Document Spanners — A Brief Overview of Concepts, Results, and Recent Developments

Markus L. Schmid and Nicole Schweikardt

Humboldt-Universität zu Berlin

Gems of PODS

PODS 2022: 41st Symposium on Principles of Database Systems

Introduction

Goal: extract structured information from texts

Idea: query texts like you would query a relational database

Introduction

Goal: extract structured information from texts

Idea: query texts like you would query a relational database

Timeline:

Since 2005: development of IMB's SystemT:
a rule-based information extraction system with an SQL-like declarative language called AQL (Annotation Query Language)

Introduction

Goal: extract structured information from texts

Idea: query texts like you would query a relational database

Timeline:

- Since 2005: development of IMB's SystemT:
  a rule-based information extraction system with an SQL-like declarative language called AQL (Annotation Query Language)
- 2010: release of SystemT as a commercial IBM product

Introduction

Goal: extract structured information from texts

Idea: query texts like you would query a relational database

Timeline:

- Since 2005: development of IMB's SystemT: a rule-based information extraction system with an SQL-like declarative language called AQL (Annotation Query Language)
- 2010: release of SystemT as a commercial IBM product
- PODS 2013: introduction of the framework of document spanners as a formalisation of AQL:

Spanners: A Formal Framework for Information Extraction

Ronald Fagin
IBM Research – Almaden
San Jose, CA, USA

Benny Kimelfeld
IBM Research – Almaden
San Jose, CA, USA

Frederick Reiss
IBM Research – Almaden
San Jose, CA, USA

Stijn Vansummeren
Université Libre de Bruxelles (ULB)
Bruxelles, Belgium

Introduction

Goal: extract structured information from texts

Idea: query texts like you would query a relational database

Timeline:

Since 2005: development of IMB's SystemT:
a rule-based information extraction system with an SQL-like declarative language called AQL (Annotation Query Language)

2010: release of SystemT as a commercial IBM product

PODS 2013: introduction of the framework of document spanners as a formalisation of AQL:

Spanners: A Formal Framework for Information Extraction

Ronald Fagin
IBM Research – Almaden
San Jose, CA, USA

Benny Kimelfeld
IBM Research – Almaden
San Jose, CA, USA

Frederick Reiss
IBM Research – Almaden
San Jose, CA, USA

Stijn Vansummeren
Université Libre de Bruxelles (ULB)
Bruxelles, Belgium

since then: in-depth investigation of this framework

Overview

Overview

Spanners: map documents to relations of spans

Spanners: map documents to relations of spans

$D = \texttt{aggagccagc}$

x	y	z
$[2, 5\rangle$	$[4, 7\rangle$	$[1, 10\rangle$
$[3, 5\rangle$	$[5, 8\rangle$	$[4, 7\rangle$
$[1, 3\rangle$	$[4, 6\rangle$	$[2, 4\rangle$

Spanners: map documents to relations of spans

a span

$D = a\,g\,g\,a\,g\,c\,c\,a\,g\,c$

x	y	z
$[2, 5\rangle$	$[4, 7\rangle$	$[1, 10\rangle$
$[3, 5\rangle$	$[5, 8\rangle$	$[4, 7\rangle$
$[1, 3\rangle$	$[4, 6\rangle$	$[2, 4\rangle$

Spanners: map documents to relations of spans

a span

$D = a\,g\,g\,a\,g\,c\,c\,a\,g\,c$

x	y	z
$[2, 5\rangle$	$[4, 7\rangle$	$[1, 10\rangle$
$[3, 5\rangle$	$[5, 8\rangle$	$[4, 7\rangle$
$[1, 3\rangle$	$[4, 6\rangle$	$[2, 4\rangle$

Spanners: map documents to relations of spans

a span

$D = \texttt{a g g a g c c a g c}$

x	y	z
$[2, 5\rangle$	$[4, 7\rangle$	$[1, 10\rangle$
$[3, 5\rangle$	$[5, 8\rangle$	$[4, 7\rangle$
$[1, 3\rangle$	$[4, 6\rangle$	$[2, 4\rangle$

Spanners: map documents to relations of spans

$D = \texttt{aggagccagc}$

x	y	z
$[2, 5\rangle$	$[4, 7\rangle$	$[1, 10\rangle$
$[3, 5\rangle$	$[5, 8\rangle$	$[4, 7\rangle$
$[1, 3\rangle$	$[4, 6\rangle$	$[2, 4\rangle$

Spanners: map documents to relations of spans

$D = \texttt{a g g a g c c a g c}$

In SystemT / AQL: 2-stage approach

1. extract a relation from a document — regular expressions with capture variables

x	y	z
$[2,5\rangle$	$[4,7\rangle$	$[1,10\rangle$
$[3,5\rangle$	$[5,8\rangle$	$[4,7\rangle$
$[1,3\rangle$	$[4,6\rangle$	$[2,4\rangle$

Spanners: map documents to relations of spans

$D = \texttt{a g g a g c c a g c}$

In SystemT / AQL: 2-stage approach

1. extract a relation from a document

x	y	z
$[2,5\rangle$	$[4,7\rangle$	$[1,10\rangle$
$[3,5\rangle$	$[5,8\rangle$	$[4,7\rangle$
$[1,3\rangle$	$[4,6\rangle$	$[2,4\rangle$

regular expressions
with capture variables

finite automata
(with marker symbols)

Spanners: map documents to relations of spans

$D = \mathtt{a\,g\,g\,a\,g\,c\,c\,a\,g\,c}$

In SystemT / AQL: 2-stage approach

1. extract a relation from a document

x	y	z
$[2,5\rangle$	$[4,7\rangle$	$[1,10\rangle$
$[3,5\rangle$	$[5,8\rangle$	$[4,7\rangle$
$[1,3\rangle$	$[4,6\rangle$	$[2,4\rangle$

regular expressions
with capture variables

finite automata
(with marker symbols)

2. manipulate this relation by algebraic operations

standard relational operators:
union $\cup$
projection π
join $\bowtie$

Spanners: map documents to relations of spans

D = a g g a g c c a g c

In SystemT / AQL: 2-stage approach

1. extract a relation from a document

x	y	z
$[2,5\rangle$	$[4,7\rangle$	$[1,10\rangle$
$[3,5\rangle$	$[5,8\rangle$	$[4,7\rangle$
$[1,3\rangle$	$[4,6\rangle$	$[2,4\rangle$

regular expressions
with capture variables

finite automata
(with marker symbols)

2. manipulate this relation by algebraic operations

standard relational operators:
union $\cup$
projection π
join $\bowtie$

text-centric operators:
string-equality selection $\varsigma^{=}$
containment

Spanners: map documents to relations of spans

D = a g g a g c c a g c

In SystemT / AQL: 2-stage approach

1. extract a relation from a document

regular expressions
with capture variables

finite automata
(with marker symbols)

$R =$

x	y	z
$[2,5\rangle$	$[4,7\rangle$	$[1,10\rangle$
$[3,5\rangle$	$[5,8\rangle$	$[4,7\rangle$
$[1,3\rangle$	$[4,6\rangle$	$[2,4\rangle$

2. manipulate this relation by algebraic operations

$\varsigma^{=}_{x,y}(R)$

selects tuples of R where the substrings described by x and y are equal

standard relational operators:
union $\cup$
projection π
join $\bowtie$

text-centric operators:
string-equality selection $\varsigma^{=}$
containment

Spanners: map documents to relations of spans

D = a g g a g c c a g c

In SystemT / AQL: 2-stage approach

1. extract a relation from a document

x	y	z
$[2,5\rangle$	$[4,7\rangle$	$[1,10\rangle$
$[3,5\rangle$	$[5,8\rangle$	$[4,7\rangle$
$[1,3\rangle$	$[4,6\rangle$	$[2,4\rangle$

regular expressions
with capture variables

finite automata
(with marker symbols)

2. manipulate this relation by algebraic operations

standard relational operators:
union $\cup$
projection π
join $\bowtie$

text-centric operators:
string-equality selection $\varsigma^{=}$
containment

conflict resolution:
resolve undesired overlapping spans

Core Spanners: map documents to relations of spans

$D = a\,g\,g\,a\,g\,c\,c\,a\,g\,c$

In SystemT / AQL: 2-stage approach

1. extract a relation from a document

x	y	z
$[2,5\rangle$	$[4,7\rangle$	$[1,10\rangle$
$[3,5\rangle$	$[5,8\rangle$	$[4,7\rangle$
$[1,3\rangle$	$[4,6\rangle$	$[2,4\rangle$

regular expressions
with capture variables

finite automata
(with marker symbols)

2. manipulate this relation by algebraic operations

standard relational operators:
union $\cup$
projection π
join $\bowtie$

text-centric operators:
string-equality selection $\varsigma^{=}$
containment

conflict resolution:
resolve undesired overlapping spans

Core-Simplification Lemma

[Fagin, Kimelfeld, Reiss, Vansummeren, PODS'13]

Every core spanner can be represented by a single variable-set automaton, followed by string-equality selections and then by a projection.

Representing spanners by languages of subword-marked words

Markers: For every variable x we use meta-symbols (markers) ${}^{x}\triangleright$ and $\triangleleft^{x}$.

Representing spanners by languages of subword-marked words

Markers: For every variable x we use meta-symbols (markers) ${}^{x}\triangleright$ and $\triangleleft^{x}$.

document	span tuple		subword-marked word
	x	y	
agcca	$[1, 2\rangle$	$[3, 4\rangle$	${}^{x}\triangleright$ a $\triangleleft^{x}$ g ${}^{y}\triangleright$ c $\triangleleft^{y}$ ca

Representing spanners by languages of subword-marked words

Markers: For every variable x we use meta-symbols (markers) ${}^{x}\triangleright$ and $\triangleleft^{x}$.

document	span tuple		subword-marked word
	x	y	
agcca	$[1,2\rangle$	$[3,4\rangle$	${}^{x}\triangleright$ a $\triangleleft^{x}$ g ${}^{y}\triangleright$ c $\triangleleft^{y}$ ca
agcca	$[1,2\rangle$	$[4,5\rangle$	${}^{x}\triangleright$ a $\triangleleft^{x}$ gc ${}^{y}\triangleright$ c $\triangleleft^{y}$ a
agcca	$[1,2\rangle$	$[3,5\rangle$	${}^{x}\triangleright$ a $\triangleleft^{x}$ g ${}^{y}\triangleright$ cc $\triangleleft^{y}$ a

Representing spanners by languages of subword-marked words

Markers: For every variable x we use meta-symbols (markers) ${}^{x}\triangleright$ and $\triangleleft^{x}$.

document	span tuple x	span tuple y	subword-marked word
agcca	$[1,2\rangle$	$[3,4\rangle$	${}^{x}\triangleright$ a $\triangleleft^{x}$ g ${}^{y}\triangleright$ c $\triangleleft^{y}$ ca
agcca	$[1,2\rangle$	$[4,5\rangle$	${}^{x}\triangleright$ a $\triangleleft^{x}$ gc ${}^{y}\triangleright$ c $\triangleleft^{y}$ a
agcca	$[1,2\rangle$	$[3,5\rangle$	${}^{x}\triangleright$ a $\triangleleft^{x}$ g ${}^{y}\triangleright$ cc $\triangleleft^{y}$ a
$\mathfrak{e}(w)$	st(w)		w

Representing spanners by languages of subword-marked words

Markers: For every variable x we use meta-symbols (markers) ${}^{x}\triangleright$ and $\triangleleft^{x}$.

document	span tuple x	span tuple y	subword-marked word
agcca	$[1,2\rangle$	$[3,4\rangle$	${}^{x}\triangleright$ a $\triangleleft^{x}$ g ${}^{y}\triangleright$ c $\triangleleft^{y}$ ca
agcca	$[1,2\rangle$	$[4,5\rangle$	${}^{x}\triangleright$ a $\triangleleft^{x}$ gc ${}^{y}\triangleright$ c $\triangleleft^{y}$ a
agcca	$[1,2\rangle$	$[3,5\rangle$	${}^{x}\triangleright$ a $\triangleleft^{x}$ g ${}^{y}\triangleright$ cc $\triangleleft^{y}$ a
$\mathfrak{e}(w)$	$\mathsf{st}(w)$		w

Observation:

- Every subword-marked language L represents a spanner $[\![L]\!]$: $[\![L]\!](\mathsf{D}) = \{\, \mathsf{st}(w) \;:\; w \in L,\ \mathfrak{e}(w) = \mathsf{D} \,\}$.

Representing spanners by languages of subword-marked words

Markers: For every variable x we use meta-symbols (markers) ${}^{x}\triangleright$ and $\triangleleft^{x}$.

document	span tuple x	span tuple y	subword-marked word
agcca	$[1,2\rangle$	$[3,4\rangle$	${}^{x}\triangleright$ a $\triangleleft^{x}$ g ${}^{y}\triangleright$ c $\triangleleft^{y}$ ca
agcca	$[1,2\rangle$	$[4,5\rangle$	${}^{x}\triangleright$ a $\triangleleft^{x}$ gc ${}^{y}\triangleright$ c $\triangleleft^{y}$ a
agcca	$[1,2\rangle$	$[3,5\rangle$	${}^{x}\triangleright$ a $\triangleleft^{x}$ g ${}^{y}\triangleright$ cc $\triangleleft^{y}$ a
$\mathfrak{e}(w)$	$\mathrm{st}(w)$		w

Observation:

- Every subword-marked language L represents a spanner $[\![L]\!]$: $[\![L]\!](\mathrm{D}) = \{\, \mathrm{st}(w) \;:\; w \in L,\ \mathfrak{e}(w) = \mathrm{D} \,\}$.
- For every spanner S there is a subword-marked language L with $S = [\![L]\!]$.

Representing spanners by languages of subword-marked words

Markers: For every variable x we use meta-symbols (markers) ${}^{x}\triangleright$ and $\triangleleft^{x}$.

document	span tuple x	span tuple y	subword-marked word
agcca	$[1,2\rangle$	$[3,4\rangle$	${}^{x}\triangleright$ a $\triangleleft^{x}$ g ${}^{y}\triangleright$ c $\triangleleft^{y}$ ca
agcca	$[1,2\rangle$	$[4,5\rangle$	${}^{x}\triangleright$ a $\triangleleft^{x}$ gc ${}^{y}\triangleright$ c $\triangleleft^{y}$ a
agcca	$[1,2\rangle$	$[3,5\rangle$	${}^{x}\triangleright$ a $\triangleleft^{x}$ g ${}^{y}\triangleright$ cc $\triangleleft^{y}$ a
$\mathfrak{e}(w)$	$\mathrm{st}(w)$		w

Observation:

- Every subword-marked language L represents a spanner $[\![L]\!]$:
  $[\![L]\!](\mathrm{D}) = \{\, \mathrm{st}(w) \;:\; w \in L,\ \mathfrak{e}(w) = \mathrm{D} \,\}$.
- For every spanner S there is a subword-marked language L with $S = [\![L]\!]$.

Regular spanners: defined by regular languages of subword-marked words

Representing spanners by languages of subword-marked words

Markers: For every variable x we use meta-symbols (markers) ${}^{x}\triangleright$ and $\triangleleft^{x}$.

document	span tuple x	span tuple y	subword-marked word
agcca	$[1,2\rangle$	$[3,4\rangle$	${}^{x}\triangleright$ a $\triangleleft^{x}$ g ${}^{y}\triangleright$ c $\triangleleft^{y}$ ca
agcca	$[1,2\rangle$	$[4,5\rangle$	${}^{x}\triangleright$ a $\triangleleft^{x}$ gc ${}^{y}\triangleright$ c $\triangleleft^{y}$ a
agcca	$[1,2\rangle$	$[3,5\rangle$	${}^{x}\triangleright$ a $\triangleleft^{x}$ g ${}^{y}\triangleright$ cc $\triangleleft^{y}$ a
$\mathfrak{e}(w)$	$\mathrm{st}(w)$		w

Observation:

- Every subword-marked language L represents a spanner $[\![L]\!]$:
  $[\![L]\!](\mathrm{D}) = \{\, \mathrm{st}(w) \;:\; w \in L,\ \mathfrak{e}(w) = \mathrm{D} \,\}$.
- For every spanner S there is a subword-marked language L with $S = [\![L]\!]$.

Regular spanners: defined by regular languages of subword-marked words (equivalent to the *variable-set automata* of [Fagin et al, PODS'13])

Representing spanners by languages of subword-marked words

Markers: For every variable x we use meta-symbols (markers) ${}^{x}\triangleright$ and $\triangleleft^{x}$.

document	span tuple x	span tuple y	subword-marked word
agcca	$[1,2\rangle$	$[3,4\rangle$	${}^{x}\triangleright$ a $\triangleleft^{x}$ g ${}^{y}\triangleright$ c $\triangleleft^{y}$ ca
agcca	$[1,2\rangle$	$[4,5\rangle$	${}^{x}\triangleright$ a $\triangleleft^{x}$ gc ${}^{y}\triangleright$ c $\triangleleft^{y}$ a
agcca	$[1,2\rangle$	$[3,5\rangle$	${}^{x}\triangleright$ a $\triangleleft^{x}$ g ${}^{y}\triangleright$ cc $\triangleleft^{y}$ a
$\mathfrak{e}(w)$	$\mathrm{st}(w)$		w

Observation:

- Every subword-marked language L represents a spanner $[\![L]\!]$:
  $[\![L]\!](\mathrm{D}) = \{\, \mathrm{st}(w) \;:\; w \in L,\ \mathfrak{e}(w) = \mathrm{D} \,\}$.
- For every spanner S there is a subword-marked language L with $S = [\![L]\!]$.

Regular spanners: defined by regular languages of subword-marked words (equivalent to the *variable-set automata* of [Fagin et al, PODS'13])

Context-free spanners: def. by context-free languages of subword-marked words

Representing spanners by languages of subword-marked words

Markers: For every variable x we use meta-symbols (markers) ${}^{x}\triangleright$ and $\triangleleft^{x}$.

document	span tuple x	span tuple y	subword-marked word
agcca	$[1,2\rangle$	$[3,4\rangle$	${}^{x}\triangleright$ a $\triangleleft^{x}$ g ${}^{y}\triangleright$ c $\triangleleft^{y}$ ca
agcca	$[1,2\rangle$	$[4,5\rangle$	${}^{x}\triangleright$ a $\triangleleft^{x}$ gc ${}^{y}\triangleright$ c $\triangleleft^{y}$ a
agcca	$[1,2\rangle$	$[3,5\rangle$	${}^{x}\triangleright$ a $\triangleleft^{x}$ g ${}^{y}\triangleright$ cc $\triangleleft^{y}$ a
$\mathfrak{e}(w)$	$\mathrm{st}(w)$		w

Observation:

- Every subword-marked language L represents a spanner $[\![L]\!]$:
  $[\![L]\!](\mathrm{D}) = \{\, \mathrm{st}(w) \;:\; w \in L,\ \mathfrak{e}(w) = \mathrm{D} \,\}$.
- For every spanner S there is a subword-marked language L with $S = [\![L]\!]$.

Regular spanners: defined by regular languages of subword-marked words (equivalent to the *variable-set automata* of [Fagin et al, PODS'13])

Context-free spanners: def. by context-free languages of subword-marked words (equivalent to the *extraction grammars* of [Peterfreund, ICDT'21])

Overview

Enumerating Results of Spanners

Goal: given spanner S and document D, compute $S(\mathrm{D})$

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

$S(\mathrm{D}) =$

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

Enumerating Results of Spanners

Goal: given spanner S and document D, compute $S(\mathrm{D})$

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

$S(\mathrm{D}) =$

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

Preprocessing phase: compute a suitable data structure

Enumerating Results of Spanners

Goal: given spanner S and document D, compute $S(\mathrm{D})$

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

$S(\mathrm{D}) =$

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

Preprocessing phase: compute a suitable data structure

Enumeration phase: use this data structure to enumerate, without repetition, all tuples in $S(\mathrm{D})$

Enumerating Results of Spanners

Goal: given spanner S and document D, compute $S(\mathrm{D})$

$$\mathrm{D} = \mathtt{a\,g\,g\,a\,g\,c\,c\,a\,g\,c} \qquad S(\mathrm{D}) =$$

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

Preprocessing phase: compute a suitable data structure

linear preprocessing time: $O(|\mathrm{D}|)$

Enumeration phase: use this data structure to enumerate, without repetition, all tuples in $S(\mathrm{D})$

Enumerating Results of Spanners

Goal: given spanner S and document D, compute $S(\mathrm{D})$

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

$S(\mathrm{D}) =$

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

Preprocessing phase: compute a suitable data structure

linear preprocessing time: $O(|\mathrm{D}|)$, i.e., $f(S)\cdot O(|\mathrm{D}|)$

Enumeration phase: use this data structure to enumerate, without repetition, all tuples in $S(\mathrm{D})$

Enumerating Results of Spanners

Goal: given spanner S and document D, compute $S(\mathrm{D})$

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

$S(\mathrm{D}) =$

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

Preprocessing phase: compute a suitable data structure

linear preprocessing time: $O(|\mathrm{D}|)$, i.e., $f(S)\cdot O(|\mathrm{D}|)$

Enumeration phase: use this data structure to enumerate, without repetition, all tuples in $S(\mathrm{D})$

constant delay: $O(1)$

Enumerating Results of Spanners

Goal: given spanner S and document D, compute $S(\mathrm{D})$

$$\mathrm{D} = \mathtt{a\,g\,g\,a\,g\,c\,c\,a\,g\,c} \qquad S(\mathrm{D}) =$$

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

Preprocessing phase: compute a suitable data structure

linear preprocessing time: $O(|\mathrm{D}|)$, i.e., $f(S)\cdot O(|\mathrm{D}|)$

Enumeration phase: use this data structure to enumerate, without repetition, all tuples in $S(\mathrm{D})$

constant delay: $O(1)$, i.e., $f(S)$

Enumerating Results of Spanners

Goal: given spanner S and document D, compute $S(\mathrm{D})$

$$\mathrm{D} = \mathtt{a\,g\,g\,a\,g\,c\,c\,a\,g\,c} \qquad S(\mathrm{D}) =$$

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

Preprocessing phase: compute a suitable data structure
linear preprocessing time: $O(|\mathrm{D}|)$, i.e., $f(S)\cdot O(|\mathrm{D}|)$

Enumeration phase: use this data structure to enumerate, without repetition, all tuples in $S(\mathrm{D})$
constant delay: $O(1)$, i.e., $f(S)$

Results for regular spanners:

- [Florenzano, Riveros, Ugarte, Vansummeren, Vrgoc, PODS'18]: linear preprocessing $f(S)\cdot O(|\mathrm{D}|)$ and constant delay $f(S)$

Enumerating Results of Spanners

Goal: given spanner S and document D, compute $S(\mathrm{D})$

$$\mathrm{D} = \mathtt{a\,g\,g\,a\,g\,c\,c\,a\,g\,c} \qquad S(\mathrm{D}) =$$

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

Preprocessing phase: compute a suitable data structure
linear preprocessing time: $O(|\mathrm{D}|)$, i.e., $f(S)\cdot O(|\mathrm{D}|)$

Enumeration phase: use this data structure to enumerate, without repetition, all tuples in $S(\mathrm{D})$
constant delay: $O(1)$, i.e., $f(S)$

Results for regular spanners:

- [Florenzano, Riveros, Ugarte, Vansummeren, Vrgoc, PODS'18]: linear preprocessing $f(S)\cdot O(|\mathrm{D}|)$ and constant delay $f(S)$
- [Amarilli, Bourhis, Mengel, Niewerth, ICDT'19]: improvement to $f(S) = \mathrm{poly}(|M|)$ where S is represented by a variable-set automaton M

Decision Problems for Spanners

Query evaluation problems:

Decision Problems for Spanners

Query evaluation problems:

Testing: Given spanner S, document D, span tuple t, decide if $t \in S(\mathrm{D})$.

Decision Problems for Spanners

Query evaluation problems:

Testing: Given spanner S, document D, span tuple t, decide if $t \in S(\mathrm{D})$.

NonEmptiness: Given spanner S, document D, decide if $S(\mathrm{D}) \neq \emptyset$.

Decision Problems for Spanners

Query evaluation problems:

Testing: Given spanner S, document D, span tuple t, decide if $t \in S(\mathrm{D})$.

NonEmptiness: Given spanner S, document D, decide if $S(\mathrm{D}) \neq \emptyset$.

Static analysis problems:

Decision Problems for Spanners

Query evaluation problems:

Testing: Given spanner S, document D, span tuple t, decide if $t \in S(\mathrm{D})$.

NonEmptiness: Given spanner S, document D, decide if $S(\mathrm{D}) \neq \emptyset$.

Static analysis problems:

Satisfiability: Given spanner S, decide if there is a document D with $S(\mathrm{D}) \neq \emptyset$.

Decision Problems for Spanners

Query evaluation problems:

Testing: Given spanner S, document D, span tuple t, decide if $t \in S(\mathrm{D})$.

NonEmptiness: Given spanner S, document D, decide if $S(\mathrm{D}) \neq \emptyset$.

Static analysis problems:

Satisfiability: Given spanner S, decide if there is a document D with $S(\mathrm{D}) \neq \emptyset$.

Equivalence: Given spanners S_1, S_2, decide if $S_1(\mathrm{D}) = S_2(\mathrm{D})$ for every document D.

Decision Problems for Spanners

Query evaluation problems:

Testing: Given spanner S, document D, span tuple t, decide if $t \in S(\mathrm{D})$.

NonEmptiness: Given spanner S, document D, decide if $S(\mathrm{D}) \neq \emptyset$.

Static analysis problems:

Satisfiability: Given spanner S, decide if there is a document D with $S(\mathrm{D}) \neq \emptyset$.

Equivalence: Given spanners S_1, S_2, decide if $S_1(\mathrm{D}) = S_2(\mathrm{D})$ for every document D.

Containment: Given spanners S_1, S_2, decide if $S_1(\mathrm{D}) \subseteq S_2(\mathrm{D})$ for every document D.

Decision Problems for Spanners

Query evaluation problems:

Testing: Given spanner S, document D, span tuple t, decide if $t \in S(\mathrm{D})$.

NonEmptiness: Given spanner S, document D, decide if $S(\mathrm{D}) \neq \emptyset$.

Static analysis problems:

Satisfiability: Given spanner S, decide if there is a document D with $S(\mathrm{D}) \neq \emptyset$.

Equivalence: Given spanners S_1, S_2, decide if $S_1(\mathrm{D}) = S_2(\mathrm{D})$ for every document D.

Containment: Given spanners S_1, S_2, decide if $S_1(\mathrm{D}) \subseteq S_2(\mathrm{D})$ for every document D.

Hierarchicality: Given spanner S, decide if S is hierarchical.[1]

[1] I.e.: **balanced** brackets $\{{}^{x}\triangleright, \triangleleft^{x} : x \in \mathcal{X}\}$ in subword-marked words.
not balanced: ${}^{x}\triangleright\,{}^{y}\triangleright\triangleleft^{x}\triangleleft^{y}$ balanced: ${}^{x}\triangleright\,{}^{y}\triangleright\triangleleft^{y}\triangleleft^{x}\,{}^{z}\triangleright\,\triangleleft^{z}$

Decision Problems for Spanners

Query evaluation problems:

Testing: Given spanner S, document D, span tuple t, decide if $t \in S(\mathrm{D})$.

NonEmptiness: Given spanner S, document D, decide if $S(\mathrm{D}) \neq \emptyset$.

Static analysis problems:

Satisfiability: Given spanner S, decide if there is a document D with $S(\mathrm{D}) \neq \emptyset$.

Equivalence: Given spanners S_1, S_2, decide if $S_1(\mathrm{D}) = S_2(\mathrm{D})$ for every document D.

Containment: Given spanners S_1, S_2, decide if $S_1(\mathrm{D}) \subseteq S_2(\mathrm{D})$ for every document D.

Hierarchicality: Given spanner S, decide if S is hierarchical.

Known:

Decision Problems for Spanners

Query evaluation problems:

Testing: Given spanner S, document D, span tuple t, decide if $t \in S(\mathrm{D})$.

NonEmptiness: Given spanner S, document D, decide if $S(\mathrm{D}) \neq \emptyset$.

Static analysis problems:

Satisfiability: Given spanner S, decide if there is a document D with $S(\mathrm{D}) \neq \emptyset$.

Equivalence: Given spanners S_1, S_2, decide if $S_1(\mathrm{D}) = S_2(\mathrm{D})$ for every document D.

Containment: Given spanners S_1, S_2, decide if $S_1(\mathrm{D}) \subseteq S_2(\mathrm{D})$ for every document D.

Hierarchicality: Given spanner S, decide if S is hierarchical.

Known:

computational complexity of these problems for regular spanners & core spanners

Decision Problems for Spanners

Query evaluation problems:

Testing: Given spanner S, document D, span tuple t, decide if $t \in S(\mathrm{D})$.

NonEmptiness: Given spanner S, document D, decide if $S(\mathrm{D}) \neq \emptyset$.

Static analysis problems:

Satisfiability: Given spanner S, decide if there is a document D with $S(\mathrm{D}) \neq \emptyset$.

Equivalence: Given spanners S_1, S_2, decide if $S_1(\mathrm{D}) = S_2(\mathrm{D})$ for every document D.

Containment: Given spanners S_1, S_2, decide if $S_1(\mathrm{D}) \subseteq S_2(\mathrm{D})$ for every document D.

Hierarchicality: Given spanner S, decide if S is hierarchical.

Known:

computational complexity of these problems for regular spanners & core spanners

In short:

- regular spanners: mild complexity

Decision Problems for Spanners

Query evaluation problems:

Testing: Given spanner S, document D, span tuple t, decide if $t \in S(\mathrm{D})$.

NonEmptiness: Given spanner S, document D, decide if $S(\mathrm{D}) \neq \emptyset$.

Static analysis problems:

Satisfiability: Given spanner S, decide if there is a document D with $S(\mathrm{D}) \neq \emptyset$.

Equivalence: Given spanners S_1, S_2, decide if $S_1(\mathrm{D}) = S_2(\mathrm{D})$ for every document D.

Containment: Given spanners S_1, S_2, decide if $S_1(\mathrm{D}) \subseteq S_2(\mathrm{D})$ for every document D.

Hierarchicality: Given spanner S, decide if S is hierarchical.

Known:
computational complexity of these problems for regular spanners & core spanners

In short:

- regular spanners: mild complexity
- core spanners: intractable

Decision Problems for Spanners

Query evaluation problems:

Testing: Given spanner S, document D, span tuple t, decide if $t \in S(\mathrm{D})$.

NonEmptiness: Given spanner S, document D, decide if $S(\mathrm{D}) \neq \emptyset$.

Static analysis problems:

Satisfiability: Given spanner S, decide if there is a document D with $S(\mathrm{D}) \neq \emptyset$.

Equivalence: Given spanners S_1, S_2, decide if $S_1(\mathrm{D}) = S_2(\mathrm{D})$ for every document D.

Containment: Given spanners S_1, S_2, decide if $S_1(\mathrm{D}) \subseteq S_2(\mathrm{D})$ for every document D.

Hierarchicality: Given spanner S, decide if S is hierarchical.

Known: Satisfiability is

Decision Problems for Spanners

Query evaluation problems:

Testing: Given spanner S, document D, span tuple t, decide if $t \in S(\mathrm{D})$.

NonEmptiness: Given spanner S, document D, decide if $S(\mathrm{D}) \neq \emptyset$.

Static analysis problems:

Satisfiability: Given spanner S, decide if there is a document D with $S(\mathrm{D}) \neq \emptyset$.

Equivalence: Given spanners S_1, S_2, decide if $S_1(\mathrm{D}) = S_2(\mathrm{D})$ for every document D.

Containment: Given spanners S_1, S_2, decide if $S_1(\mathrm{D}) \subseteq S_2(\mathrm{D})$ for every document D.

Hierarchicality: Given spanner S, decide if S is hierarchical.

Known: Satisfiability is

- in $O(|M|)$ if S is a regular spanner represented by a var.-set automaton M

Decision Problems for Spanners

Query evaluation problems:

Testing: Given spanner S, document D, span tuple t, decide if $t \in S(\mathrm{D})$.

NonEmptiness: Given spanner S, document D, decide if $S(\mathrm{D}) \neq \emptyset$.

Static analysis problems:

Satisfiability: Given spanner S, decide if there is a document D with $S(\mathrm{D}) \neq \emptyset$.

Equivalence: Given spanners S_1, S_2, decide if $S_1(\mathrm{D}) = S_2(\mathrm{D})$ for every document D.

Containment: Given spanners S_1, S_2, decide if $S_1(\mathrm{D}) \subseteq S_2(\mathrm{D})$ for every document D.

Hierarchicality: Given spanner S, decide if S is hierarchical.

Known: Satisfiability is

- in $O(|M|)$ if S is a regular spanner represented by a var.-set automaton M
- PSpace-complete for core spanners S [Freydenberger, Holldack, ICDT'16]

Decision Problems for Spanners

Query evaluation problems:

Testing: Given spanner S, document D, span tuple t, decide if $t \in S(\mathrm{D})$.

NonEmptiness: Given spanner S, document D, decide if $S(\mathrm{D}) \neq \emptyset$.

Static analysis problems:

Satisfiability: Given spanner S, decide if there is a document D with $S(\mathrm{D}) \neq \emptyset$.

Equivalence: Given spanners S_1, S_2, decide if $S_1(\mathrm{D}) = S_2(\mathrm{D})$ for every document D.

Containment: Given spanners S_1, S_2, decide if $S_1(\mathrm{D}) \subseteq S_2(\mathrm{D})$ for every document D.

Hierarchicality: Given spanner S, decide if S is hierarchical.

Known: Satisfiability is

- in $O(|M|)$ if S is a regular spanner represented by a var.-set automaton M
- PSpace-complete for core spanners S [Freydenberger, Holldack, ICDT'16]
  Idea: intersection non-emptiness of regular languages is PSpace-hard

Decision Problems for Spanners

Query evaluation problems:

Testing: Given spanner S, document D, span tuple t, decide if $t \in S(\mathrm{D})$.

NonEmptiness: Given spanner S, document D, decide if $S(\mathrm{D}) \neq \emptyset$.

Static analysis problems:

Satisfiability: Given spanner S, decide if there is a document D with $S(\mathrm{D}) \neq \emptyset$.

Equivalence: Given spanners S_1, S_2, decide if $S_1(\mathrm{D}) = S_2(\mathrm{D})$ for every document D.

Containment: Given spanners S_1, S_2, decide if $S_1(\mathrm{D}) \subseteq S_2(\mathrm{D})$ for every document D.

Hierarchicality: Given spanner S, decide if S is hierarchical.

Known: Satisfiability is

- in $O(|M|)$ if S is a regular spanner represented by a var.-set automaton M
- PSpace-complete for core spanners S [Freydenberger, Holldack, ICDT'16]
  Idea: intersection non-emptiness of regular languages is PSpace-hard

$$r_1 \cap \cdots \cap r_n \neq \emptyset \iff \varsigma^{=}_{x_1,\ldots,x_n}({}^{x_1}\triangleright\, r_1 \,\triangleleft^{x_1} \cdots {}^{x_n}\triangleright\, r_n \triangleleft^{x_n}) \text{ is satisfiable}$$

Overview

Recent Developments

- Evaluating spanners on
  compressed representations of documents

Recent Developments

- Evaluating spanners on
  compressed representations of documents

- Beyond regular spanners:

  Search for classes of spanners that

  - are more expressive than the regular spanners
  - but avoid the high complexity of the class of core spanners

Beyond regular spanners:

Refl-Spanners

Ref-Words and Ref-Languages

A ref-word over Σ and $\mathcal{X}$ is ...

... a subword-marked word w over $\Sigma' = \mathcal{X} \cup \Sigma \cup \{{}^{x}\triangleright, \triangleleft^{x} : x \in \mathcal{X}\}$ such that

$$\forall x \in \mathcal{X}: \quad w = w_1\, x\, w_2 \quad \Longrightarrow \quad w_1 = v_1\, {}^{x}\triangleright\, v_2 \triangleleft^{x}\, v_3 \quad \text{and} \quad x \notin v_2\,.$$

Ref-Words and Ref-Languages

A ref-word over Σ and $\mathcal{X}$ is ...

... a subword-marked word w over $\Sigma' = \mathcal{X} \cup \Sigma \cup \{{}^{x}\triangleright, \triangleleft^{x} : x \in \mathcal{X}\}$ such that

$$\forall x \in \mathcal{X}: \quad w = w_1\, x\, w_2 \quad \Longrightarrow \quad w_1 = v_1\, {}^{x}\triangleright\, v_2 \triangleleft^{x}\, v_3 \quad \text{and} \quad x \notin v_2\,.$$

Example

ref-words	subword-marked words, but no ref-words
$\text{ag}\, {}^{x}\triangleright\, \text{ag} \triangleleft^{x}\, \text{c}\, {}^{y}\triangleright\, \text{xaa} \triangleleft^{y}\, \text{y}$	

Ref-Words and Ref-Languages

A ref-word over Σ and $\mathcal{X}$ is ...

... a subword-marked word w over $\Sigma' = \mathcal{X} \cup \Sigma \cup \{{}^{x}\triangleright, \triangleleft^{x} : x \in \mathcal{X}\}$ such that

$$\forall x \in \mathcal{X}: \quad w = w_1\, x\, w_2 \quad \Longrightarrow \quad w_1 = v_1\, {}^{x}\triangleright\, v_2 \triangleleft^{x}\, v_3 \quad \text{and} \quad x \notin v_2\,.$$

Example

ref-words	subword-marked words, but no ref-words
ag ${}^{x}\triangleright$ ag $\triangleleft^{x}$ c ${}^{y}\triangleright$ xaa $\triangleleft^{y}$ y	
a ${}^{x}\triangleright$ ag ${}^{y}\triangleright$ ag $\triangleleft^{x}$ a $\triangleleft^{y}$ xy	

Ref-Words and Ref-Languages

A ref-word over Σ and $\mathcal{X}$ is ...

... a subword-marked word w over $\Sigma' = \mathcal{X} \cup \Sigma \cup \{{}^{x}\triangleright, \triangleleft^{x} : x \in \mathcal{X}\}$ such that

$$\forall x \in \mathcal{X}: \quad w = w_1\, x\, w_2 \quad \Longrightarrow \quad w_1 = v_1\ {}^{x}\triangleright\ v_2 \triangleleft^{x}\ v_3 \quad \text{and} \quad x \notin v_2\,.$$

Example

ref-words	subword-marked words, but no ref-words
ag ${}^{x}\triangleright$ ag $\triangleleft^{x}$ c ${}^{y}\triangleright$ xaa $\triangleleft^{y}$ y	axg ${}^{x}\triangleright$ ag $\triangleleft^{x}$ c ${}^{y}\triangleright$ xaa $\triangleleft^{y}$ y
a ${}^{x}\triangleright$ ag ${}^{y}\triangleright$ ag $\triangleleft^{x}$ a $\triangleleft^{y}$ xy	

Ref-Words and Ref-Languages

A ref-word over Σ and $\mathcal{X}$ is ...

... a subword-marked word w over $\Sigma' = \mathcal{X} \cup \Sigma \cup \{{}^{x}\triangleright, \triangleleft^{x} : x \in \mathcal{X}\}$ such that

$$\forall x \in \mathcal{X}: \quad w = w_1\, x\, w_2 \quad \Longrightarrow \quad w_1 = v_1\ {}^{x}\triangleright\ v_2 \triangleleft^{x}\ v_3 \quad \text{and} \quad x \notin v_2\,.$$

Example

ref-words	subword-marked words, but no ref-words
ag ${}^{x}\triangleright$ ag $\triangleleft^{x}$ c ${}^{y}\triangleright$ xaa $\triangleleft^{y}$ y	axg ${}^{x}\triangleright$ ag $\triangleleft^{x}$ c ${}^{y}\triangleright$ xaa $\triangleleft^{y}$ y
a ${}^{x}\triangleright$ ag ${}^{y}\triangleright$ ag $\triangleleft^{x}$ a $\triangleleft^{y}$ xy	aa ${}^{x}\triangleright$ ag $\triangleleft^{x}$ c ${}^{y}\triangleright$ ya$\triangleleft^{y}$

Ref-Words and Ref-Languages

A ref-word over Σ and $\mathcal{X}$ is ...

... a subword-marked word w over $\Sigma' = \mathcal{X} \cup \Sigma \cup \{{}^{x}\triangleright, \triangleleft^{x} : x \in \mathcal{X}\}$ such that

$$\forall x \in \mathcal{X}: \quad w = w_1\, x\, w_2 \quad \Longrightarrow \quad w_1 = v_1\, {}^{x}\triangleright\, v_2 \triangleleft^{x}\, v_3 \quad \text{and} \quad x \notin v_2\,.$$

Example

ref-words	subword-marked words, but no ref-words
ag ${}^{x}\triangleright$ ag $\triangleleft^{x}$ c ${}^{y}\triangleright$ xaa $\triangleleft^{y}$ y	axg ${}^{x}\triangleright$ ag $\triangleleft^{x}$ c ${}^{y}\triangleright$ xaa $\triangleleft^{y}$ y
a ${}^{x}\triangleright$ ag ${}^{y}\triangleright$ ag $\triangleleft^{x}$ a $\triangleleft^{y}$ xy	aa ${}^{x}\triangleright$ ag $\triangleleft^{x}$ c ${}^{y}\triangleright$ ya$\triangleleft^{y}$

A ref-language is a set of ref-words.

Dereferencing Ref-Words

Dereference-Function

Let w be a ref-word (over Σ and $\mathcal{X}$).

$\mathfrak{d}(w) =$ replace all x by whatever is in ${}^{x}\triangleright \cdots \triangleleft^{x}$.

Dereferencing Ref-Words

Dereference-Function

Let w be a ref-word (over Σ and $\mathcal{X}$).

$\mathfrak{d}(w) =$ replace all x by whatever is in ${}^{x}\triangleright \cdots \triangleleft^{x}$.

Example

${}^{x_1}\triangleright\, a\, {}^{x_2}\triangleright\, aa \triangleleft^{x_1}\, cx_1\, {}^{x_3}\triangleright\, ac \triangleleft^{x_2}\, x_2 a x_1 \triangleleft^{x_3}$

Dereferencing Ref-Words

Dereference-Function

Let w be a ref-word (over Σ and $\mathcal{X}$).

$\mathfrak{d}(w) =$ replace all x by whatever is in ${}^{x}\triangleright \cdots \triangleleft^{x}$.

Example

${}^{x_1}\triangleright\, a\, {}^{x_2}\triangleright\, aa \triangleleft^{x_1}\, c x_1\, {}^{x_3}\triangleright\, ac \triangleleft^{x_2}\, x_2 a x_1 \triangleleft^{x_3}$

Dereferencing Ref-Words

Dereference-Function

Let w be a ref-word (over Σ and $\mathcal{X}$).

$\mathfrak{d}(w)$ = replace all x by whatever is in ${}^{x}\triangleright \cdots \triangleleft^{x}$.

Example

${}^{x_1}\triangleright$ a ${}^{x_2}\triangleright$ aa $\triangleleft^{x_1}$ caaa ${}^{x_3}\triangleright$ ac $\triangleleft^{x_2}$ $x_2$aaaa$\triangleleft^{x_3}$

Dereferencing Ref-Words

Dereference-Function

Let w be a ref-word (over Σ and $\mathcal{X}$).

$\mathfrak{d}(w) =$ replace all x by whatever is in ${}^{x}\triangleright \cdots \triangleleft^{x}$.

Example

${}^{x_1}\triangleright$ a ${}^{x_2}\triangleright$ aa $\triangleleft^{x_1}$ caaa ${}^{x_3}\triangleright$ ac $\triangleleft^{x_2}$ $x_2$aaaa$\triangleleft^{x_3}$

Dereferencing Ref-Words

Dereference-Function

Let w be a ref-word (over Σ and $\mathcal{X}$).

$\mathfrak{d}(w)$ = replace all x by whatever is in ${}^{x}\triangleright \cdots \triangleleft^{x}$.

Example

${}^{x_1}\triangleright$ a ${}^{x_2}\triangleright$ aa $\triangleleft^{x_1}$ caaa ${}^{x_3}\triangleright$ ac $\triangleleft^{x_2}$ aacaaaacaaaa$\triangleleft^{x_3}$

Dereferencing Ref-Words

Dereference-Function

Let w be a ref-word (over Σ and $\mathcal{X}$).

$\mathfrak{d}(w) =$ replace all x by whatever is in ${}^{x}\triangleright \cdots \triangleleft^{x}$.

Example

${}^{x_1}\triangleright$ a ${}^{x_2}\triangleright$ aa $\triangleleft^{x_1}$ caaa ${}^{x_3}\triangleright$ ac $\triangleleft^{x_2}$ aacaaaacaaaa$\triangleleft^{x_3}$

ref-word	subword-marked word	document	span-tuple
w	$\mathfrak{d}(w)$	$\mathfrak{e}(\mathfrak{d}(w))$	$\mathrm{st}(\mathfrak{d}(w))$

Dereferencing Ref-Words

Dereference-Function

Let w be a ref-word (over Σ and $\mathcal{X}$).

$\mathfrak{d}(w)$ = replace all x by whatever is in ${}^{x}\triangleright \cdots \triangleleft^{x}$.

Example

${}^{x_1}\triangleright$ a ${}^{x_2}\triangleright$ aa $\triangleleft^{x_1}$ caaa ${}^{x_3}\triangleright$ ac $\triangleleft^{x_2}$ aacaaaacaaaa$\triangleleft^{x_3}$

ref-word	subword-marked word	document	span-tuple
w	$\mathfrak{d}(w)$	$\mathfrak{e}(\mathfrak{d}(w))$	st$(\mathfrak{d}(w))$

Example

$$
\begin{aligned}
w &= {}^{x_1}\triangleright \text{ a } {}^{x_2}\triangleright \text{ aa } \triangleleft^{x_1} \text{ c}x_1 \ {}^{x_3}\triangleright \text{ ac } \triangleleft^{x_2} x_2\text{a}x_1 \triangleleft^{x_3} \\
\mathfrak{d}(w) &= {}^{x_1}\triangleright \text{ a } {}^{x_2}\triangleright \text{ aa } \triangleleft^{x_1} \text{ caaa } {}^{x_3}\triangleright \text{ ac } \triangleleft^{x_2} \text{ aacaaaacaaaa} \triangleleft^{x_3} \\
\mathfrak{e}(\mathfrak{d}(w)) &= \text{aaacaaaacaacaaaacaaaa} \\
\text{st}(\mathfrak{d}(w)) &= ([1,4\rangle, [2,10\rangle, [8,22\rangle)
\end{aligned}
$$

Refl-Spanners

Refl-Spanners

A spanner S is a refl-spanner if $S = [\![\mathfrak{d}(L)]\!]$ for a regular ref-language L.

Refl-Spanners

Refl-Spanners

A spanner S is a refl-spanner if $S = [\![\mathfrak{d}(L)]\!]$ for a regular ref-language L.

Example

Regular spanner: $r = \Sigma^* \, {}^{x}\!\triangleright \, (a \vee g)^+ \triangleleft^{x} \, \Sigma^* c \Sigma^* \, {}^{y}\!\triangleright \, (a \vee g)^+ \triangleleft^{y} \, \Sigma^*$

String-equ. sel.: $\zeta^{=}_{x,y}$

Refl-Spanners

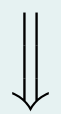
Refl-Spanners

A spanner S is a refl-spanner if $S = [\![\mathfrak{d}(L)]\!]$ for a regular ref-language L.

Example

Regular spanner: $r = \Sigma^* \, {}^{x}\!\triangleright \, (a \vee g)^+ \triangleleft^{x} \, \Sigma^* c \Sigma^* \, {}^{y}\!\triangleright \, (a \vee g)^+ \triangleleft^{y} \, \Sigma^*$

String-equ. sel.: $\varsigma^{=}_{x,y}$



Refl-spanner: $r = \Sigma^* \, {}^{x}\!\triangleright \, (a \vee g)^+ \triangleleft^{x} \, \Sigma^* c \Sigma^* \, {}^{y}\!\triangleright \, x \triangleleft^{y} \, \Sigma^*$

Expressive Power of Refl-Spanners

Theorem (Schmid, Schweikardt, ICDT'21)

- Every regular spanner is a refl-spanner.

Expressive Power of Refl-Spanners

Theorem (Schmid, Schweikardt, ICDT'21)

- Every regular spanner is a refl-spanner.
- Every reference-bounded[1] refl-spanner is a core spanner.

---

[1] no unbounded number of references, like in ${}^{x}\triangleright\,(a \vee g)^{*}\,\triangleleft^{x}\,x^{*}$

Expressive Power of Refl-Spanners

Theorem (Schmid, Schweikardt, ICDT'21)

- Every regular spanner is a refl-spanner.
- Every reference-bounded[1] refl-spanner is a core spanner.
- Core spanners with non-overlapping string-equality selections can be expressed by refl-spanners with span fusions.[2]

---

[1] no unbounded number of references, like in ${}^{x}\triangleright\,(a \vee g)^* \triangleleft^{x} x^*$

[2] $\biguplus_{\{x_1,x_2\}\to z}$(

x_1	x_2	x_3
$[1,4\rangle$	$[4,7\rangle$	$[1,10\rangle$
$[3,5\rangle$	$[5,8\rangle$	$[4,7\rangle$
$[1,3\rangle$	$[3,10\rangle$	$[2,4\rangle$

) =

z	x_3
$[1,7\rangle$	$[1,10\rangle$
$[3,8\rangle$	$[4,7\rangle$
$[1,10\rangle$	$[2,4\rangle$

Complexity of Evaluation of Refl-Spanners

Problem	Regular sp.	Refl-sp.[4]	Core sp.[6]
Testing	$O(\|w\|\|M\| \log \|\mathcal{X}\|)$	$O(\|w\|\|M\| \log \|\mathcal{X}\|)$	NP-c
NonEmptiness	$O(\|w\|\|M\|)$	NP-c	NP-h
Satisfiability	$O(\|M\|)$	$O(\|M\|)$	PSpace-c
Containment	PSpace-c[3]	PSpace-c[5]	undec.
Equivalence	PSpace-c[3]	PSpace-c[5]	undec.
Hierarchicality	$O(\|M\|\|\mathcal{X}\|^3)$	$O(\|M\|\|\mathcal{X}\|^3)$	PSpace-c

[3] Maturana et al., PODS'18.
[4] Schmid, Schweikardt, ICDT'21.
[5] For strongly reference extracting refl-spanners.
[6] Freydenberger, Holldack, ICDT'16.

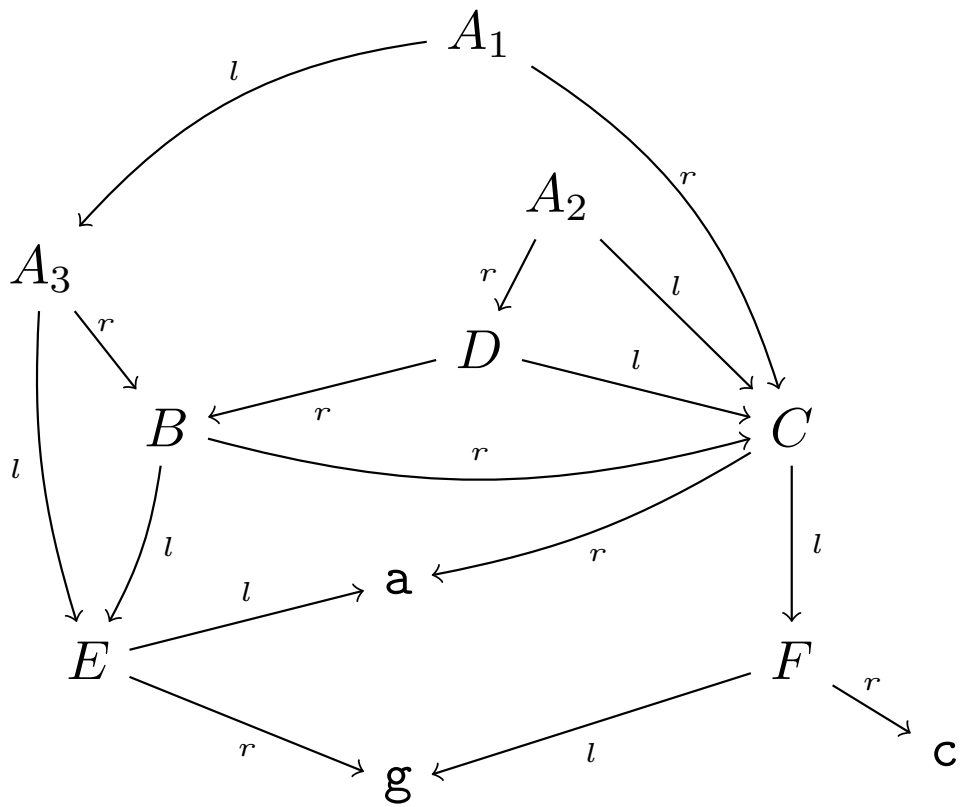
Evaluating spanners on compressed representations of documents

SLP-Represented Document Databases

$$\mathsf{DDB} \;=\; \{\, \underbrace{\mathtt{agaggcagca}}_{\mathsf{D_1}},\; \underbrace{\mathtt{gcagcaaggca}}_{\mathsf{D_2}},\; \underbrace{\mathtt{agaggca}}_{\mathsf{D_3}} \,\}$$

SLP-Represented Document Databases

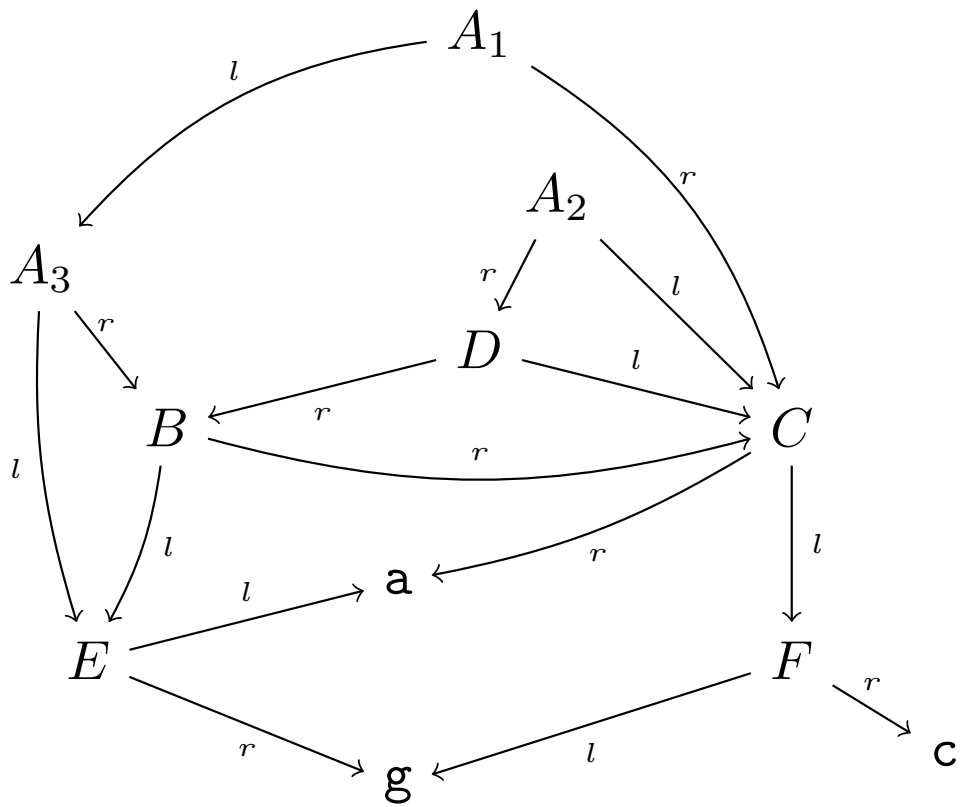
$$\mathsf{DDB} \;=\; \{\; \underbrace{\mathtt{agaggcagca}}_{\mathsf{D_1}} \;,\; \underbrace{\mathtt{gcagcaaggca}}_{\mathsf{D_2}} \;,\; \underbrace{\mathtt{agaggca}}_{\mathsf{D_3}} \;\}$$



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

SLP-Represented Document Databases

$$\mathsf{DDB} \;=\; \{\; \underbrace{\mathtt{agaggcagca}}_{\mathsf{D_1}} \;,\; \underbrace{\mathtt{gcagcaaggca}}_{\mathsf{D_2}} \;,\; \underbrace{\mathtt{agaggca}}_{\mathsf{D_3}} \;\}$$



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

$A_1 \to A_3 C \to EBC \to \cdots \to D_1$
$A_2 \to \cdots \to D_2$

Complex Document Editing

CDE-Expression

An expression over documents, using
typical text-editor copy-and-paste operations.

Complex Document Editing

CDE-Expression

An expression over documents, using
typical text-editor copy-and-paste operations.

Example

$e = \text{concat}(D_1, \text{insert}(D_3, \text{extract}(D_7, 5, 21), 12))$

Complex Document Editing

CDE-Expression

An expression over documents, using
typical text-editor copy-and-paste operations.

Example

$e = \text{concat}(\text{D}_1, \text{insert}(\text{D}_3, \text{extract}(\text{D}_7, 5, 21), 12))$

Main Result (Schmid, Schweikardt, PODS'21, PODS'22):

- Represent a document database DDB by a suitably balanced SLP $\mathcal{S}$.
- For a given regular spanner R use time $O(|\mathcal{S}|)$ to build a data structure that enables to efficiently enumerate the result of R on each document in DDB
- add the document D_e described by a CDE-expression e into this data structure in time $\approx O(\log |D_e|)$

More Precisely: Main Result

Theorem[7] (Schmid, Schweikardt, PODS'21, PODS'22)

Let DDB be represented by a strongly balanced[8] SLP $\mathcal{S}$, and let Q be a finite set of regular document spanners.

Enumeration: In time $O(|Q| \cdot |\mathcal{S}|)$, we can compute a data structure such that, for any $R \in Q$ and $D \in \text{DDB}$, we can enumerate $[\![R]\!](D)$ with delay $O(\log |D|)$.

[7] All bounds in data complexity.

[8] A property that implies that paths are bounded logarithmically in the size of the data.

More Precisely: Main Result

Theorem[7] (Schmid, Schweikardt, PODS'21, PODS'22)

Let DDB be represented by a strongly balanced[8] SLP $\mathcal{S}$, and let Q be a finite set of regular document spanners.

Enumeration: In time $O(|Q|\cdot|\mathcal{S}|)$, we can compute a data structure such that, for any $R \in Q$ and $D \in \text{DDB}$, we can enumerate $[\![R]\!](D)$ with delay $O(\log|D|)$.

Updates: Given a CDE-expression e over DDB, we can extend $\mathcal{S}$ into a strongly balanced SLP $\mathcal{S}'$ for

$$\text{DDB}' \;=\; \text{DDB} \cup \{\text{eval}(e)\}$$

in time $O(|e|\cdot \log d)$,

where d is the maximum length of any intermediate document represented by a subexpression of e.
We can update the data structure in time $O(|Q|\cdot|e|\cdot \log d)$ such that afterwards, upon input of any D $\in$ DDB$'$ and any $R \in Q$, we can enumerate $[\![R]\!](D)$ with delay $O(\log|D|)$.

---

[7] All bounds in data complexity.
[8] A property that implies that paths are bounded logarithmically in the size of the data.

Thank you very much

for your attention!