Languages of lossless seeds

Karel Břinda
karel.brinda@univ-mlv.fr

LIGM Université Paris-Est Marne-la-Vallée, France

May 27, 2014
14th International Conference Automata and Formal Languages

Introduction

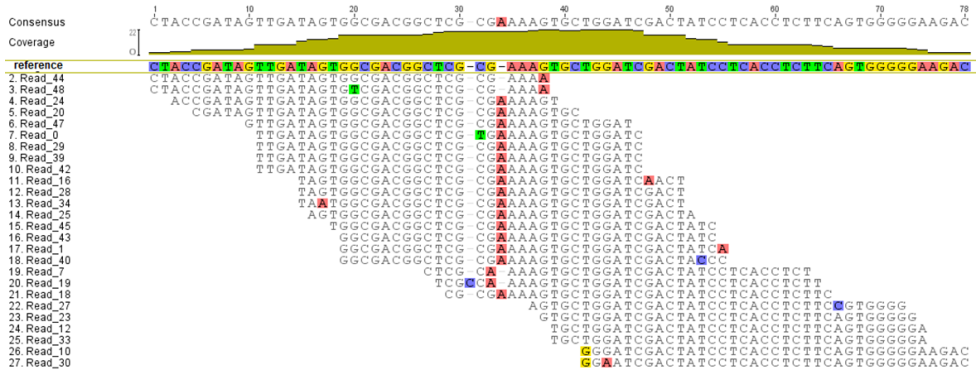
- Lossless seeds are objects used to speed-up approximate string matching algorithms

Introduction

- Lossless seeds are objects used to speed-up approximate string matching algorithms
- They are widely used in programs for Next-Generation Sequencing data processing

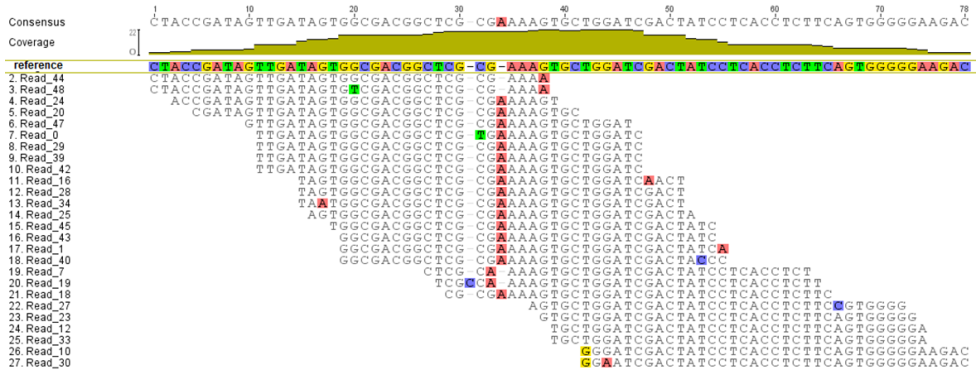
Introduction

- Lossless seeds are objects used to speed-up approximate string matching algorithms
- They are widely used in programs for Next-Generation Sequencing data processing
  - e.g., in short-read mapping



Introduction

- Lossless seeds are objects used to speed-up approximate string matching algorithms



- They are widely used in programs for NEXT-GENERATION SEQUENCING data processing
  - e.g., in short-read mapping

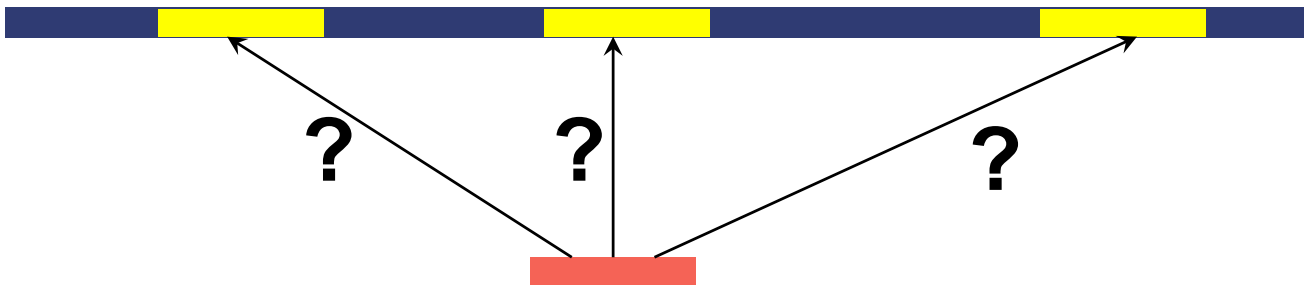
We show a connection between

- lossless seeds
- symbolic dynamics
- formal languages

Text filtration – general principle

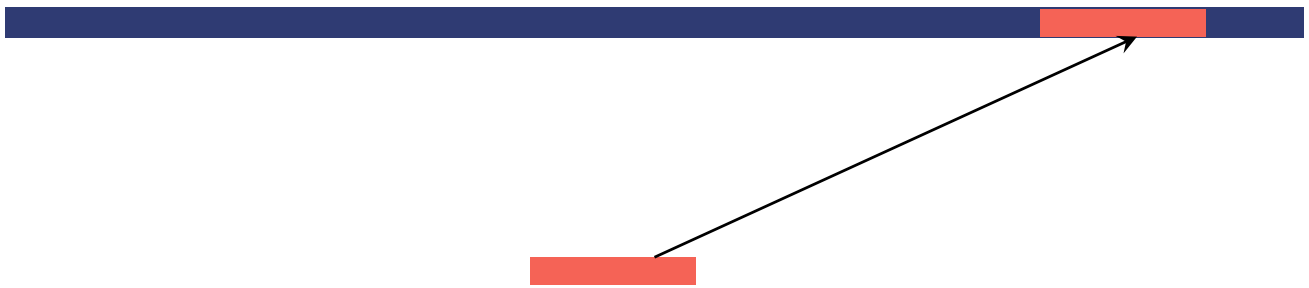
Text filtration – general principle

Potential matches



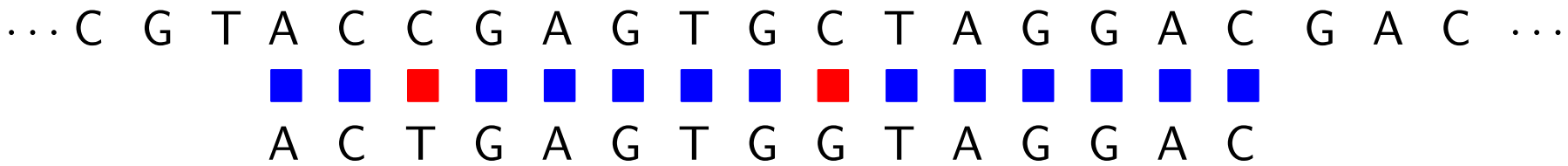
Text filtration – general principle

True matches



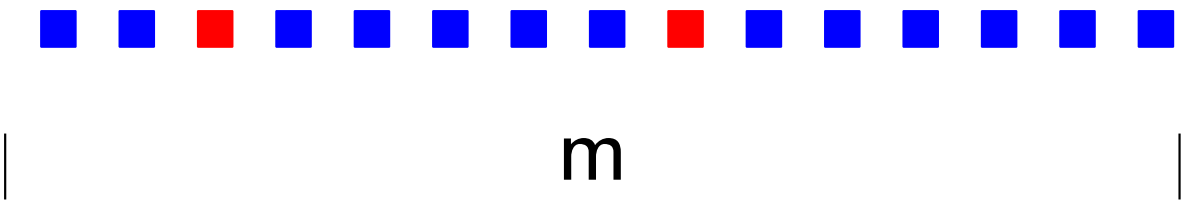
Gapless similarities

Similarities can be defined through **Hamming distance**



Gapless similarities

Similarities can be defined through **Hamming distance**



(m, k)-**problem**

m: length of compared strings
k: number of allowed mismatches

Seeds

Principle of lossless seeds

Two strings of the same length m in Hamming distance $\leq k$ must share some patterns – so-called **seeds**

Seeds

Principle of lossless seeds

Two strings of the same length m in Hamming distance $\leq k$ must share some patterns – so-called **seeds**

\# ... **matching symbol** (matching position)
\- ... **joker symbol** (matching or mismatching position)

Seeds

Principle of lossless seeds

Two strings of the same length m in Hamming distance $\leq k$ must share some patterns – so-called **seeds**

\# ... **matching symbol** (matching position)
\- ... **joker symbol** (matching or mismatching position)

Seeds can be: continuous vs. spaced, lossy vs. lossless

Lossless seeds

Example ($(15, 2)$-problem)

Seeds solving this problem are, e.g.:

- `#####`
- `##-#--##-#`

Lossless seeds

Example ($(15, 2)$-problem)

Seeds solving this problem are, e.g.:

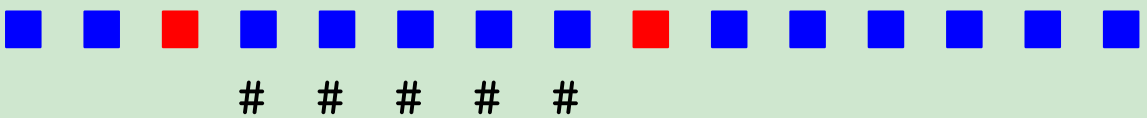
- `#####`
- `##-#--##-#`

Lossless seeds

Example ($(15, 2)$-problem)

Seeds solving this problem are, e.g.:

- `#####`
- `##-#--##-#`



Lossless seeds

Example ($(15, 2)$-problem)

Seeds solving this problem are, e.g.:

- `#####`
- `##-#--##-#`

Seed design

- We are given $m > k > 0$, a typical task is to find seeds solving the (m, k)-problem
  - lossless seeds must detect **all** $\binom{m}{k}$ error combinations

Seed design

- We are given $m > k > 0$, a typical task is to find seeds solving the (m, k)-problem
  - lossless seeds must detect **all** $\binom{m}{k}$ error combinations
- Higher weight (number of #'s) $\implies$ **higher selectivity** of the filter
  - we want to reach the **maximal possible** weight

Seed design

- We are given $m > k > 0$, a typical task is to find seeds solving the (m, k)-problem
  - lossless seeds must detect **all** $\binom{m}{k}$ error combinations
- Higher weight (number of #'s) $\implies$ **higher selectivity** of the filter
  - we want to reach the **maximal possible** weight
- Designed seeds are then **used as parameters** in the programs
  - they are designed for a specific task

Spaced seeds – articles overview

- **Introduced:**
  [Burkhardt and Kärkkäinen, 2001, Burkhardt and Kärkkäinen, 2002]
- **Asymptotic properties:** [Kucherov et al., 2005, Farach-Colton et al., 2007]
- **Complexity:** [Nicolas and Rivals, 2005, Nicolas and Rivals, 2008]
- **Seed design:** [Kucherov et al., 2005], [Lin et al., 2008], [Chen et al., 2009], [Egidi and Manzini, 2011, Egidi and Manzini, 2014b], [Egidi and Manzini, 2013], [B., 2013], [Egidi and Manzini, 2014a]
- **Generalizations:**
  - Seed families: [Kucherov et al., 2005]
  - Vector seeds: [Brejová et al., 2005]
  - Subset seeds: [Noé and Kucherov, 2005]
- **Programs which use them** (examples): ZOOM [Lin et al., 2008], PerM [Chen et al., 2009], Shrimp2 [David et al., 2011]
- **List of all papers:** [Noé, 2014]

An important issue – structural properties of lossless seeds

What are structural properties of seeds?

- If seeds have a good structure, we can help decrease memory consumption of short-read mappers (smaller hashtables)

An important issue – structural properties of lossless seeds

What are structural properties of seeds?

- If seeds have a good structure, we can help decrease memory consumption of short-read mappers (smaller hashtables)
- Observed in many articles: many good seeds are repetitions of short patterns

An important issue – structural properties of lossless seeds

What are structural properties of seeds?

- If seeds have a good structure, we can help decrease memory consumption of short-read mappers (smaller hashtables)
- Observed in many articles: many good seeds are repetitions of short patterns
- Can be optimal seeds obtained from short patterns in any case?

Conjecture [B., 2013]

Fix m. For every $0 < s < m$, a seed of length s solving the $(m, 2)$-problem with highest possible weight can be obtained from a cyclic seed[a] of length at most $\ell + 1$.

[a] Defined in [Kucherov et al., 2005].

An important issue – structural properties of lossless seeds

What are structural properties of seeds?

- If seeds have a good structure, we can help decrease memory consumption of short-read mappers (smaller hashtables)
- Observed in many articles: many good seeds are repetitions of short patterns
- Can be optimal seeds obtained from short patterns in any case?

Conjecture [B., 2013]

Fix m. For every $0 < s < m$, a seed of length s solving the $(m, 2)$-problem with highest possible weight can be obtained from a cyclic seed[a] of length at most $\ell + 1$.

[a] Defined in [Kucherov et al., 2005].

To find the structure – **approach based on subshifts** – seeds form languages of certain sofic subshifts

Parameters

Fixed params	**Object of study**
m and k	seeds solving the (m, k)-problem
ℓ and k	seeds Q solving $(\lvert Q\rvert + \ell, k)$-problems

Parameters

Fixed params	Object of study
m and k	seeds solving the (m, k)-problem
ℓ and k	seeds Q solving $(\lvert Q\rvert + \ell, k)$-problems

Asymptotic relations between ℓ and m for optimal seeds

Theorem

For a fixed positive k, optimal seeds must satisfy $\ell \in \Theta(m^{\frac{k}{k+1}})$

Subshifts – notation

$\mathcal{A} = \{\#, -\}$	**seed alphabet**
$\mathbf{w}$	a **bi-infinite word** (from $\mathcal{A}^{\mathbb{Z}}$)
$\mathbf{w}[m, n]$	**factor** $\mathbf{w}_m \mathbf{w}_{m+1} \cdots \mathbf{w}_n$ of the bi-infinite word $\mathbf{w}$
σ	**shift operation**, $(\sigma(\mathbf{u}))_i = \mathbf{u}_{i+1}$
S_X	**subshift** of all bi-infinite words over $\mathcal{A}$, which do not contain any word from X as a factor (X: set of finite words)
$\oplus$	**"seed OR"**

```
-#--###-#
    ⊕
##--#-##-
    =
##--#####
```

Approach based on subshifts

The main idea: Fix ℓ and k, find subshifts of the full shift $\mathcal{A}^{\mathbb{Z}}$ with seeds as languages.

Approach based on subshifts

The main idea: Fix ℓ and k, find subshifts of the full shift $\mathcal{A}^{\mathbb{Z}}$ with seeds as languages.

Theorem

Let:

- m *and* k *be positive integers,*
- Q *be a seed such that* $|Q| < m$,
- $\ell := m - |Q|$,
- $\mathbf{w} := \cdots --|Q--\cdots$.

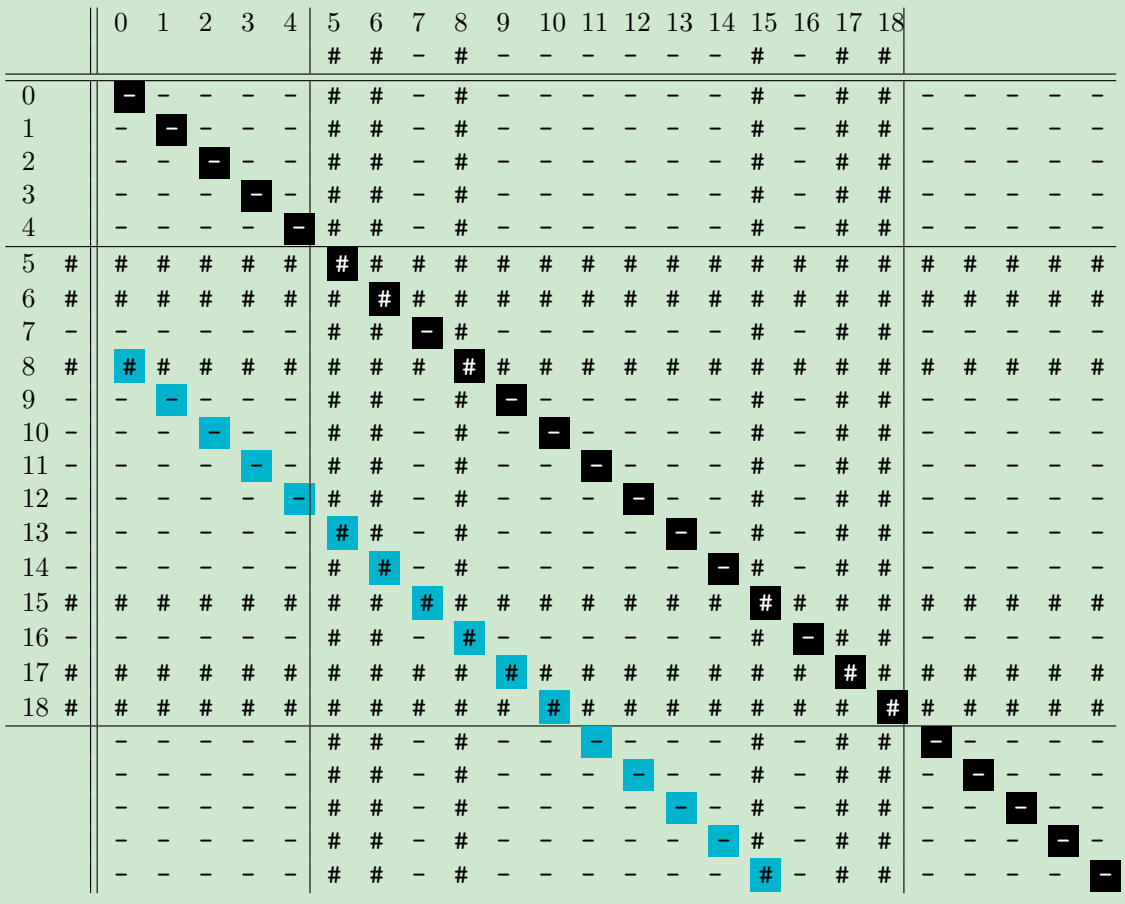
The seed Q *does not solve the* (m, k)*-problem if and only if*

$$(\oplus(\sigma^{i_1}(\mathbf{w}), \ldots, \sigma^{i_k}(\mathbf{w}))[0, \ell] = \#^{\ell+1}.$$

for some integers $i_1, \ldots, i_k$.

$(m, 2)$-problems – Laser method [B., 2013]

Example ($(19, 2)$-problem, $Q =$ ##–#––––––#–##)



The bi-infinite word $\oplus(\sigma^5(\mathbf{w}), \sigma^{13}(\mathbf{w})$

$(m, 2)$-problems – Laser method [B., 2013]

Example ($(19, 2)$-problem, $Q =$ ##–#––––––#–##)

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18					
							#	#	–	#	–	–	–	–	–	–	#	–	#	#					
0		–	–	–	–	–	#	#	–	#	–	–	–	–	–	–	#	–	#	#	–	–	–	–	–
1		–	–	–	–	–	#	#	–	#	–	–	–	–	–	–	#	–	#	#	–	–	–	–	–
2		–	–	–	–	–	#	#	–	#	–	–	–	–	–	–	#	–	#	#	–	–	–	–	–
3		–	–	–	–	–	#	#	–	#	–	–	–	–	–	–	#	–	#	#	–	–	–	–	–
4		–	–	–	–	–	#	#	–	#	–	–	–	–	–	–	#	–	#	#	–	–	–	–	–
5	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
6	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
7	–	–	–	–	–	–	#	#	–	#	–	–	–	–	–	–	#	–	#	#	–	–	–	–	–
8	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
9	–	–	–	–	–	–	#	#	–	#	–	–	–	–	–	–	#	–	#	#	–	–	–	–	–
10	–	–	–	–	–	–	#	#	–	#	–	–	–	–	–	–	#	–	#	#	–	–	–	–	–
11	–	–	–	–	–	–	#	#	–	#	–	–	–	–	–	–	#	–	#	#	–	–	–	–	–
12	–	–	–	–	–	–	#	#	–	#	–	–	–	–	–	–	#	–	#	#	–	–	–	–	–
13	–	–	–	–	–	–	#	#	–	#	–	–	–	–	–	–	#	–	#	#	–	–	–	–	–
14	–	–	–	–	–	–	#	#	–	#	–	–	–	–	–	–	#	–	#	#	–	–	–	–	–
15	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
16	–	–	–	–	–	–	#	#	–	#	–	–	–	–	–	–	#	–	#	#	–	–	–	–	–
17	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
18	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
		–	–	–	–	–	#	#	–	#	–	–	–	–	–	–	#	–	#	#	–	–	–	–	–
		–	–	–	–	–	#	#	–	#	–	–	–	–	–	–	#	–	#	#	–	–	–	–	–
		–	–	–	–	–	#	#	–	#	–	–	–	–	–	–	#	–	#	#	–	–	–	–	–
		–	–	–	–	–	#	#	–	#	–	–	–	–	–	–	#	–	#	#	–	–	–	–	–
		–	–	–	–	–	#	#	–	#	–	–	–	–	–	–	#	–	#	#	–	–	–	–	–

$\Longrightarrow$ Q does not solve the problem

Extending seeds to bi-infinite words

Definition

w is (ℓ, k)-valid $\iff$ $\forall i_1, \ldots, i_k \in \mathbb{Z}$:

$$(\oplus(\sigma^{i_1}(\mathbf{w}), \ldots, \sigma^{i_k}(\mathbf{w}))[0, \ell] \neq \#^{\ell+1}.$$

Important property: Every factor Q of **w** solves the $(|Q| + \ell, k)$-problem.

Extending seeds to bi-infinite words

Definition

w is (ℓ, k)-valid $\iff$ $\forall i_1, \ldots, i_k \in \mathbb{Z}$:

$$(\oplus(\sigma^{i_1}(\mathbf{w}), \ldots, \sigma^{i_k}(\mathbf{w}))[0, \ell] \neq \#^{\ell+1}.$$

Important property: Every factor Q of **w** solves the $(|Q| + \ell, k)$-problem.

For given positive k and ℓ, we define on $\mathcal{A}^{\ell+1}$ a k-nary **compatibility relation** C_k^{ℓ} (details in the paper) such that:

Extending seeds to bi-infinite words

Definition

w is (ℓ, k)-valid $\iff$ $\forall i_1, \ldots, i_k \in \mathbb{Z}$:

$$(\oplus(\sigma^{i_1}(\mathbf{w}), \ldots, \sigma^{i_k}(\mathbf{w}))[0, \ell] \neq \#^{\ell+1}.$$

Important property: Every factor Q of **w** solves the $(|Q| + \ell, k)$-problem.

For given positive k and ℓ, we define on $\mathcal{A}^{\ell+1}$ a k-nary **compatibility relation** C_k^{ℓ} (details in the paper) such that:

A bi-infinite word **w** is (ℓ, k)-valid $\iff$ all its k factors $u^{(1)}, \ldots, u^{(k)}$ of length $\ell + 1$ satisfy $\mathrm{C}_k^{\ell}(u^{(1)}, \ldots, u^{(k)})$

Example

Example ($\ell = 5$, $k = 2$)

- $Q^{(1)} =$ ##–#–– solves the $(11, 2)$-problem
- $Q^{(2)} =$ ––#–## solves the $(11, 2)$-problem
- $Q^{(3)} =$ ##–#––––––––––#–## does not solve the $(25, 2)$-problem
  - because $\neg \mathrm{C}^{\frac{5}{2}}(Q^{(1)}, Q^{(2)})$
    - reason: $Q^{(1)} \oplus Q^{(2)} =$ ###### $= \#^{\ell+1}$

Generating sets

Definition

For given positive integers ℓ and k, a subset G of $\mathcal{A}^{\ell+1}$ is called (ℓ, k)-**generating set** if the following conditions are satisfied:

1. for all $v^{(1)}, \ldots, v^{(k)} \in G$, it holds $\mathrm{C}_k^{\ell}(v^{(1)}, \ldots, v^{(k)})$;
2. it cannot contain any other word from $\mathcal{A}^{\ell+1}$.

Generating sets

Definition

For given positive integers ℓ and k, a subset G of $\mathcal{A}^{\ell+1}$ is called (ℓ, k)-**generating set** if the following conditions are satisfied:

1. for all $v^{(1)}, \ldots, v^{(k)} \in G$, it holds $\mathrm{C}_k^{\ell}(v^{(1)}, \ldots, v^{(k)})$;
2. it cannot contain any other word from $\mathcal{A}^{\ell+1}$.

G **fully determines a subshift:** $S_{\mathcal{A}^{\ell+1} \setminus G}$

$\Longrightarrow$ it is a subshift of finite type

$\Longrightarrow$ its language is recognized by a finite automaton

Generating sets

Definition

For given positive integers ℓ and k, a subset G of $\mathcal{A}^{\ell+1}$ is called (ℓ, k)-**generating set** if the following conditions are satisfied:

1. for all $v^{(1)}, \ldots, v^{(k)} \in G$, it holds $\mathrm{C}_k^{\ell}(v^{(1)}, \ldots, v^{(k)})$;
2. it cannot contain any other word from $\mathcal{A}^{\ell+1}$.

G **fully determines a subshift:** $S_{\mathcal{A}^{\ell+1} \setminus G}$

$\Longrightarrow$ it is a subshift of finite type

$\Longrightarrow$ its language is recognized by a finite automaton

Definition

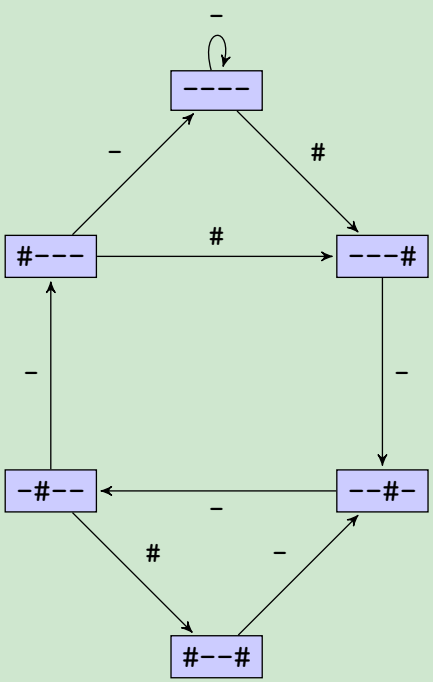
A seed Q is **generated** by an (ℓ, k)-generating set G if it is a factor of some bi-infinite word from $S_{\mathcal{A}^{\ell+1} \setminus G}$.

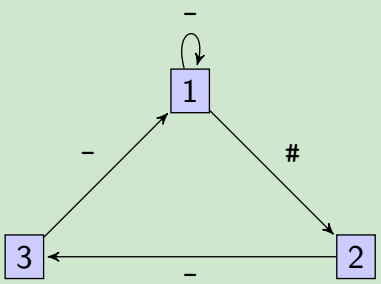
Recognizing automata

Example ($k = 2$, $\ell = 3$)

Generating set $G = \{----, \#---, -\#--, --\#-, ---\#, \#--\#\}$

Automaton from de-Bruijn graph ...and after minimization.





Words from $S_{\mathcal{A}^{\ell+1} \setminus G}$ are bi-infinite paths in such graphs

The main result

Theorem

For given ℓ and k, the set of all (ℓ, k)-valid bi-infinite words is a sofic subshift.

The main result

For given positive ℓ and k:

- Every (ℓ, k)-valid bi-infinite word is generated by some (ℓ, k)-generating set.
- Set of bi-infinite words generated by the same (ℓ, k)-generating set is a subshift of finite type.
- There exist only finitely many (ℓ, k)-generating sets.

$\Downarrow$ $\Downarrow$ $\Downarrow$ $\Downarrow$ $\Downarrow$

Theorem

For given ℓ and k, the set of all (ℓ, k)-valid bi-infinite words is a sofic subshift.

How to find generating sets

1. $k = 1$: The only simple case.

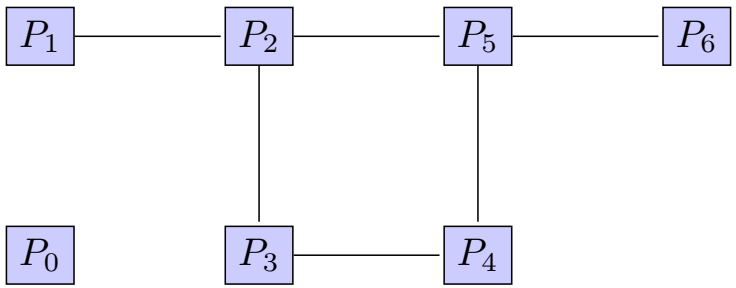
Lemma

Let ℓ and k be positive integers. The unique (ℓ, k)-generating set is the set $\mathcal{A}^{\ell+1} \setminus \{\#^{\ell+1}\}$

2. $k = 2$: $(m, 2)$-generating sets $=$ maximal (w.r.t. to inclusion) independent sets of a graph constructed from the relation C_2^{ℓ}

3. $k > 2$: we would need hypergraphs

Example - $(5,2)$-generating sets search



$P_0 = \{$------ ; -----# , ----#- , ---#-- , --#--- , -#---- , #----- ;
----## , ---##- , --##-- , -##--- , ##---- ;
---#-# , --#-#- , -#-#-- , #-#--- ;
--#--# , -#--#- , #--#-- ; -#---# , #---#- ;
#---## ; ##---# ; #----# $\}$,

$P_1 = \{$#--#-# $\}$, $P_2 = \{$--##-# , -##-#- , ##-#-- $\}$,

$P_3 = \{$-##--# , ##--#- $\}$, $P_4 = \{$-#--## , #--##- $\}$,

$P_5 = \{$--#-## , -#-##- , #-##-- $\}$, $P_6 = \{$#-#--# $\}$.

Concluding remarks

- Every (ℓ, k)-generating set corresponds to a subshift of finite type
- The subshift of all (ℓ, k)-valid bi-infinite words is sofic
- For given ℓ and k, the set of seeds Q solving $(|Q| + \ell, k)$-problems is a regular language
- Cycles in its recognizing automaton correspond to cyclic seeds from [Kucherov et al., 2005]
- For a fixed k, the relation between m and ℓ for optimal seed is $\ell \in \Theta(m^{\frac{k}{k+1}}) \implies$ when $m \to +\infty$, then $\ell \to +\infty$, too

Thank you for your attention!

References I

B., K. (2013).
Lossless seeds for approximate string matching.
Master's thesis, FNSPE Czech Technical University in Prague, Czech Republic.

Brejová, B., Brown, D. G., and Vinař, T. (2005).
Vector seeds: An extension to spaced seeds.
Journal of Computer and System Sciences, 70(3):364–380.
Special Issue on Bioinformatics {II}.

Burkhardt, S. and Kärkkäinen, J. (2001).
Better filtering with gapped q-grams.
In *Proceedings of the 12th Symposium on Combinatorial Pattern Matching (CPM)*, volume 2089 of *Lecture Notes in Computer Science*, pages 73–85. Springer.

Burkhardt, S. and Kärkkäinen, J. (2002).
Better filtering with gapped q-grams.
Fundamenta Informaticae, 56(1-2):51–70.

References II

Chen, Y., Souaiaia, T., and Chen, T. (2009).

PerM: efficient mapping of short sequencing reads with periodic full sensitive spaced seeds.

Bioinformatics, 25(19):2514–2521.

David, M., Dzamba, M., Lister, D., Ilie, L., and Brudno, M. (2011).

Shrimp2: Sensitive yet practical short read mapping.

Bioinformatics.

Egidi, L. and Manzini, G. (2011).

Spaced seeds design using perfect rulers.

In *Proceedings of the 18th International Symposium on String Processing and Information Retrieval (SPIRE), Pisa (Italy)*, volume 7024 of *Lecture Notes in Computer Science*, pages 32–43. Springer.

Egidi, L. and Manzini, G. (2013).

Better spaced seeds using quadratic residues.

Journal of Computer and System Sciences, 79(7):1144–1155.

References III


Egidi, L. and Manzini, G. (2014a).
Design and analysis of periodic multiple seeds.
Theoretical Computer Science, 522:62–76.


Egidi, L. and Manzini, G. (2014b).
Spaced seeds design using perfect rulers.
Fundamenta Informaticae, 131(2):187–203.


Farach-Colton, M., Landau, G. M., Cenk Sahinalp, S., and Tsur, D. (2007).
Optimal spaced seeds for faster approximate string matching.
Journal of Computer and System Sciences, 73(7):1035–1044.


Kucherov, G., Noé, L., and Roytberg, M. A. (2005).
Multiseed lossless filtration.
IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB), 2(1):51–61.


Lin, H., Zhang, Z., Zhang, M. Q., Ma, B., and Li, M. (2008).
ZOOM! Zillions Of Oligos Mapped.
Bioinformatics, 24(21):2431–2437.

References IV

Nicolas, F. and Rivals, É. (2005).

Hardness of optimal spaced seed design.

In Apostolico, A., Crochemore, M., and Park, K., editors, *Proceedings of the 16th Annual Symposium on Combinatorial Pattern Matching (CPM), Jeju Island (Korea)*, volume 3537 of *Lecture Notes in Computer Science*, pages 144–155. Springer.

Nicolas, F. and Rivals, É. (2008).

Hardness of optimal spaced seed design.

Journal of Computer and System Sciences, 74(5):831–849.

Noé, L. (2014).

Spaced seeds bibliography on http://www.lifl.fr/~noe/spaced_seeds.html.

Noé, L. and Kucherov, G. (2005).

YASS: enhancing the sensitivity of DNA similarity search.

Nucleic Acids Research, 33(suppl. 2):W540–W543.