Regularity Preserving but not Reflecting Encodings

Jörg Endrullis Clemens Grabmayer Dimitri Hendriks

Department of Computer Science
VU University Amsterdam

LICS 2015

Kyoto, July 9, 2015

The problem

An *encoding* is an injective word function $f : A \to \Gamma^*$.

The problem

An *encoding* is an injective word function $f : A \to \Gamma^*$.

Does there exist a bijective encoding $f : \Sigma^* \to \Gamma^*$ *(*Σ, Γ *finite alphabets) such that*

The problem

An *encoding* is an injective word function $f : A \to \Gamma^*$.

Does there exist a bijective encoding $f : \Sigma^* \to \Gamma^*$ (Σ, Γ finite alphabets) such that

- *the image function $f[\_]$ of f preserves language regularity:*

$$\forall L \subseteq \Sigma^* (\, L \textit{ is regular} \implies f[L] \textit{ is regular}\,)\ ,$$

The problem

An *encoding* is an injective word function $f : A \to \Gamma^*$.

Does there exist a bijective encoding $f : \Sigma^* \to \Gamma^*$ (Σ, Γ finite alphabets) such that

- *the image function $f[\_]$ of f preserves language regularity:*

$$\forall L \subseteq \Sigma^* (\, L \text{ is regular} \implies f[L] \text{ is regular} \,)\,,$$

- *but the image function $f^{-1}[\cdot]$ for the inverse function f^{-1} does not:*

$$\exists L' \subseteq \Gamma^* (\, L' \text{ is regular} \;\wedge\; f^{-1}[L] \text{ is not regular} \,) \qquad ?$$

Outline

- the problem
  - more motivation: number encodings and c-automaticity
- the solution
  - main theorem
  - encoding, and extension lemmas
  - proof sketch/idea
- consequences for comparing models of computation
- summary

c-automatic functions

a number encoding $c : \mathbb{N} \to \Gamma^*$

c-automatic functions

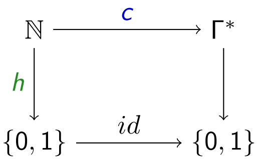
a number encoding $c : \mathbb{N} \to \Gamma^*$

$$
\begin{array}{ccc}
\mathbb{N} & \xrightarrow{\quad c \quad} & \Gamma^* \\
\downarrow & & \downarrow \\
\{0,1\} & \xrightarrow{\quad id \quad} & \{0,1\}
\end{array}
$$

c-automatic functions

For a number encoding $c : \mathbb{N} \to \Gamma^*$, a function $h : \mathbb{N} \to \{0,1\}$ is *c-automatic*

c-automatic function



c-automatic functions

For a number encoding $c : \mathbb{N} \to \Gamma^*$, a function $h : \mathbb{N} \to \{0,1\}$ is *c-automatic* if there is a DFA-computable $h^c : \Gamma^* \to \{0,1\}$ such that:

c-automatic function

$$\begin{array}{ccc} \mathbb{N} & \xrightarrow{\;c\;} & \Gamma^* \\ {\scriptstyle h}\downarrow & & \downarrow{\scriptstyle h^c} \\ \{0,1\} & \xrightarrow{\;id\;} & \{0,1\} \end{array}$$

computable by a DFA

c-automatic functions

For a number encoding $c : \mathbb{N} \to \Gamma^*$, a function $h : \mathbb{N} \to \{0,1\}$ is *c-automatic* if there is a DFA-computable $h^c : \Gamma^* \to \{0,1\}$ such that:

c-automatic function
$\triangleq$ infinite sequence

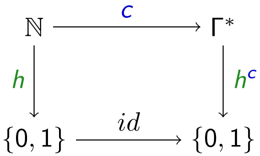
$$\begin{array}{ccc} \mathbb{N} & \xrightarrow{\;c\;} & \Gamma^* \\ {\scriptstyle h}\downarrow & & \downarrow{\scriptstyle h^c} \\ \{0,1\} & \xrightarrow{\;id\;} & \{0,1\} \end{array}$$

computable by a DFA

c-automatic functions

For a number encoding $c : \mathbb{N} \to \Gamma^*$, a function $h : \mathbb{N} \to \{0,1\}$ is *c-automatic* if there is a DFA-computable $h^c : \Gamma^* \to \{0,1\}$ such that:

c-automatic function
$\triangleq$ infinite sequence
$\triangleq$ subset of $\mathbb{N}$



computable by a DFA

c-automatic functions

For a number encoding $c : \mathbb{N} \to \Gamma^*$, a function $h : \mathbb{N} \to \{0,1\}$ is *c-automatic* if there is a DFA-computable $h^c : \Gamma^* \to \{0,1\}$ such that:

c-automatic function
$\triangleq$ infinite sequence
$\triangleq$ subset of $\mathbb{N}$

$$\begin{array}{ccc} \mathbb{N} & \xrightarrow{\;c\;} & \Gamma^* \\ {\scriptstyle h}\downarrow & & \downarrow{\scriptstyle h^c} \\ \{0,1\} & \xrightarrow{\;id\;} & \{0,1\} \end{array}$$

computable by a DFA
$\triangleq$ regular language

c-automatic functions

For a number encoding $c : \mathbb{N} \to \Gamma^*$, a function $h : \mathbb{N} \to \{0,1\}$ is *c-automatic* if there is a DFA-computable $h^c : \Gamma^* \to \{0,1\}$ such that:

c-automatic function
$\triangleq$ infinite sequence
$\triangleq$ subset of $\mathbb{N}$

$$\begin{array}{ccc} \mathbb{N} & \xrightarrow{\;c\;} & \Gamma^* \\ {\scriptstyle h}\downarrow & & \downarrow{\scriptstyle h^c} \\ \{0,1\} & \xrightarrow{\;id\;} & \{0,1\} \end{array}$$

computable by a DFA
$\triangleq$ regular language

Gives rise to a *subsumption pre-order* $\leq$:

$c \leq d$ if all c-automatic functions are d-automatic.

c-automatic functions

For a number encoding $c : \mathbb{N} \to \Gamma^*$, a function $h : \mathbb{N} \to \{0, 1\}$ is *c-automatic* if there is a DFA-computable $h^c : \Gamma^* \to \{0, 1\}$ such that:

c-automatic function
$\triangleq$ infinite sequence
$\triangleq$ subset of $\mathbb{N}$

$$\begin{array}{ccc} \mathbb{N} & \xrightarrow{\;c\;} & \Gamma^* \\ {\scriptstyle h}\downarrow & & \downarrow{\scriptstyle h^c} \\ \{0,1\} & \xrightarrow{\;id\;} & \{0,1\} \end{array}$$

computable by a DFA
$\triangleq$ regular language

Gives rise to a *subsumption pre-order* $\leq$:

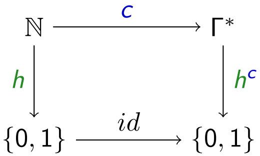
$c \leq d$ if all c-automatic functions are d-automatic.

Q: How to characterize number encodings c, d such that $c \leq d$?

c-automatic functions

For a number encoding $c : \mathbb{N} \to \Gamma^*$, a function $h : \mathbb{N} \to \{0,1\}$ is *c-automatic* if there is a DFA-computable $h^c : \Gamma^* \to \{0,1\}$ such that:

c-automatic function
$\triangleq$ infinite sequence
$\triangleq$ subset of $\mathbb{N}$



computable by a DFA
$\triangleq$ regular language

Gives rise to a *subsumption pre-order* $\leq$:

$c \leq d$ if all c-automatic functions are d-automatic.

Q: How to characterize number encodings c, d such that $c \leq d$?

A: For *bijective* number encodings c and d:

$$c \leq d \iff (d \circ c^{-1})[\_] \text{ preserves regularity}$$

c-automatic functions

For a number encoding $c : \mathbb{N} \to \Gamma^*$, a function $h : \mathbb{N} \to \{0,1\}$ is *c-automatic* if there is a DFA-computable $h^c : \Gamma^* \to \{0,1\}$ such that:

c-automatic function
$\triangleq$ infinite sequence
$\triangleq$ subset of $\mathbb{N}$

$$\begin{array}{ccc} \mathbb{N} & \xrightarrow{\ c\ } & \Gamma^* \\ {\scriptstyle h}\downarrow & & \downarrow{\scriptstyle h^c} \\ \{0,1\} & \xrightarrow{\ id\ } & \{0,1\} \end{array}$$

computable by a DFA
$\triangleq$ regular language

Gives rise to a *subsumption pre-order* $\leq$:

$c \leq d$ if all c-automatic functions are d-automatic.

Q: How to characterize number encodings c, d such that $c \leq d$?

A: For *bijective* number encodings c and d:

$$c \leq d \iff (d \circ c^{-1})[\_] \text{ preserves regularity}$$

Q: Does $\leq$ induce a proper hierarchy? $\exists c, d.\ c \leq d \wedge d \not\leq c$?

c-automatic functions

For a number encoding $c : \mathbb{N} \to \Gamma^*$, a function $h : \mathbb{N} \to \{0,1\}$ is *c-automatic* if there is a DFA-computable $h^c : \Gamma^* \to \{0,1\}$ such that:

c-automatic function
$\triangleq$ infinite sequence
$\triangleq$ subset of $\mathbb{N}$

$$\begin{array}{ccc} \mathbb{N} & \xrightarrow{\;c\;} & \Gamma^* \\ h \downarrow & & \downarrow h^c \\ \{0,1\} & \xrightarrow{\;id\;} & \{0,1\} \end{array}$$

computable by a DFA
$\triangleq$ regular language

Gives rise to a *subsumption pre-order* $\leq$:

$c \leq d$ if all c-automatic functions are d-automatic.

Q: How to characterize number encodings c, d such that $c \leq d$?

A: For *bijective* number encodings c and d :

$$c \leq d \iff (d \circ c^{-1})[\_] \text{ preserves regularity}$$

Q: Does $\leq$ induce a proper hierarchy? $\exists c, d.\ c \leq d \wedge d \not\leq c$?

A: For *bijective* encodings: Yes, if answer to the initial problem is yes!

Solution

Main theorem

Main Theorem

Let Σ, Γ *be finite alphabets, with* $|\Gamma| \geq 2$.

For every countable class $\mathcal{C}$ *of languages over* Σ,
there exists a bijective encoding $g : \Sigma^* \to \Gamma^*$ *such that:*

$$\forall L \in \mathcal{C} \left(g[L] \text{ is regular} \right).$$

Main theorem

Main Theorem

Let Σ, Γ be finite alphabets, with $|\Gamma| \geq 2$.

For every countable class $\mathcal{C}$ of languages over Σ, there exists a bijective encoding $g : \Sigma^* \to \Gamma^*$ such that:

$$\forall L \in \mathcal{C} \left(g[L] \text{ is regular} \right) .$$

Thus the answer to the initial problem is 'Yes!', because for $\mathcal{C} = Rec$:

Main theorem

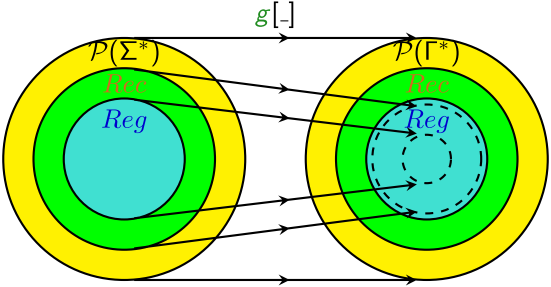
Main Theorem

Let Σ, Γ *be finite alphabets, with* $|\Gamma| \geq 2$.

For every countable class $\mathcal{C}$ *of languages over* Σ,
there exists a bijective encoding $g : \Sigma^* \to \Gamma^*$ *such that:*

$$\forall L \in \mathcal{C} \; \big(\, g[L] \text{ is regular} \, \big) \, .$$

Thus the answer to the initial problem is 'Yes!', because for $\mathcal{C} = Rec$:



Encoding and extension lemmas

Encoding Lemma (weakening main theorem to injective encodings)

Let Σ, Γ alphabets with $|\Gamma| \geq 2$.
Then for every countable class $\mathcal{C}$ of languages over Σ,
there exists an encoding $f : \Sigma^* \to \Gamma^*$ such that:

$$\forall L \in \mathcal{C} \, \big(\, f[L] \text{ is relatively regular in } f[\Sigma^*] \, \big)$$

Encoding and extension lemmas

Encoding Lemma (weakening main theorem to injective encodings)

Let Σ, Γ alphabets with $|\Gamma| \geq 2$.
Then for every countable class $\mathcal{C}$ of languages over Σ,
there exists an encoding $f : \Sigma^* \to \Gamma^*$ such that:

$$\forall L \in \mathcal{C} \; \big(\, f[L] \text{ is relatively regular in } f[\Sigma^*] \, \big)$$

We say that L *is relatively regular in* M
if $L = M \cap R$ for some regular language R.
(if a finite automaton can decide $w \in L$ for all $w \in M$).

Encoding and extension lemmas

Encoding Lemma (weakening main theorem to injective encodings)

Let Σ, Γ alphabets with $|\Gamma| \geq 2$.
Then for every countable class $\mathcal{C}$ of languages over Σ,
there exists an encoding $f : \Sigma^* \to \Gamma^*$ such that:

$$\forall L \in \mathcal{C} \ (\ f[L] \text{ is relatively regular in } f[\Sigma^*]\ )$$

We say that L *is relatively regular in* M
if $L = M \cap R$ for some regular language R.
(if a finite automaton can decide $w \in L$ for all $w \in M$).

Extension Lemma (from injective to bijective encodings)

For every injection $f : \Sigma^* \to \Gamma^*$ there is a bijection $g : \Sigma^* \to \Gamma^*$ s.th.:

$$\forall L \subseteq \Sigma^* \ (\ f[L] \text{ is relatively regular in } f[\Sigma^*] \implies g[L] \text{ is regular}\ )$$

Encoding lemma: proof

Encoding Lemma (injective encodings)

Let Σ, Γ be an alphabet with $|\Gamma| \geq 2$.

Then for every countable class $\mathcal{C}$ of languages over Σ, there exists an encoding $f : \Sigma^* \to \Gamma^*$ such that:

$$\forall L \in \mathcal{C} \left(f[L] \text{ is relatively regular in } f[\Sigma^*] \right)$$

Encoding lemma: proof

Encoding Lemma (injective encodings)

Let Σ, Γ be an alphabet with $|\Gamma| \geq 2$.

Then for every countable class $\mathcal{C}$ of languages over Σ, there exists an encoding $f : \Sigma^* \to \Gamma^*$ such that:

$$\forall L \in \mathcal{C} \left(f[L] \text{ is relatively regular in } f[\Sigma^*] \right)$$

Proof.

Let $L_1, L_2, L_3, L_4, \ldots$ be an enumeration of $\mathcal{C}$, and $w_1, w_2, w_3, \ldots$ of Σ^*.

Encoding lemma: proof

Encoding Lemma (injective encodings)

Let Σ, Γ be an alphabet with $|\Gamma| \geq 2$.

Then for every countable class $\mathcal{C}$ of languages over Σ,
there exists an encoding $f : \Sigma^* \to \Gamma^*$ such that:

$$\forall L \in \mathcal{C} \left(f[L] \text{ is relatively regular in } f[\Sigma^*] \right)$$

Proof.

Let $L_1, L_2, L_3, L_4, \ldots$ be an enumeration of $\mathcal{C}$, and $w_1, w_2, w_3, \ldots$ of Σ^*.
Suppose $\{0, 1\} \subseteq \Gamma$, and define $L(w) = 1$ if $w \in L$, and else $L(w) = 0$.

Encoding lemma: proof

Encoding Lemma (injective encodings)

Let Σ, Γ be an alphabet with $|\Gamma| \geq 2$.

Then for every countable class $\mathcal{C}$ of languages over Σ,
there exists an encoding $f : \Sigma^* \to \Gamma^*$ such that:

$$\forall L \in \mathcal{C} \left(f[L] \text{ is relatively regular in } f[\Sigma^*] \right)$$

Proof.

Let $L_1, L_2, L_3, L_4, \ldots$ be an enumeration of $\mathcal{C}$, and $w_1, w_2, w_3, \ldots$ of Σ^*.
Suppose $\{0, 1\} \subseteq \Gamma$, and define $L(w) = 1$ if $w \in L$, and else $L(w) = 0$.
Define $f : \Sigma^* \to \Gamma^*$ by:

$$\begin{aligned}
f(w_1) &= L_1(w_1) \\
f(w_2) &= L_1(w_2)\, L_2(w_2) \\
f(w_3) &= L_1(w_3)\, L_2(w_3)\, L_3(w_3) \\
f(w_4) &= L_1(w_4)\, L_2(w_4)\, L_3(w_4)\, L_4(w_4) \\
&\vdots
\end{aligned}$$

Encoding lemma: proof

Encoding Lemma (injective encodings)

Let Σ, Γ be an alphabet with $|\Gamma| \geq 2$.

Then for every countable class $\mathcal{C}$ of languages over Σ,
there exists an encoding $f : \Sigma^* \to \Gamma^*$ such that:

$$\forall L \in \mathcal{C} \; \big(f[L] \text{ is relatively regular in } f[\Sigma^*] \big)$$

Proof.

Let $L_1, L_2, L_3, L_4, \ldots$ be an enumeration of $\mathcal{C}$, and $w_1, w_2, w_3, \ldots$ of Σ^*.
Suppose $\{0, 1\} \subseteq \Gamma$, and define $L(w) = 1$ if $w \in L$, and else $L(w) = 0$.
Define $f : \Sigma^* \to \Gamma^*$ by:

$$\begin{aligned} f(w_1) &= L_1(w_1) \\ f(w_2) &= L_1(w_2)\, L_2(w_2) \\ f(w_3) &= L_1(w_3)\, L_2(w_3)\, L_3(w_3) \\ f(w_4) &= L_1(w_4)\, L_2(w_4)\, L_3(w_4)\, L_4(w_4) \\ &\vdots \end{aligned}$$

For almost all $u \in f[\Sigma^*]$: $u \in f[L_n] \iff u \in \Gamma^{n-1} 1 \Gamma^*$.

Extension lemma: proof idea

Extension Lemma (from injective to bijective encodings)

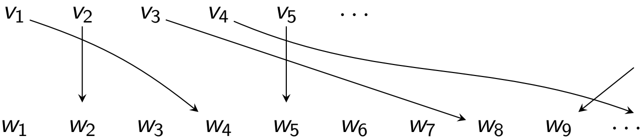
Let Σ^* be a countably infinite set.
For every injection $f : \Sigma^* \to \Gamma^*$ there is a bijection $g : \Sigma^* \to \Gamma^*$ s.th.:

$$\forall L \subseteq \Sigma^* \left(f[L] \text{ is relatively regular in } f[\Sigma^*] \implies g[L] \text{ is regular} \right)$$

Extension lemma: proof idea

- Let $v_1, v_2, v_3, \ldots$ be an enumeration of all words Σ^*.
- Let $w_1, w_2, w_3, \ldots$ be an enumeration of all words Γ^*.
- Let $A_1, A_2, A_3, \ldots$ an enumeration of all finite automata over Γ.

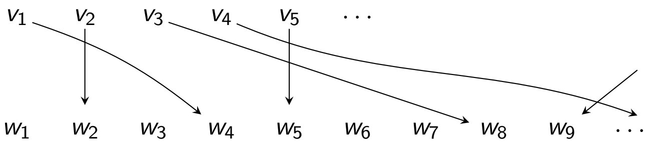
We start with an injective function f:



Extension lemma: proof idea

- Let $v_1, v_2, v_3, \ldots$ be an enumeration of all words Σ^*.
- Let $w_1, w_2, w_3, \ldots$ be an enumeration of all words Γ^*.
- Let $A_1, A_2, A_3, \ldots$ an enumeration of all finite automata over Γ.

We start with an injective function f:

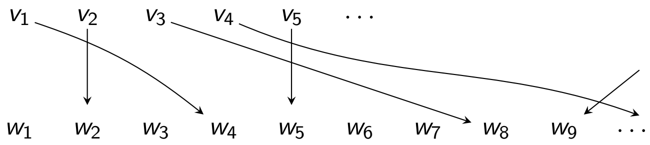


Idea: Change arrows such that every element is in the image, but so that:

Extension lemma: proof idea

- Let $v_1, v_2, v_3, \ldots$ be an enumeration of all words Σ^*.
- Let $w_1, w_2, w_3, \ldots$ be an enumeration of all words Γ^*.
- Let $A_1, A_2, A_3, \ldots$ an enumeration of all finite automata over Γ.

We start with an injective function f:



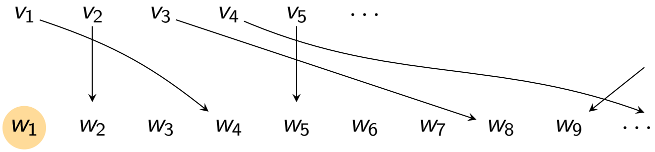
Idea: Change arrows such that every element is in the image, but so that:

- the language whose image is accepted by A_n is changed only at finitely many words (preserving relative regularity in the limit)

Extension lemma: proof idea

- Let $v_1, v_2, v_3, \ldots$ be an enumeration of all words Σ^*.
- Let $w_1, w_2, w_3, \ldots$ be an enumeration of all words Γ^*.
- Let $A_1, A_2, A_3, \ldots$ an enumeration of all finite automata over Γ.

We start with an injective function f:



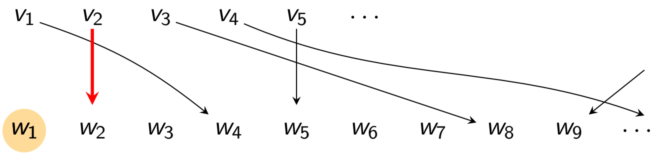
Idea: Change arrows such that every element is in the image, but so that:

- w_1 needs to be part of the image

Extension lemma: proof idea

- Let $v_1, v_2, v_3, \ldots$ be an enumeration of all words Σ^*.
- Let $w_1, w_2, w_3, \ldots$ be an enumeration of all words Γ^*.
- Let $A_1, A_2, A_3, \ldots$ an enumeration of all finite automata over Γ.

We start with an injective function f:



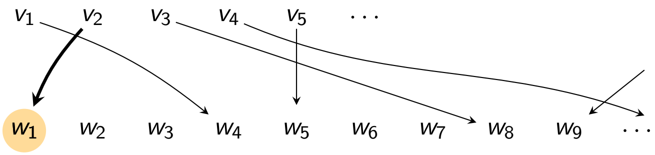
Idea: Change arrows such that every element is in the image, but so that:

- w_1 needs to be part of the image
- we pick the arrow $v_2 \to w_2$

Extension lemma: proof idea

- Let $v_1, v_2, v_3, \ldots$ be an enumeration of all words Σ^*.
- Let $w_1, w_2, w_3, \ldots$ be an enumeration of all words Γ^*.
- Let $A_1, A_2, A_3, \ldots$ an enumeration of all finite automata over Γ.

We start with an injective function f:



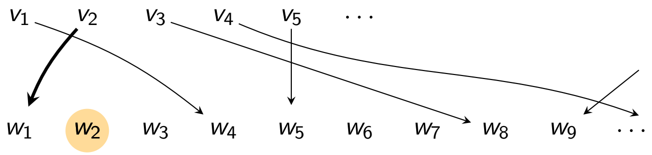
Idea: Change arrows such that every element is in the image, but so that:

- w_1 needs to be part of the image
- we pick the arrow $v_2 \to w_2$
- we redirect this arrow to target w_1

Extension lemma: proof idea

- Let $v_1, v_2, v_3, \ldots$ be an enumeration of all words Σ^*.
- Let $w_1, w_2, w_3, \ldots$ be an enumeration of all words Γ^*.
- Let $A_1, A_2, A_3, \ldots$ an enumeration of all finite automata over Γ.

We start with an injective function f:



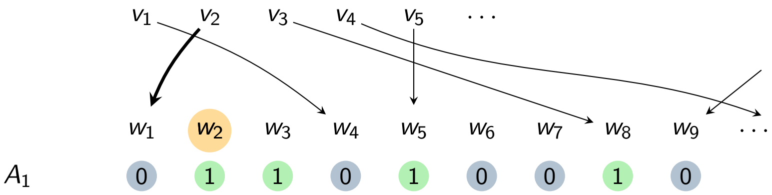
Idea: Change arrows such that every element is in the image, but so that:

- w_2 needs to be part of the image

Extension lemma: proof idea

- Let $v_1, v_2, v_3, \ldots$ be an enumeration of all words Σ^*.
- Let $w_1, w_2, w_3, \ldots$ be an enumeration of all words Γ^*.
- Let $A_1, A_2, A_3, \ldots$ an enumeration of all finite automata over Γ.

We start with an injective function f:



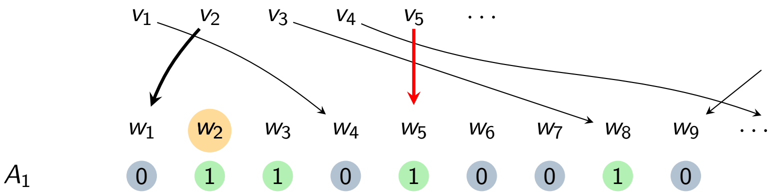
Idea: Change arrows such that every element is in the image, but so that:

- w_2 needs to be part of the image
- we take into account acceptance of the automaton A_1

Extension lemma: proof idea

- Let $v_1, v_2, v_3, \ldots$ be an enumeration of all words Σ^*.
- Let $w_1, w_2, w_3, \ldots$ be an enumeration of all words Γ^*.
- Let $A_1, A_2, A_3, \ldots$ an enumeration of all finite automata over Γ.

We start with an injective function f:



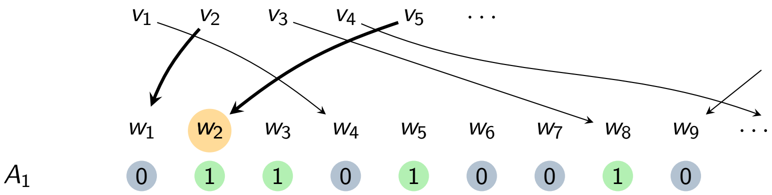
Idea: Change arrows such that every element is in the image, but so that:

- w_2 needs to be part of the image
- we take into account acceptance of the automaton A_1
- we pick the first arrow whose target has equal acceptance to w_2 (accepted by A_1)

Extension lemma: proof idea

- Let $v_1, v_2, v_3, \ldots$ be an enumeration of all words Σ^*.
- Let $w_1, w_2, w_3, \ldots$ be an enumeration of all words Γ^*.
- Let $A_1, A_2, A_3, \ldots$ an enumeration of all finite automata over Γ.

We start with an injective function f:



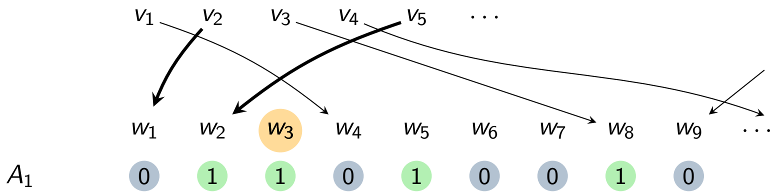
Idea: Change arrows such that every element is in the image, but so that:

- w_2 needs to be part of the image
- we take into account acceptance of the automaton A_1
- we pick the first arrow whose target has equal acceptance to w_2 (accepted by A_1)
- we redirect the arrow to w_2

Extension lemma: proof idea

- Let $v_1, v_2, v_3, \ldots$ be an enumeration of all words Σ^*.
- Let $w_1, w_2, w_3, \ldots$ be an enumeration of all words Γ^*.
- Let $A_1, A_2, A_3, \ldots$ an enumeration of all finite automata over Γ.

We start with an injective function f:



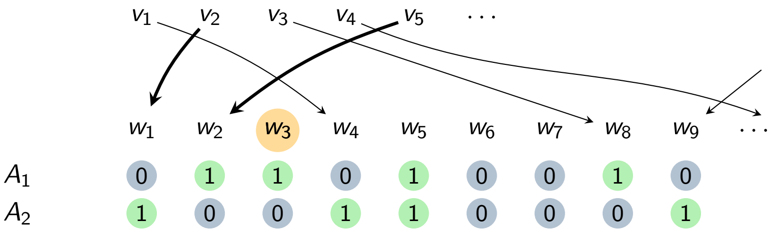
Idea: Change arrows such that every element is in the image, but so that:

- w_3 needs to be part of the image

Extension lemma: proof idea

- Let $v_1, v_2, v_3, \ldots$ be an enumeration of all words Σ^*.
- Let $w_1, w_2, w_3, \ldots$ be an enumeration of all words Γ^*.
- Let $A_1, A_2, A_3, \ldots$ an enumeration of all finite automata over Γ.

We start with an injective function f:



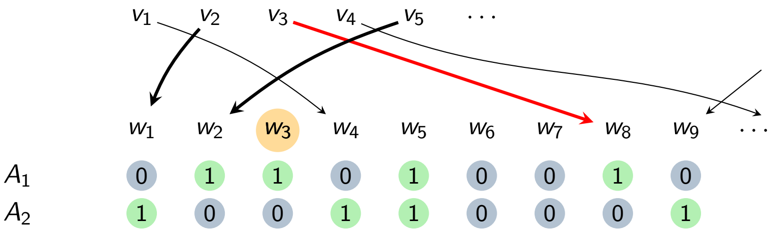
Idea: Change arrows such that every element is in the image, but so that:

- w_3 needs to be part of the image
- we take into account acceptance of A_1 and A_2

Extension lemma: proof idea

- Let $v_1, v_2, v_3, \ldots$ be an enumeration of all words Σ^*.
- Let $w_1, w_2, w_3, \ldots$ be an enumeration of all words Γ^*.
- Let $A_1, A_2, A_3, \ldots$ an enumeration of all finite automata over Γ.

We start with an injective function f:



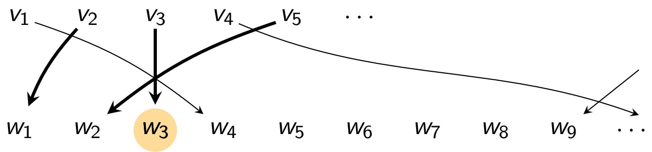
Idea: Change arrows such that every element is in the image, but so that:

- w_3 needs to be part of the image
- we take into account acceptance of A_1 and A_2
- we pick the first arrow whose target has equal acceptance to w_3 (accepted by A_1 and rejected by A_2)

Extension lemma: proof idea

- Let $v_1, v_2, v_3, \ldots$ be an enumeration of all words Σ^*.
- Let $w_1, w_2, w_3, \ldots$ be an enumeration of all words Γ^*.
- Let $A_1, A_2, A_3, \ldots$ an enumeration of all finite automata over Γ.

We start with an injective function f:



	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9
A_1	0	1	1	0	1	0	0	1	0
A_2	1	0	0	1	1	0	0	0	1

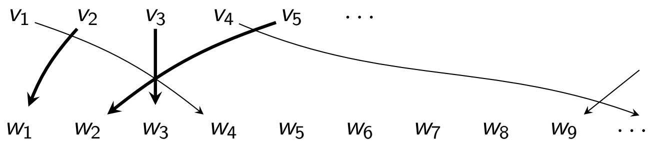
Idea: Change arrows such that every element is in the image, but so that:

- w_3 needs to be part of the image
- we take into account acceptance of A_1 and A_2
- we pick the first arrow whose target has equal acceptance to w_3 (accepted by A_1 and rejected by A_2)
- we redirect the arrow to w_3

Extension lemma: proof idea

- Let $v_1, v_2, v_3, \ldots$ be an enumeration of all words Σ^*.
- Let $w_1, w_2, w_3, \ldots$ be an enumeration of all words Γ^*.
- Let $A_1, A_2, A_3, \ldots$ an enumeration of all finite automata over Γ.

We start with an injective function f:



	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9
A_1	0	1	1	0	1	0	0	1	0
A_2	1	0	0	1	1	0	0	0	1

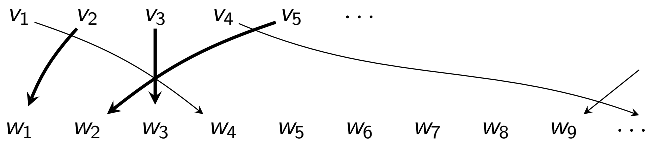
Idea: Change arrows such that every element is in the image, but so that:

. . . and so forth.

Extension lemma: proof idea

- Let $v_1, v_2, v_3, \ldots$ be an enumeration of all words Σ^*.
- Let $w_1, w_2, w_3, \ldots$ be an enumeration of all words Γ^*.
- Let $A_1, A_2, A_3, \ldots$ an enumeration of all finite automata over Γ.

We start with an injective function f:



	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9
A_1	0	1	1	0	1	0	0	1	0
A_2	1	0	0	1	1	0	0	0	1

Idea: Change arrows such that every element is in the image, but so that:

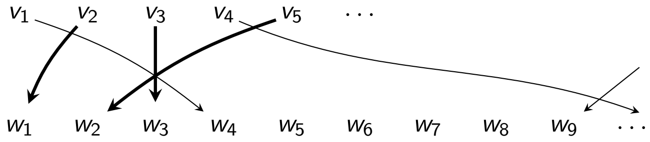
In the $(n+1)$-th step, we make w_{n+1} part of the image

- without altering acceptance of language images by $A_1, A_2, \ldots, A_n$.

Extension lemma: proof idea

- Let $v_1, v_2, v_3, \ldots$ be an enumeration of all words Σ^*.
- Let $w_1, w_2, w_3, \ldots$ be an enumeration of all words Γ^*.
- Let $A_1, A_2, A_3, \ldots$ an enumeration of all finite automata over Γ.

We start with an injective function f:



	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9
A_1	0	1	1	0	1	0	0	1	0
A_2	1	0	0	1	1	0	0	0	1

Idea: Change arrows such that every element is in the image, but so that:

In the $(n+1)$-th step, we make w_{n+1} part of the image

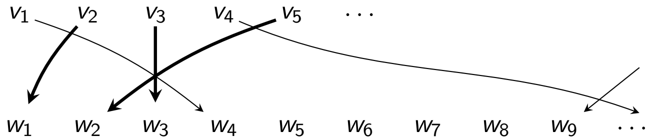
- without altering acceptance of language images by $A_1, A_2, \ldots, A_n$.

Thus the language whose image is accepted by A_n is only disturbed with respect to finitely many words.

Extension lemma: proof idea

- Let $v_1, v_2, v_3, \ldots$ be an enumeration of all words Σ^*.
- Let $w_1, w_2, w_3, \ldots$ be an enumeration of all words Γ^*.
- Let $A_1, A_2, A_3, \ldots$ an enumeration of all finite automata over Γ.

We start with an injective function f:



	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9
A_1	0	1	1	0	1	0	0	1	0
A_2	1	0	0	1	1	0	0	0	1

Idea: Change arrows such that every element is in the image, but so that:

In the $(n+1)$-th step, we make w_{n+1} part of the image

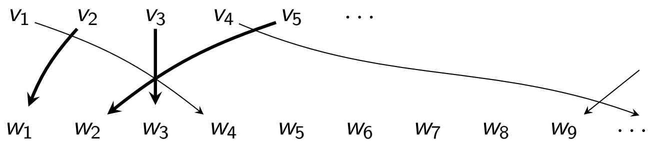
- without altering acceptance of language images by $A_1, A_2, \ldots, A_n$.

(Relative) regularity is preserved in steps and in the limit!

Extension lemma: proof idea

- Let $v_1, v_2, v_3, \dots$ be an enumeration of all words Σ^*.
- Let $w_1, w_2, w_3, \dots$ be an enumeration of all words Γ^*.
- Let $A_1, A_2, A_3, \dots$ an enumeration of all finite automata over Γ.

We start with an injective function f:



	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9
A_1	0	1	1	0	1	0	0	1	0
A_2	1	0	0	1	1	0	0	0	1

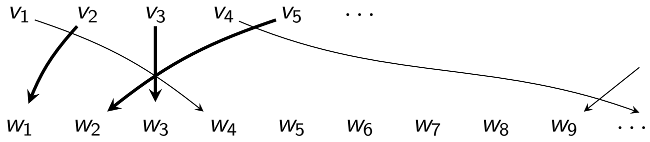
Complication

What if in the induction step there is no arrow to a target with the same acceptance behavior as w_{n+1} for $A_1, \dots, A_n$?

Extension lemma: proof idea

- Let $v_1, v_2, v_3, \ldots$ be an enumeration of all words Σ^*.
- Let $w_1, w_2, w_3, \ldots$ be an enumeration of all words Γ^*.
- Let $A_1, A_2, A_3, \ldots$ an enumeration of all finite automata over Γ.

We start with an injective function f:



	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9
A_1	0	1	1	0	1	0	0	1	0
A_2	1	0	0	1	1	0	0	0	1

Complication

What if in the induction step there is no arrow to a target with the same acceptance behavior as w_{n+1} for $A_1, \ldots, A_n$?

- We show that, if not, A_n can be changed (once) to A'_n with almost the same acceptance behavior such that the choice is possible.

Consequences for comparing models of computation

Comparing computational power via encodings

- Simulation of models of computation $\mathcal{M}_1 = \langle D_1, \mathcal{F}_1 \rangle$, $\mathcal{M}_2 = \langle D_2, \mathcal{F}_2 \rangle$:

Comparing computational power via encodings

- Simulation of functions:

  function f_2 *simulates* function f_1 via *encoding* ρ if:

$$
\mathcal{M}_1 \qquad
\begin{array}{ccc}
D_1 & \xrightarrow{\ \rho\ } & D_2 \\
{\scriptstyle f_1}\downarrow & & \downarrow{\scriptstyle f_2} \\
D_1 & \xrightarrow[\ \rho\ ]{} & D_2
\end{array}
\qquad \mathcal{M}_2
$$

- Simulation of models of computation $\mathcal{M}_1 = \langle D_1, \mathcal{F}_1 \rangle$, $\mathcal{M}_2 = \langle D_2, \mathcal{F}_2 \rangle$:

Comparing computational power via encodings

- Simulation of functions:

  function f_2 *simulates* function f_1 via *encoding* ρ if:

$$\mathcal{M}_1 \qquad \begin{array}{ccc} D_1 & \xrightarrow{\rho} & D_2 \\ {\scriptstyle f_1}\downarrow & & \downarrow{\scriptstyle f_2} \\ D_1 & \xrightarrow[\rho]{} & D_2 \end{array} \qquad \mathcal{M}_2$$

- Simulation of models of computation $\mathcal{M}_1 = \langle D_1, \mathcal{F}_1 \rangle$, $\mathcal{M}_2 = \langle D_2, \mathcal{F}_2 \rangle$:

  $\mathcal{M}_2$ *can simulate* $\mathcal{M}_1$ *via* ρ ($\mathcal{M}_1 \lesssim_\rho \mathcal{M}_2$), if:

Comparing computational power via encodings

- Simulation of functions:

  function f_2 *simulates* function f_1 via *encoding* ρ if:

$$\mathcal{M}_1 \qquad \forall f_1 \in \mathcal{F}_1 \qquad \begin{array}{ccc} D_1 & \xrightarrow{\ \rho\ } & D_2 \\ {\scriptstyle f_1}\downarrow & & \downarrow{\scriptstyle f_2} \\ D_1 & \xrightarrow[\ \rho\ ]{} & D_2 \end{array} \qquad \mathcal{M}_2$$

- Simulation of models of computation $\mathcal{M}_1 = \langle D_1, \mathcal{F}_1 \rangle$, $\mathcal{M}_2 = \langle D_2, \mathcal{F}_2 \rangle$:

  $\mathcal{M}_2$ *can simulate* $\mathcal{M}_1$ *via* ρ ($\mathcal{M}_1 \lesssim_\rho \mathcal{M}_2$), if:

Comparing computational power via encodings

- Simulation of functions:

  function f_2 *simulates* function f_1 via *encoding* ρ if:

$$\mathcal{M}_1 \qquad \forall f_1 \in \mathcal{F}_1 \qquad \begin{array}{ccc} D_1 & \xrightarrow{\quad\rho\quad} & D_2 \\ {\scriptstyle f_1}\downarrow & & \downarrow{\scriptstyle f_2} \\ D_1 & \xrightarrow[\quad\rho\quad]{} & D_2 \end{array} \qquad \exists f_2 \in \mathcal{F}_2 \qquad \mathcal{M}_2$$

- Simulation of models of computation $\mathcal{M}_1 = \langle D_1, \mathcal{F}_1 \rangle$, $\mathcal{M}_2 = \langle D_2, \mathcal{F}_2 \rangle$:

  $\mathcal{M}_2$ *can simulate* $\mathcal{M}_1$ *via* ρ ($\mathcal{M}_1 \lesssim_\rho \mathcal{M}_2$), if:

Comparing computational power via encodings

- Simulation of functions:

  function f_2 *simulates* function f_1 via *encoding* ρ if:

$$
\mathcal{M}_1 \qquad \forall f_1 \in \mathcal{F}_1 \qquad
\begin{array}{ccc}
D_1 & \xrightarrow{\ \rho\ } & D_2 \\
{\scriptstyle f_1}\downarrow & & \downarrow{\scriptstyle f_2} \\
D_1 & \xrightarrow[\ \rho\ ]{} & D_2
\end{array}
\qquad \exists f_2 \in \mathcal{F}_2 \qquad \mathcal{M}_2
$$

- Simulation of models of computation $\mathcal{M}_1 = \langle D_1, \mathcal{F}_1 \rangle$, $\mathcal{M}_2 = \langle D_2, \mathcal{F}_2 \rangle$:

  $\mathcal{M}_2$ *can simulate* $\mathcal{M}_1$ *via* ρ ($\mathcal{M}_1 \lesssim_\rho \mathcal{M}_2$), if:

$$\forall f_1 \in \mathcal{F}_1 \; \exists f_2 \in \mathcal{F}_2 \; \big(f_2 \text{ simulates } f_1 \text{ via } \rho \big)$$

Weak requirements on encodings (Boker/Dershowitz)

Traditional requirements on encodings are:

- *informally computable*/*effective*/*mechanizable in principle*
- *computable* with respect to a specific model (Turing machine, . . .)

Weak requirements on encodings (Boker/Dershowitz)

Traditional requirements on encodings are:

- *informally computable*/*effective*/*mechanizable in principle*
- *computable* with respect to a specific model (Turing machine, . . .)

Boker/Dershowitz: want a 'robust definition that does not itself depend on the notion of computability', and therefore suggest as encodings:

Weak requirements on encodings (Boker/Dershowitz)

Traditional requirements on encodings are:

- *informally computable*/*effective*/*mechanizable in principle*
- *computable* with respect to a specific model (Turing machine, . . .)

Boker/Dershowitz: want a 'robust definition that does not itself depend on the notion of computability', and therefore suggest as encodings:

(i) *injective* functions

(ii) *bijective* functions

Weak requirements on encodings (Boker/Dershowitz)

Traditional requirements on encodings are:

- *informally computable*/*effective*/*mechanizable in principle*
- *computable* with respect to a specific model (Turing machine, . . .)

Boker/Dershowitz: want a 'robust definition that does not itself depend on the notion of computability', and therefore suggest as encodings:

(i) *injective* functions

(ii) *bijective* functions

Definition (power subsumption pre-order [Boker/Dershowitz 2006])

(i) $\mathcal{M}_1 \lesssim \mathcal{M}_2$ if: there is an injective ρ such that $\mathcal{M}_1 \lesssim_\rho \mathcal{M}_2$

(ii) $\mathcal{M}_1 \lesssim_{\text{bijective}} \mathcal{M}_2$ if: there is a bijective ρ such that $\mathcal{M}_1 \lesssim_\rho \mathcal{M}_2$

Anomalies for decision models

Our main result implies anomalies of these definitions.

$\mathcal{M} = \langle D, \mathcal{F} \rangle$ is a *decision model* if $\{0,1\} \subseteq D$, $\forall f \in \mathcal{F}\,(f[D] \subseteq \{0,1\})$.

Anomalies for decision models

Our main result implies anomalies of these definitions.

$\mathcal{M} = \langle D, \mathcal{F} \rangle$ is a *decision model* if $\{0,1\} \subseteq D$, $\forall f \in \mathcal{F}\, (f[D] \subseteq \{0,1\})$.

Corollary (of Main Theorem)

Let Σ and Γ with $\{0,1\} \subseteq \Sigma, \Gamma$ be alphabets.

Then for every countable decision model $\mathcal{M} = \langle \Sigma^*, \mathcal{F} \rangle$, it holds:

$$\mathcal{M} \lesssim \mathsf{DFA}(\Gamma) \qquad \mathcal{M} \lesssim_{\text{bijective}} \mathsf{DFA}(\Gamma)$$

Anomalies for decision models

Our main result implies anomalies of these definitions.

$\mathcal{M} = \langle D, \mathcal{F} \rangle$ is a *decision model* if $\{0,1\} \subseteq D, \quad \forall f \in \mathcal{F}\,(f[D] \subseteq \{0,1\})$.

Corollary (of Main Theorem)

Let Σ and Γ with $\{0,1\} \subseteq \Sigma, \Gamma$ be alphabets.

Then for every countable decision model $\mathcal{M} = \langle \Sigma^*, \mathcal{F} \rangle$, it holds:

$$\mathcal{M} \lesssim \mathsf{DFA}(\Gamma) \qquad \mathcal{M} \lesssim_{\text{bijective}} \mathsf{DFA}(\Gamma)$$

$\mathsf{TMD}(\Sigma)$: class of Turing machine deciders with input alphabet Σ

Anomaly (example)

$$\mathsf{TMD}(\Sigma) \lesssim_{\text{bijective}} \mathsf{DFA}(\Gamma)$$

Extensions and moral of these anomalies

Anomaly (example)

$$\mathsf{TMD}(\Sigma) \lesssim_{\text{bijective}} \mathsf{DFA}(\Gamma)$$

This anomaly for two decision models via bijective encodings:

- can be extended to some moc's with unbounded output domain,

Extensions and moral of these anomalies

Anomaly (example)

$\mathsf{TMD}(\Sigma) \lesssim_{\text{bijective}} \mathsf{DFA}(\Gamma)$

This anomaly for two decision models via bijective encodings:

- can be extended to some moc's with unbounded output domain,
  - but: $\underbrace{\mathsf{TM}(\Sigma)}_{\text{Turing machines}} \not\lesssim_{\text{bijective}} \underbrace{(2\text{-})\mathsf{FST}(\Sigma)}_{\text{(2-way) finite-state transducers}}$

Extensions and moral of these anomalies

Anomaly (example)

$$\mathsf{TMD}(\Sigma) \lesssim_{\text{bijective}} \mathsf{DFA}(\Gamma)$$

This anomaly for two decision models via bijective encodings:

- can be extended to some moc's with unbounded output domain,
  - but: $$\underbrace{\mathsf{TM}(\Sigma)}_{\text{Turing machines}} \not\lesssim_{\text{bijective}} \underbrace{(2\text{-})\mathsf{FST}(\Sigma)}_{\text{(2-way) finite-state transducers}}$$
- depends on uncomputable encodings,

Extensions and moral of these anomalies

Anomaly (example)

$$\mathsf{TMD}(\Sigma) \lesssim_{\text{bijective}} \mathsf{DFA}(\Gamma)$$

This anomaly for two decision models via bijective encodings:

- can be extended to some moc's with unbounded output domain,
  - but: $\underbrace{\mathsf{TM}(\Sigma)}_{\text{Turing machines}} \not\lesssim_{\text{bijective}} \underbrace{(2\text{-})\mathsf{FST}(\Sigma)}_{\text{(2-way) finite-state transducers}}$
- depends on uncomputable encodings,
- highlights that uncomputable encodings must be excluded.

Extensions and moral of these anomalies

Anomaly (example)

$$\mathsf{TMD}(\Sigma) \lesssim_{\text{bijective}} \mathsf{DFA}(\Gamma)$$

This anomaly for two decision models via bijective encodings:

- can be extended to some moc's with unbounded output domain,
  - but: $\underbrace{\mathsf{TM}(\Sigma)}_{\text{Turing machines}} \not\lesssim_{\text{bijective}} \underbrace{(2\text{-})\mathsf{FST}(\Sigma)}_{\text{(2-way) finite-state transducers}}$
- depends on uncomputable encodings,
- highlights that uncomputable encodings must be excluded.
  - Sometimes the structure of the models $\mathcal{M}_1$ and $\mathcal{M}_2$ excludes uncomputable, bijective encodings ρ such that $\mathcal{M}_1 \lesssim_\rho \mathcal{M}_2$.
  - We give a sufficient condition for this, extending work by Shapiro (1982).

Summary

we solved a problem in language theory:

- there exist bijective word encodings
  that are regularity preserving, but not reflecting.

by showing:

- for all countable sets $\mathcal{C}$ of languages,
  there is a bijective encoding g
  such that $g[L]$ is regular for all $L \in \mathcal{C}$.

some consequences:

- for comparing models of computation via encodings:
  - use of *unrestricted* bijective encodings leads to anomalies
- in the paper: for c-automatic sequences