Document Spanners: From Expressive Power to Decision Problems

Dominik D. Freydenberger

Bayreuth University,
Bayreuth, Germany

Based on a joint work with
Mario Holldack
ICDT 2016

A Formal Model for AQL

Core Spanners

- introduced by Fagin, Kimelfeld, Reiss, Vansummeren (PODS 2013/ JACM)
- formalization of AQL (Annotation Query Language), used in IBM's SystemT

A Formal Model for AQL

Core Spanners

- introduced by Fagin, Kimelfeld, Reiss, Vansummeren (PODS 2013/ JACM)
- formalization of AQL (Annotation Query Language), used in IBM's SystemT
- basic idea: complicated searches in large texts

A Formal Model for AQL

Core Spanners

- introduced by Fagin, Kimelfeld, Reiss, Vansummeren (PODS 2013/ JACM)
- formalization of AQL (Annotation Query Language), used in IBM's SystemT
- basic idea: complicated searches in large texts
- regex + {projection, union, join, string equality selection}

The model (1/3)

Example Query

Find all wines that can be paired with multiple cheeses.

The model (1/3)

Example Query

Find all wines that can be paired with multiple cheeses.

(if you prefer, imagine something with companies and products instead)

The model (1/3)

Example Query

Find all wines that can be paired with multiple cheeses.

(if you prefer, imagine something with companies and products instead)

Step 1: Searching for multiple words at once

- "find all mentions of a wine"

The model (1/3)

Example Query

Find all wines that can be paired with multiple cheeses.

(if you prefer, imagine something with companies and products instead)

Step 1: Searching for multiple words at once

- "find all mentions of a wine"
- use a regular expression 🍷,
  🍷 := (`pinot noir` | `seyval blanc` | ⋯ | `chardonnay`)

The model (1/3)

Example Query

Find all wines that can be paired with multiple cheeses.

(if you prefer, imagine something with companies and products instead)

Step 1: Searching for multiple words at once

- "find all mentions of a wine"
- use a regular expression 🍷,
  🍷 := (`pinot noir` | `seyval blanc` | ⋯ | `chardonnay`)

- 🍷 can be written by hand, with machine learning,...

The model (2/3)

Step 2: Searching for pairs of related words

- “find all mentions of a wine close to a cheese”

The model (2/3)

Step 2: Searching for pairs of related words

- "find all mentions of a wine close to a cheese"
- use an extractor (regular expression with captures):

$$\texttt{P} := (X\{🧀\} \,.^{\leq 10}\, Y\{🍷\}) \mid (Y\{🍷\} \,.^{\leq 10}\, X\{🧀\})$$

$$🍷 := (\texttt{pinot noir} \mid \texttt{seyval blanc} \mid \cdots \mid \texttt{chardonnay})$$
$$🧀 := (\texttt{red leicester} \mid \texttt{stilton} \mid \cdots \mid \texttt{sage derby})$$

The model (2/3)

Step 2: Searching for pairs of related words

- "find all mentions of a wine close to a cheese"
- use an extractor (regular expression with captures):

$$\texttt{P} := (X\{🧀\} \,.^{\leq 10}\, Y\{🍷\}) \mid (Y\{🍷\} \,.^{\leq 10}\, X\{🧀\})$$

$$🍷 := (\texttt{pinot noir} \mid \texttt{seyval blanc} \mid \cdots \mid \texttt{chardonnay})$$
$$🧀 := (\texttt{red leicester} \mid \texttt{stilton} \mid \cdots \mid \texttt{sage derby})$$

- captures are used to mark positions
- extractor turns text into a table of positions, columns X and Y

The model (2/3)

Step 2: Searching for pairs of related words

- "find all mentions of a wine close to a cheese"
- use an extractor (regular expression with captures):

$$\texttt{P} := (X\{🧀\} \,.^{\leq 10}\, Y\{🍷\}) \mid (Y\{🍷\} \,.^{\leq 10}\, X\{🧀\})$$

$$🍷 := (\texttt{pinot noir} \mid \texttt{seyval blanc} \mid \cdots \mid \texttt{chardonnay})$$
$$🧀 := (\texttt{red leicester} \mid \texttt{stilton} \mid \cdots \mid \texttt{sage derby})$$

- captures are used to mark positions
- extractor turns text into a table of positions, columns X and Y
- next step: combine tables like in a relational database

The model (3/3)

Step 3: Relational search

- “find all wines that are mentioned close to two different cheeses”

The model (3/3)

Step 3: Relational search

- "find all wines that are mentioned close to two different cheeses"
- $\mathrm{P}_i := ((X_i\{🧀\} .^{\leq 10} Y_i\{🍷\}) \mid (Y_i\{🍷\} .^{\leq 10} X_i\{🧀\}))$, $i \in \{1, 2\}$

The model (3/3)

Step 3: Relational search

- "find all wines that are mentioned close to two different cheeses"
- $\mathsf{P}_i := ((X_i\{🧀\} \mathrel{.^{\leq 10}} Y_i\{🍷\}) \mid (Y_i\{🍷\} \mathrel{.^{\leq 10}} X_i\{🧀\}))$, $i \in \{1, 2\}$
- $\pi_{Y_1} \zeta^{=}_{Y_1, Y_2} \zeta^{\neq}_{X_1, X_2} (\mathsf{P}_1 \times \mathsf{P}_2)$

The model (3/3)

Step 3: Relational search

- "find all wines that are mentioned close to two different cheeses"
- $\mathsf{P}_i := ((X_i\{🧀\} .^{\leq 10} Y_i\{🍷\}) \mid (Y_i\{🍷\} .^{\leq 10} X_i\{🧀\}))$, $i \in \{1, 2\}$
- $\pi_{Y_1} \zeta^{=}_{Y_1, Y_2} \zeta^{\neq}_{X_1, X_2} (\mathsf{P}_1 \times \mathsf{P}_2)$

Summary

- use extractors to turn text into tables
- combine tables with projection, join, string comparison selection (and union, which is not in this example)

Our results

We showed:

- evaluation of core spanners is NP-complete

Our results

We showed:

- evaluation of core spanners is NP-complete
- satisfiability of core spanners is PSPACE-complete

Our results

We showed:

- evaluation of core spanners is NP-complete
- satisfiability of core spanners is PSPACE-complete
- minimization of core spanners is undecidable (and so is much more)

Our results

We showed:

- evaluation of core spanners is NP-complete
- satisfiability of core spanners is PSPACE-complete
- minimization of core spanners is undecidable (and so is much more)

Common idea

- connect core spanners to other models: pattern languages, regex, word equations

Our results

We showed:

- evaluation of core spanners is NP-complete
- satisfiability of core spanners is PSPACE-complete
- minimization of core spanners is undecidable (and so is much more)

Common idea

- connect core spanners to other models: pattern languages, regex, word equations
- get results for free (or with little effort)

Our results

We showed:

- evaluation of core spanners is NP-complete
- satisfiability of core spanners is PSPACE-complete
- minimization of core spanners is undecidable (and so is much more)

Common idea

- connect core spanners to other models: pattern languages, regex, word equations
- get results for free (or with little effort)

Remember

If your model has string equality, at least some of this will work for you.

Last Slide

Thank you for your attention!

Sequel at ICDT 2017

A Logic for Document Spanners