *and a homomorphism* $B_d(\Sigma) \to G$*?*

Temporal equations

We arrived at this problem because of a problem in temporal logic:

Problem (Unifiability in temporal logic of next)

Given a system of equations in the temporal logic of next X, *does it have a unifier in this logic?*

Unifiers defined

Let $x, y, \ldots$ be variables and $p_1, p_2, \ldots$ be propositional constants.

A formula is an expression built from these with $\vee$, $\neg$, $\bot$, and X.

Unifiers defined

Let $x, y, \ldots$ be variables and $p_1, p_2, \ldots$ be propositional constants.

A formula is an expression built from these with $\vee$, $\neg$, $\bot$, and X.

A unifier of a formula $\varphi(x, y, \ldots)$ is a substitution $x \mapsto \sigma_x$, $y \mapsto \sigma_y$, $\ldots$, where the σ's are variable-free formulas, and such that $\varphi(\sigma_x, \sigma_y, \ldots)$ is a valid formula.

Unifiers defined

Let $x, y, \dots$ be variables and $p_1, p_2, \dots$ be propositional constants.

A formula is an expression built from these with $\vee$, $\neg$, $\bot$, and X.

A unifier of a formula $\varphi(x, y, \dots)$ is a substitution $x \mapsto \sigma_x$, $y \mapsto \sigma_y$, $\dots$, where the σ's are variable-free formulas, and such that $\varphi(\sigma_x, \sigma_y, \dots)$ is a valid formula.

The depth of a substitution is the maximum nesting of X in the formulas σ_x, for x a variable.

Temporal equations

Problem (Unifiability in temporal logic of next)

Given a system of equations in the temporal logic of next X*, does it have a unifier in this logic?*

Temporal equations

Problem (Unifiability in temporal logic of next)

Given a system of equations in the temporal logic of next X*, does it have a unifier in this logic?*

For example, the system

$$\begin{cases} x = \neg \mathrm{X} p \wedge \mathrm{X}\mathrm{X}(x \vee y) \\ y = x \to p \end{cases}$$

Temporal equations

Problem (Unifiability in temporal logic of next)

Given a system of equations in the temporal logic of next X*, does it have a unifier in this logic?*

For example, the system

$$\begin{cases} x = \neg \mathrm{X} p \wedge \mathrm{X}\mathrm{X}(x \vee y) \\ y = x \rightarrow p \end{cases}$$

gives the formula

$$[x \leftrightarrow (\neg \mathrm{X} p \wedge \mathrm{X}\mathrm{X}(x \vee y))] \wedge [y \leftrightarrow (x \rightarrow p)]$$

which has a (unique) unifier:

Temporal equations

Problem (Unifiability in temporal logic of next)

Given a system of equations in the temporal logic of next X*, does it have a unifier in this logic?*

For example, the system

$$\begin{cases} x = \neg \mathrm{X} p \wedge \mathrm{X}\mathrm{X}(x \vee y) \\ y = x \to p \end{cases}$$

gives the formula

$$[x \leftrightarrow (\neg \mathrm{X} p \wedge \mathrm{X}\mathrm{X}(x \vee y))] \wedge [y \leftrightarrow (x \to p)]$$

which has a (unique) unifier: $x \mapsto \neg \mathrm{X} p$, $y \mapsto p \vee \mathrm{X} p$.

Temporal equations

Problem (Unifiability in temporal logic of next)

Given a system of equations in the temporal logic of next X, *does it have a unifier in this logic?*

For example, the system

$$\begin{cases} x = \neg \mathrm{X} p \wedge \mathrm{X}\mathrm{X}(x \vee y) \\ y = x \rightarrow p \end{cases}$$

gives the formula

$$[x \leftrightarrow (\neg \mathrm{X} p \wedge \mathrm{X}\mathrm{X}(x \vee y))] \wedge [y \leftrightarrow (x \rightarrow p)]$$

which has a (unique) unifier: $x \mapsto \neg \mathrm{X} p$, $y \mapsto p \vee \mathrm{X} p$.

Negative example: $x \leftrightarrow \neg \mathrm{X} x$ does not have a unifier.

Temporal equations

Problem (Unifiability in temporal logic of next)

Given a system of equations in the temporal logic of next X*, does it have a unifier in this logic?*

For example, the system

$$\begin{cases} x = \neg \mathrm{X} p \wedge \mathrm{X}\mathrm{X}(x \vee y) \\ y = x \rightarrow p \end{cases}$$

gives the formula

$$[x \leftrightarrow (\neg \mathrm{X} p \wedge \mathrm{X}\mathrm{X}(x \vee y))] \wedge [y \leftrightarrow (x \rightarrow p)]$$

which has a (unique) unifier: $x \mapsto \neg \mathrm{X} p$, $y \mapsto p \vee \mathrm{X} p$.

Negative example: $x \leftrightarrow \neg \mathrm{X} x$ does not have a unifier. (Why?)

Unifiers and homomorphisms

For any formula φ, with propositional constants in a finite set C, we compute a certain finite graph $G(\varphi)$ with a 2^C-labeling on the edges, and we prove:

Unifiers and homomorphisms

For any formula φ, with propositional constants in a finite set C, we compute a certain finite graph $G(\varphi)$ with a 2^C-labeling on the edges, and we prove:

Theorem

The set of depth $\leq d$ unifiers of φ

is in bijection with

the set of homomorphisms from $B_d(2^C)$ to $G(\varphi)$.

Unifiers and homomorphisms

For any formula φ, with propositional constants in a finite set C, we compute a certain finite graph $G(\varphi)$ with a 2^C-labeling on the edges, and we prove:

Theorem

The set of depth $\leq d$ unifiers of φ

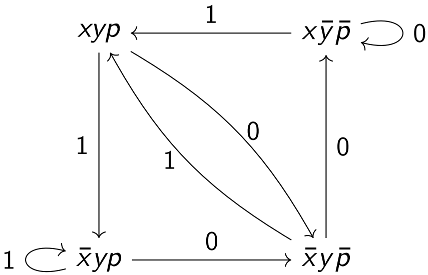
is in bijection with

the set of homomorphisms from $B_d(2^C)$ to $G(\varphi)$.

In particular, decidability of the de Bruijn graph mapping problem implies decidability of the unifiability problem.
(It is in fact equivalent.)

Example of the graph associated to a formula

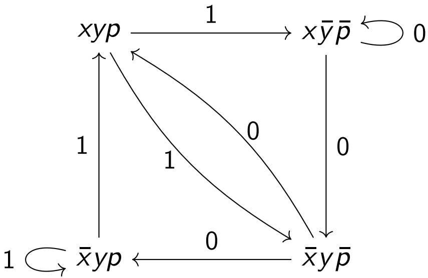
$$\varphi \ : \ [x \leftrightarrow \neg \mathrm{X} p] \wedge [y \leftrightarrow (x \rightarrow p)]$$



$G(\varphi)$

Example of the graph associated to a formula

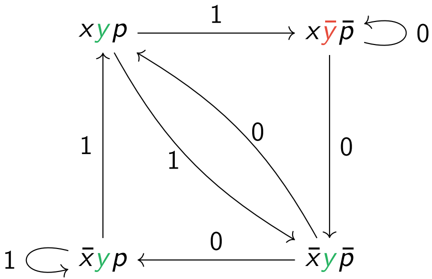
$$\varphi \ : \ [x \leftrightarrow \neg \mathrm{X} p] \wedge [y \leftrightarrow (x \rightarrow p)]$$



$G(\varphi)$ reversed

Example of the graph associated to a formula

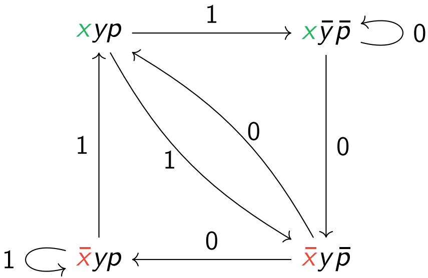
$$\varphi \ : \ [x \leftrightarrow \neg \mathrm{X} p] \wedge [y \leftrightarrow (x \rightarrow p)]$$



$G(\varphi)$ reversed

Example of the graph associated to a formula

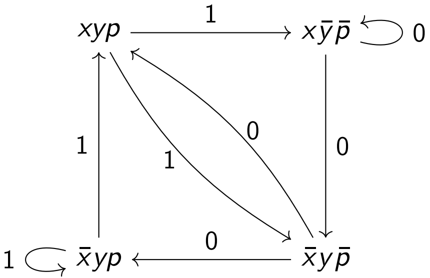
$$\varphi \;:\; [x \leftrightarrow \neg \mathrm{X} p] \wedge [y \leftrightarrow (x \rightarrow p)]$$



$G(\varphi)$ reversed

Example of the graph associated to a formula

$$\varphi \; : \; [x \leftrightarrow \neg \mathrm{X} p] \wedge [y \leftrightarrow (x \rightarrow p)]$$



$G(\varphi)$ reversed

The graph $G(\varphi)$ is an image of B_2 (in a unique way).

Deterministic de Bruijn graph images

Let Σ a finite alphabet and G a Σ-labeled graph.

Definition

A graph G is an image if there exist $d \geq 1$ and a surjective homomorphism $B_d(\Sigma) \twoheadrightarrow G$.

Deterministic de Bruijn graph images

Let Σ a finite alphabet and G a Σ-labeled graph.

Definition

A graph G is an image if there exist $d \geq 1$ and a surjective homomorphism $B_d(\Sigma) \twoheadrightarrow G$.

Observation: It suffices to characterize images.

Deterministic de Bruijn graph images

Let Σ a finite alphabet and G a Σ-labeled graph.

Definition

A graph G is an image if there exist $d \geq 1$ and a surjective homomorphism $B_d(\Sigma) \twoheadrightarrow G$.

Observation: It suffices to characterize images.

Definition

- A node u of G is a w-sink, for $w \in \Sigma^*$, if, for every node x of G, we have a path $x \xrightarrow{w} u$.

Deterministic de Bruijn graph images

Let Σ a finite alphabet and G a Σ-labeled graph.

Definition

A graph G is an image if there exist $d \geq 1$ and a surjective homomorphism $B_d(\Sigma) \twoheadrightarrow G$.

Observation: It suffices to characterize images.

Definition

- A node u of G is a w-sink, for $w \in \Sigma^*$, if, for every node x of G, we have a path $x \xrightarrow{w} u$.
- G is d-synchronizing, for $d \geq 1$, if G has a w-sink for every w of length d.

Deciding deterministic images

Proposition (Bleak, Cameron, Maissel, Navas, Olukoya 2016)

Let G be a deterministic graph. Then:

G is an image of B_d

if, and only if,

G is strongly connected and G is d-synchronizing.

Deciding deterministic images

Proposition (Bleak, Cameron, Maissel, Navas, Olukoya 2016)

Let G be a deterministic graph. Then:

G is an image of B_d

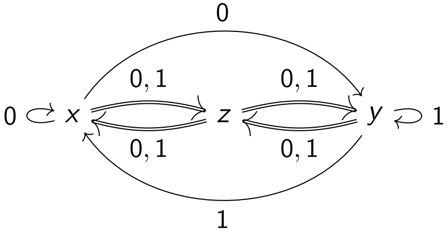
if, and only if,

G is strongly connected and G is d-synchronizing.

This characterization yields a linear time decision procedure for deterministic input graphs:

1. Set $G_0 := G$.
2. Set $G_{n+1} := G_n/{\equiv_n}$, where $u \equiv_n u'$ iff $u \cdot a = u' \cdot a$ for all a.
3. When $G_{n+1} = G_n$, output 'yes' iff G_n has a single node.

A non-deterministic example: 'The hamburger'



This graph is strongly connected and 2-synchronizing.

However, it is not an image of B_2.

In fact, it is not an image of B_d for any d.

Power-connectedness: intuition

The de Bruijn graphs possess a *very* strong connectedness property, which transfers to images: B_d is power-connected.

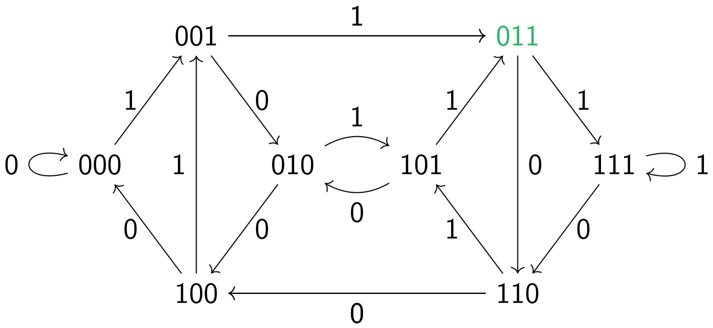
Power-connectedness: intuition

The de Bruijn graphs possess a *very* strong connectedness property, which transfers to images: B_d is power-connected.

Intuitively: 'You can find out where you are within d steps'.

Power-connectedness: intuition

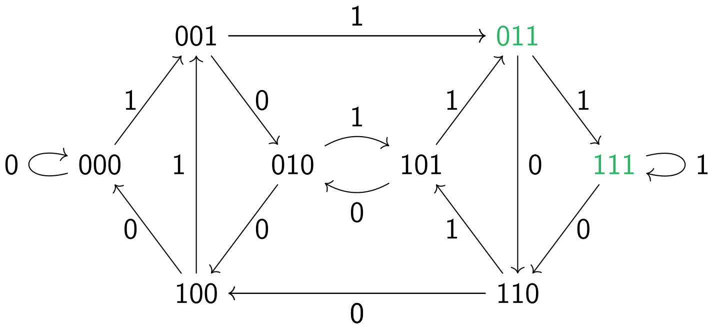
The de Bruijn graphs possess a *very* strong connectedness property, which transfers to images: B_d is power-connected. Intuitively: 'You can find out where you are within d steps'.



Letters read: 011

Power-connectedness: intuition

The de Bruijn graphs possess a *very* strong connectedness property, which transfers to images: B_d is power-connected. Intuitively: 'You can find out where you are within d steps'.

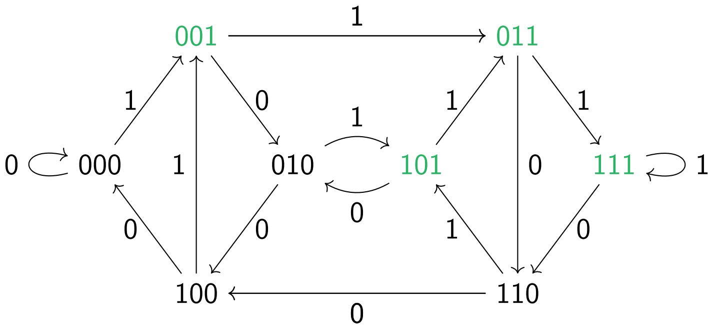


Letters read: 11

Power-connectedness: intuition

The de Bruijn graphs possess a *very* strong connectedness property, which transfers to images: B_d is power-connected.

Intuitively: 'You can find out where you are within d steps'.

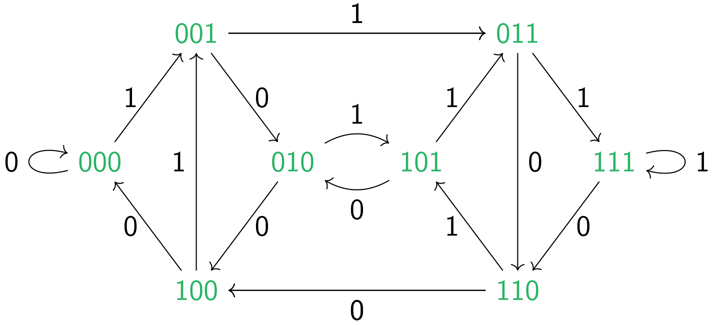


Letters read: 1

Power-connectedness: intuition

The de Bruijn graphs possess a *very* strong connectedness property, which transfers to images: B_d is power-connected.

Intuitively: 'You can find out where you are within d steps'.



Letters read:

Power-connectedness defined

Let $H = (V_H, E_H)$ be a Σ-graph.

- A node $u \in V_H$ is a predecessor of a set $S \subseteq V_H$ if, for every $a \in \Sigma$, there exists $s \in S$ such that $u \xrightarrow{a} s$.

Power-connectedness defined

Let $H = (V_H, E_H)$ be a Σ-graph.

- A node $u \in V_H$ is a predecessor of a set $S \subseteq V_H$ if, for every $a \in \Sigma$, there exists $s \in S$ such that $u \xrightarrow{a} s$.
- A set $C \subseteq V_H$ is closed if it contains all its predecessors.

Power-connectedness defined

Let $H = (V_H, E_H)$ be a Σ-graph.

- A node $u \in V_H$ is a predecessor of a set $S \subseteq V_H$ if, for every $a \in \Sigma$, there exists $s \in S$ such that $u \xrightarrow{a} s$.
- A set $C \subseteq V_H$ is closed if it contains all its predecessors.

Let $G = (V_G, E_G)$ be a Σ-graph.

- The power graph of G is the graph with nodes $\mathcal{P}(V_G)$ and

$$S \xrightarrow{a} T \iff \forall x \in S, \exists y \in T \text{ such that } x \xrightarrow{a}_G y \ .$$

Power-connectedness defined

Let $H = (V_H, E_H)$ be a Σ-graph.

- A node $u \in V_H$ is a predecessor of a set $S \subseteq V_H$ if, for every $a \in \Sigma$, there exists $s \in S$ such that $u \xrightarrow{a} s$.
- A set $C \subseteq V_H$ is closed if it contains all its predecessors.

Let $G = (V_G, E_G)$ be a Σ-graph.

- The power graph of G is the graph with nodes $\mathcal{P}(V_G)$ and

$$S \xrightarrow{a} T \iff \forall x \in S, \exists y \in T \text{ such that } x \xrightarrow{a}_G y \ .$$

- G is power-connected if, in its power graph, the node V_G is in the closure of the set of nodes $\{\{u\} \ : \ u \in V_G\}$.

Power-connectedness: necessity

Proposition

Any image of B_d is power-connected.

Power-connectedness: necessity

Proposition

Any image of B_d is power-connected.

Proposition

There is an exponential-time algorithm that determines whether or not a graph is power-connectd.

Power-connectedness: necessity

Proposition

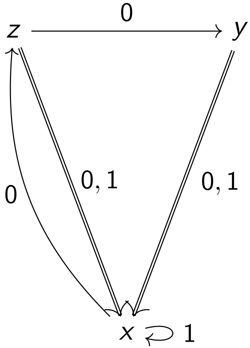
Any image of B_d is power-connected.

Proposition

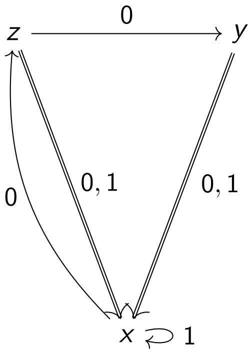
There is an exponential-time algorithm that determines whether or not a graph is power-connectd.

The hamburger fails to be power-connected, and thus cannot be an image.

Another non-deterministic example: 'The cone of fries'



Another non-deterministic example: ‘The cone of fries’



This graph is power-connected, but cannot have a homomorphism from any B_d: There is no self-loop labeled 0.

Cycle-connectedness: intuition

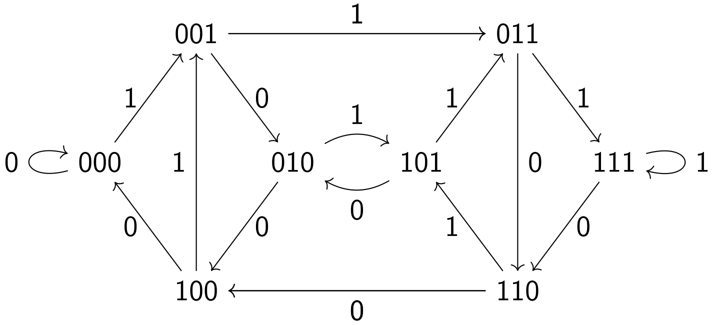
A *different* very strong connectedness property of de Bruijn graphs:

We have arbitrarily long cycles, reachable from anywhere.

Cycle-connectedness: intuition

A *different* very strong connectedness property of de Bruijn graphs:

We have arbitrarily long cycles, reachable from anywhere.

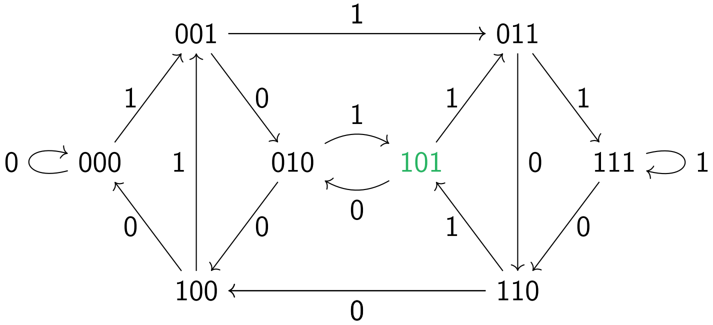


A cycle labeled 1100101

Cycle-connectedness: intuition

A *different* very strong connectedness property of de Bruijn graphs:

We have arbitrarily long cycles, reachable from anywhere.

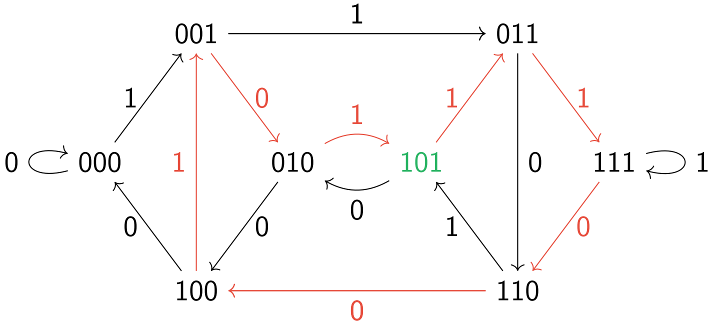


A cycle labeled 1100101

Cycle-connectedness: intuition

A *different* very strong connectedness property of de Bruijn graphs:

We have arbitrarily long cycles, reachable from anywhere.



A cycle labeled 1100101

Cycle-connectedness defined

Let $G = (V_G, E_G)$ be a Σ-graph.

- A *w*-cycle, for $w \in \Sigma^+$, is a path $u \xrightarrow{w} u$, for some $u \in V_G$.

Cycle-connectedness defined

Let $G = (V_G, E_G)$ be a Σ-graph.

- A *w*-cycle, for $w \in \Sigma^+$, is a path $u \xrightarrow{w} u$, for some $u \in V_G$.
- A reachable *w*-cycle is a w-cycle $u \xrightarrow{w} u$ such that, for some $k \geq 1$, there exists, for every node v, a path $v \xrightarrow{w^k} u$.

Cycle-connectedness defined

Let $G = (V_G, E_G)$ be a Σ-graph.

- A *w*-cycle, for $w \in \Sigma^+$, is a path $u \xrightarrow{w} u$, for some $u \in V_G$.
- A reachable *w*-cycle is a *w*-cycle $u \xrightarrow{w} u$ such that, for some $k \geq 1$, there exists, for every node v, a path $v \xrightarrow{w^k} u$.
- G is cycle-connected if, for every $w \in \Sigma^+$, there exists a reachable *w*-cycle.

Cycle-connectedness defined

Let $G = (V_G, E_G)$ be a Σ-graph.

- A w-cycle, for $w \in \Sigma^+$, is a path $u \xrightarrow{w} u$, for some $u \in V_G$.
- A reachable w-cycle is a w-cycle $u \xrightarrow{w} u$ such that, for some $k \geq 1$, there exists, for every node v, a path $v \xrightarrow{w^k} u$.
- G is cycle-connected if, for every $w \in \Sigma^+$, there exists a reachable w-cycle.

Proposition

Any image of B_d is cycle-connected.

Proposition

There is an exponential-time algorithm that determines whether or not a graph is cycle-connected.

Characterization theorem

Theorem

A Σ-graph G is an image of a de Bruijn graph if, and only if, G has a subgraph that is cycle-connected and power-connected.

Characterization theorem

Theorem

A Σ-graph G is an image of a de Bruijn graph if, and only if, G has a subgraph that is cycle-connected and power-connected.

The two conditions are independent from each other:

- the hamburger is cycle-connected and not power-connected,
- the cone of fries is power-connected and not cycle-connected.

The two conditions can be checked in exponential time.

Proof overview

- Given a cycle- and power-connected graph G, we need to define a homomorphism $h\colon B_d \to G$, for some d.

Proof overview

- Given a cycle- and power-connected graph G, we need to define a homomorphism $h\colon B_d \to G$, for some d.
- Any node of the form a^d *must* go to a node with a-loop.

Proof overview

- Given a cycle- and power-connected graph G, we need to define a homomorphism $h\colon B_d \to G$, for some d.
- Any node of the form a^d *must* go to a node with a-loop.
- If $w \in \Sigma^d$ has a long part with small period, then w must go 'close' to a cycle.

Proof overview

- Given a cycle- and power-connected graph G, we need to define a homomorphism $h\colon B_d \to G$, for some d.
- Any node of the form a^d *must* go to a node with a-loop.
- If $w \in \Sigma^d$ has a long part with small period, then w must go 'close' to a cycle.
- Problem: How to *synchronize* different cycles?

Proof overview

- Given a cycle- and power-connected graph G, we need to define a homomorphism $h\colon B_d \to G$, for some d.
- Any node of the form a^d *must* go to a node with a-loop.
- If $w \in \Sigma^d$ has a long part with small period, then w must go 'close' to a cycle.
- Problem: How to *synchronize* different cycles?
- For example,

$$0\,01\,01\,01\,01\,01 \text{ and } 10\,10\,10\,10\,10\,1$$

  should go to two nodes that have a common 0-successor.

Proof overview

- Given a cycle- and power-connected graph G, we need to define a homomorphism $h\colon B_d \to G$, for some d.
- Any node of the form a^d *must* go to a node with a-loop.
- If $w \in \Sigma^d$ has a long part with small period, then w must go 'close' to a cycle.
- Problem: How to *synchronize* different cycles?
- For example,

  $$0\,01\,01\,01\,01\,01 \text{ and } 10\,10\,10\,10\,10\,1$$

  should go to two nodes that have a common 0-successor.
- Idea: Use *minimizers*: If w is not highly periodic, single out a position for starting a new synchronization process. (Schleimer et al. 2003, Roberts et al. 2004)

Proof ingredients

- Dichotomy Lemma. For any sufficiently long word w:
  1. either w contains a long periodic suffix,
  2. or w has a minimizer for long enough to synchronize.

Proof ingredients

- Dichotomy Lemma. For any sufficiently long word w:
  1. either w contains a long periodic suffix,
  2. or w has a minimizer for long enough to synchronize.

- The proof uses the critical factorization theorem from combinatorics on words (Lothaire).

Proof ingredients

- Dichotomy Lemma. For any sufficiently long word w:
  1. either w contains a long periodic suffix,
  2. or w has a minimizer for long enough to synchronize.

- The proof uses the critical factorization theorem from combinatorics on words (Lothaire).

- From there, a (so far) somewhat *ad hoc* construction of the homomorphism.

Final remarks

- We establish that the mapping problem is in exp-time, and the unifiability problem in 2-exp-time. Hardness?

Final remarks

- We establish that the mapping problem is in exp-time, and the unifiability problem in 2-exp-time. Hardness?
- A connection to symbolic dynamics: Our main result implies that *it is decidable if a two-sided edge shift has a sliding block code section* (One-sided case: Salo and Törmä 2015.)

Final remarks

- We establish that the mapping problem is in exp-time, and the unifiability problem in 2-exp-time. Hardness?
- A connection to symbolic dynamics: Our main result implies that *it is decidable if a two-sided edge shift has a sliding block code section* (One-sided case: Salo and Törmä 2015.)
- The general method is not limited, in principle, to de Bruijn graphs and temporal logic of next. One might try to use it for unifiability for other logics, e.g., modal logic **K** (hypergraphs!)

Final remarks

- We establish that the mapping problem is in exp-time, and the unifiability problem in 2-exp-time. Hardness?
- A connection to symbolic dynamics: Our main result implies that *it is decidable if a two-sided edge shift has a sliding block code section* (One-sided case: Salo and Törmä 2015.)
- The general method is not limited, in principle, to de Bruijn graphs and temporal logic of next. One might try to use it for unifiability for other logics, e.g., modal logic **K** (hypergraphs!)
- Thanks for your attention! (Who else is ready for lunch...?)