Query Evaluation Over SLP-Represented Document Databases With Complex Document Editing

Markus L. Schmid, Nicole Schweikardt

HU Berlin, Germany

PODS 2022

The Data Model

Document Databases

- Finite alphabet $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \ldots\}$.
- *Documents*: strings over Σ, e. g., $\mathsf{D} = \mathtt{abaacbca}$.
- *Document Databases*: Sets of documents, e. g.,

$$\begin{aligned} \mathsf{DDB} &= \{\mathsf{D}_1, \mathsf{D}_2, \mathsf{D}_3\} \\ &= \{\mathtt{ababbcabca}, \mathtt{bcabcaabbca}, \mathtt{ababbca}\}. \end{aligned}$$

Information Extraction

Evaluation Task

Given a document database DDB,
evaluate a given query q over some $\mathrm{D} \in \mathrm{DDB}$.

Information Extraction

Evaluation Task

Given a document database DDB,
evaluate a given query q over some $\mathrm{D} \in \mathrm{DDB}$.

Prominent choice as query class:

Document spanners

Information Extraction

Evaluation Task

Given a document database DDB,
evaluate a given query q over some $\mathrm{D} \in \mathrm{DDB}$.

Prominent choice as query class:

Document spanners

$\mathrm{D} = \mathtt{a\,b\,b\,a\,b\,c\,c\,a\,b\,c}$

Information Extraction

Evaluation Task

Given a document database DDB,
evaluate a given query q over some $\mathrm{D} \in \mathrm{DDB}$.

Prominent choice as query class:

Document spanners

$\mathrm{D} = \mathtt{a\,b\,b\,a\,b\,c\,c\,a\,b\,c}$ $\Longrightarrow$

x	y	z
$[2,5\rangle$	$[4,7\rangle$	$[1,10\rangle$
$[3,5\rangle$	$[5,8\rangle$	$[4,7\rangle$
$[1,3\rangle$	$[3,10\rangle$	$[2,4\rangle$
$\vdots$	$\vdots$	$\vdots$

Information Extraction

Evaluation Task

Given a document database DDB,
evaluate a given query q over some $D \in DDB$.

Prominent choice as query class:

Document spanners

$D = \mathtt{a\,b\,b\,a\,b\,c\,c\,a\,b\,c}$ $\Longrightarrow$

x	y	z
$[2,5\rangle$	$[4,7\rangle$	$[1,10\rangle$
$[3,5\rangle$	$[5,8\rangle$	$[4,7\rangle$
$[1,3\rangle$	$[3,10\rangle$	$[2,4\rangle$
$\vdots$	$\vdots$	$\vdots$

Information Extraction

Evaluation Task

Given a document database DDB,
evaluate a given query q over some $\mathrm{D} \in \mathrm{DDB}$.

Prominent choice as query class:

Document spanners

$\mathrm{D} = \texttt{a b b a b c c a b c} \quad \Longrightarrow$

x	y	z
$[2,5\rangle$	$[4,7\rangle$	$[1,10\rangle$
$[3,5\rangle$	$[5,8\rangle$	$[4,7\rangle$
$[1,3\rangle$	$[3,10\rangle$	$[2,4\rangle$
$\vdots$	$\vdots$	$\vdots$

Information Extraction

Evaluation Task

Given a document database DDB,
evaluate a given query q over some $\mathrm{D} \in \mathrm{DDB}$.

Prominent choice as query class:

Document spanners

$\mathrm{D} = \texttt{a b b a b c c a b c}$ $\Longrightarrow$

x	y	z
$[2,5\rangle$	$[4,7\rangle$	$[1,10\rangle$
$[3,5\rangle$	$[5,8\rangle$	$[4,7\rangle$
$[1,3\rangle$	$[3,10\rangle$	$[2,4\rangle$
$\vdots$	$\vdots$	$\vdots$

Compressed Strings

- In practice, strings have many redundancies ($\leadsto$ are highly compressible).
- Many practical (dictionary based) compression schemes for strings exist.
- Good compression rates, low (near linear-time) running times.

Algorithmics on Compressed Strings

String-Problem P

Input: A string w.
Task: Solve P for w.
Running time: $f(|w|)$.

Algorithmics on Compressed Strings

String-Problem P (compressed)

Input: A **compressed form** S of string w.
Task: Solve P for w (**without explicitly constructing** w).
Running time: $f(|S|)$.

Algorithmics on Compressed Strings

String-Problem P (compressed)

Input: A **compressed form** S of string w.
Task: Solve P for w (**without explicitly constructing** w).
Running time: $f(|S|)$.

uncompressed	vs.	compressed
$f(\|w\|)$	vs.	$g(\|S\|)$

Algorithmics on Compressed Strings

String-Problem P (compressed)

Input: A **compressed form** S of string w.
Task: Solve P for w (**without explicitly constructing** w).
Running time: $f(|S|)$.

uncompressed	vs.	compressed
$f(\|w\|)$	vs.	$g(\|S\|)$
$O(\|w\|)$	vs.	$\mathrm{poly}(\|S\|)$

Algorithmics on Compressed Strings

String-Problem P (compressed)

Input: A **compressed form** S of string w.
Task: Solve P for w (**without explicitly constructing** w).
Running time: $f(|S|)$.

uncompressed	vs.	compressed
$f(\lvert w\rvert)$	vs.	$g(\lvert S\rvert)$
$O(\lvert w\rvert)$	vs.	poly($\lvert S\rvert$)
$O(\lvert w\rvert)$	vs.	polylog($\lvert w\rvert$)

Straight-Line Programs (SLPs)

Main idea: Represent a string w by a context-free grammar $\mathcal{S}$ (in chomsky normal form) for language $\{w\}$.

Straight-Line Programs (SLPs)

Main idea: Represent a string w by a context-free grammar $\mathcal{S}$ (in chomsky normal form) for language $\{w\}$.

Example

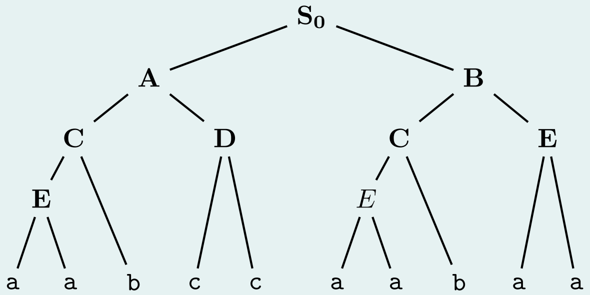
$$S_0 \to AB, \quad A \to CD, \quad B \to CE,$$
$$C \to E\mathtt{b}, \quad D \to \mathtt{cc}, \quad E \to \mathtt{aa}$$

Straight-Line Programs (SLPs)

Main idea: Represent a string w by a context-free grammar $\mathcal{S}$ (in chomsky normal form) for language $\{w\}$.

Example

$$S_0 \to AB, \qquad A \to CD, \qquad B \to CE\,,$$
$$C \to E\mathtt{b}, \qquad D \to \mathtt{cc}, \qquad E \to \mathtt{aa}$$

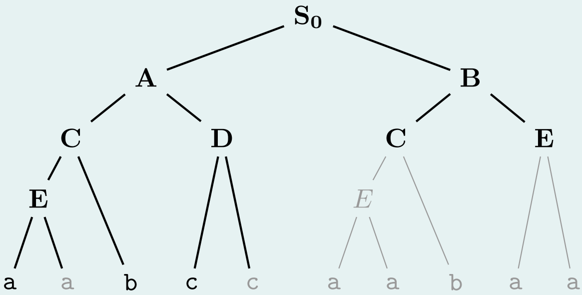


Straight-Line Programs (SLPs)

Main idea: Represent a string w by a context-free grammar $\mathcal{S}$ (in chomsky normal form) for language $\{w\}$.

Example

$$S_0 \to AB, \qquad A \to CD, \qquad B \to CE\,,$$
$$C \to E\mathtt{b}, \qquad D \to \mathtt{cc}, \qquad E \to \mathtt{aa}$$



Good Properties of SLPs

SLPs are intensely researched in TCS and many things are known:

- Exponential compression rates.
- SLPs are mathematically easy to handle ($\Rightarrow$ good for theoretical considerations).
- High practical relevance (SLPs cover many practically applied dictionary-based compression schemes).
- Many approximations and heuristics exist that efficiently compute small SLPs.
- SLPs are suitable for algorithmics on compressed strings: comparison, pattern matching, membership in a regular language, retrieving subwords, etc.

Straight-Line Programs for Document Databases

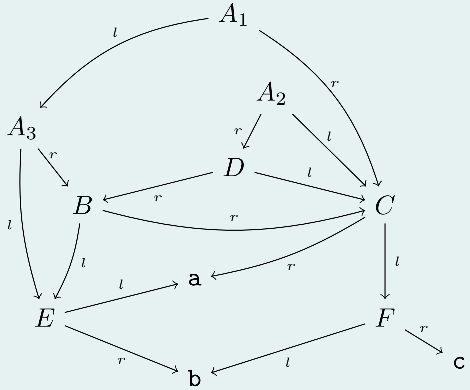
Example

$\mathrm{DDB} = \{\underbrace{\texttt{ababbcabca}}_{D_1}, \underbrace{\texttt{bcabcaabbca}}_{D_2}, \underbrace{\texttt{ababbca}}_{D_3}\}.$

Straight-Line Programs for Document Databases

Example

$\mathsf{DDB} = \{\underbrace{\mathtt{ababbcabca}}_{\mathsf{D}_1}, \underbrace{\mathtt{bcabcaabbca}}_{\mathsf{D}_2}, \underbrace{\mathtt{ababbca}}_{\mathsf{D}_3}\}.$

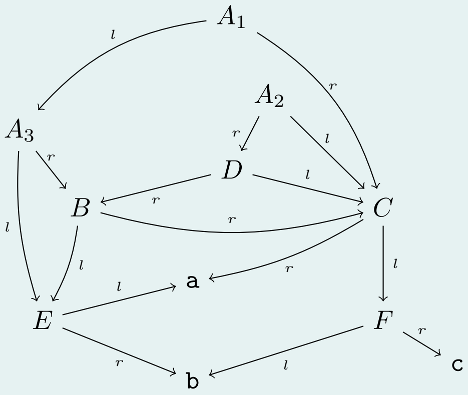


$A_1 \to A_3 C$
$A_2 \to CD$
$C \to F\mathtt{a}$
$\vdots$

Straight-Line Programs for Document Databases

Example

$\mathsf{DDB} = \{\underbrace{\texttt{ababbcabca}}_{\mathsf{D}_1 = \mathfrak{D}(A_1)}, \underbrace{\texttt{bcabcaabbca}}_{\mathsf{D}_2 = \mathfrak{D}(A_2)}, \underbrace{\texttt{ababbca}}_{\mathsf{D}_3 = \mathfrak{D}(A_3)}\}$.



$A_1 \rightarrow A_3 C$
$A_2 \rightarrow CD$
$C \rightarrow F\mathsf{a}$
$\vdots$

Spanner Evaluation Over SLP-Compressed Documents

What about database theory (i. e., information extraction)?

Theorem (S., Schweikardt, PODS'21)

Let D be a document represented by an SLP $\mathcal{S}$ and let M be a document spanner. The set $[\![M]\!](\mathrm{D})$ can be enumerated with preprocessing time $\mathrm{O}(|\mathcal{S}|)$ and delay $\mathrm{O}(\log |\mathrm{D}|)$.[1]

[1] In data-complexity.

Spanner Evaluation Over SLP-Compressed Documents

What about database theory (i. e., information extraction)?

Theorem (S., Schweikardt, PODS'21)

Let D be a document represented by an SLP $\mathcal{S}$ and let M be a document spanner. The set $[\![M]\!](\mathrm{D})$ can be enumerated with preprocessing time $\mathrm{O}(|\mathcal{S}|)$ and delay $\mathrm{O}(\log|\mathrm{D}|)$.[1]

Remark

Delay $\mathrm{O}(\log|\mathrm{D}|)$ independent of the size $|\mathcal{S}|$ is possible since $\mathcal{S}$ is **balanced** in the preprocessing.

(**Balanced** SLPs will play a central role in the following).

[1] In data-complexity.

This Paper: Dynamic Setting

Basic Setting

Given:

- A document database DDB represented by an SLP $\mathcal{S}$.
- Data structures for enumerating spanners $M_1, M_2, \ldots$ on documents of DDB.

This Paper: Dynamic Setting

Basic Setting

Given:

- A document database DDB represented by an SLP $\mathcal{S}$.
- Data structures for enumerating spanners $M_1, M_2, \ldots$ on documents of DDB.

Task:

- "Update" DDB by directly updating $\mathcal{S}$.

This Paper: Dynamic Setting

Basic Setting

Given:

- A document database DDB represented by an SLP $\mathcal{S}$.
- Data structures for enumerating spanners $M_1, M_2, \ldots$ on documents of DDB.

Task:

- "Update" DDB by directly updating $\mathcal{S}$.

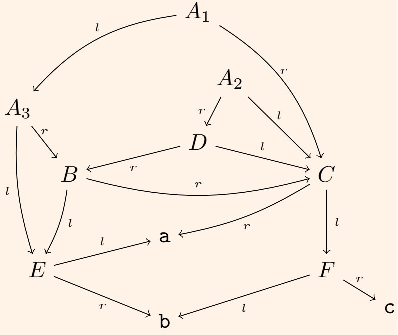
Additional conditions:

- Do not decompress $\mathcal{S}$.
- Also update the data structures for enumeration.
- Maintain the **"balancedness"** property of $\mathcal{S}$.

Balanced SLPs

Notations for Balancedness

Let A be a node of an SLP $\mathcal{S}$ that represents a document database.

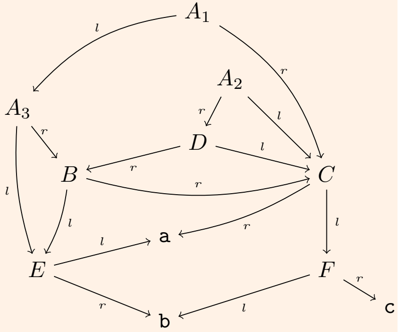


Balanced SLPs

Notations for Balancedness

Let A be a node of an SLP $\mathcal{S}$ that represents a document database.

ord(A) is the length of a longest path from A to a sink node.
(E. g., ord$(A_3) = 4$)



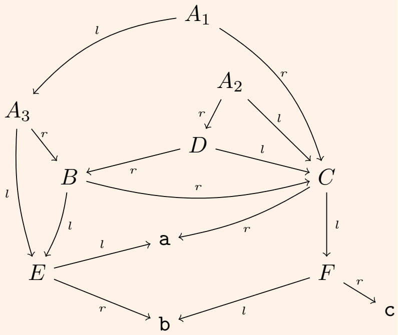
Balanced SLPs

Notations for Balancedness

Let A be a node of an SLP $\mathcal{S}$ that represents a document database.

ord(A) is the length of a longest path from A to a sink node.
(E. g., $\mathrm{ord}(A_3) = 4$)

Let $A \to BC$.
$\mathrm{bal}(A) = \mathrm{ord}(B) - \mathrm{ord}(C)$.
(E. g., $\mathrm{bal}(A_3) = -2$)



Balanced SLPs

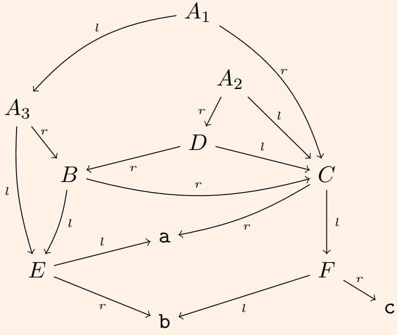
Notations for Balancedness

Let A be a node of an SLP $\mathcal{S}$ that represents a document database.

$\mathrm{ord}(A)$ is the length of a longest path from A to a sink node.
(E. g., $\mathrm{ord}(A_3) = 4$)

Let $A \to BC$.
$\mathrm{bal}(A) = \mathrm{ord}(B) - \mathrm{ord}(C)$.
(E. g., $\mathrm{bal}(A_3) = -2$)

A is c-**shallow** (for a constant $c \in \mathbb{N}$) if $\mathrm{ord}(A) \leq c \cdot \log |\mathfrak{D}(A)|$.

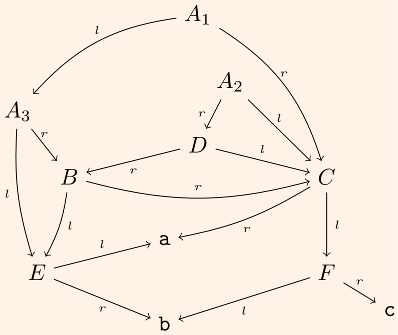


Balanced SLPs

Notations for Balancedness

Let A be a node of an SLP $\mathcal{S}$ that represents a document database.

ord(A) is the length of a longest path from A to a sink node.
(E. g., $\mathrm{ord}(A_3) = 4$)



Let $A \to BC$.
$\mathrm{bal}(A) = \mathrm{ord}(B) - \mathrm{ord}(C)$.
(E. g., $\mathrm{bal}(A_3) = -2$)

A is c-**shallow** (for a constant $c \in \mathbb{N}$) if $\mathrm{ord}(A) \leq c \cdot \log |\mathfrak{D}(A)|$.

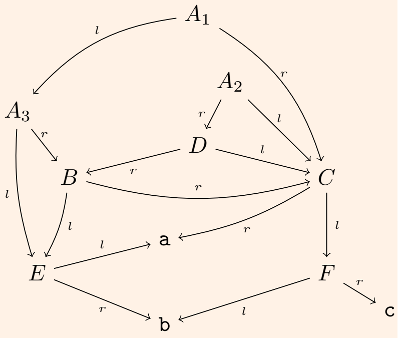
A is **balanced** if $\mathrm{bal}(A) \in \{-1, 0, 1\}$.

Balanced SLPs

Notations for Balancedness

Let A be a node of an SLP $\mathcal{S}$ that represents a document database.

$\mathrm{ord}(A)$ is the length of a longest path from A to a sink node.
(E. g., $\mathrm{ord}(A_3) = 4$)



Let $A \to BC$.
$\mathrm{bal}(A) = \mathrm{ord}(B) - \mathrm{ord}(C)$.
(E. g., $\mathrm{bal}(A_3) = -2$)

A is c-**shallow** (for a constant $c \in \mathbb{N}$) if $\mathrm{ord}(A) \leq c \cdot \log |\mathfrak{D}(A)|$.

A is **balanced** if $\mathrm{bal}(A) \in \{-1, 0, 1\}$.

$\mathcal{S}$ is c-**shallow** or **strongly balanced** if all nodes are c-**shallow** or **balanced**, respectively.

Balanced SLPs

Important Results About SLP-Balancing

A given SLP $\mathcal{S}$ for a single document D ...

... can be made c-shallow in time $O(|\mathcal{S}|)$.

(Ganardi, Jez, Lohrey, JACM 2021)

Balanced SLPs

Important Results About SLP-Balancing

A given SLP $\mathcal{S}$ for a single document D …

… can be made c-shallow in time $O(|\mathcal{S}|)$.

(Ganardi, Jez, Lohrey, JACM 2021)

… can be made strongly balanced in time $O(|\mathcal{S}| \cdot \log |\mathrm{D}|)$.

(Rytter, TCS 2003)

Balanced SLPs

Important Results About SLP-Balancing

A given SLP $\mathcal{S}$ for a single document D ...

... can be made c-shallow in time $O(|\mathcal{S}|)$.

(Ganardi, Jez, Lohrey, JACM 2021)

... can be made strongly balanced in time $O(|\mathcal{S}| \cdot \log |\mathrm{D}|)$.

(Rytter, TCS 2003)

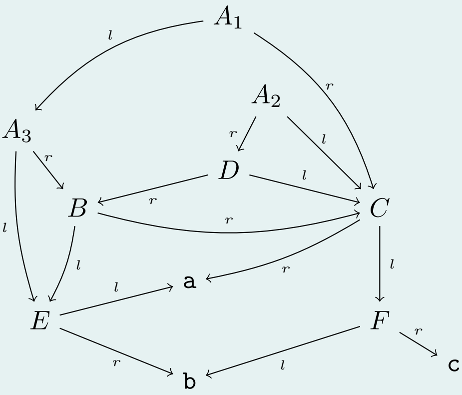
... can in general not be made strongly balanced without a size increase by a factor of $\Omega(\log |\mathrm{D}|)$.

(Ganardi, ESA 2021)

Updating SLP-Represented Document Databases

Example

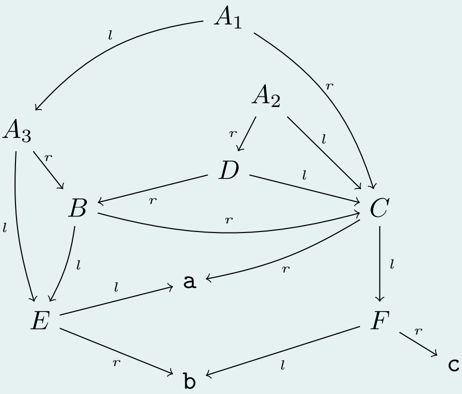
DDB $= \{\mathtt{ababbcabca}, \mathtt{bcabcaabbca}, \mathtt{ababbca}\}$.



Updating SLP-Represented Document Databases

Example

$\mathsf{DDB} = \{\mathtt{ababbcabca}, \mathtt{bcabcaabbca}, \mathtt{ababbca}\}$.

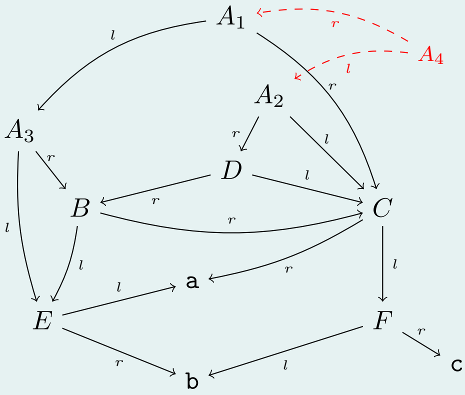


Add document
$\mathsf{D}_4 = \mathsf{D}_2\,\mathsf{D}_1$

Updating SLP-Represented Document Databases

Example

$\mathsf{DDB} = \{\texttt{ababbcabca}, \texttt{bcabcaabbca}, \texttt{ababbca}\}$.

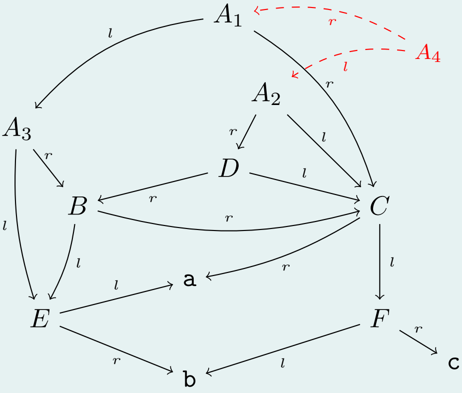


Add document
$\mathsf{D}_4 = \mathsf{D}_2\,\mathsf{D}_1$

Updating SLP-Represented Document Databases

Example

DDB $= \{\texttt{ababbcabca}, \texttt{bcabcaabbca}, \texttt{ababbca}\}$.



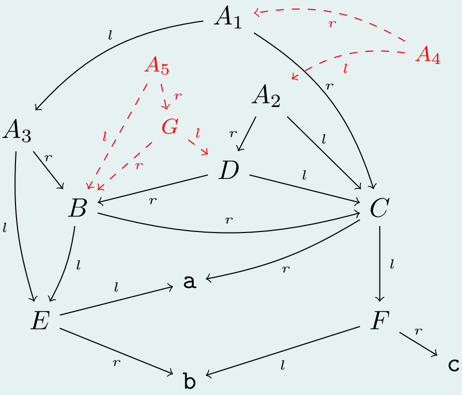
Add document
$\mathsf{D}_4 = \mathsf{D}_2\,\mathsf{D}_1$

Add document
$\mathsf{D}_5 = \mathfrak{D}(B)\,\mathfrak{D}(D)\,\mathfrak{D}(B)$

Updating SLP-Represented Document Databases

Example

DDB $= \{\texttt{ababbcabca}, \texttt{bcabcaabbca}, \texttt{ababbca}\}$.



Add document
$\mathsf{D}_4 = \mathsf{D}_2\,\mathsf{D}_1$

Add document
$\mathsf{D}_5 = \mathfrak{D}(B)\,\mathfrak{D}(D)\,\mathfrak{D}(B)$

Complex Document Editing

Notation

Let D be a document and $i, j \in \{1, 2, \ldots, |\,\mathrm{D}\,|\}$.
$\mathrm{D}[i..j]$ is D's factor from position i to position j.

Complex Document Editing

Notation

Let D be a document and $i, j \in \{1, 2, \ldots, |\,\mathrm{D}\,|\}$.
$\mathrm{D}[i..j]$ is D's factor from position i to position j.

Basic CDE-Algebra Operations

$$
\begin{aligned}
\mathrm{concat}(\mathrm{D}, \mathrm{D}') &= \mathrm{D} \cdot \mathrm{D}', \\
\mathrm{extract}(\mathrm{D}, i, j) &= \mathrm{D}[i..j]\,, \\
\mathrm{delete}(\mathrm{D}, i, j) &= \mathrm{D}[1..i-1] \cdot \mathrm{D}[j+1..|\,\mathrm{D}\,|]\,, \\
\mathrm{insert}(\mathrm{D}, \mathrm{D}', k) &= \mathrm{D}[1..k-1] \cdot \mathrm{D}' \cdot \mathrm{D}[k..|\,\mathrm{D}\,|]\,, \\
\mathrm{copy}(\mathrm{D}, i, j, k) &= \mathrm{D}[1..k-1] \cdot \mathrm{D}[i..j] \cdot \mathrm{D}[k..|\,\mathrm{D}\,|]\,.
\end{aligned}
$$

Complex Document Editing

Notation

Let D be a document and $i, j \in \{1, 2, \ldots, |\,\mathrm{D}\,|\}$.
$\mathrm{D}[i..j]$ is D's factor from position i to position j.

Basic CDE-Algebra Operations

$$
\begin{aligned}
\mathrm{concat}(\mathrm{D}, \mathrm{D}') &= \mathrm{D} \cdot \mathrm{D}', \\
\mathrm{extract}(\mathrm{D}, i, j) &= \mathrm{D}[i..j], \\
\mathrm{delete}(\mathrm{D}, i, j) &= \mathrm{D}[1..i-1] \cdot \mathrm{D}[j+1..|\,\mathrm{D}\,|], \\
\mathrm{insert}(\mathrm{D}, \mathrm{D}', k) &= \mathrm{D}[1..k-1] \cdot \mathrm{D}' \cdot \mathrm{D}[k..|\,\mathrm{D}\,|], \\
\mathrm{copy}(\mathrm{D}, i, j, k) &= \mathrm{D}[1..k-1] \cdot \mathrm{D}[i..j] \cdot \mathrm{D}[k..|\,\mathrm{D}\,|].
\end{aligned}
$$

CDE-Expressions

Expressions over DDB's documents using the basic CDE-algebra operations.

E. g., $\varphi = \mathrm{concat}(\mathrm{D}_1, \mathrm{insert}(\mathrm{D}_3, \mathrm{extract}(\mathrm{D}_7, 5, 21), 12))$

Our Main Result

CDE Extension Theorem

Let DDB be represented by a strongly balanced SLP $\mathcal{S}$, let φ be a CDE-expression over DDB.

We can construct a strongly balanced SLP $\mathcal{S}'$ for

$$\mathsf{DDB} \cup \{\mathsf{eval}(\varphi)\}$$

in time

$$\mathsf{O}(|\varphi|^2 + |\varphi| \cdot \log(\mathsf{d}_{\mathsf{max}}))\,,$$

where $\mathsf{d}_{\mathsf{max}} = \{|\,\mathsf{D}\,|\mid \mathsf{D} \in \mathsf{DDB}\}$.

Our Main Result

Corollary

Let DDB be represented by a strongly balanced SLP $\mathcal{S}$.

Let $\mathcal{M}$ be a class of regular spanners such that every $M \in \mathcal{M}$ can be enumerated on every $\mathsf{D} \in \mathsf{DDB}$ with delay $\mathsf{O}(\log(|\,\mathsf{D}\,|))$.

Given a CDE-expression φ over DDB, we can construct in time

$$\mathsf{O}(|\mathcal{M}| \cdot (|\varphi|^2 + |\varphi| \cdot \log(\mathsf{d}_{\mathsf{max}})))$$

a strongly balanced SLP $\mathcal{S}'$ for

$$\mathsf{DDB}' = \mathsf{DDB} \cup \{\mathsf{eval}(\varphi)\}$$

and data structures such that we can now enumerate every $M \in \mathcal{M}$ on every $\mathsf{D} \in \mathsf{DDB}'$ with delay $\mathsf{O}(\log(|\,\mathsf{D}\,|))$.

Proof Ideas

Proof Roadmap

- For each of the basic CDE-operation, prove a lemma that "handles" this operation.
  (I. e., "For nodes B and C, create a node that derives concat$(\mathfrak{D}(B), \mathfrak{D}(C))$", "For node B and i, j, create a node that derives extract$(\mathfrak{D}(B), i, j)$", etc.)

Proof Ideas

Proof Roadmap

- For each of the basic CDE-operation, prove a lemma that "handles" this operation.
  (I. e., "For nodes B and C, create a node that derives concat$(\mathfrak{D}(B), \mathfrak{D}(C))$", "For node B and i, j, create a node that derives extract$(\mathfrak{D}(B), i, j)$", etc.)
- For a given CDE-expression φ, we can then apply these lemmas "bottom-up" (along φ's syntax tree).

Proof Ideas

Proof Roadmap

- For each of the basic CDE-operation, prove a lemma that "handles" this operation.
  (I. e., "For nodes B and C, create a node that derives $\text{concat}(\mathfrak{D}(B), \mathfrak{D}(C))$", "For node B and i, j, create a node that derives $\text{extract}(\mathfrak{D}(B), i, j)$", etc.)
- For a given CDE-expression φ, we can then apply these lemmas "bottom-up" (along φ's syntax tree).
- Lemmas for $\text{concat}(\cdot, \cdot)$ and $\text{extract}(\cdot, \cdot, \cdot)$ are sufficient (since all CDE-operations can be defined by applications of these two operations).

Proof Sketch for Concatenation Lemma

$$\begin{array}{cccc} & B & & & C & \\ \swarrow & & \searrow & \swarrow & & \searrow \\ L_B & & R_B & L_C & & R_C \end{array}$$

Proof Sketch for Concatenation Lemma

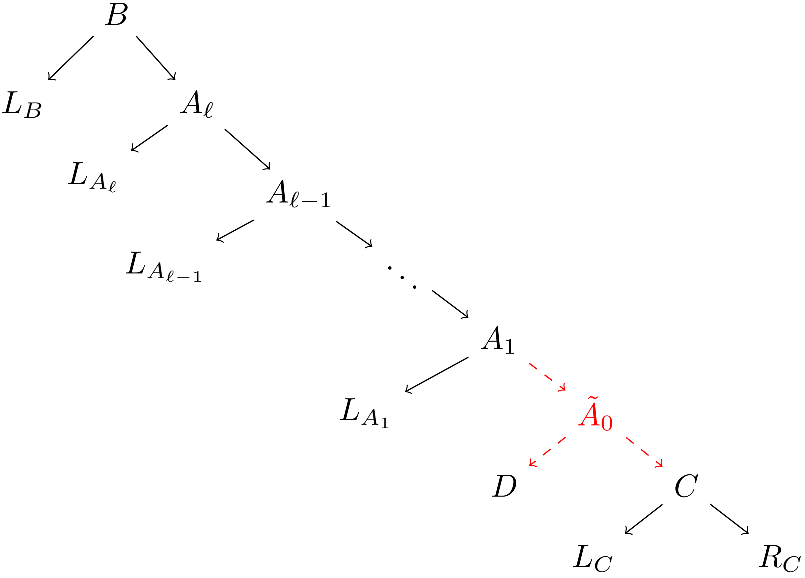
$$\begin{tikzcd}
 & & & \color{red}{\tilde{A}} \arrow[dll, dashed, red] \arrow[drr, dashed, red] & & & \\
 & B \arrow[dl] \arrow[dr] & & & & C \arrow[dl] \arrow[dr] & \\
L_B & & R_B & & L_C & & R_C
\end{tikzcd}$$

Proof Sketch for Concatenation

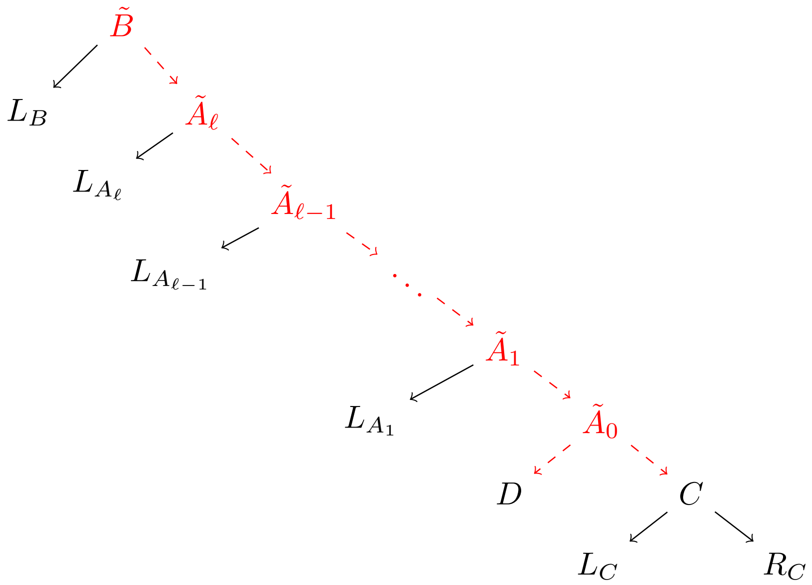
$$
\begin{tikzcd}
 & B \arrow[dl] \arrow[dr] & & & & & \\
L_B & & A_\ell \arrow[dl] \arrow[dr] & & & & \\
 & L_{A_\ell} & & A_{\ell-1} \arrow[dl] \arrow[dr] & & & \\
 & & L_{A_{\ell-1}} & & \ddots \arrow[dr] & & \\
 & & & & & A_1 \arrow[dl] \arrow[dr] & \\
 & & & & L_{A_1} & & D
\end{tikzcd}
\qquad
\begin{tikzcd}
 & C \arrow[dl] \arrow[dr] & \\
L_C & & R_C
\end{tikzcd}
$$

$\mathsf{ord}(D) \in \{\mathsf{ord}(C), \mathsf{ord}(C)+1\}$

Proof Sketch for Concatenation



Proof Sketch for Concatenation



Rotations

$$
\begin{tikzcd}
 & A^{(k+3)} \arrow[dl] \arrow[dr] & & \\
B_{(k)} & & C^{(k+2)} \arrow[dl] \arrow[dr] & \\
 & & D_{(k)} & & E_{(k+1)}
\end{tikzcd}
$$

Rotations

$$\begin{tikzcd}
& A^{(k+3)} \arrow[dl] \arrow[dr] & & \\
B_{(k)} & & C^{(k+2)} \arrow[dl] \arrow[dr] & \\
& D_{(k)} & & E_{(k+1)}
\end{tikzcd}$$

$$\Downarrow$$

$$\begin{tikzcd}
& & A^{(k+2)} \arrow[dl] \arrow[dr] & \\
& {}^{(k+1)}C \arrow[dl] \arrow[dr] & & E_{(k+1)} \\
B_{(k)} & & D_{(k)} &
\end{tikzcd}$$

Related Question Discussed in the Paper

- Adding a completely new document (in compressed form or plain text).
- Building an SLP-represented document database from scratch.
- Retrieving SLP-represented documents for the spans extracted by a spanner.
- Overall setting works for other evaluation problems like testing, non-emptiness, etc.
- Extends to general finite transducers as extractors.

Thank you very much for your attention.