Skyline Operators for Document Spanners

Antoine Amarilli[1], Sébastien Labbé[2], Benny Kimelfeld[3], Stefan Mengel[4]

March 27th, 2024

[1]Télécom Paris

[2]Technion

[3]École normale supérieure

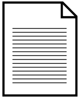
[4]CNRS CRIL

Document spanners

We use document spanners, a declarative formalism for information extraction tasks

Document spanners

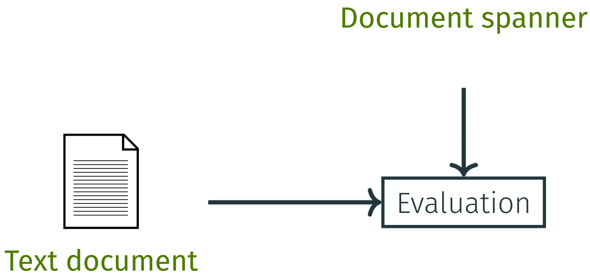
We use **document spanners**, a declarative formalism for information extraction tasks



Text document

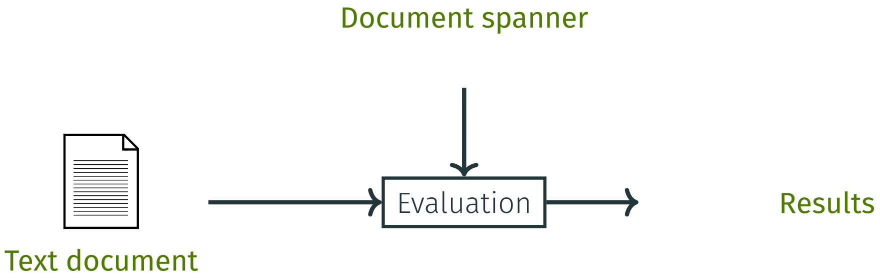
Document spanners

We use **document spanners**, a declarative formalism for information extraction tasks



Document spanners

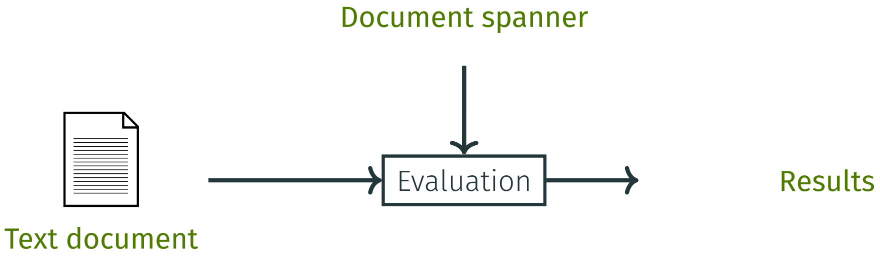
We use **document spanners**, a declarative formalism for information extraction tasks



Document spanners

We use document spanners, a declarative formalism for information extraction tasks

“Extract all email addresses in the document”

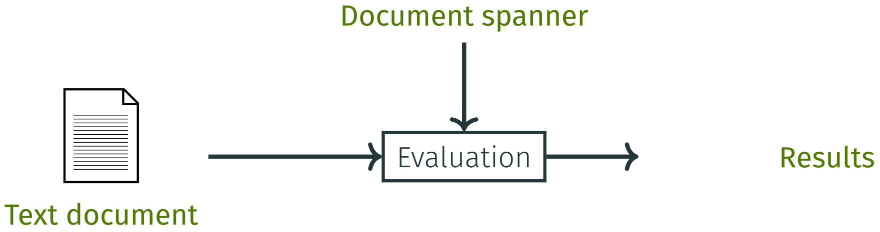


Document spanners

We use **document spanners**, a declarative formalism for information extraction tasks

$$\texttt{[a-z]}^{+}\ \texttt{@}\ \texttt{[a-z]}^{+}\ \texttt{.}\ \texttt{[a-z]}^{+}$$

“Extract all email addresses in the document”



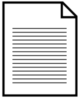
Document spanners

We use **document spanners**, a declarative formalism for information extraction tasks

$$\vdash_{\mathit{email}}\ \texttt{[a-z]}^{+}\ \texttt{@}\ \texttt{[a-z]}^{+}\ \texttt{.}\ \texttt{[a-z]}^{+}\ \dashv_{\mathit{email}}$$

“Extract all email addresses in the document”

Document spanner



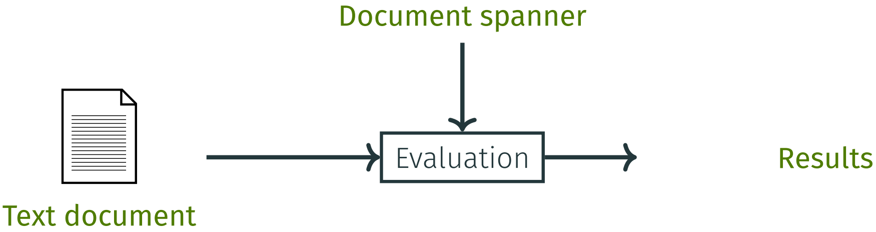
Text document

$\longrightarrow$ $\boxed{\text{Evaluation}}$ $\longrightarrow$ Results

We use **document spanners**, a declarative formalism for information extraction tasks

$$\Sigma^* \ \textvisiblespace \ \vdash_{\mathit{email}} \ \texttt{[a-z]}^+ \ \texttt{@} \ \texttt{[a-z]}^+ \ \texttt{.} \ \texttt{[a-z]}^+ \ \dashv_{\mathit{email}} \ \textvisiblespace \ \Sigma^*$$

"Extract all email addresses in the document"

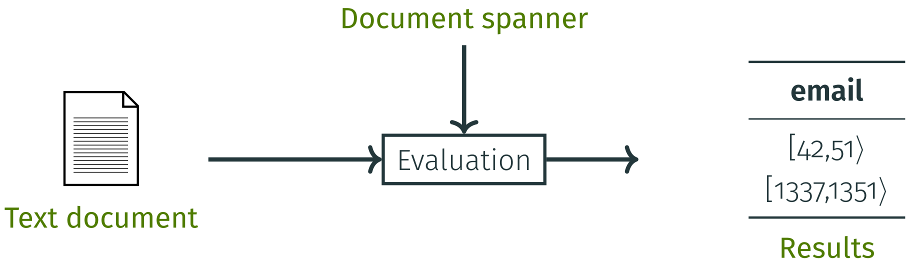


Document spanners

We use **document spanners**, a declarative formalism for information extraction tasks

$$\Sigma^* \ \sqcup \ \vdash_{\mathit{email}} \ \texttt{[a-z]}^+ \ \texttt{@} \ \texttt{[a-z]}^+ \ \texttt{.} \ \texttt{[a-z]}^+ \ \dashv_{\mathit{email}} \ \sqcup \ \Sigma^*$$

“Extract all email addresses in the document”



Document spanners

We use document spanners, a declarative formalism for information extraction tasks

"Extract all last names with possibly a phone number"

Document spanner

Results

Text document

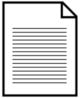
Document spanners

We use **document spanners**, a declarative formalism for information extraction tasks

$$\vdash_{name}\ \texttt{[A-Z][a-z]}^{+}\ \dashv_{name}$$

“Extract all last names with possibly a phone number”

Document spanner



Text document

Evaluation

Results

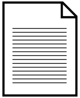
Document spanners

We use document spanners, a declarative formalism for information extraction tasks

$$\vdash_{name} \texttt{[A-Z][a-z]}^{+} \dashv_{name} \left(\epsilon \mid \sqcup \vdash_{phone} \qquad\qquad \dashv_{phone} \right)$$

"Extract all last names with possibly a phone number"

Document spanner



Text document

Evaluation

Results

Document spanners

We use **document spanners**, a declarative formalism for information extraction tasks

$$\vdash_{name}\ \texttt{[A-Z][a-z]}^{+}\ \dashv_{name}\ (\ \epsilon\ |\ \sqcup\ \vdash_{phone}\ \texttt{[0-9]}^{+}\ \dashv_{phone})$$

"Extract all last names with possibly a phone number"

Document spanner

Results

Text document

Document spanners

We use **document spanners**, a declarative formalism for information extraction tasks

$$\Sigma^* \ \vdash_{name} \texttt{[A-Z][a-z]}^+ \dashv_{name} (\ \epsilon \mid \sqcup \ \vdash_{phone} \texttt{[0-9]}^+ \dashv_{phone}) \sqcup \Sigma^*$$

"Extract all last names with possibly a phone number"

Document spanner

Results

Text document

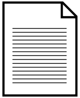
Document spanners

We use **document spanners**, a declarative formalism for information extraction tasks

$$\Sigma^* \vdash_{name} \texttt{[A-Z][a-z]}^{+} \dashv_{name} (\, \epsilon \mid \sqcup \vdash_{phone} \texttt{[0-9]}^{+} \dashv_{phone}) \sqcup \Sigma^*$$

"Extract all last names with possibly a phone number"

Document spanner



Text document

Evaluation

name	phone
$[145,152\rangle$	–
$[2034,2048\rangle$	$[2049,2059\rangle$

Results

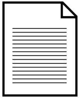
Document spanners

We use **document spanners**, a declarative formalism for information extraction tasks

$$\Sigma^* \ \vdash_{name} \texttt{[A-Z][a-z]}^+ \dashv_{name} (\ \epsilon \mid \sqcup \ \vdash_{phone} \texttt{[0-9]}^+ \dashv_{phone}) \sqcup \Sigma^*$$

"Extract all last names with possibly a phone number"

Document spanner



Text document

Evaluation

name	phone
[145,152⟩	–
[2034,2048⟩	[2049,2059⟩

Results

- Several **formalisms** to express document spanners
  - → Focus: **regular spanners** expressed as **Variable-Set Automata** (VAs) or regex-formulas

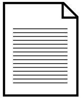
Document spanners

We use **document spanners**, a declarative formalism for information extraction tasks

$$\Sigma^* \ \vdash_{\mathit{name}} \texttt{[A-Z][a-z]}^{+} \dashv_{\mathit{name}} \left(\epsilon \mid \sqcup \ \vdash_{\mathit{phone}} \texttt{[0-9]}^{+} \dashv_{\mathit{phone}} \right) \sqcup \Sigma^*$$

"Extract all last names with possibly a phone number"

Document spanner



Text document

Evaluation

name	**phone**
[145,152⟩	–
[2034,2048⟩	[2049,2059⟩

Results

- Several **formalisms** to express document spanners
  - → Focus: **regular spanners** expressed as **Variable-Set Automata** (VAs) or regex-formulas
- Well-studied task: **efficient evaluation**, including enumeration algorithms

Maximal matches

Standard semantics: extract all mappings of the spanner variables. But...

Maximal matches

Standard semantics: extract all mappings of the spanner variables. But...

A3 **You**

How can I obtain all matches of a regular expression in a string?

ChatGPT

To obtain all matches of a regular expression in a string, you can use the appropriate function provided by your programming language or library.

python Copy code

```python
pattern = r'your_pattern_here'
input_string = "your_input_string_here"
matches = re.findall(pattern, input_string)
```

Maximal matches

Standard semantics: extract **all mappings** of the spanner variables. But…

`re.findall(pattern, string, flags=0)`

Return all non-overlapping matches of *pattern* in *string*, as a list of strings or tuples. The *string* is scanned left-to-right, and matches are returned in the order found. Empty matches are included in the result.

The result depends on the number of capturing groups in the pattern. If there are no groups, return a list of strings matching the whole pattern. If there is exactly one group, return a list of strings matching that group. If multiple groups are present, return a list of tuples of strings matching the groups. Non-capturing groups do not affect the form of the result.

```
>>> re.findall(r'\bf[a-z]*', 'which foot or hand fell fastest')
['foot', 'fell', 'fastest']
```

python — Copy code

```python
pattern = r'your_pattern_here'
input_string = "your_input_string_here"
matches = re.findall(pattern, input_string)
```

Maximal matches

Specifically, we may want:

- *"Extract all email addresses"*

  $\Sigma^* \ \sqcup \ \vdash_{\textit{email}} \ \texttt{[a-z]}^+ \ \texttt{@} \ \texttt{[a-z]}^+ \ \texttt{.} \ \texttt{[a-z]}^+ \ \dashv_{\textit{email}} \ \sqcup \ \Sigma^*$

Maximal matches

Specifically, we may want:

- *"Extract all email addresses"*, without worrying about delimiters
  $\Sigma^* \ \sqcup \ \vdash_{\textit{email}} \ \texttt{[a-z]}^+ \ \texttt{@} \ \texttt{[a-z]}^+ \ \texttt{.} \ \texttt{[a-z]}^+ \ \dashv_{\textit{email}} \ \sqcup \ \Sigma^*$

Maximal matches

Specifically, we may want:

- *“Extract all email addresses”*, without worrying about delimiters
  $\Sigma^* \quad \vdash_{\mathit{email}}$ `[a-z]`$^+$ `@` `[a-z]`$^+$ `.` `[a-z]`$^+$ $\dashv_{\mathit{email}} \quad \Sigma^*$

Maximal matches

Specifically, we may want:

- *"Extract all* ***maximal*** *email addresses"*, without worrying about delimiters

  $\Sigma^* \quad \vdash_{\textit{email}}\ \texttt{[a-z]}^+\ \texttt{@}\ \texttt{[a-z]}^+\ \texttt{.}\ \texttt{[a-z]}^+\ \dashv_{\textit{email}} \quad \Sigma^*$

Maximal matches

Specifically, we may want:

- *"Extract all **maximal** email addresses"*, without worrying about delimiters
  $\Sigma^* \quad \vdash_{\mathit{email}}\ \texttt{[a-z]}^+\ \texttt{@}\ \texttt{[a-z]}^+\ \texttt{.}\ \texttt{[a-z]}^+\ \dashv_{\mathit{email}} \quad \Sigma^*$

- *"Extract all **maximal matches of** last names with possibly a phone number"*
  $\rightarrow$ If the number is given, do not extract a match without it

Maximal matches

Specifically, we may want:

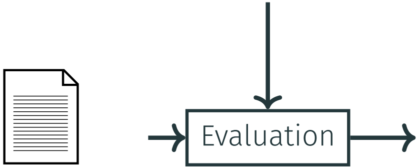
- *"Extract all **maximal** email addresses"*, without worrying about delimiters
  $\Sigma^* \quad \vdash_{\textit{email}}$ `[a-z]`$^+$ `@` `[a-z]`$^+$ `.` `[a-z]`$^+$ $\dashv_{\textit{email}} \quad \Sigma^*$
- *"Extract all **maximal matches of** last names with possibly a phone number"*
  $\rightarrow$ If the number is given, do not extract a match without it

Skyline under a domination relation: the results which are **maximal**, i.e., not dominated

Naive skyline computation

$$\Sigma^* \vdash_{email} \texttt{[a-z]}^+ \texttt{@} \texttt{[a-z]}^+ \texttt{.} \texttt{[a-z]}^+ \dashv_{email} \Sigma^*$$

Document spanner

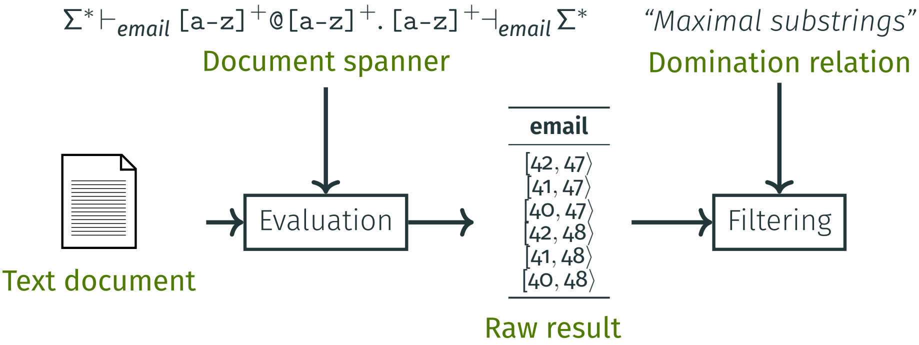


Text document

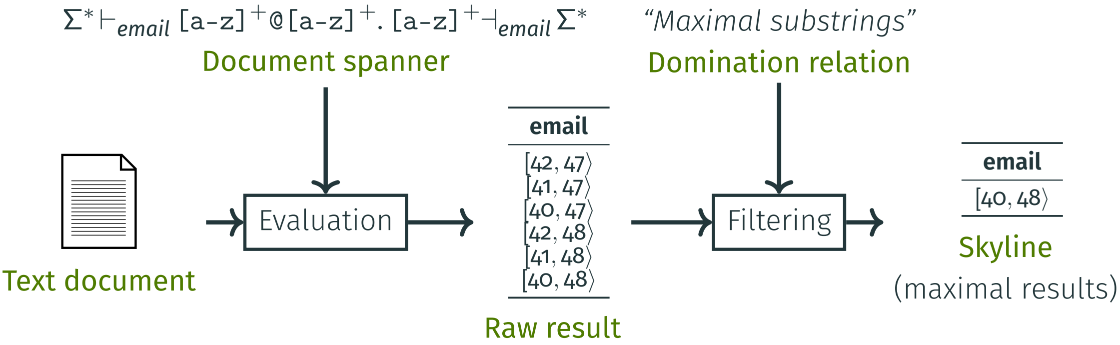
email
$[42, 47\rangle$
$[41, 47\rangle$
$[40, 47\rangle$
$[42, 48\rangle$
$[41, 48\rangle$
$[40, 48\rangle$

Raw result

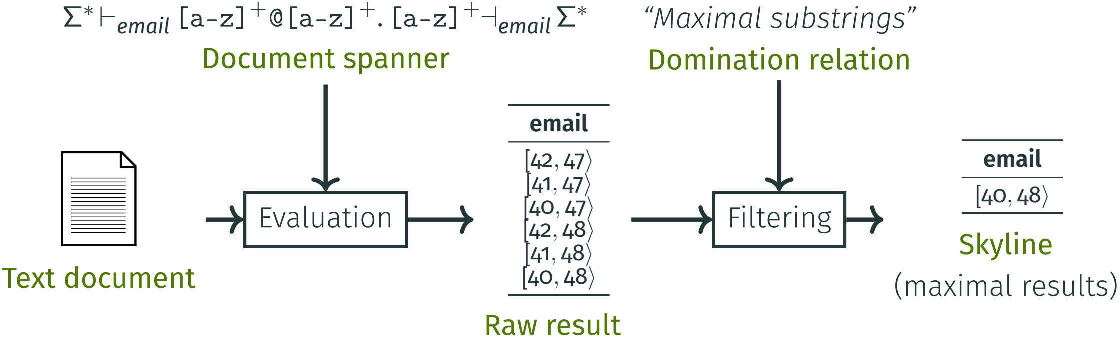
Naive skyline computation



Naive skyline computation



Naive skyline computation



- Can we be **more efficient**, i.e., avoiding materializing the raw result?
- Can we **merge** both steps, i.e., compile the domination relation in the spanner?

Paper contributions and talk outline

- Introduce and formalize the skyline problem for regular spanners

  $\rightarrow$ Propose a general framework to express domination relations

Paper contributions and talk outline

- **Introduce** and **formalize** the skyline problem for regular spanners
  $\rightarrow$ Propose a **general framework** to express domination relations

- Study if we can **compile** the skyline operator **into the spanner**
  $\rightarrow$ **Expressiveness:** is it **possible**?
  $\rightarrow$ **State complexity:** does it **blow up** the spanner representation?

Paper contributions and talk outline

- Introduce and formalize the skyline problem for regular spanners
  $\rightarrow$ Propose a general framework to express domination relations
- Study if we can compile the skyline operator into the spanner
  $\rightarrow$ Expressiveness: is it possible?
  $\rightarrow$ State complexity: does it blow up the spanner representation?
- Study the problem of efficiently evaluating the skyline operator
  $\rightarrow$ In data complexity and combined complexity

Table of contents

Basics of spanners

- **Document:** string over an alphabet

Basics of spanners

- **Document:** string over an alphabet

$$d = \begin{array}{cccccccccccc} \mathtt{J} & \mathtt{o} & \mathtt{h} & \mathtt{n} & \sqcup & \mathtt{4} & \mathtt{5} & \mathtt{6} & \mathtt{1} & \mathtt{2} & \mathtt{3} & \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \end{array}$$

Basics of spanners

- **Document:** string over an alphabet

$$d = \begin{array}{cccccccccccc} \texttt{J} & \texttt{o} & \texttt{h} & \texttt{n} & \textvisiblespace & \texttt{4} & \texttt{5} & \texttt{6} & \texttt{1} & \texttt{2} & \texttt{3} & \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \end{array}$$

- **Span:** interval of positions
  $\rightarrow$ ex: $[0, 4\rangle$, $[5, 11\rangle$

Basics of spanners

- **Document:** string over an alphabet

$$d = \texttt{J}\ \texttt{o}\ \texttt{h}\ \texttt{n}\ \textvisiblespace\ \texttt{4}\ \texttt{5}\ \texttt{6}\ \texttt{1}\ \texttt{2}\ \texttt{3}$$

$d =$	J	o	h	n	␣	4	5	6	1	2	3	
	0	1	2	3	4	5	6	7	8	9	10	11

- **Span:** interval of positions
  - → ex: $[0, 4\rangle$, $[5, 11\rangle$
- **Mapping** over a set of variables X: **partial** function from X to spans
  - → ex: for $X = \{x, y, z\}$, map x to $[0, 4\rangle$ and leave y and z unassigned

Basics of spanners

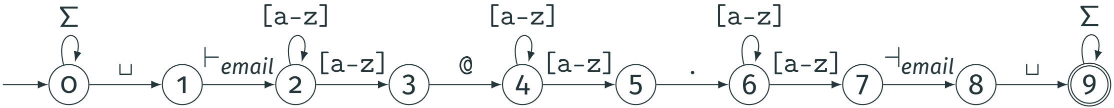
- **Document:** string over an alphabet

$$d = \begin{array}{cccccccccccc} \texttt{J} & \texttt{o} & \texttt{h} & \texttt{n} & \llcorner\!\lrcorner & \texttt{4} & \texttt{5} & \texttt{6} & \texttt{1} & \texttt{2} & \texttt{3} & \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \end{array}$$

- **Span:** interval of positions
  - $\rightarrow$ ex: $[0, 4\rangle$, $[5, 11\rangle$
- **Mapping** over a set of variables X: **partial** function from X to spans
  - $\rightarrow$ ex: for $X = \{x, y, z\}$, map x to $[0, 4\rangle$ and leave y and z unassigned
- **Spanner**: function that maps each document to a set of mappings

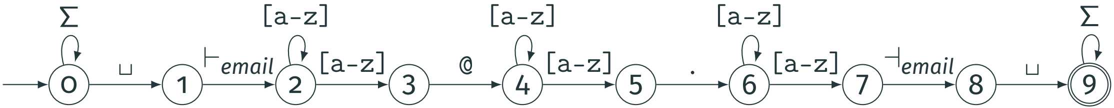
Defining spanners

Regular spanners: those that can be expressed as variable-set automata (VAs; always assumed to be sequential)



Defining spanners

Regular spanners: those that can be expressed as **variable-set automata** (VAs; always assumed to be sequential)

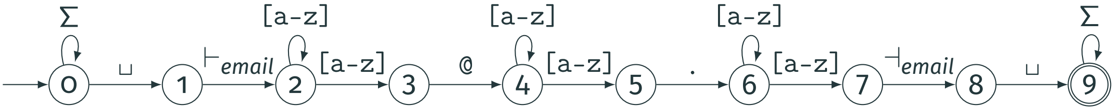


In practice, often more convenient to write in the subclass of **regex-formulas**:

$$\Sigma^* \vdash_{email}\ \texttt{[a-z]}^+\ \texttt{@}\ \texttt{[a-z]}^+\ \texttt{.}\ \texttt{[a-z]}^+\ \dashv_{email}\ \Sigma^*$$

Defining spanners

Regular spanners: those that can be expressed as **variable-set automata** (VAs; always assumed to be sequential)



In practice, often more convenient to write in the subclass of **regex-formulas**:

$$\Sigma^* \vdash_{email} \texttt{[a-z]}^+ \ \texttt{@} \ \texttt{[a-z]}^+ \ \texttt{.} \ \texttt{[a-z]}^+ \dashv_{email} \Sigma^*$$

Other **more general classes**:

- **Core spanners**: featuring **string equality selection**
- **Generalized core spanners**: featuring **difference**

Defining the skyline operator

- A spanner A applied to a document d returns a set of mappings

Defining the skyline operator

- A spanner A applied to a document d returns a **set of mappings**
- **Domination relation:** a **partial order** $\leq$ on the mappings

Defining the skyline operator

- A spanner A applied to a document d returns a **set of mappings**
- **Domination relation:** a **partial order** $\leq$ on the mappings
- **Skyline** $\eta_{\leq}(A)$ of A under $\leq$: mappings not **strictly dominated** by another mapping

Defining the skyline operator

- A spanner A applied to a document d returns a **set of mappings**
- **Domination relation:** a **partial order** $\leq$ on the mappings
- **Skyline** $\eta_{\leq}(A)$ of A under $\leq$: mappings not **strictly dominated** by another mapping

But which domination relations make sense on mappings m and m'?

Defining the skyline operator

- A spanner A applied to a document d returns a **set of mappings**
- **Domination relation:** a **partial order** $\leq$ on the mappings
- **Skyline** $\eta_{\leq}(A)$ of A under $\leq$: mappings not **strictly dominated** by another mapping

But which domination relations make sense on mappings m and m'?

- **Trivial domination relation:** no mapping dominates another

Defining the skyline operator

- A spanner A applied to a document d returns a **set of mappings**
- **Domination relation:** a **partial order** $\leq$ on the mappings
- **Skyline** $\eta_{\leq}(A)$ of A under $\leq$: mappings not **strictly dominated** by another mapping

But which domination relations make sense on mappings m and m'?

- **Trivial domination relation:** no mapping dominates another
- **Span inclusion relation:** "larger spans are better"
  - $\rightarrow$ If m and m' assign the same variables and $m(x)$ is subspan of $m'(x)$ for all x, then $m \leq m'$

Defining the skyline operator

- A spanner A applied to a document d returns a **set of mappings**
- **Domination relation:** a **partial order** $\leq$ on the mappings
- **Skyline** $\eta_{\leq}(A)$ of A under $\leq$: mappings not **strictly dominated** by another mapping

But which domination relations make sense on mappings m and m'?

- **Trivial domination relation:** no mapping dominates another
- **Span inclusion relation:** "larger spans are better"
  $\rightarrow$ If m and m' assign the same variables and $m(x)$ is subspan of $m'(x)$ for all x, then $m \leq m'$
- **Variable inclusion relation:** "assigning more variables is better"
  $\rightarrow$ If m and m' agree on common variables and m' assigns more variables, then $m \leq m'$

Defining the skyline operator

- A spanner A applied to a document d returns a **set of mappings**
- **Domination relation:** a **partial order** $\leq$ on the mappings
- **Skyline** $\eta_{\leq}(A)$ of A under $\leq$: mappings not **strictly dominated** by another mapping

But which domination relations make sense on mappings m and m'?

- **Trivial domination relation:** no mapping dominates another
- **Span inclusion relation:** “larger spans are better”
  $\rightarrow$ If m and m' assign the same variables and $m(x)$ is subspan of $m'(x)$ for all x, then $m \leq m'$
- **Variable inclusion relation:** “assigning more variables is better”
  $\rightarrow$ If m and m' agree on common variables and m' assigns more variables, then $m \leq m'$
- **Span length relation:** “longer spans are better”

Defining the skyline operator

- A spanner A applied to a document d returns a **set of mappings**
- **Domination relation:** a **partial order** $\leq$ on the mappings
- **Skyline** $\eta_{\leq}(A)$ of A under $\leq$: mappings not **strictly dominated** by another mapping

But which domination relations make sense on mappings m and m'?

- **Trivial domination relation:** no mapping dominates another
- **Span inclusion relation:** "larger spans are better"
  $\rightarrow$ If m and m' assign the same variables and $m(x)$ is subspan of $m'(x)$ for all x, then $m \leq m'$
- **Variable inclusion relation:** "assigning more variables is better"
  $\rightarrow$ If m and m' agree on common variables and m' assigns more variables, then $m \leq m'$
- **Span length relation:** "longer spans are better"

Can we have a **unified framework** covering those?

Examples of domination relations

Extracted mappings:

x	y
$[1, 2\rangle$	$[2, 3\rangle$
—	$[2, 3\rangle$
$[0, 2\rangle$	$[2, 3\rangle$
$[4, 6\rangle$	$[4, 10\rangle$

Examples of domination relations

Extracted mappings:

x	y
$[1, 2\rangle$	$[2, 3\rangle$
—	$[2, 3\rangle$
$[0, 2\rangle$	$[2, 3\rangle$
$[4, 6\rangle$	$[4, 10\rangle$

Skyline under variable inclusion:

x	y
$[1, 2\rangle$	$[2, 3\rangle$
$[0, 2\rangle$	$[2, 3\rangle$
$[4, 6\rangle$	$[4, 10\rangle$

Examples of domination relations

Extracted mappings:

x	y
$[1,2\rangle$	$[2,3\rangle$
—	$[2,3\rangle$
$[0,2\rangle$	$[2,3\rangle$
$[4,6\rangle$	$[4,10\rangle$

Skyline under variable inclusion:

x	y
$[1,2\rangle$	$[2,3\rangle$
$[0,2\rangle$	$[2,3\rangle$
$[4,6\rangle$	$[4,10\rangle$

Skyline under span inclusion:

x	y
—	$[2,3\rangle$
$[0,2\rangle$	$[2,3\rangle$
$[4,6\rangle$	$[4,10\rangle$

Examples of domination relations

Extracted mappings:

x	y
$[1, 2\rangle$	$[2, 3\rangle$
—	$[2, 3\rangle$
$[0, 2\rangle$	$[2, 3\rangle$
$[4, 6\rangle$	$[4, 10\rangle$

Skyline under variable inclusion:

x	y
$[1, 2\rangle$	$[2, 3\rangle$
$[0, 2\rangle$	$[2, 3\rangle$
$[4, 6\rangle$	$[4, 10\rangle$

Skyline under span inclusion:

x	y
—	$[2, 3\rangle$
$[0, 2\rangle$	$[2, 3\rangle$
$[4, 6\rangle$	$[4, 10\rangle$

Skyline under span length:

x	y
—	$[2, 3\rangle$
$[4, 6\rangle$	$[4, 10\rangle$

Formalizing domination relations

- We want to express a domination relation: a partial order on mappings
  - → Say the domain is $X = \{x, y\}$

Formalizing domination relations

- We want to express a domination relation: a partial order on mappings
  - $\rightarrow$ Say the domain is $X = \{x, y\}$
- A domination pair (m, m') is a pair of mappings m and m' such that $m \leq m'$

Formalizing domination relations

- We want to express a **domination relation**: a partial order on mappings
  - $\rightarrow$ Say the domain is $X = \{x, y\}$
- A **domination pair** (m, m') is a pair of mappings m and m' such that $m \leq m'$
- **Idea:** domination pair (m, m') can be seen as a mapping μ, if we **rename variables**!
  - Variables are $X \cup X^\dagger$, i.e., $\{x, y, x^\dagger, y^\dagger\}$
  - Variables of X are mapped by μ like in m
  - For each variable $z \in X$, variable $z^\dagger$ is mapped by μ like $m'(z)$

Formalizing domination relations

- We want to express a **domination relation**: a partial order on mappings
  $\rightarrow$ Say the domain is $X = \{x, y\}$
- A **domination pair** (m, m') is a pair of mappings m and m' such that $m \leq m'$
- **Idea:** domination pair (m, m') can be seen as a mapping μ, if we **rename variables**!
  - Variables are $X \cup X^{\dagger}$, i.e., $\{x, y, x^{\dagger}, y^{\dagger}\}$
  - Variables of X are mapped by μ like in m
  - For each variable $z \in X$, variable $z^{\dagger}$ is mapped by μ like $m'(z)$
- Example:
  - Mapping m maps x to $[42,51\rangle$ and does not map y
  - Mapping m' maps x to $[42,51\rangle$ and maps y to $[52,58\rangle$
  - Then μ maps x and $x^{\dagger}$ to $[42,51\rangle$, does not map y, and maps $y^{\dagger}$ to $[52,58\rangle$

Formalizing domination relations

- We want to express a domination relation: a partial order on mappings
  - $\rightarrow$ Say the domain is $X = \{x, y\}$

- A domination pair (m, m') is a pair of mappings m and m' such that $m \leq m'$

- Idea: domination pair (m, m') can be seen as a mapping μ, if we rename variables!
  - Variables are $X \cup X^\dagger$, i.e., $\{x, y, x^\dagger, y^\dagger\}$
  - Variables of X are mapped by μ like in m
  - For each variable $z \in X$, variable $z^\dagger$ is mapped by μ like $m'(z)$

- Example:
  - Mapping m maps x to $[42{,}51\rangle$ and does not map y
  - Mapping m' maps x to $[42{,}51\rangle$ and maps y to $[52{,}58\rangle$
  - Then μ maps x and $x^\dagger$ to $[42{,}51\rangle$, does not map y, and maps $y^\dagger$ to $[52{,}58\rangle$

$\rightarrow$ We can define the domination relation as a spanner D, called a domination rule:
  - $\rightarrow$ Definition of spanner D: given d, extract all mappings μ that code a domination pair

Expressing domination relations via domination rules

Consider the example domination relations on a single variable x

- Trivial domination relation: no mapping dominates another

Expressing domination relations via domination rules

Consider the example domination relations on a single variable x

- Trivial domination relation: no mapping dominates another

$$\Sigma^* \;\; \vdash_{x^\dagger} \vdash_x \;\; \Sigma^* \;\; \dashv_x \dashv_{x^\dagger} \;\; \Sigma^*$$

Expressing domination relations via domination rules

Consider the example domination relations on a single variable x

- Trivial domination relation: no mapping dominates another

$$\Sigma^* \vdash_{x^\dagger} \vdash_x \Sigma^* \dashv_x \dashv_{x^\dagger} \Sigma^* \vee \Sigma^*$$

Expressing domination relations via domination rules

Consider the example domination relations on a single variable x

- Trivial domination relation: no mapping dominates another

$$\Sigma^* \;\vdash_{x^\dagger} \vdash_x \;\Sigma^*\; \dashv_x \dashv_{x^\dagger} \;\Sigma^* \;\vee\; \Sigma^*$$

- Span inclusion relation: “larger spans are better”

Expressing domination relations via domination rules

Consider the example domination relations on a single variable x

- Trivial domination relation: no mapping dominates another

$$\Sigma^* \ \vdash_{x^\dagger} \vdash_x \ \Sigma^* \ \dashv_x \dashv_{x^\dagger} \ \Sigma^* \ \vee \ \Sigma^*$$

- Span inclusion relation: “larger spans are better”

$$\Sigma^* \ \vdash_{x^\dagger} \ \Sigma^* \ \vdash_x \ \Sigma^* \ \dashv_x \ \Sigma^* \ \dashv_{x^\dagger} \ \Sigma^* \ \vee \ \Sigma^*$$

Expressing domination relations via domination rules

Consider the example domination relations on a single variable x

- **Trivial domination relation:** no mapping dominates another

$$\Sigma^* \ \vdash_{x^\dagger} \vdash_x \ \Sigma^* \ \dashv_x \dashv_{x^\dagger} \ \Sigma^* \ \vee \ \Sigma^*$$

- **Span inclusion relation:** "larger spans are better"

$$\Sigma^* \ \vdash_{x^\dagger} \ \Sigma^* \ \vdash_x \ \Sigma^* \ \dashv_x \ \Sigma^* \ \dashv_{x^\dagger} \ \Sigma^* \ \vee \ \Sigma^*$$

- **Variable inclusion relation:** "assigning more variables is better"

Expressing domination relations via domination rules

Consider the example domination relations on a single variable x

- Trivial domination relation: no mapping dominates another

$$\Sigma^* \vdash_{x^\dagger} \vdash_x \Sigma^* \dashv_x \dashv_{x^\dagger} \Sigma^* \vee \Sigma^*$$

- Span inclusion relation: “larger spans are better”

$$\Sigma^* \vdash_{x^\dagger} \Sigma^* \vdash_x \Sigma^* \dashv_x \Sigma^* \dashv_{x^\dagger} \Sigma^* \vee \Sigma^*$$

- Variable inclusion relation: “assigning more variables is better”

$$\Sigma^* \vdash_{x^\dagger} \Sigma^* \dashv_{x^\dagger} \Sigma^*$$

Expressing domination relations via domination rules

Consider the example domination relations on a single variable x

- Trivial domination relation: no mapping dominates another

$$\Sigma^* \ \vdash_{x^\dagger} \vdash_x \ \Sigma^* \ \dashv_x \dashv_{x^\dagger} \ \Sigma^* \ \vee \ \Sigma^*$$

- Span inclusion relation: "larger spans are better"

$$\Sigma^* \ \vdash_{x^\dagger} \ \Sigma^* \ \vdash_x \ \Sigma^* \ \dashv_x \ \Sigma^* \ \dashv_{x^\dagger} \ \Sigma^* \ \vee \ \Sigma^*$$

- Variable inclusion relation: "assigning more variables is better"

$$\Sigma^* \ \vdash_{x^\dagger} \ \Sigma^* \ \dashv_{x^\dagger} \ \Sigma^* \ \vee \ \Sigma^* \ \vdash_{x^\dagger} \vdash_x \ \Sigma^* \ \dashv_x \dashv_{x^\dagger} \ \Sigma^*$$

Expressing domination relations via domination rules

Consider the example domination relations on a single variable x

- **Trivial domination relation:** no mapping dominates another

$$\Sigma^* \ \vdash_{x^\dagger} \vdash_x \ \Sigma^* \ \dashv_x \dashv_{x^\dagger} \ \Sigma^* \ \vee \ \Sigma^*$$

- **Span inclusion relation:** "larger spans are better"

$$\Sigma^* \ \vdash_{x^\dagger} \ \Sigma^* \ \vdash_x \ \Sigma^* \ \dashv_x \ \Sigma^* \ \dashv_{x^\dagger} \ \Sigma^* \ \vee \ \Sigma^*$$

- **Variable inclusion relation:** "assigning more variables is better"

$$\Sigma^* \ \vdash_{x^\dagger} \ \Sigma^* \ \dashv_{x^\dagger} \ \Sigma^* \ \vee \ \Sigma^* \ \vdash_{x^\dagger} \vdash_x \ \Sigma^* \ \dashv_x \dashv_{x^\dagger} \ \Sigma^* \ \vee \ \Sigma^*$$

Expressing domination relations via domination rules

Consider the example domination relations on a single variable x

- **Trivial domination relation:** no mapping dominates another

$$\Sigma^* \ \vdash_{x^\dagger} \vdash_x \ \Sigma^* \ \dashv_x \dashv_{x^\dagger} \ \Sigma^* \ \vee \ \Sigma^*$$

- **Span inclusion relation:** “larger spans are better”

$$\Sigma^* \ \vdash_{x^\dagger} \ \Sigma^* \ \vdash_x \ \Sigma^* \ \dashv_x \ \Sigma^* \ \dashv_{x^\dagger} \ \Sigma^* \ \vee \ \Sigma^*$$

- **Variable inclusion relation:** “assigning more variables is better”

$$\Sigma^* \ \vdash_{x^\dagger} \ \Sigma^* \ \dashv_{x^\dagger} \ \Sigma^* \ \vee \ \Sigma^* \ \vdash_{x^\dagger} \vdash_x \ \Sigma^* \ \dashv_x \dashv_{x^\dagger} \ \Sigma^* \ \vee \ \Sigma^*$$

- **Span length relation:** “longer spans are better”

Expressing domination relations via domination rules

Consider the example domination relations on a single variable x

- Trivial domination relation: no mapping dominates another

$$\Sigma^* \vdash_{x^\dagger} \vdash_x \Sigma^* \dashv_x \dashv_{x^\dagger} \Sigma^* \vee \Sigma^*$$

- Span inclusion relation: “larger spans are better”

$$\Sigma^* \vdash_{x^\dagger} \Sigma^* \vdash_x \Sigma^* \dashv_x \Sigma^* \dashv_{x^\dagger} \Sigma^* \vee \Sigma^*$$

- Variable inclusion relation: “assigning more variables is better”

$$\Sigma^* \vdash_{x^\dagger} \Sigma^* \dashv_{x^\dagger} \Sigma^* \vee \Sigma^* \vdash_{x^\dagger} \vdash_x \Sigma^* \dashv_x \dashv_{x^\dagger} \Sigma^* \vee \Sigma^*$$

- Span length relation: “longer spans are better”
  $\rightarrow$ Not expressible as a regular spanner

Variable-wise rules

What about spanners extracting more than one variable?

Variable-wise rules

What about spanners extracting **more than one variable**?

- Spanner description generally **exponential in the number of variables**...

$$\Sigma^* \ \vdash_{y^\dagger} \vdash_y \vdash_{x^\dagger} \vdash_x \ \Sigma^* \ \dashv_x \dashv_{x^\dagger} \dashv_y \dashv_{y^\dagger} \ \Sigma^*$$

Variable-wise rules

What about spanners extracting more than one variable?

- Spanner description generally exponential in the number of variables...

$$\Sigma^* \ \vdash_{y^\dagger}\vdash_y\vdash_{x^\dagger}\vdash_x \ \Sigma^* \ \dashv_x\dashv_{x^\dagger}\dashv_y\dashv_{y^\dagger} \ \Sigma^* \vee \Sigma^* \ \vdash_{x^\dagger}\vdash_x \ \Sigma^* \ \dashv_x\dashv_{x^\dagger} \ \Sigma^*$$

Variable-wise rules

What about spanners extracting **more than one variable**?

- Spanner description generally **exponential in the number of variables**...

$$\Sigma^* \vdash_{y^\dagger} \vdash_y \vdash_{x^\dagger} \vdash_x \Sigma^* \dashv_x \dashv_{x^\dagger} \dashv_y \dashv_{y^\dagger} \Sigma^* \vee \Sigma^* \vdash_{x^\dagger} \vdash_x \Sigma^* \dashv_x \dashv_{x^\dagger} \Sigma^* \vee \Sigma^* \vdash_{y^\dagger} \vdash_y \Sigma^* \dashv_y \dashv_{y^\dagger} \Sigma^*$$

Variable-wise rules

What about spanners extracting **more than one variable**?

- Spanner description generally **exponential in the number of variables**...

$$\Sigma^* \vdash_{y^\dagger} \vdash_y \vdash_{x^\dagger} \vdash_x \Sigma^* \dashv_x \dashv_{x^\dagger} \dashv_y \dashv_{y^\dagger} \Sigma^* \vee \Sigma^* \vdash_{x^\dagger} \vdash_x \Sigma^* \dashv_x \dashv_{x^\dagger} \Sigma^* \vee \Sigma^* \vdash_{y^\dagger} \vdash_y \Sigma^* \dashv_y \dashv_{y^\dagger} \Sigma^* \vee \Sigma^*$$

Variable-wise rules

What about spanners extracting **more than one variable**?

- Spanner description generally **exponential in the number of variables**...

$$\Sigma^* \ \vdash_{y^\dagger}\vdash_y\vdash_{x^\dagger}\vdash_x \ \Sigma^* \ \dashv_x\dashv_{x^\dagger}\dashv_y\dashv_{y^\dagger} \ \Sigma^* \vee \Sigma^* \ \vdash_{x^\dagger}\vdash_x \ \Sigma^* \ \dashv_x\dashv_{x^\dagger} \ \Sigma^* \vee \Sigma^* \ \vdash_{y^\dagger}\vdash_y \ \Sigma^* \ \dashv_y\dashv_{y^\dagger} \ \Sigma^* \vee \Sigma^*$$

- Better idea: **product** of copies of the same single-variable rule

$$\left(\Sigma^* \ \vdash_{x^\dagger}\vdash_x \ \Sigma^* \ \dashv_x\dashv_{x^\dagger} \ \Sigma^* \vee \Sigma^*\right) \times \left(\Sigma^* \ \vdash_{y^\dagger}\vdash_y \ \Sigma^* \ \dashv_y\dashv_{y^\dagger} \ \Sigma^* \vee \Sigma^*\right)$$

Variable-wise rules

What about spanners extracting **more than one variable**?

- Spanner description generally **exponential in the number of variables**...

$$\Sigma^* \vdash_{y^\dagger} \vdash_y \vdash_{x^\dagger} \vdash_x \Sigma^* \dashv_x \dashv_{x^\dagger} \dashv_y \dashv_{y^\dagger} \Sigma^* \vee \Sigma^* \vdash_{x^\dagger} \vdash_x \Sigma^* \dashv_x \dashv_{x^\dagger} \Sigma^* \vee \Sigma^* \vdash_{y^\dagger} \vdash_y \Sigma^* \dashv_y \dashv_{y^\dagger} \Sigma^* \vee \Sigma^*$$

- Better idea: **product** of copies of the same single-variable rule

$$\left(\Sigma^* \vdash_{x^\dagger} \vdash_x \Sigma^* \dashv_x \dashv_{x^\dagger} \Sigma^* \vee \Sigma^*\right) \times \left(\Sigma^* \vdash_{y^\dagger} \vdash_y \Sigma^* \dashv_y \dashv_{y^\dagger} \Sigma^* \vee \Sigma^*\right)$$

$\rightarrow$ A rule is **variable-wise** if it is a product of copies of one single-variable rule

Variable-wise rules

What about spanners extracting **more than one variable**?

- Spanner description generally **exponential in the number of variables**...

$$\Sigma^* \vdash_{y^\dagger} \vdash_y \vdash_{x^\dagger} \vdash_x \Sigma^* \dashv_x \dashv_{x^\dagger} \dashv_y \dashv_{y^\dagger} \Sigma^* \vee \Sigma^* \vdash_{x^\dagger} \vdash_x \Sigma^* \dashv_x \dashv_{x^\dagger} \Sigma^* \vee \Sigma^* \vdash_{y^\dagger} \vdash_y \Sigma^* \dashv_y \dashv_{y^\dagger} \Sigma^* \vee \Sigma^*$$

- Better idea: **product** of copies of the same single-variable rule

$$\left(\Sigma^* \vdash_{x^\dagger} \vdash_x \Sigma^* \dashv_x \dashv_{x^\dagger} \Sigma^* \vee \Sigma^*\right) \times \left(\Sigma^* \vdash_{y^\dagger} \vdash_y \Sigma^* \dashv_y \dashv_{y^\dagger} \Sigma^* \vee \Sigma^*\right)$$

$\to$ A rule is **variable-wise** if it is a product of copies of one single-variable rule

- $\to$ Covers all examples so far

Summary of problems

We have:

- Spanner A on variables X describing which mappings to extract
  $\rightarrow$ e.g., expressed as a variable-set automaton (VA)

Summary of problems

We have:

- Spanner A on variables X describing which mappings to extract
  - $\rightarrow$ e.g., expressed as a variable-set automaton (VA)
- Variable-wise domination rule D to say which mappings dominate which mappings
  - $\rightarrow$ e.g., expressed as a VA on variables x and $x^\dagger$
  - $\rightarrow$ Implicitly extended to X and $X^\dagger$ by taking the product

Summary of problems

We have:

- Spanner A on variables X describing which mappings to extract
  - $\rightarrow$ e.g., expressed as a variable-set automaton (VA)
- Variable-wise domination rule D to say which mappings dominate which mappings
  - $\rightarrow$ e.g., expressed as a VA on variables x and $x^{\dagger}$
  - $\rightarrow$ Implicitly extended to X and $X^{\dagger}$ by taking the product
- Document d

Summary of problems

We have:

- **Spanner** A on variables X describing which mappings to extract
  - $\rightarrow$ e.g., expressed as a **variable-set automaton** (VA)
- **Variable-wise domination rule** D to say which mappings dominate which mappings
  - $\rightarrow$ e.g., expressed as a VA on variables x and $x^\dagger$
  - $\rightarrow$ Implicitly extended to X and $X^\dagger$ by taking the product
- **Document** d

We want to evaluate the **skyline** $\eta_D(A)$ on d:

$\rightarrow$ Compute the mappings extracted by A on d which are maximal according to D

Summary of problems

We have:

- **Spanner** A on variables X describing which mappings to extract
  - $\rightarrow$ e.g., expressed as a **variable-set automaton** (VA)
- **Variable-wise domination rule** D to say which mappings dominate which mappings
  - $\rightarrow$ e.g., expressed as a VA on variables x and $x^\dagger$
  - $\rightarrow$ Implicitly extended to X and $X^\dagger$ by taking the product
- **Document** d

We want to evaluate the **skyline** $\eta_D(A)$ on d:

$\rightarrow$ Compute the mappings extracted by A on d which are maximal according to D

We can compute the skyline in two ways:

- **Compilation:** from A and D, compute a VA A' extracting $\eta_D(A)$.
  - $\rightarrow$ This is **independent from the document d**!

Summary of problems

We have:

- Spanner A on variables X describing which mappings to extract
  - $\rightarrow$ e.g., expressed as a variable-set automaton (VA)
- Variable-wise domination rule D to say which mappings dominate which mappings
  - $\rightarrow$ e.g., expressed as a VA on variables x and $x^\dagger$
  - $\rightarrow$ Implicitly extended to X and $X^\dagger$ by taking the product
- Document d

We want to evaluate the skyline $\eta_D(A)$ on d:

$\rightarrow$ Compute the mappings extracted by A on d which are maximal according to D

We can compute the skyline in two ways:

- Compilation: from A and D, compute a VA A' extracting $\eta_D(A)$.
  - $\rightarrow$ This is independent from the document d!
- Evaluation: from A and D and d, compute the skyline directly

Table of contents

Compilation: Expressiveness results

Remember the task:

- Spanner A describing which mappings to extract
- Domination rule D expressed as a VA

$\rightarrow$ Can we compute a spanner A' that extracts precisely the skyline $\eta_D(A)$ of A under D?

Compilation: Expressiveness results

Remember the task:

- Spanner A describing which mappings to extract
- Domination rule D expressed as a VA

$\rightarrow$ Can we compute a spanner A' that extracts precisely the skyline $\eta_D(A)$ of A under D?

For regular spanners, this is possible:

Theorem

Given a VA $\boldsymbol{A}$ and a domination rule expressed as a VA $\boldsymbol{D}$, we can compute a VA $\boldsymbol{A'}$ extracting the skyline $\eta_{\boldsymbol{D}}(\boldsymbol{A})$

Proof idea: the skyline operator can be defined via regular operations

Compilation: Expressiveness results

Remember the task:

- **Spanner** A describing which mappings to extract
- **Domination rule** D expressed as a VA

$\rightarrow$ Can we compute a spanner $\boldsymbol{A'}$ that extracts precisely the **skyline** $\eta_D(A)$ **of** A **under** D?

For **regular spanners**, this is possible:

Theorem

Given a VA $\mathbf{A}$ *and a domination rule expressed as a VA* $\mathbf{D}$, *we can compute a VA* $\mathbf{A'}$ *extracting the skyline* $\eta_{\mathbf{D}}(\mathbf{A})$

Proof idea: the skyline operator can be defined via **regular operations**

For **core spanners**: not possible!

$\rightarrow$ Already in the case where D is the **span inclusion** or **variable inclusion** rule

Compilation: State complexity results

Remember the task:

- Spanner A describing which mappings to extract
- Domination rule D expressed as a VA

$\rightarrow$ We can compute a spanner A' that extracts precisely the skyline $\eta_D(A)$ of A under D?

What is the complexity?

Compilation: State complexity results

Remember the task:

- Spanner A describing which mappings to extract
- Domination rule D expressed as a VA

$\rightarrow$ We can compute a spanner A' that extracts precisely the skyline $\eta_D(A)$ of A under D?

What is the complexity?

- Construction from the previous slide is exponential

Compilation: State complexity results

Remember the task:

- Spanner A describing which mappings to extract
- Domination rule D expressed as a VA

$\rightarrow$ We can compute a spanner A' that extracts precisely the skyline $\eta_D(A)$ of A under D?

What is the complexity?

- Construction from the previous slide is exponential
- This blowup is unavoidable, at least in the case of variable inclusion!

Compilation: State complexity results

Remember the task:

- Spanner A describing which mappings to extract
- Domination rule D expressed as a VA

$\rightarrow$ We can compute a spanner A' that extracts precisely the skyline $\eta_D(A)$ of A under D?

What is the complexity?

- Construction from the previous slide is exponential
- This blowup is unavoidable, at least in the case of variable inclusion!

Theorem

Given a VA A with n states, a VA A' computing the skyline $\eta(A)$ under variable inclusion needs $2^{\Omega(n)}$ states in general

Proof technique via nFBDDs

Table of contents

Computation: Getting the results of the skyline

- Input:
  - Regular spanner A describing which mappings to extract
  - Domination rule D expressed as a VA
  - Document d

Computation: Getting the results of the skyline

- Input:
  - Regular spanner A describing which mappings to extract
  - Domination rule D expressed as a VA
  - Document d
- Output: the skyline of A on d under D, i.e., the mappings extracted by A on d which are maximal according to the order on mappings described by D

Computation: Getting the results of the skyline

- Input:
  - Regular spanner A describing which mappings to extract
  - Domination rule D expressed as a VA
  - Document d
- Output: the skyline of A on d under D, i.e., the mappings extracted by A on d which are maximal according to the order on mappings described by D

Two different perspectives:

Computation: Getting the results of the skyline

- Input:
  - Regular spanner A describing which mappings to extract
  - Domination rule D expressed as a VA
  - Document d
- Output: the skyline of A on d under D, i.e., the mappings extracted by A on d which are maximal according to the order on mappings described by D

Two different perspectives:

- Data complexity: VAs A and D fixed, the input is the document d

Computation: Getting the results of the skyline

- Input:
  - Regular spanner A describing which mappings to extract
  - Domination rule D expressed as a VA
  - Document d
- Output: the skyline of A on d under D, i.e., the mappings extracted by A on d which are maximal according to the order on mappings described by D

Two different perspectives:

- Data complexity: VAs A and D fixed, the input is the document d
- Combined complexity for fixed rule: fix D, the input is the VA A and the document d

Evaluation in data complexity

The skyline is always tractable to compute in data complexity:

Theorem

For any fixed VA A and domination rule D, given a document d, we can compute the skyline of A on d under D in PTIME in d

Two ways to see it:

Evaluation in data complexity

The skyline is always tractable to compute in data complexity:

Theorem

For any fixed VA $\mathbf{A}$ *and domination rule* $\mathbf{D}$*, given a document* $\mathbf{d}$*, we can compute the skyline of* $\mathbf{A}$ *on* $\mathbf{d}$ *under* $\mathbf{D}$ *in PTIME in d*

Two ways to see it:

- Naive materialization:
  - Compute the set S of mappings of A on d

Evaluation in data complexity

The skyline is always **tractable to compute in data complexity**:

Theorem

*For any fixed VA A and domination rule D, given a **document d**, we can compute the skyline of A on d under D in PTIME in d*

Two ways to see it:

- **Naive materialization:**
  - Compute the **set S of mappings of A on d**
  - **Materialize** the domination relation $\leq$ (pairs of mappings) by running D on d

Evaluation in data complexity

The skyline is always **tractable to compute in data complexity**:

Theorem

*For any fixed VA A and domination rule D, given a **document** d, we can compute the skyline of A on d under D in PTIME in d*

Two ways to see it:

- **Naive materialization:**
  - Compute the **set S of mappings of A on d**
  - **Materialize** the domination relation $\leq$ (pairs of mappings) by running D on d
  - **Filter** the mappings of S to keep only the maximal ones under $\leq$

Evaluation in data complexity

The skyline is always **tractable to compute in data complexity**:

Theorem

*For any fixed VA $\mathbf{A}$ and domination rule $\mathbf{D}$, given a **document $\mathbf{d}$**, we can compute the skyline of $\mathbf{A}$ on $\mathbf{d}$ under $\mathbf{D}$ in **PTIME in d***

Two ways to see it:

- **Naive materialization:**
  - Compute the **set S of mappings of A on d**
  - **Materialize** the domination relation $\leq$ (pairs of mappings) by running D on d
  - **Filter** the mappings of S to keep only the maximal ones under $\leq$
- **Compilation** using previous results:
  - **Rewrite** the VA A and domination rule D to a VA A' computing the skyline of A under D

Evaluation in data complexity

The skyline is always **tractable to compute in data complexity**:

Theorem

*For any fixed VA $\mathbf{A}$ and domination rule $\mathbf{D}$, given a **document $\mathbf{d}$**, we can compute the skyline of $\mathbf{A}$ on $\mathbf{d}$ under $\mathbf{D}$ in **PTIME in d***

Two ways to see it:

- **Naive materialization:**
  - Compute the **set S of mappings of A on d**
  - **Materialize** the domination relation $\leq$ (pairs of mappings) by running D on d
  - **Filter** the mappings of S to keep only the maximal ones under $\leq$
- **Compilation** using previous results:
  - **Rewrite** the VA A and domination rule D to a VA A' computing the skyline of A under D
  - Then, simply **run A' on d** to compute the maximal mappings

Evaluation in combined complexity

Computing the skyline is **intractable** even under the variable inclusion rule

Theorem

The following problem is NP-hard: given $\boldsymbol{n} \in \mathbb{N}$, a VA $\boldsymbol{A}$, and document $\boldsymbol{d}$, decide if $\boldsymbol{A}$ has more than n mappings on d that are maximal for variable inclusion

Evaluation in combined complexity

Computing the skyline is **intractable** even under the variable inclusion rule

Theorem

The following problem is NP-hard: given $n \in \mathbb{N}$, a VA A, and document d, decide if A has more than n mappings on d that are maximal for variable inclusion

- As a consequence: unless $\mathrm{P} = \mathrm{NP}$, **no output-polynomial algorithm**

Evaluation in combined complexity

Computing the skyline is **intractable** even under the variable inclusion rule

Theorem

The following problem is NP-hard: given $n \in \mathbb{N}$, a VA A, and document d, decide if A has more than n mappings on d that are maximal for variable inclusion

- As a consequence: unless $\mathrm{P} = \mathrm{NP}$, **no output-polynomial algorithm**
- Hardness also holds for the **span inclusion** rule

Evaluation in combined complexity

Computing the skyline is **intractable** even under the variable inclusion rule

Theorem

The following problem is NP-hard: given $\boldsymbol{n} \in \mathbb{N}$, a VA $\boldsymbol{A}$, and document $\boldsymbol{d}$, decide if $\boldsymbol{A}$ has more than n mappings on d that are maximal for variable inclusion

- As a consequence: unless $\mathrm{P} = \mathrm{NP}$, **no output-polynomial algorithm**
- Hardness also holds for the **span inclusion** rule
- Hardness also holds if the **input document is fixed**

Which domination rules are hard?

- **Skyline computation** is intractable in combined complexity for the **variable inclusion** and **span inclusion** rules
- Of course it is tractable for the **trivial domination rule** (= no skyline)

$\rightarrow$ Can we get a **dichotomy?**

Which domination rules are hard?

- **Skyline computation** is intractable in combined complexity for the **variable inclusion** and **span inclusion** rules
- Of course it is tractable for the **trivial domination rule** (= no skyline)

→ Can we get a **dichotomy?**

In the paper:

- **Sufficient condition for hardness:** whenever a rule captures unboundedly many comparable pairs that are "disjoint", then skyline computation is hard
- **Dichotomy** on a subset of domination rules based on a variant of this condition
- **Troubling asymmetry:** there is a domination rule $\leq$ such that:
  - Computing the skyline under $\leq$ is **easy**
  - Computing the skyline under $\geq$ is **hard**!

Table of contents

Summary and further work

- We have studied skyline computation for document spanners, with a spanner-based framework to express domination rules
- Regular spanners are closed under skyline, but unavoidable exponential blowup
- Evaluation tractable in data complexity but hard in combined complexity

Summary and further work

- We have studied **skyline computation for document spanners**, with a **spanner-based framework** to express domination rules
- Regular spanners are **closed** under skyline, but unavoidable exponential blowup
- Evaluation tractable in data complexity but **hard in combined complexity**

Main open questions:

- Are there **other applications** of the nFBDD correspondence?
  - $\rightarrow$ In the paper: exponential blowup for the **join** of **schemaless regex formulas**
- Can we get a **dichotomy** on all single-variable variable-wise rules?
- Same question for the **state complexity blowup**?
- Is it the **same criterion** for **state complexity** and **computational complexity**?

Summary and further work

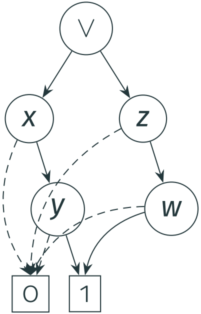
- We have studied **skyline computation for document spanners**, with a **spanner-based framework** to express domination rules
- Regular spanners are **closed** under skyline, but unavoidable exponential blowup
- Evaluation tractable in data complexity but **hard in combined complexity**

Main open questions:

- Are there **other applications** of the nFBDD correspondence?
  - $\rightarrow$ In the paper: exponential blowup for the **join** of **schemaless regex formulas**
- Can we get a **dichotomy** on all single-variable variable-wise rules?
- Same question for the **state complexity blowup**?
- Is it the **same criterion** for **state complexity** and **computational complexity**?

Thanks for your attention!

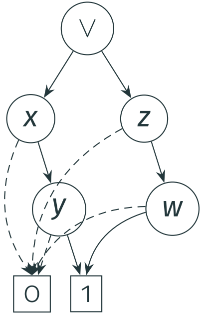
Proof technique: nFBDDs (aka NROBPs)



(Courtesy of Tim Van Bremen)

- nFBDDs are a formalism to represent Boolean functions
  - Intuitively, OBDDs with nondeterminism and without variable order

Proof technique: nFBDDs (aka NROBPs)

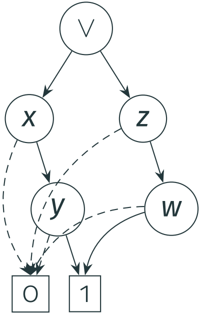


(Courtesy of Tim Van Bremen)

- nFBDDs are a formalism to represent Boolean functions
  - Intuitively, OBDDs with nondeterminism and without variable order
- Given an nFBDD representing a Boolean function ϕ on variables X, we can easily compute a VA A_ϕ and document d such that the mappings extracted by A_ϕ correspond to the satisfying assignments of ϕ

  → For any Boolean valuation $\nu \colon X \to \{0, 1\}$, then ν satisfies ϕ if and only if A_ϕ extracts a mapping on d that assigns $\{x \in X \mid \nu(x) = 1\}$

Proof technique: nFBDDs (aka NROBPs)

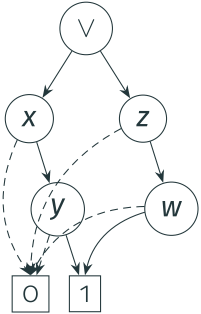


(Courtesy of Tim Van Bremen)

- nFBDDs are a formalism to represent Boolean functions
  - Intuitively, OBDDs with nondeterminism and without variable order
- Given an nFBDD representing a Boolean function ϕ on variables X, we can easily compute a VA A_ϕ and document d such that the **mappings extracted by A_ϕ** correspond to the **satisfying assignments of ϕ**

  $\rightarrow$ For any Boolean valuation $\nu\colon X \to \{0, 1\}$, then ν satisfies ϕ if and only if A_ϕ extracts a mapping on d that assigns $\{x \in X \mid \nu(x) = 1\}$
- But nFBDDs are **exponentially less concise** than other representations (read-3 monotone 2-CNF formulas)

Proof technique: nFBDDs (aka NROBPs)



(Courtesy of Tim Van Bremen)

- nFBDDs are a formalism to represent Boolean functions
  - Intuitively, OBDDs with nondeterminism and without variable order

- Given an nFBDD representing a Boolean function ϕ on variables X, we can easily compute a VA A_ϕ and document d such that the mappings extracted by A_ϕ correspond to the satisfying assignments of ϕ

  $\rightarrow$ For any Boolean valuation $\nu\colon X \to \{0,1\}$, then ν satisfies ϕ if and only if A_ϕ extracts a mapping on d that assigns $\{x \in X \mid \nu(x) = 1\}$

- But nFBDDs are exponentially less concise than other representations (read-3 monotone 2-CNF formulas)

- For such a formula ψ, we can build a VA A_ψ whose skyline under variable inclusion corresponds to the satisfying assignments of ψ

  $\rightarrow$ not expressible as a small nFBDD, hence not expressible by a small VA

Hardness sketch

- Reduction from SAT of a CNF Φ with variables $X = \{x_1, \ldots, x_n\}$ and m clauses
  $\rightarrow$ W.l.o.g., each variable occurs positively and negatively

Hardness sketch

- Reduction from SAT of a CNF Φ with variables $X = \{x_1, \ldots, x_n\}$ and m clauses
  $\rightarrow$ W.l.o.g., each variable occurs positively and negatively
- The variables of the spanner A correspond to **literals**:
  - Variable $p_{i,j}$ whenever variable $x_i \in X$ occurs **positively** in clause j
  - Variable $n_{i,j}$ whenever variable $x_i \in X$ occurs **negatively** in clause j

Hardness sketch

- Reduction from SAT of a CNF Φ with variables $X = \{x_1, \ldots, x_n\}$ and m clauses
  - $\rightarrow$ W.l.o.g., each variable occurs positively and negatively
- The variables of the spanner A correspond to literals:
  - Variable $p_{i,j}$ whenever variable $x_i \in X$ occurs positively in clause j
  - Variable $n_{i,j}$ whenever variable $x_i \in X$ occurs negatively in clause j
- Define the spanner A as a union $r \cup r'$, where:
  - r captures one mapping per valuation of X, i.e., for each variable $x_i \in X$:
    - either assign all the variables $p_{i,j}$ corresponding to positive literals of x_i
    - or assign all the variables $n_{i,j}$ corresponding to negative literals $y_{i,j}$ of x_i

Hardness sketch

- Reduction from SAT of a CNF Φ with variables $X = \{x_1, \ldots, x_n\}$ and m clauses
  - $\rightarrow$ W.l.o.g., each variable occurs positively and negatively
- The variables of the spanner A correspond to **literals**:
  - Variable $p_{i,j}$ whenever variable $x_i \in X$ occurs **positively** in clause j
  - Variable $n_{i,j}$ whenever variable $x_i \in X$ occurs **negatively** in clause j
- Define the spanner A as a union $r \cup r'$, where:
  - r captures **one mapping per valuation of X**, i.e., for each variable $x_i \in X$:
    - either assign all the variables $p_{i,j}$ corresponding to **positive literals** of x_i
    - or assign all the variables $n_{i,j}$ corresponding to **negative literals** $y_{i,j}$ of x_i
  - r' captures **one mapping per clause j**: all literals $p_{i,j'}$ and $n_{i,j'}$ with $j' \neq j$

Hardness sketch

- Reduction from **SAT** of a **CNF Φ** with variables $X = \{x_1, \ldots, x_n\}$ and m clauses
  $\rightarrow$ W.l.o.g., each variable occurs positively and negatively
- The variables of the spanner A correspond to **literals**:
  - Variable $p_{i,j}$ whenever variable $x_i \in X$ occurs **positively** in clause j
  - Variable $n_{i,j}$ whenever variable $x_i \in X$ occurs **negatively** in clause j
- Define the spanner A as a union $r \cup r'$, where:
  - r captures **one mapping per valuation of X**, i.e., for each variable $x_i \in X$:
    - either assign all the variables $p_{i,j}$ corresponding to **positive literals** of x_i
    - or assign all the variables $n_{i,j}$ corresponding to **negative literals** $y_{i,j}$ of x_i
  - r' captures **one mapping per clause j**: all literals $p_{i,j'}$ and $n_{i,j'}$ with $j' \neq j$

Example: for $\Phi = (x_1 \vee x_2) \wedge (\neg x_2 \vee \neg x_3) \wedge (\neg x_1 \wedge x_3)$:

- r assigns: $p_{1,1}$ or $n_{1,3}$; and $p_{2,1}$ or $n_{2,2}$; and $p_{3,3}$ or $n_{3,2}$
- r' assigns: all but $p_{1,1}$ and $p_{2,1}$; or all but $n_{2,2}$ and $n_{3,2}$; or all but $n_{1,3}$ and $p_{3,3}$

Hardness sketch (cont'd)

- r captures **one mapping per valuation of X**, i.e., for each variable $x_i \in X$:
  - either assign all the variables $p_{i,j}$ corresponding to positive literals of x_i
  - or assign all the variables $n_{i,j}$ corresponding to negative literals $y_{i,j}$ of x_i
- r' captures **one mapping per clause j**: all literals $p_{i,j'}$ and $n_{i,j'}$ with $j' \neq j$

Example: for $\Phi = (x_1 \lor x_2) \land (\neg x_2 \lor \neg x_3) \land (\neg x_1 \land x_3)$:

- r assigns: $p_{1,1}$ or $n_{1,3}$; and $p_{2,1}$ or $n_{2,2}$; and $p_{3,3}$ or $n_{3,2}$
- r' assigns: all but $p_{1,1}$ and $p_{2,1}$; or all but $n_{2,2}$ and $n_{3,2}$; or all but $n_{1,3}$ and $p_{3,3}$

What is the **skyline** of $r \cup r'$?

Hardness sketch (cont'd)

- r captures **one mapping per valuation of X**, i.e., for each variable $x_i \in X$:
  - either assign all the variables $p_{i,j}$ corresponding to positive literals of x_i
  - or assign all the variables $n_{i,j}$ corresponding to negative literals $y_{i,j}$ of x_i
- r' captures **one mapping per clause j**: all literals $p_{i,j'}$ and $n_{i,j'}$ with $j' \neq j$

Example: for $\Phi = (x_1 \vee x_2) \wedge (\neg x_2 \vee \neg x_3) \wedge (\neg x_1 \wedge x_3)$:

- r assigns: $p_{1,1}$ or $n_{1,3}$; and $p_{2,1}$ or $n_{2,2}$; and $p_{3,3}$ or $n_{3,2}$
- r' assigns: all but $p_{1,1}$ and $p_{2,1}$; or all but $n_{2,2}$ and $n_{3,2}$; or all but $n_{1,3}$ and $p_{3,3}$

What is the **skyline** of $r \cup r'$?

- The m mappings of m' are **maximal** (incomparable and not covered by m)

Hardness sketch (cont'd)

- r captures **one mapping per valuation of X**, i.e., for each variable $x_i \in X$:
  - either assign all the variables $p_{i,j}$ corresponding to positive literals of x_i
  - or assign all the variables $n_{i,j}$ corresponding to negative literals $y_{i,j}$ of x_i
- r' captures **one mapping per clause j**: all literals $p_{i,j'}$ and $n_{i,j'}$ with $j' \neq j$

Example: for $\Phi = (x_1 \lor x_2) \land (\neg x_2 \lor \neg x_3) \land (\neg x_1 \land x_3)$:

- r assigns: $p_{1,1}$ or $n_{1,3}$; and $p_{2,1}$ or $n_{2,2}$; and $p_{3,3}$ or $n_{3,2}$
- r' assigns: all but $p_{1,1}$ and $p_{2,1}$; or all but $n_{2,2}$ and $n_{3,2}$; or all but $n_{1,3}$ and $p_{3,3}$

What is the **skyline** of $r \cup r'$?

- The m mappings of m' are **maximal** (incomparable and not covered by m)
- Other than that:
  - If there is a **maximal mapping from m**, then it is not covered by r' so contains **one literal per clause**: Φ is **satisfiable**
  - Otherwise, all assignments violate some clause: Φ is **unsatisfiable**