Enumerating Regular Tree Languages Under Bounded Delay

Antoine Amarilli Arthur Lombardo Mikaël Monet

University of Lille

ENS Paris

5 november 2026

Bounded Delay Enumeration

- It is possible to have a bounded delay enumeration if there is a bound on the size of the outputs.

Bounded Delay Enumeration

- It is possible to have a bounded delay enumeration if there is a bound on the size of the outputs.

- For outputs with unbounded size, one proposition is to not charge the time needed to write the current output.

Bounded Delay Enumeration

- It is possible to have a bounded delay enumeration if there is a bound on the size of the outputs.

- For outputs with unbounded size, one proposition is to not charge the time needed to write the current output.

- We will consider a different framework where we get the next solution by applying a constant number of edits to the previous one.

Bounded Delay Enumeration of Word Languages

- An **alphabet** is a finite set of symbols Σ.
- A **word** is a finite sequence over Σ.

Bounded Delay Enumeration of Word Languages

- An **alphabet** is a finite set of symbols Σ.
- A **word** is a finite sequence over Σ.
- A **regular word language** is a set of words recognized by an NFA.

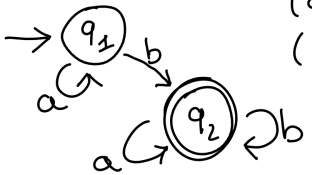
$A = (Q, \Sigma, I, F, \Delta)$

with $Q = \{q_1, q_2\}$,

$\Sigma = \{a, b\}$,

$I = \{q_1\}$, $F = \{q_2\}$,

$\Delta = \{(a, q_1, q_1), (a, q_2, q_2), (b, *, q_2)\}$



Bounded Delay Enumeration of Word Languages

- An **alphabet** is a finite set of symbols Σ.
- A **word** is a finite sequence over Σ.
- A **regular word language** is a set of words recognized by an NFA.
- **Known result (Amarilli & Monet):** given an NFA A, one can decide in polynomial time whether $L(A)$ admits a bounded-delay enumeration with push-pop edits.

$A = (Q, \Sigma, I, F, \Delta)$

with $Q = \{q_1, q_2\}$,

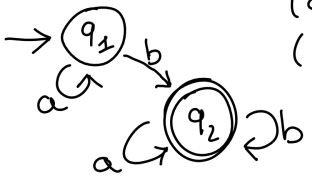
$\Sigma = \{a, b\}$,

$I = \{q_1\}$, $F = \{q_2\}$,

$\Delta = \{(a, q_1, q_1),$

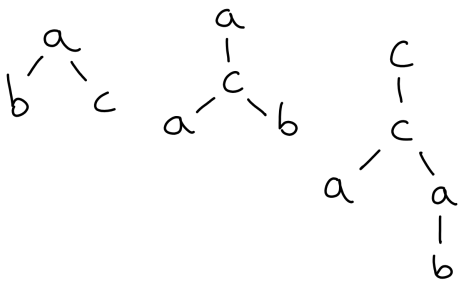
$(a, q_2, q_2),$

$(b, *, q_2)\}$



Regular Tree Languages and NFTAs

- **Trees:** Branching structure over an alphabet.
(Finite, rooted, ordered)



Regular Tree Languages and NFTAs

- **Trees:** Branching structure over an alphabet.
(Finite, rooted, ordered)

- **NFTA:** Bottom-up tree automaton (recognizes a regular tree language).

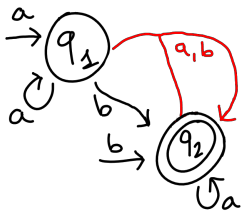
$$A = (Q, \Sigma, F, \Delta)$$

with $Q = \{q_1, q_2\}$,

$\Sigma = \{a, b\}$,

$F = \{q_2\}$,

$\Delta = \{(a, q_1),$
$(a, q_1, q_1),$
$(a, q_1, q_2, q_2),$
$\dots\}$



Regular Tree Languages and NFTAs

- **Trees:** Branching structure over an alphabet.
(Finite, rooted, ordered)

- **NFTA:** Bottom-up tree automaton (recognizes a regular tree language).

- **Goal:** Enumerate a given regular tree language in bounded delay via edits

$$A = (Q, \Sigma, F, \Delta)$$

with $Q = \{q_1, q_2\}$,

$\Sigma = \{a, b\}$,

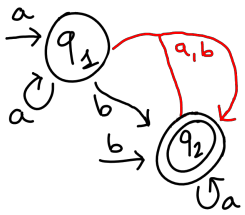
$F = \{q_2\}$,

$\Delta = \{(a, q_1),$

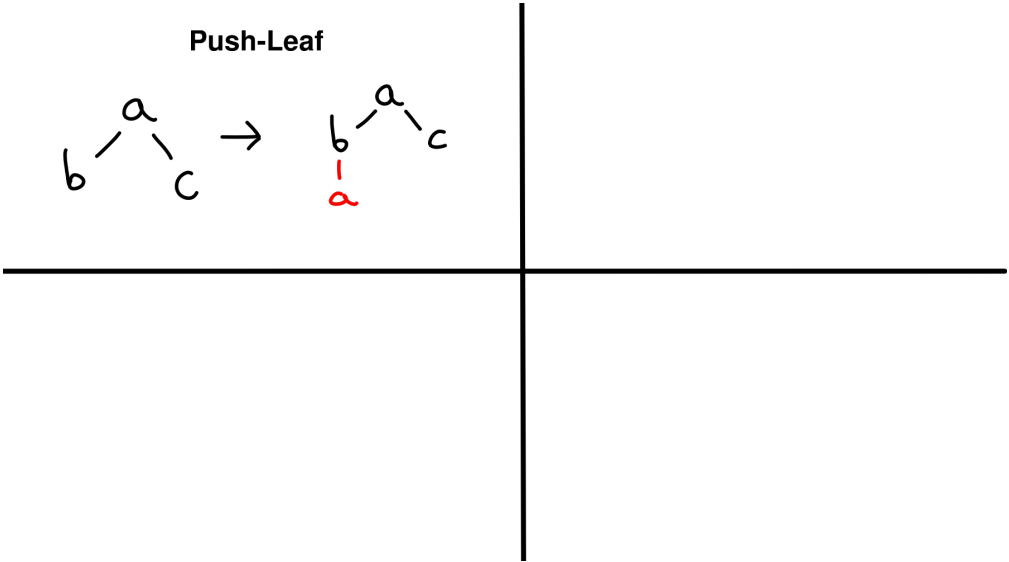
$(a, q_1, q_1),$

$(a, q_1, q_2, q_2),$

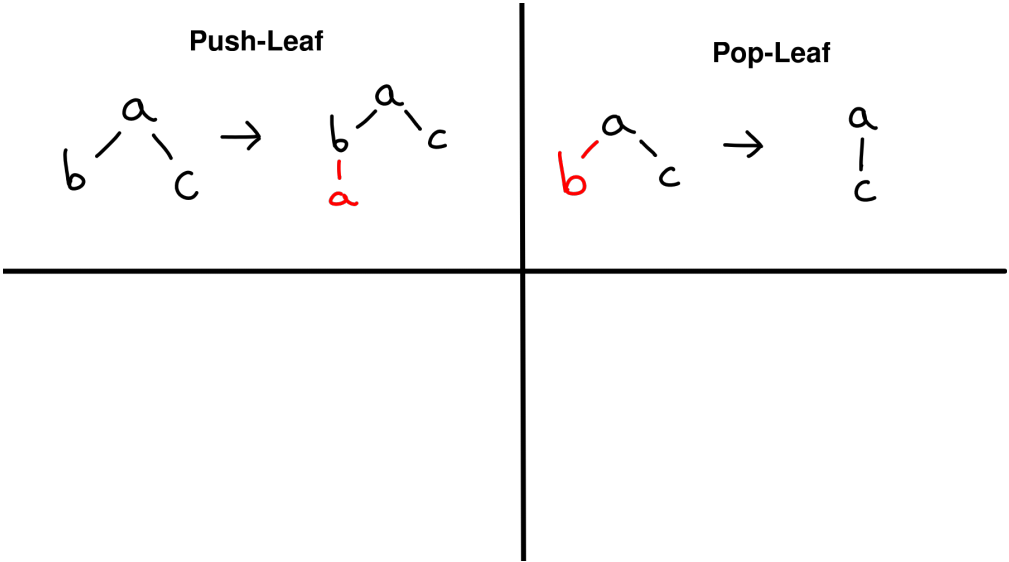
$\dots\}$



Push-Pop Tree Edits

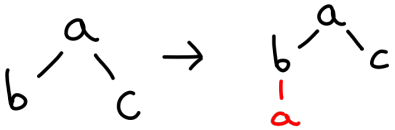


Push-Pop Tree Edits

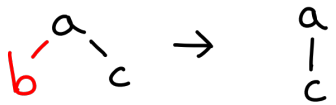


Push-Pop Tree Edits

Push-Leaf



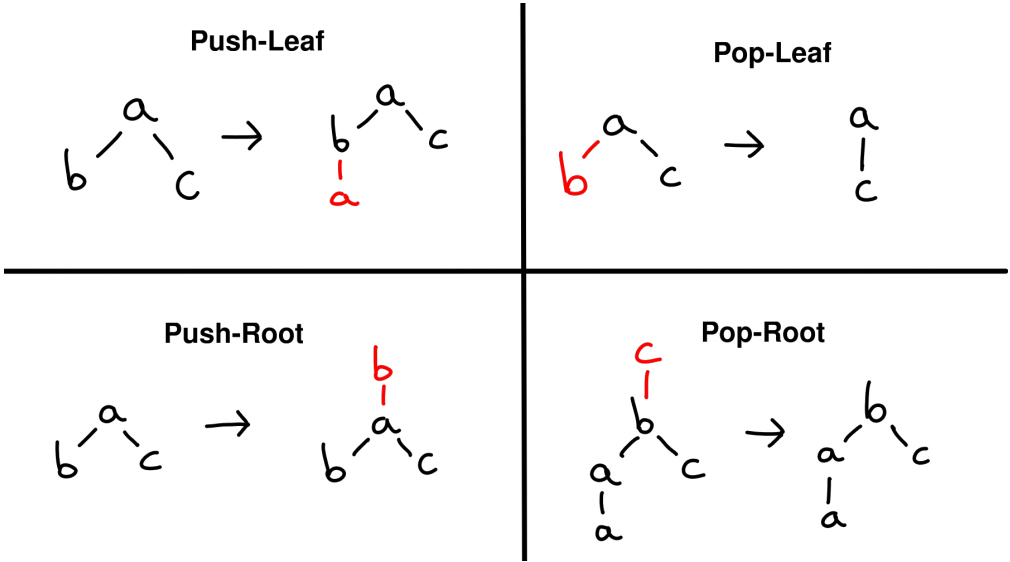
Pop-Leaf



Push-Root



Push-Pop Tree Edits



Orderability vs Enumeration

Push-pop distance between t and t': Minimum amount of push-pop edits required to transform t into t'.

Orderability vs Enumeration

Push-pop distance between t and t': Minimum amount of push-pop edits required to transform t into t'.

For $d \in \mathbb{N}$, a ***d*-order** of a language L is a sequence S of trees such that:

- Every tree of L appears exactly once in S.
- The distance between two consecutive trees is at most d.

Orderability vs Enumeration

Push-pop distance between t and t': Minimum amount of push-pop edits required to transform t into t'.

For $d \in \mathbb{N}$, a ***d*-order** of a language L is a sequence S of trees such that:

- Every tree of L appears exactly once in S.
- The distance between two consecutive trees is at most d.

A language L is ***d*-orderable** if it has a d-order.

Orderability vs Enumeration

Push-pop distance between t and t': Minimum amount of push-pop edits required to transform t into t'.

For $d \in \mathbb{N}$, a ***d*-order** of a language L is a sequence S of trees such that:

- Every tree of L appears exactly once in S.
- The distance between two consecutive trees is at most d.

A language L is ***d*-orderable** if it has a d-order.

Orderability is **necessary** for bounded-delay enumeration by edits.

Orderability vs Enumeration

Push-pop distance between t and t': Minimum amount of push-pop edits required to transform t into t'.

For $d \in \mathbb{N}$, a ***d*-order** of a language L is a sequence S of trees such that:

- Every tree of L appears exactly once in S.
- The distance between two consecutive trees is at most d.

A language L is ***d*-orderable** if it has a d-order.

Orderability is **necessary** for bounded-delay enumeration by edits.

Amarilli and Monet showed that we can efficiently decide if a regular word language is orderable. They also provided an algorithm that effectively enumerates an orderable language.

Main Result and Examples

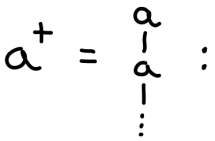
Theorem 1

Given an NFTA A, we can decide in **polynomial time** whether $L(A)$ is orderable.

Main Result and Examples

Theorem 1

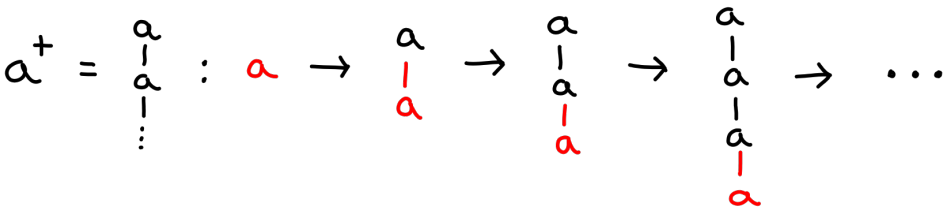
Given an NFTA A, we can decide in **polynomial time** whether $L(A)$ is orderable.



Main Result and Examples

Theorem 1

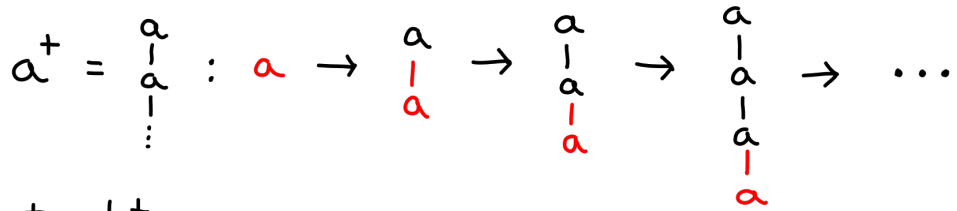
Given an NFTA A, we can decide in **polynomial time** whether $L(A)$ is orderable.



Main Result and Examples

Theorem 1

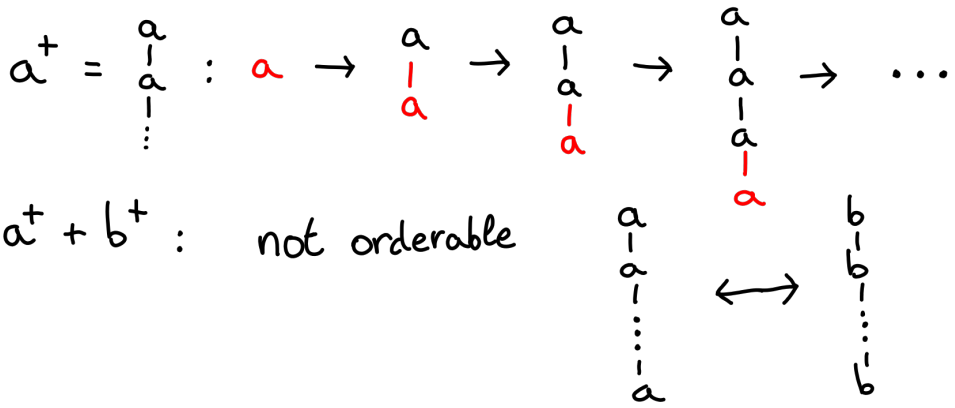
Given an NFTA A, we can decide in **polynomial time** whether $L(A)$ is orderable.



Main Result and Examples

Theorem 1

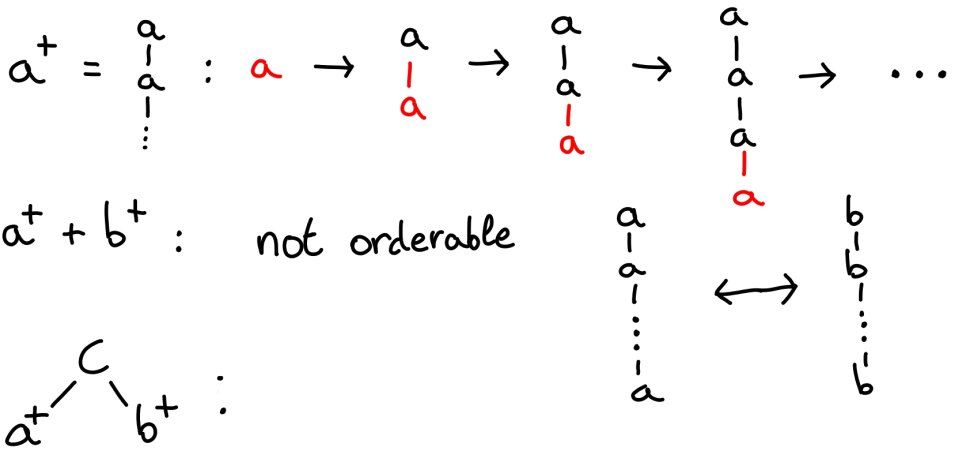
Given an NFTA A, we can decide in **polynomial time** whether $L(A)$ is orderable.



Main Result and Examples

Theorem 1

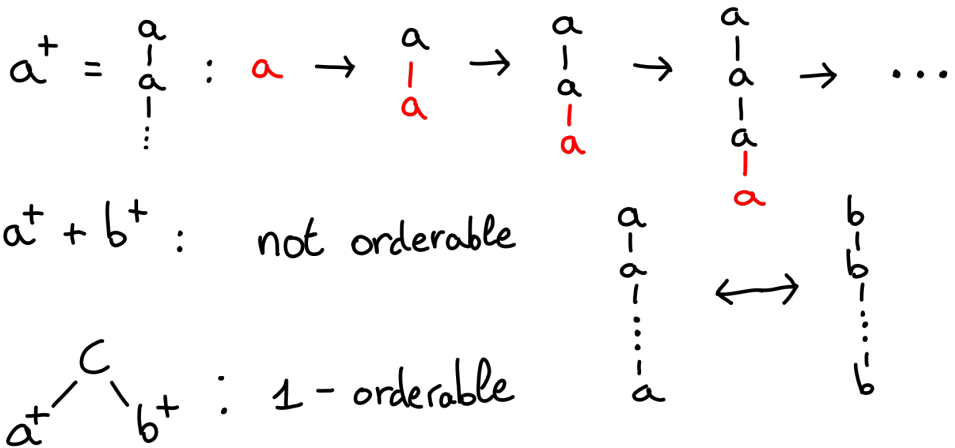
Given an NFTA A, we can decide in **polynomial time** whether $L(A)$ is orderable.



Main Result and Examples

Theorem 1

Given an NFTA A, we can decide in **polynomial time** whether $L(A)$ is orderable.



The d-Graph of a Language

The ***d*-graph** of a language L for some $d \in \mathbb{N}$ is the graph with vertex set L and an edge between $u, v \in L$ if the distance between u and v is at most d.

The d-Graph of a Language

The ***d*-graph** of a language L for some $d \in \mathbb{N}$ is the graph with vertex set L and an edge between $u, v \in L$ if the distance between u and v is at most d.

1-Graph of a^+:



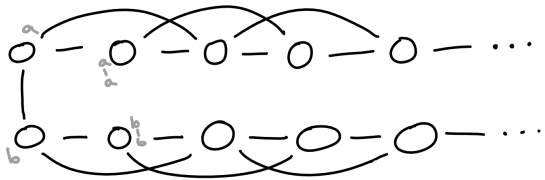
The d-Graph of a Language

The ***d*-graph** of a language L for some $d \in \mathbb{N}$ is the graph with vertex set L and an edge between $u, v \in L$ if the distance between u and v is at most d.

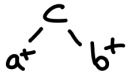
1-Graph of a^+:



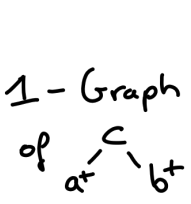
2-Graph of $a^+ + b^+$:

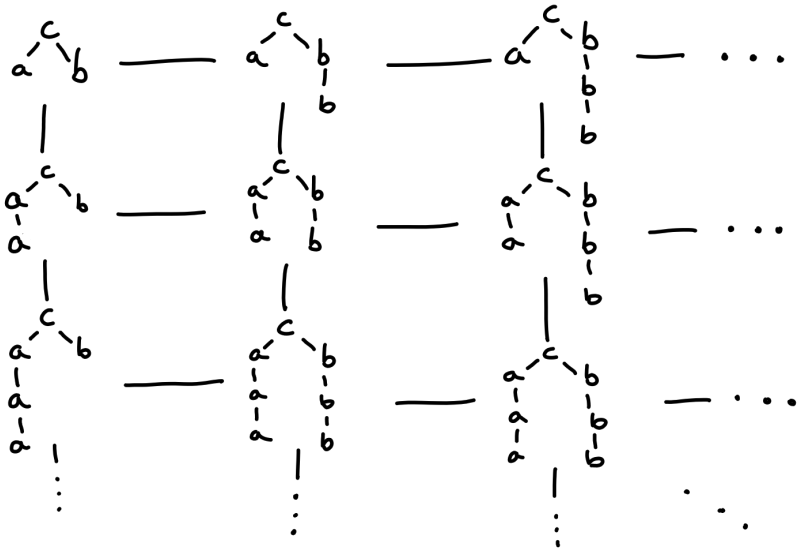


Another Example



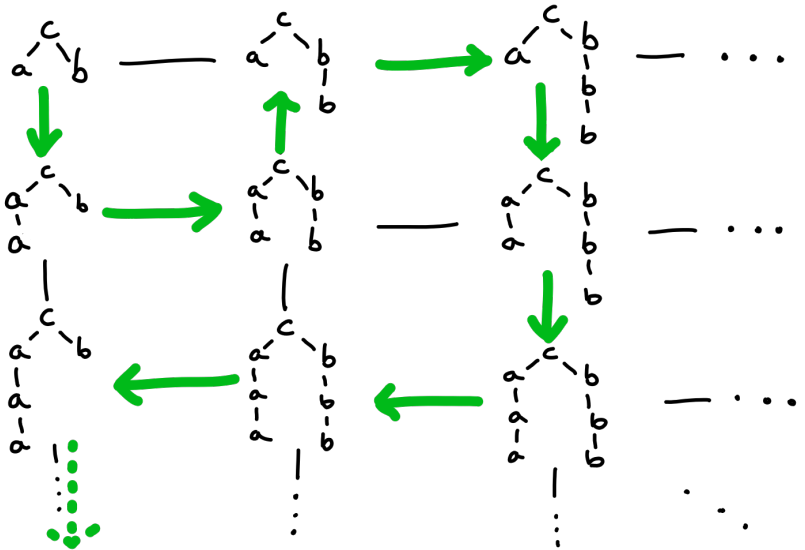
Another Example





Another Example

1 - Graph of $c(a^+, b^+)$



Connectivity (almost) implies orderability

Finite Graphs:

Connectivity (almost) implies orderability

Finite Graphs:

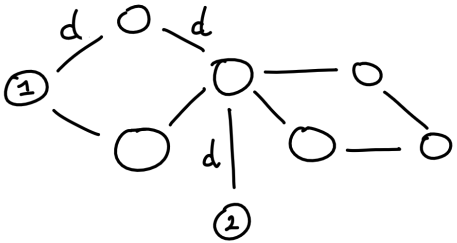
Known Theorem [Karaganis]:
The cube of any connected graph has a Hamiltonian cycle.

Connectivity (almost) implies orderability

Finite Graphs:

Known Theorem [Karaganis]:
The cube of any connected graph has a Hamiltonian cycle.

Corollary: Any language with a finite and connected d-graph is orderable.



Connectivity (almost) implies orderability

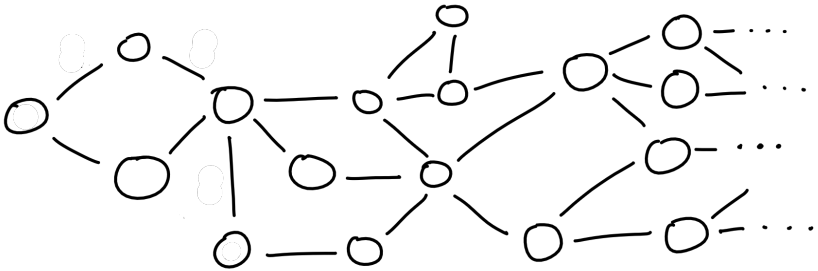
Finite Graphs:

Known Theorem [Karaganis]: The cube of any connected graph has a Hamiltonian cycle.

Corollary: Any language with a finite and connected d-graph is orderable.

Infinite Graphs:

Language graphs are generally infinite.



Connectivity (almost) implies orderability

Finite Graphs:

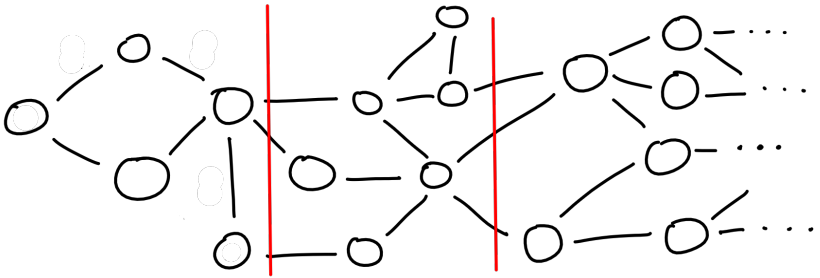
Known Theorem [Karaganis]: The cube of any connected graph has a Hamiltonian cycle.

Corollary: Any language with a finite and connected d-graph is orderable.

Infinite Graphs:

Language graphs are generally infinite.

Idea: Partition the d-graph into finite **strata** of trees of a certain height range.

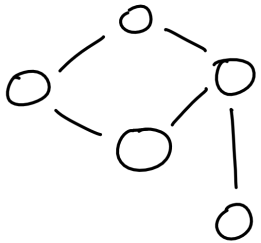


Connectivity (almost) implies orderability

Finite Graphs:

Known Theorem [Karaganis]: The cube of any connected graph has a Hamiltonian cycle.

Corollary: Any language with a finite and connected d-graph is orderable.



Infinite Graphs:

Language graphs are generally infinite.

Idea: Partition the d-graph into finite **strata** of trees of a certain height range.

If every stratum is connected, then the language is orderable.

Connectivity (almost) implies orderability

Finite Graphs:

Known Theorem [Karaganis]: The cube of any connected graph has a Hamiltonian cycle.

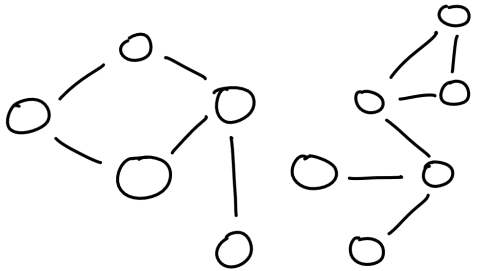
Corollary: Any language with a finite and connected d-graph is orderable.

Infinite Graphs:

Language graphs are generally infinite.

Idea: Partition the d-graph into finite **strata** of trees of a certain height range.

If every stratum is connected, then the language is orderable.



Connectivity (almost) implies orderability

Finite Graphs:

Known Theorem [Karaganis]: The cube of any connected graph has a Hamiltonian cycle.

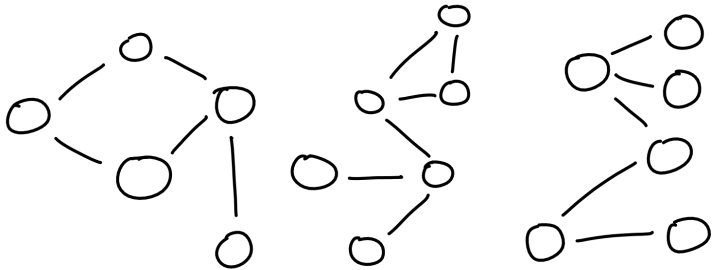
Corollary: Any language with a finite and connected d-graph is orderable.

Infinite Graphs:

Language graphs are generally infinite.

Idea: Partition the d-graph into finite **strata** of trees of a certain height range.

If every stratum is connected, then the language is orderable.



Actual Main Result

Theorem 2

Every tree language L can be partitioned into $L_1, \ldots, L_p$ such that each L_i is push-pop orderable.

Actual Main Result

Theorem 2

Every tree language L can be partitioned into $L_1, \ldots, L_p$ such that each L_i is push-pop orderable.

(Example: $a^+ + b^+$)

Actual Main Result

Theorem 2

Every tree language L can be partitioned into $L_1, \ldots, L_p$ such that each L_i is push-pop orderable.

(Example: $a^+ + b^+$)

We can find the smallest $p \in \mathbb{N}$ such that this is true in polynomial time.

Actual Main Result

Theorem 2

Every tree language L can be partitioned into $L_1, \ldots, L_p$ such that each L_i is push-pop orderable.

(Example: $a^+ + b^+$)

We can find the smallest $p \in \mathbb{N}$ such that this is true in polynomial time.

Even if using the Levenshtein distance, the minimal size of the partition is the same.

Conclusion

Open Questions:

Conclusion

Open Questions:

- **Effective Enumeration:** Can we explicitly compute the sequence in bounded delay?

Conclusion

Open Questions:

- **Effective Enumeration:** Can we explicitly compute the sequence in bounded delay?
- **Distances:** Which languages are orderable for other distances (e.g., Push-Pop-Leaf)?

Conclusion

Open Questions:

- **Effective Enumeration:** Can we explicitly compute the sequence in bounded delay?
- **Distances:** Which languages are orderable for other distances (e.g., Push-Pop-Leaf)?
- **Grammars:** Can these results be extended to context-free tree grammars?