Encoding Co-Lex Orders of Finite State-Automata in Linear Space

36th Annual Symposium on Combinatorial Pattern Matching

R. Becker[1], N. Cotumaccio[2], S. Kim[1],
N. Prezza[1], **C. Tosoni**[1]

1. Ca' Foscari University of Venice
2. University of Helsinki

Milan, Bicocca University

17/06/2025

The Burrows-Wheeler transform

The **BWT** is a famous reversible string transformation invented by Burrows and Wheeler [1].

1. **Enhance the compressibility** of strings.
2. Allows the implementation of **indexes for pattern matching**.

$$banana\$ \longrightarrow annb\$aa$$

1. *Burrows M., Wheeler D.: A block-sorting lossless data compression algorithm. SRS Research Report (1994)*

Extending the BWT to NFAs

The BWT has been extended to **nondeterministic finite automata** (NFAs) using **Wheeler orders** [2].

Wheeler order

Total order $\leq$ of nodes **consistent with the strings** reaching the states.

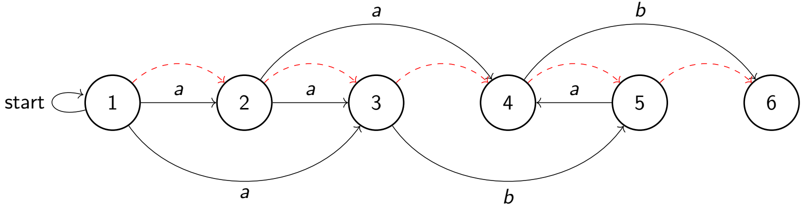
2. *Gagie et al.: Wheeler graphs: A framework for BWT-based data structures. Theor. Comput. Sci. (2017)*

Extending the BWT to NFAs

The BWT has been extended to **nondeterministic finite automata** (NFAs) using **Wheeler orders** [2].

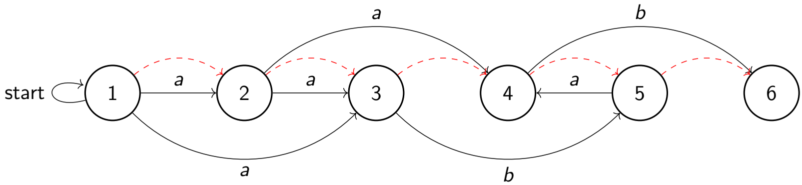
Wheeler order

Total order $\leq$ of nodes **consistent with the strings** reaching the states.



2. *Gagie et al.: Wheeler graphs: A framework for BWT-based data structures. Theor. Comput. Sci. (2017)*

Extending the BWT to NFAs



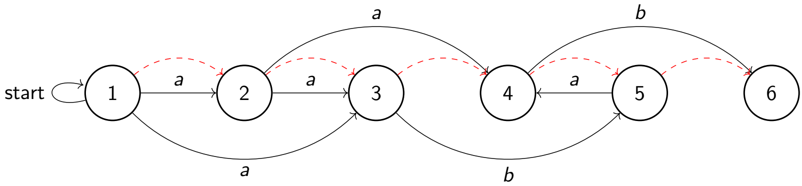
What does "**consistent**" mean?

- **Initial state** is the **first node** of $\leq$ (node u_1)

Strings reaching u_3 and u_4 $\rightarrow$ $I_{u_3} = \{a, aa\}$ and $I_{u_4} = \{aa, aba, aaba\}$

- $u_3 \leq u_4 \implies$ excluding the strings in common $\{aa\}$, strings in I_{u_3} **are smaller than those in** I_{u_4}.

Extending the BWT to NFAs



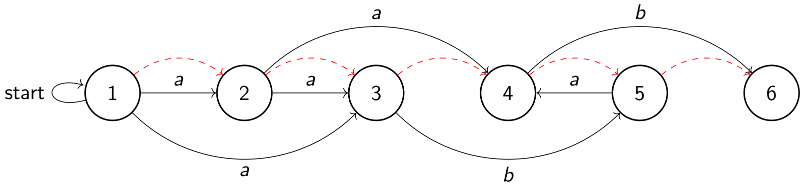
What does “**consistent**” mean?

- **Initial state** is the **first node** of $\leq$ (node u_1)

Strings reaching u_3 and u_4 $\rightarrow$ $I_{u_3} = \{a, aa\}$ and $I_{u_4} = \{aa, aba, aaba\}$

- $u_3 \leq u_4 \implies$ excluding the strings in common $\{aa\}$, strings in I_{u_3} **are smaller than those in** I_{u_4}.

Extending the BWT to NFAs



What does "**consistent**" mean?

- **Initial state** is the **first node** of $\leq$ (node u_1)

Strings reaching u_3 and u_4 $\rightarrow$ $I_{u_3} = \{a, aa\}$ and $I_{u_4} = \{aa, aba, aaba\}$

- $u_3 \leq u_4 \implies$ excluding the strings in common $\{aa\}$, strings in I_{u_3} **are smaller than those in** I_{u_4}.

Wheeler orders: pros and cons

Pros

- NFAs admitting a Wheeler order can be **efficiently compressed and indexed**.

Cons

- Most NFAs do not admit a Wheeler order.
- Recognizing if this is the case is an **NP-complete problem** [3]!

3. *Gibney, Thankachan: On the Hardness and Inapproximability of Recognizing Wheeler Graphs. ESA. (2019)*

Wheeler orders: pros and cons

Pros

- NFAs admitting a Wheeler order can be **efficiently compressed and indexed**.

Cons

- Most NFAs do not admit a Wheeler order.
- Recognizing if this is the case is an **NP-complete problem** [3]!

3. *Gibney, Thankachan: On the Hardness and Inapproximability of Recognizing Wheeler Graphs. ESA. (2019)*

Wheeler orders: pros and cons

Pros

- NFAs admitting a Wheeler order can be **efficiently compressed and indexed**.

Cons

- Most NFAs do not admit a Wheeler order.
- Recognizing if this is the case is an **NP-complete problem** [3]!

3. *Gibney, Thankachan: On the Hardness and Inapproximability of Recognizing Wheeler Graphs. ESA. (2019)*

CFS orders

To address these issues → **CFS orders**! [4]

Coarsest Forward-Stable co-lex (CFS) orders

A CFS order $\leq_{FS}$ is a **partial preorder** on an NFA's states **consistent with the strings reaching them**.

State-Of-The-Art for **indexing** and **compressing** NFAs!

- **Exists** and is **unique** for each NFA.
- **Polynomial time** to compute it.

4. *Becker et al.: Indexing Finite-State Automata Using Forward-Stable Partitions. SPIRE (2024)*

CFS orders

To address these issues $\rightarrow$ **CFS orders**! [4]

Coarsest Forward-Stable co-lex (CFS) orders

A CFS order $\leq_{FS}$ is a **partial preorder** on an NFA's states **consistent with the strings reaching them**.

State-Of-The-Art for **indexing** and **compressing** NFAs!

- **Exists** and is **unique** for each NFA.
- **Polynomial time** to compute it.

4. *Becker et al.: Indexing Finite-State Automata Using Forward-Stable Partitions. SPIRE (2024)*

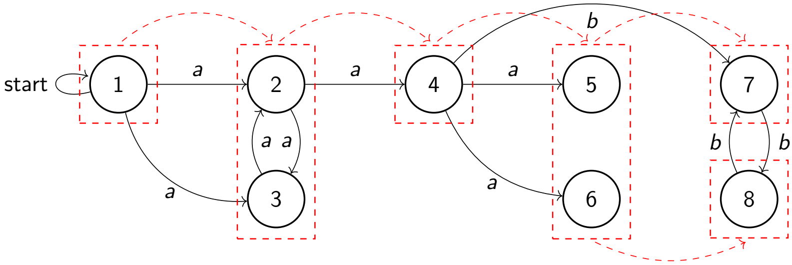
CFS orders

Definition of CFS order $\leq_{FS}$

1) Compute the **quotient NFA** defined by **coarsest forward stable partition**.

2) For every **transitions** $\mathbf{v} \xrightarrow{\mathbf{a}} \mathbf{u}$, $\mathbf{v}' \xrightarrow{\mathbf{a}'} \mathbf{u}'$ in the **quotient NFA**:

- $\mathbf{a} < \mathbf{a}' \implies \mathbf{u} < \mathbf{u}'$
- $(\mathbf{a} = \mathbf{a}') \wedge (\mathbf{v} < \mathbf{v}') \implies \mathbf{u} \leq \mathbf{u}'$



Complexity of CFS Orders

The CFS order can be computed in **polynomial time**, more precisely in $\mathbf{O(m^2)}$ time, with **m** being the number of **transitions in the NFA**.

- Unfeasible for **big data applications** (i.e., to index **pagenome graphs**)

Main open problem: devise a (near) **linear time algorithm** :D

Complexity of CFS Orders for DFAs

For the special case of **DFAs**, a $\mathbf{O(m \log n)}$ time algorithm to compute CFS orders **is already known**. [5]

This algorithm was preceded by a $\mathbf{O(n)}$ **linear space representation** to encode them. [6]

$m \rightarrow$ Number of transitions.

$n \rightarrow$ Number of states.

5. *Becker et al.: Sorting Finite Automata via Partition Refinement. ESA. (2023)*

6. *Kim et al.: Faster Prefix-Sorting Algorithms for Deterministic Finite Automata. CPM. (2023)*

Complexity of CFS Orders for DFAs

For the special case of **DFAs**, a $\mathbf{O(m \log n)}$ time algorithm to compute CFS orders **is already known**. [5]

This algorithm was preceded by a $\mathbf{O(n)}$ **linear space representation** to encode them. [6]

$m \rightarrow$ Number of transitions.

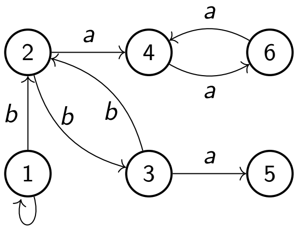
$n \rightarrow$ Number of states.

5. *Becker et al.: Sorting Finite Automata via Partition Refinement. ESA. (2023)*

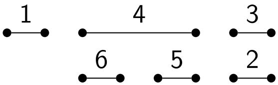
6. *Kim et al.: Faster Prefix-Sorting Algorithms for Deterministic Finite Automata. CPM. (2023)*

Complexity of CFS Orders for DFAs

Indeed, **in DFAs** the CFS order $\leq_{FS}$ can be **encoded using intervals**



Interval order of $\leq_{FS}$



However, this is **not possible in the case of nondeterminism**

Our main result

Theorem

There exists an **$O(n)$ representation** of $\leq_{FS}$

- Arbitrary partial preorders on V ($|V| = n$) require **$\Omega(n^2)$ bits to be represented** (binary matrix)
- This special class can be encoded in just linear space

Step towards a subquadratic algorithm?

Our main result

Theorem

There exists an **$\mathbf{O(n)}$ representation** of $\leq_{FS}$

- Arbitrary partial preorders on V ($|V| = n$) require **$\mathbf{\Omega(n^2)}$ bits to be represented** (binary matrix)
- This special class can be encoded in just linear space

Step towards a subquadratic algorithm?

Our main result

Theorem

There exists an **O(n) representation** of $\leq_{FS}$

- Arbitrary partial preorders on V ($|V| = n$) require **$\Omega(n^2)$ bits to be represented** (binary matrix)
- This special class can be encoded in just linear space

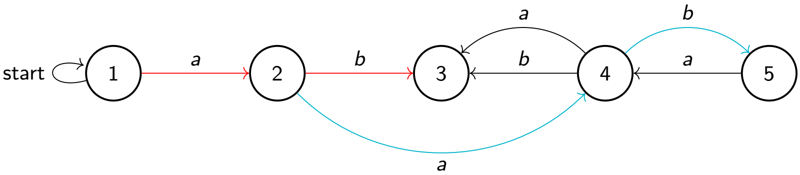
Step towards a subquadratic algorithm?

Characterizing $\leq_{FS}$

Preceding pairs [7]

Let $\mathcal{A}$ be an NFA and Q its set of states.

Consider $(w, z), (u, v) \in Q \times Q$, we say that (w, z) **precedes** (u, v), denoted $(\mathbf{w}, \mathbf{z}) \rightrightarrows (\mathbf{u}, \mathbf{v})$, if there exist α such that $\mathbf{w} \xrightarrow{\alpha} \mathbf{u}$ and $\mathbf{z} \xrightarrow{\alpha} \mathbf{v}$



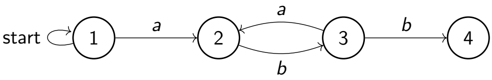
Since $1 \xrightarrow{a} 2 \xrightarrow{b} 3$ and $2 \xrightarrow{a} 4 \xrightarrow{b} 5$, we have $(\mathbf{1}, \mathbf{2}) \rightrightarrows (\mathbf{3}, \mathbf{5})$.

7. *Cotumaccio N.: Graphs can be succinctly indexed for pattern matching in $O(|E|^2 + |V|^{5/2})$ time. DCC. (2022)*

Characterizing $\leq_{FS}$

input-consistency: Incoming transitions are **labeled with same character**.

We denote with $\lambda(\mathbf{u})$ the label of the incoming edges of u.



Ex. $\lambda(3) = b$

Corollary

For every states $u, v \in Q$;

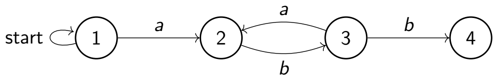
$\mathbf{u} \leq_{\mathbf{FS}} \mathbf{v} \iff$ for each **preceding pair** $(w, z) \rightrightarrows (u, v)$ we have $\lambda(\mathbf{w}) \leq \lambda(\mathbf{z})$.

Ex. $\neg(4 \leq_{FS} 3)$ because $(3, 2) \rightrightarrows (4, 3)$ and $\lambda(3) > \lambda(2)$

Characterizing $\leq_{FS}$

input-consistency: Incoming transitions are **labeled with same character**.

We denote with $\lambda(\mathbf{u})$ the label of the incoming edges of u.



Ex. $\lambda(3) = b$

Corollary

For every states $u, v \in Q$;

$\mathbf{u} \leq_{\mathbf{FS}} \mathbf{v} \iff$ for each **preceding pair** $(w, z) \rightrightarrows (u, v)$ we have $\lambda(\mathbf{w}) \leq \lambda(\mathbf{z})$.

Ex. $\neg(4 \leq_{FS} 3)$ because $(3, 2) \rightrightarrows (4, 3)$ and $\lambda(3) > \lambda(2)$

Overview

Encoding $\leq_{FS}$: $O(n)$ space

- **Linear Extension** $\leq$
- Lexicographically smallest (largest) string **inf** u (**sup** u) and its respective walk **Inf-walk** u (**Sup-walk** u)
- **Inf** (**Sup**) **Conflicts** of u

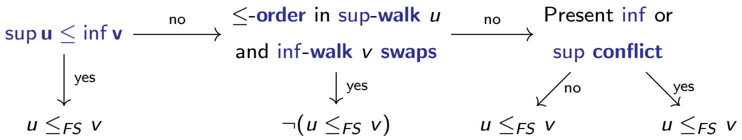
If $u < v$:

Overview

Encoding $\leq_{FS}$: $O(n)$ space

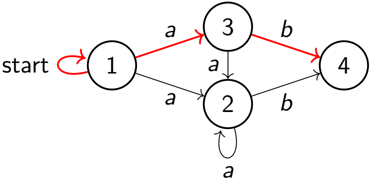
- **Linear Extension** $\leq$
- Lexicographically smallest (largest) string **inf** u (**sup** u) and its respective walk **Inf-walk** u (**Sup-walk** u)
- **Inf** (**Sup**) **Conflicts** of u

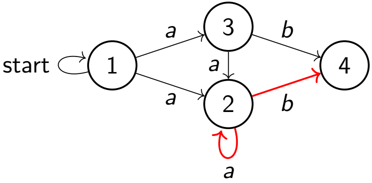
If $u < v$:



Infima/Suprema

The **Infima** (**Suprema**) are the **smallest** (**largest**) **strings reaching the states** from the initial state. **We can**: compute them in $\mathbf{O(m \log n)}$ [5] stores them in $\mathbf{O(n)}$ [6]





$\inf 4 = ba\#\#\#\#\ldots = ba\#^\omega$

an **infimum walk**: $1 \xrightarrow{\#} 1 \xrightarrow{a} 3 \xrightarrow{b} 4$

$\sup 4 = baaaaa\ldots = ba^\omega$

a **supremum walk**: $2 \xrightarrow{a} 2 \xrightarrow{b} 4$

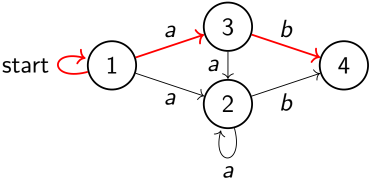
Lemma

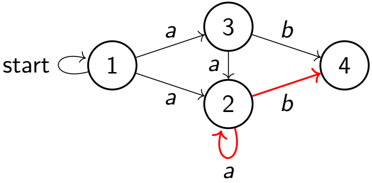
For every two states u, v, $\sup \mathbf{u} \leq \inf \mathbf{v} \implies \mathbf{u} \leq_{\mathbf{FS}} \mathbf{v}$

5. *Sorting Finite Automata via Partition Refinement.* 6. *Faster Prefix-Sorting Algorithms for DFAs.*

Infima/Suprema

The **Infima** (**Suprema**) are the **smallest** (**largest**) **strings reaching the states** from the initial state. **We can**: compute them in $\mathbf{O(m \log n)}$ [5] stores them in $\mathbf{O(n)}$ [6]





$\inf 4 = ba\#\#\#\#\ldots = ba\#^\omega$

an **infimum walk**: $1 \xrightarrow{\#} 1 \xrightarrow{a} 3 \xrightarrow{b} 4$

$\sup 4 = baaaaa\ldots = ba^\omega$

a **supremum walk**: $2 \xrightarrow{a} 2 \xrightarrow{b} 4$

Lemma

For every two states u, v, $\sup \mathbf{u} \leq \inf \mathbf{v} \implies \mathbf{u} \leq_{\mathbf{FS}} \mathbf{v}$

5. *Sorting Finite Automata via Partition Refinement.* 6. *Faster Prefix-Sorting Algorithms for DFAs.*

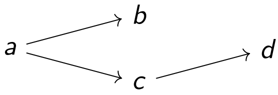
Linear Extension

Second part of our encoding: a linear extension! ($O(n)$ space)

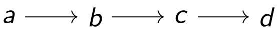
Linear Extension

A total order $\leq$ is a **linear extension** of $\leq_{FS}$ if:

$$\mathbf{u} \leq_{\mathbf{FS}} \mathbf{v} \implies \mathbf{u} \leq \mathbf{v}$$



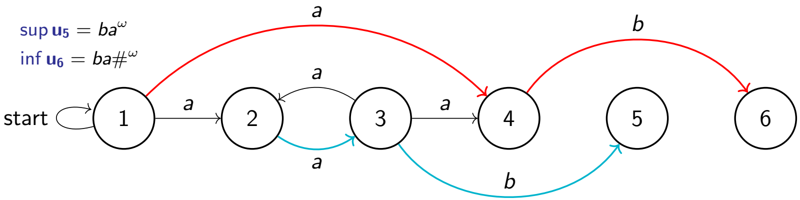
possible **partial order**



possible **linear extension**

The linear extension allows us **to reconstruct** $\leq_{FS}$ when $\sup \mathbf{u} > \inf \mathbf{v}$.

Inf/Sup-Walks Cross



In this case the **linear extension** is $1 \to 2 \to 3 \to 4 \to 5 \to 6$.

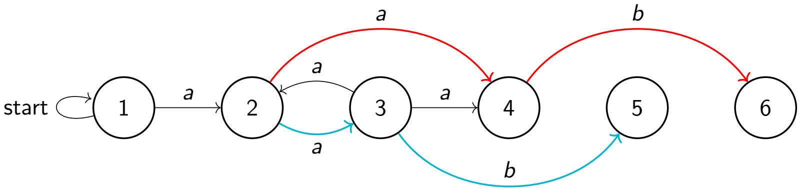
We have sup $\mathbf{5}$ > inf $\mathbf{6}$

Supremum walk to 5: $\mathbf{2 \xrightarrow{a} 3 \xrightarrow{b} 5}$

Infimum walk to 6: $\mathbf{1 \xrightarrow{a} 4 \xrightarrow{b} 6}$

The order swaps!: $2 > 1 \implies 5 \leq_{FS} 6$ does not hold.

Inf/Sup-Walks Meet

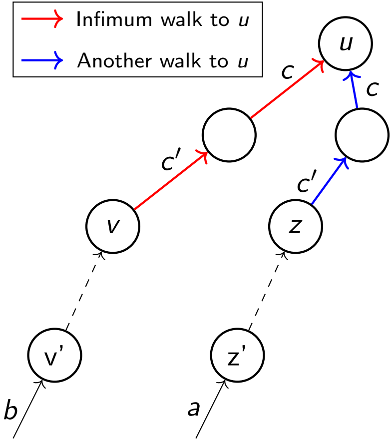


Similar as before, however now the supremum and the infimum **meet at the same node 2**

Special case! We don't know whether $5 \leq_{FS} 6$ holds

We treat this special case separately

Definition Inf/Sup Conflicts



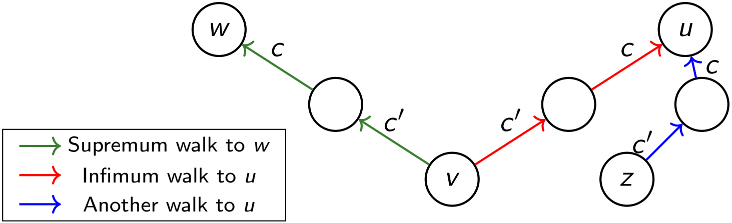
Node **v** is in **Inf conflict** with **u** because:

- **v** is in the **infimum walk** of u
- Node **z** can reach u with the **same labels c′c**
- $v \leq_{FS} z$ **does not hold**

Inf/Sup Conflicts for $\leq_{FS}$

special case seen before: the **preceding pair** of (v, z) is also a **preceding pair** of (w, u) $\implies$ **w $\leq_{FS}$ u does not hold**

w $\leq_{FS}$ u $\iff$ v not in **sup conflict** with **w** or in **inf conflict** with **u**



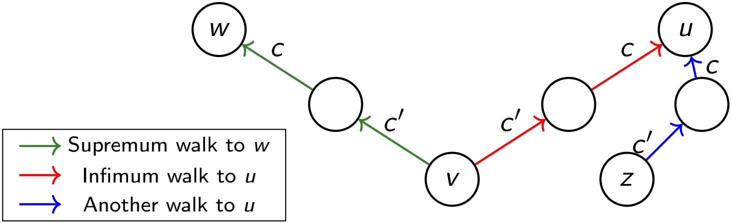
Lemma 22

It is possible to store the **inf/sup conflicts of every state** in **O(n)** space

Inf/Sup Conflicts for $\leq_{FS}$

special case seen before: the **preceding pair** of (v, z) is also a **preceding pair** of (w, u) $\implies$ **w $\leq_{FS}$ u does not hold**

w $\leq_{FS}$ u $\iff$ v not in **sup conflict** with **w** or in **inf conflict** with **u**



Lemma 22

It is possible to store the **inf/sup conflicts of every state** in **O(n)** space

Representation of $\leq_{FS}$

For every state **u** we save:

- An **infimum walk** to **u**
- A **supremum walk** to **u**
- Its position in the **linear extension**
- Its **inf conflicts**
- Its **sup conflicts**

All of these occupy $O(n)$ total space

Thank you for your attention ☺

Funded by ERC StG "REGINDEX: Compressed indexes for regular languages with applications to computational pan-genomics" grant nr 101039208. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.