Bounded-delay enumeration of regular languages

Mikaël Monet, Antoine Amarilli

STACS 2023, Hamburg, Germany

March 8, 2023

My co-author

Joint work with Antoine Amarilli

arXiv: https://arxiv.org/abs/2209.14878

Outline

Introduction

Gray code for n-bit words

- Gray code over n-bit words: an ordering

$$w_1, w_2, \ldots, w_{2^n}$$

of $(a+b)^n$ such that w_i, w_{i+1} differ by exactly one bit.

Gray code for n-bit words

- Gray code over n-bit words: an ordering

$$w_1, w_2, \dots, w_{2^n}$$

  of $(a+b)^n$ such that w_i, w_{i+1} differ by exactly one bit.
- Concatenate Gray codes for $n = 0, 1, 2, \dots$: we obtain an ordering $w_1, w_2, \dots$ of $(a+b)^*$ where consecutive words are at Levenshtein distance one.

Gray code for n-bit words

- Gray code over n-bit words: an ordering

$$w_1, w_2, \dots, w_{2^n}$$

  of $(a+b)^n$ such that w_i, w_{i+1} differ by exactly one bit.
- Concatenate Gray codes for $n = 0, 1, 2, \dots$: we obtain an ordering $w_1, w_2, \dots$ of $(a+b)^*$ where consecutive words are at Levenshtein distance one.
- In general, let $L \subseteq \Sigma^*$ be any language over some alphabet Σ. We say that L is orderable for the Levenshtein distance if there exists $d \in \mathbb{N}$ and an ordering

$$w_1, w_2, \dots$$

  of L such that consecutive words are at Levenshtein distance at most $\leq d$.

Orderability for the Levenshtein distance

Definition

A language $L \subseteq \Sigma^*$ is orderable for the Levenshtein distance if there exists $d \in \mathbb{N}$ and an ordering $w_1, w_2, \ldots$ of L such that any two consecutive words at at Levenshtein distance at most d.

Examples: Are these languages orderable for the Levenshtein distance?

- any finite language

Orderability for the Levenshtein distance

Definition

A language $L \subseteq \Sigma^*$ is orderable for the Levenshtein distance if there exists $d \in \mathbb{N}$ and an ordering $w_1, w_2, \ldots$ of L such that any two consecutive words at at Levenshtein distance at most d.

Examples: Are these languages orderable for the Levenshtein distance?

- any finite language yes

Orderability for the Levenshtein distance

Definition

A language $L \subseteq \Sigma^*$ is orderable for the Levenshtein distance if there exists $d \in \mathbb{N}$ and an ordering $w_1, w_2, \ldots$ of L such that any two consecutive words at at Levenshtein distance at most d.

Examples: Are these languages orderable for the Levenshtein distance?

- any finite language yes
- for $k \in \mathbb{N}$, the language $(a^k)^*$

Orderability for the Levenshtein distance

Definition

A language $L \subseteq \Sigma^*$ is orderable for the Levenshtein distance if there exists $d \in \mathbb{N}$ and an ordering $w_1, w_2, \ldots$ of L such that any two consecutive words at at Levenshtein distance at most d.

Examples: Are these languages orderable for the Levenshtein distance?

- any finite language yes
- for $k \in \mathbb{N}$, the language $(a^k)^*$ yes

Orderability for the Levenshtein distance

Definition

A language $L \subseteq \Sigma^*$ is orderable for the Levenshtein distance if there exists $d \in \mathbb{N}$ and an ordering $w_1, w_2, \ldots$ of L such that any two consecutive words at at Levenshtein distance at most d.

Examples: Are these languages orderable for the Levenshtein distance?

- any finite language yes
- for $k \in \mathbb{N}$, the language $(a^k)^*$ yes
- $a^* b^*$

Orderability for the Levenshtein distance

Definition

A language $L \subseteq \Sigma^*$ is orderable for the Levenshtein distance if there exists $d \in \mathbb{N}$ and an ordering $w_1, w_2, \ldots$ of L such that any two consecutive words at at Levenshtein distance at most d.

Examples: Are these languages orderable for the Levenshtein distance?

- any finite language — yes
- for $k \in \mathbb{N}$, the language $(a^k)^*$ — yes
- $a^* b^*$ — yes (Hamiltonian path in the $\mathbb{N} \times \mathbb{N}$ grid)

Orderability for the Levenshtein distance

Definition

A language $L \subseteq \Sigma^*$ is orderable for the Levenshtein distance if there exists $d \in \mathbb{N}$ and an ordering $w_1, w_2, \ldots$ of L such that any two consecutive words at at Levenshtein distance at most d.

Examples: Are these languages orderable for the Levenshtein distance?

- any finite language — yes
- for $k \in \mathbb{N}$, the language $(a^k)^*$ — yes
- $a^* b^*$ — yes (Hamiltonian path in the $\mathbb{N} \times \mathbb{N}$ grid)
- $a^* + b^*$

Orderability for the Levenshtein distance

Definition

A language $L \subseteq \Sigma^*$ is orderable for the Levenshtein distance if there exists $d \in \mathbb{N}$ and an ordering $w_1, w_2, \ldots$ of L such that any two consecutive words at at Levenshtein distance at most d.

Examples: Are these languages orderable for the Levenshtein distance?

- any finite language — yes
- for $k \in \mathbb{N}$, the language $(a^k)^*$ — yes
- $a^* b^*$ — yes (Hamiltonian path in the $\mathbb{N} \times \mathbb{N}$ grid)
- $a^* + b^*$ — no!

Other distances: definitions

We can also consider other distances in this definition:

- the push-pop distance. Defined like the Levenshtein distance, but the basic operations are:
  - popL and popR, to delete the last (resp., the first) letter of the word; and
  - $\mathrm{pushL}(\alpha)$ and $\mathrm{pushR}(\alpha)$ for $\alpha \in \Sigma$, to add the letter α at the beginning (resp., at the end) the word.

Other distances: definitions

We can also consider other distances in this definition:

- the push-pop distance. Defined like the Levenshtein distance, but the basic operations are:
  - popL and popR, to delete the last (resp., the first) letter of the word; and
  - $\mathrm{pushL}(\alpha)$ and $\mathrm{pushR}(\alpha)$ for $\alpha \in \Sigma$, to add the letter α at the beginning (resp., at the end) the word.
- the push-pop-right distance. Defined like the push-pop distance, but only allows popR and $\mathrm{pushR}(\alpha)$ for $\alpha \in \Sigma$.

Questions

We focus on regular languages

Questions

We focus on regular languages

- What are the regular languages that are orderable:
  - for the Levenshtein distance?
  - for the push-pop distance?
  - for the push-pop-right distance?

Questions

We focus on regular languages

- What are the regular languages that are orderable:
  - for the Levenshtein distance?
  - for the push-pop distance?
  - for the push-pop-right distance?
- Can we recognize them? (e.g., given a DFA)

Questions

We focus on regular languages

- What are the regular languages that are orderable:
  - for the Levenshtein distance?
  - for the push-pop distance?
  - for the push-pop-right distance?
- Can we recognize them? (e.g., given a DFA)
- Can we always partition a regular language into a finite number of orderable languages? (as in $a^* + b^*$)

Questions

We focus on regular languages

- What are the regular languages that are orderable:
  - for the Levenshtein distance?
  - for the push-pop distance?
  - for the push-pop-right distance?
- Can we recognize them? (e.g., given a DFA)
- Can we always partition a regular language into a finite number of orderable languages? (as in $a^* + b^*$)
- When L is orderable, can we design an enumeration algorithm for it? With what delay? (poly, constant?)

Main results

Main results (Levenshtein and push-pop)

Let L be regular. We show:

- There exists $t \in \mathbb{N}$ and regular languages $L_1, \ldots, L_t$ such that

$$L = L_1 \sqcup \ldots \sqcup L_t$$

  and each L_i is orderable for the push-pop distance

Main results (Levenshtein and push-pop)

Let L be regular. We show:

- There exists $t \in \mathbb{N}$ and regular languages $L_1, \dots, L_t$ such that

  $$L = L_1 \sqcup \dots \sqcup L_t$$

  and each L_i is orderable for the push-pop distance
- This t is optimal, even for the Levenshtein distance: L cannot be partitioned into less than t orderable languages for the Levenshtein distance.

Main results (Levenshtein and push-pop)

Let L be regular. We show:

- There exists $t \in \mathbb{N}$ and regular languages $L_1, \ldots, L_t$ such that

$$L = L_1 \sqcup \ldots \sqcup L_t$$

  and each L_i is orderable for the push-pop distance
- This t is optimal, even for the Levenshtein distance: L cannot be partitioned into less than t orderable languages for the Levenshtein distance.
  - → This shows L is orderable for Levenshtein iff it is for push-pop!

Main results (Levenshtein and push-pop)

Let L be regular. We show:

- There exists $t \in \mathbb{N}$ and regular languages $L_1, \ldots, L_t$ such that

  $$L = L_1 \sqcup \ldots \sqcup L_t$$

  and each L_i is orderable for the push-pop distance
- This t is optimal, even for the Levenshtein distance: L cannot be partitioned into less than t orderable languages for the Levenshtein distance.
  - → This shows L is orderable for Levenshtein iff it is for push-pop!
- When L is orderable for push-pop then, in a suitable pointer machine model, we have an algorithm that outputs push-pop edit scripts to enumerate L, with bounded delay (i.e., independent from the current word length)

Enumeration algorithms with push-pop edit scripts

Let L regular, e.g., $(\epsilon + a)b^*$. GOAL: enumerate L with a delay that is independent from the length of the current word.

Enumeration algorithms with push-pop edit scripts

Let L regular, e.g., $(\epsilon + a)b^*$. GOAL: enumerate L (in a certain sense) with a delay that is independent from the length of the current word.

Enumeration algorithms with push-pop edit scripts

Let L regular, e.g., $(\epsilon + a)b^*$. GOAL: enumerate L (in a certain sense) with a delay that is independent from the length of the current word. Example of a push-pop program for this language:

```
int main{
  output();
  while (true) {
    pushR(b); output();
    pushL(a); output();
    popL();
  }
}
```

The current word is maintained on a doubly-ended queue

Enumeration algorithms with push-pop edit scripts

Let L regular, e.g., $(\epsilon + a)b^*$. GOAL: enumerate L (in a certain sense) with a delay that is independent from the length of the current word. Example of a push-pop program for this language:

```
int main{
  output();
  while (true) {
    pushR(b); output();
    pushL(a); output();
    popL();
  }
}
```

The current word is maintained on a doubly-ended queue

An edit script is a sequence of push or pop operations executed between two `output()` instructions. This push-pop program enumerates $(\epsilon + a)b^*$ with bounded delay.

Defining the magic t

Result statement

Theorem

For a regular language L, there exist regular $L_1, \ldots, L_t$ such that

$$L = L_1 \sqcup \ldots \sqcup L_t$$

and each L_i is orderable for the push-pop distance. Moreover L cannot be partitioned into less than t orderable languages for the Levenshtein distance.

Result statement

Theorem

For a regular language L, there exist regular $L_1, \ldots, L_t$ such that

$$L = L_1 \sqcup \ldots \sqcup L_t$$

and each L_i is orderable for the push-pop distance. Moreover L cannot be partitioned into less than t orderable languages for the Levenshtein distance.

We will now define this number t and show that it is optimal

Connectivity and compatibility of loopable states

Let $A = (Q, \Sigma, q_0, F, \delta)$ be a DFA for L. For $q \in Q$, define A_q to be A where the initial state and final state is q.

Definition: loopable state

A state $q \in Q$ is loopable if $\mathrm{L}(A_q) \neq \{\epsilon\}$. In other words, when there is a non-empty run that starts and ends at q.

Connectivity and compatibility of loopable states

Let $A = (Q, \Sigma, q_0, F, \delta)$ be a DFA for L. For $q \in Q$, define A_q to be A where the initial state and final state is q.

Definition: loopable state

A state $q \in Q$ is loopable if $\mathrm{L}(A_q) \neq \{\epsilon\}$. In other words, when there is a non-empty run that starts and ends at q.

Definition: connectivity

Two loopable states $q, q' \in Q$ are connected when there is a directed path in A from q to q', or a directed path in A from q' to q

Connectivity and compatibility of loopable states

Let $A = (Q, \Sigma, q_0, F, \delta)$ be a DFA for L. For $q \in Q$, define A_q to be A where the initial state and final state is q.

Definition: loopable state

A state $q \in Q$ is loopable if $\mathrm{L}(A_q) \neq \{\epsilon\}$. In other words, when there is a non-empty run that starts and ends at q.

Definition: connectivity

Two loopable states $q, q' \in Q$ are connected when there is a directed path in A from q to q', or a directed path in A from q' to q

Definition: compatibility

Two loopable states $q, q' \in Q$ are compatible when $\mathrm{L}(A_q) \cap \mathrm{L}(A_{q'}) \neq \{\epsilon\}$.

Interchangeability of loopable states

Definition: interchangeability

Interchangeability is the equivalence relation on loopable states that is defined to be the transitive closure of the union of the connectivity and compatibility relations.

In other words, two loopable states $q, q' \in Q$ are interchangeable if there is a sequence $q = q_0, \dots, q_n = q'$ of loopable states such that for all $0 \leq i < n$, the states q_i and q_{i+1} are either connected or compatible.

Interchangeability of loopable states

Definition: interchangeability

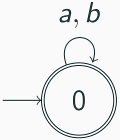
Interchangeability is the equivalence relation on loopable states that is defined to be the transitive closure of the union of the connectivity and compatibility relations.

In other words, two loopable states $q, q' \in Q$ are interchangeable if there is a sequence $q = q_0, \ldots, q_n = q'$ of loopable states such that for all $0 \leq i < n$, the states q_i and q_{i+1} are either connected or compatible.

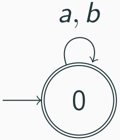
We then define t to be the number of interchangeable classes

Some examples follow

Example: $(a + b)^*$

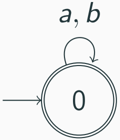


Example: $(a + b)^*$



- Loopable states: 0

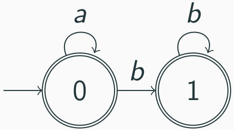
Example: $(a + b)^*$



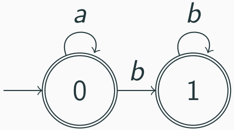
- Loopable states: 0

$\implies$ $t = 1$

Example: $a^* b^*$

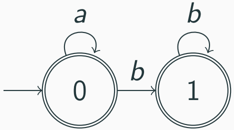


Example: $a^* b^*$



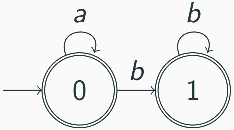
- Loopable states: 0 and 1

Example: $a^* b^*$



- Loopable states: 0 and 1
- 0 and 1 are connected, hence interchangeable

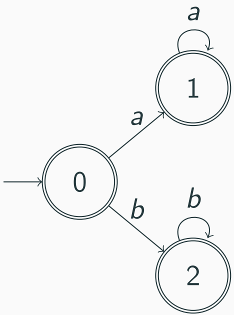
Example: $a^* b^*$



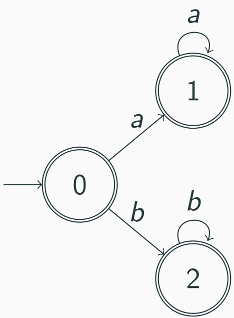
- Loopable states: 0 and 1
- 0 and 1 are connected, hence interchangeable

$\Longrightarrow \quad t = 1$

Example: $a^* + b^*$

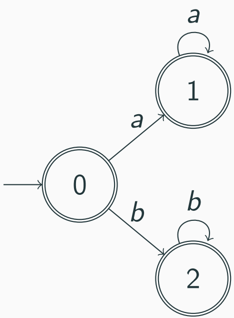


Example: $a^* + b^*$



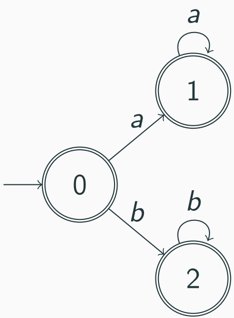
- Loopable states: 1 and 2

Example: $a^* + b^*$



- Loopable states: 1 and 2
- 1 and 2 are neither connected, nor compatible, so they are not interchangeable

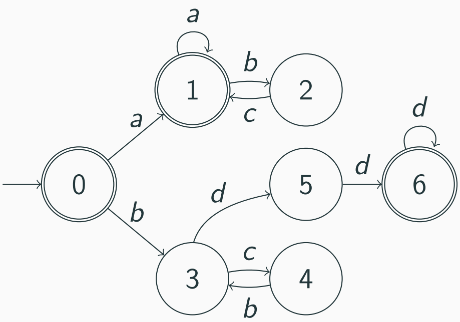
Example: $a^* + b^*$



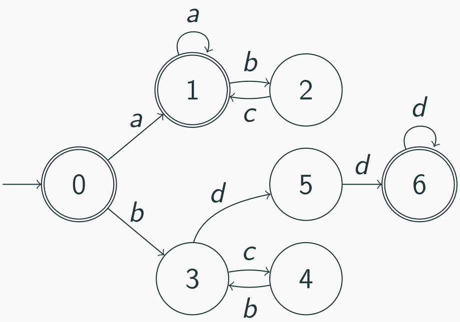
- Loopable states: 1 and 2
- 1 and 2 are neither connected, nor compatible, so they are not interchangeable

$\Longrightarrow$ $t = 2$

Example: compatibility

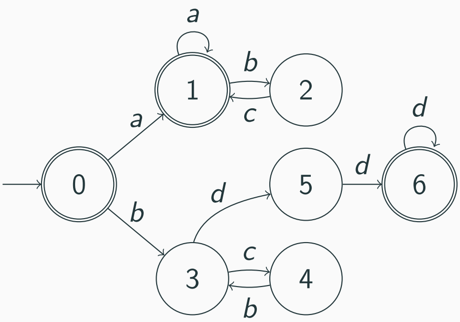


Example: compatibility



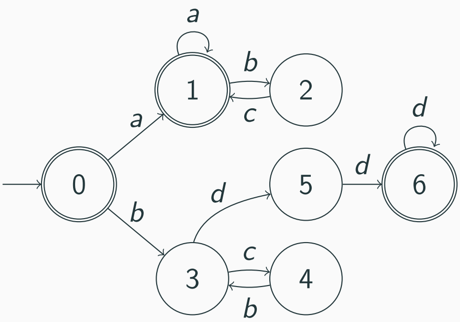
- Loopable states: $1, 2, 3, 4$ and 6

Example: compatibility



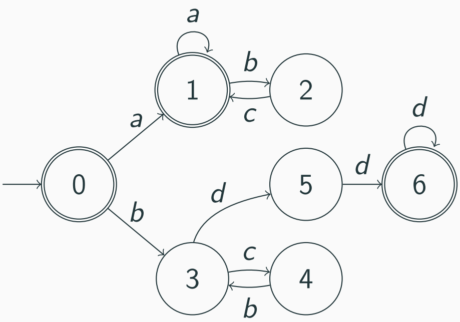
- Loopable states: $1, 2, 3, 4$ and 6
- 1 and 2 are connected hence interchangeable

Example: compatibility



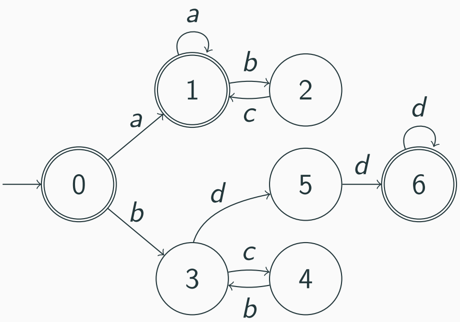
- Loopable states: $1, 2, 3, 4$ and 6
- 1 and 2 are connected hence interchangeable
- 4, 3 and 6 are connected hence interchangeable

Example: compatibility



- Loopable states: $1, 2, 3, 4$ and 6
- 1 and 2 are connected hence interchangeable
- 4, 3 and 6 are connected hence interchangeable
- 1 and 4 are compatible (with the word bc), hence interchangeable

Example: compatibility



- Loopable states: $1, 2, 3, 4$ and 6
- 1 and 2 are connected hence interchangeable
- 4, 3 and 6 are connected hence interchangeable
- 1 and 4 are compatible (with the word bc), hence interchangeable

$\Longrightarrow$ $t = 1$

Proof sketch of the upper bound

Upper bound: existence of an ordering

Theorem

Let A be a DFA A and t its magic number. We can partition $\mathrm{L}(A)$ into

$$L = L_1 \sqcup \ldots \sqcup L_t$$

each L_i is orderable for the Levenshtein distance.

Upper bound: existence of an ordering

Theorem

Let A be a DFA A and t its magic number. We can partition $\mathrm{L}(A)$ into

$$L = L_1 \sqcup \ldots \sqcup L_t$$

each L_i is orderable for the Levenshtein distance.

Enough to show:

Upper bound: existence

Let A be a DFA that has only one class of interchangeable loopable states ($t = 1$). Then $\mathrm{L}(A)$ is orderable for the push-pop distance.

Upper bound: existence of an ordering

Theorem

Let A be a DFA A and t its magic number. We can partition $\mathrm{L}(A)$ into

$$L = L_1 \sqcup \ldots \sqcup L_t$$

each L_i is orderable for the Levenshtein distance.

Enough to show:

Upper bound: existence

Let A be a DFA that has only one class of interchangeable loopable states ($t = 1$). Then $\mathrm{L}(A)$ is orderable for the push-pop distance.

Let δ_{pp} denote the push-pop distance on Σ^*

The graph $G_{L,d}$

Definition

Let L a language and $d \in \mathbb{N}$. Define the graph $G_{L,d}$ whose nodes are words of L and where two words are connected by an edge if they are at push-pop distance $\leq d$.

The graph $G_{L,d}$

Definition

Let L a language and $d \in \mathbb{N}$. Define the graph $G_{L,d}$ whose nodes are words of L and where two words are connected by an edge if they are at push-pop distance $\leq d$.

- Note: if L is orderable with distance d, then $G_{L,d}$ is connex.

The graph $G_{L,d}$

Definition

Let L a language and $d \in \mathbb{N}$. Define the graph $G_{L,d}$ whose nodes are words of L and where two words are connected by an edge if they are at push-pop distance $\leq d$.

- Note: if L is orderable with distance d, then $G_{L,d}$ is connex.

→ the converse is not true! ($G_{(a^*+b^*),1}$ is connex but $a^* + b^*$ is not even orderable)

The graph $G_{L,d}$

Definition

Let L a language and $d \in \mathbb{N}$. Define the graph $G_{L,d}$ whose nodes are words of L and where two words are connected by an edge if they are at push-pop distance $\leq d$.

- Note: if L is orderable with distance d, then $G_{L,d}$ is connex.

→ the converse is not true! ($G_{(a^*+b^*),1}$ is connex but $a^* + b^*$ is not even orderable)

- We show a kind of converse for finite languages in the next slide

$G_{L,d}$ connex implies orderability with distance $3d$ for finite languages

Proposition

For any finite language L, if $G_{L,d}$ is connex then L is orderable with distance $3d$.

$G_{L,d}$ connex implies orderability with distance $3d$ for finite languages

Proposition

For any finite language L, if $G_{L,d}$ is connex then L is orderable with distance $3d$.

Proof: take a spanning tree T of $G_{L,d}$. Apply `visit_even` to the root of T:

```
void visit_even(node n){
  enumerate(n);
  for (child ch of n)
      visit_odd(ch);
}
void visit_odd(node n){
  for (child ch of n)
      visit_even(ch);
  enumerate(n);
}
```

This yields an ordering of the nodes of $G_{L,d}$ where consecutive nodes are at distance at most 3.

Hence the corresponding words are at distance $\leq 3d$ for δ_{pp}.

Using this for infinite languages

Definition

For L a language and $i, \ell \in \mathbb{N}$, define the *i-th ℓ-stratum* of L as

$$S_i = \{w \in L \mid (i-1)\ell \leq |w| < i\ell\}$$

Using this for infinite languages

Definition

For L a language and $i, \ell \in \mathbb{N}$, define the *i-th ℓ-stratum* of L as

$$S_i = \{w \in L \mid (i-1)\ell \leq |w| < i\ell\}$$

We can show (technical):

Proposition

Let $L = \mathrm{L}(A)$ with A having only one interchangeable class of loopable states ($t = 1$). Letting $\ell = 8|A|^2$ and $d = 16|A|^2$, the graph $G_{S_i,d}$ of any ℓ-stratum is connex.

Using this for infinite languages

Definition

For L a language and $i, \ell \in \mathbb{N}$, define the *i-th ℓ-stratum* of L as

$$S_i = \{w \in L \mid (i-1)\ell \leq |w| < i\ell\}$$

We can show (technical):

Proposition

Let $L = \mathrm{L}(A)$ with A having only one interchangeable class of loopable states ($t = 1$). Letting $\ell = 8|A|^2$ and $d = 16|A|^2$, the graph $G_{S_i,d}$ of any ℓ-stratum is connex.

We conclude by concatenating orderings for $S_1, S_2, \ldots$ obtained with the enumeration technique of the previous slide, with well-chosen starting and ending points.

Conclusion

Main results (Levenshtein and push-pop)

Let L be regular. Then:

- There exists $t \in \mathbb{N}$ and regular languages $L_1, \ldots, L_t$ such that

$$L = L_1 \sqcup \ldots \sqcup L_t$$

  and each L_i is orderable for the push-pop distance

Main results (Levenshtein and push-pop)

Let L be regular. Then:

- There exists $t \in \mathbb{N}$ and regular languages $L_1, \ldots, L_t$ such that

$$L = L_1 \sqcup \ldots \sqcup L_t$$

  and each L_i is orderable for the push-pop distance
- This t is optimal, even for the Levenshtein distance: L cannot be partitioned into less than t orderable languages for the Levenshtein distance.

Main results (Levenshtein and push-pop)

Let L be regular. Then:

- There exists $t \in \mathbb{N}$ and regular languages $L_1, \ldots, L_t$ such that

$$L = L_1 \sqcup \ldots \sqcup L_t$$

  and each L_i is orderable for the push-pop distance
- This t is optimal, even for the Levenshtein distance: L cannot be partitioned into less than t orderable languages for the Levenshtein distance.
  - → This shows that L is orderable for Levenshtein iff it is for push-pop!

Main results (Levenshtein and push-pop)

Let L be regular. Then:

- There exists $t \in \mathbb{N}$ and regular languages $L_1, \ldots, L_t$ such that

$$L = L_1 \sqcup \ldots \sqcup L_t$$

  and each L_i is orderable for the push-pop distance
- This t is optimal, even for the Levenshtein distance: L cannot be partitioned into less than t orderable languages for the Levenshtein distance.
  - → This shows that L is orderable for Levenshtein iff it is for push-pop!
- When L is orderable for push-pop then, in a suitable pointer machine model, we have an algorithm that outputs push-pop edit scripts to enumerate L, with bounded delay (i.e., independent from the current word length)

Other results

Other results:

- It is *NP*-hard, given a DFA A such that $\mathrm{L}(A)$ is orderable (for Levenshtein or push-pop), to determine the minimal d such that $\mathrm{L}(A)$ is orderable for distance d.
- A regular language is partitionable into finitely many orderable languages for the push-pop-right distance if and only if it is slender.
  - Further, the optimal number of languages can also be computed from the automaton
  - We can also enumerate in bounded delay

Future work

Open questions and future work:

- Make the delay polynomial in $|A|$? (currently it is exp)
- What about enumeration in radix order? in lexicographic order?
- What about the push-left pop-right distance? the padded Hamming distance?
- What about regular tree languages?
- Other uses of the enumeration model?
- Implementation and real-life use-cases?

Future work

Open questions and future work:

- Make the delay polynomial in $|A|$? (currently it is exp)
- What about enumeration in radix order? in lexicographic order?
- What about the push-left pop-right distance? the padded Hamming distance?
- What about regular tree languages?
- Other uses of the enumeration model?
- Implementation and real-life use-cases?

Thanks for your attention!