Querying Graph Patterns

Pablo Barceló  Leonid Libkin  Juan Reutter
U. de Chile    U. of Edinburgh  U. of Edinburgh
Graph Patterns

Graph Pattern

Labeled Graph $G$
Graph Patterns

Graph Pattern

Labeled Graph $G$
Graph Patterns

Applications in biology, social networks, semantic web, and many others.
We often end up querying patterns rather than graphs.
We often end up querying patterns rather than graphs.

Crime Network \[\xrightarrow{\text{Query}}\] Graph Pattern
We often end up querying patterns rather than graphs.
We often end up querying patterns rather than graphs
We often end up querying patterns rather than graphs
We often end up querying patterns rather than graphs.
We study how to query partially defined graph data
We study how to query **partially defined graph data**

- This data is represented by means of patterns, with some additional features.
Representing partially defined data: Relations

<table>
<thead>
<tr>
<th></th>
<th>S</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>E</td>
<td>A</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>z</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x</td>
</tr>
</tbody>
</table>

1 x z
Representing partially defined data: XML

\[
\begin{array}{ccc}
S & \rightarrow & R \\
D & E \\
4 & 5 \\
z & 8 \\
y & y \\
\end{array}
\]

\[
\begin{array}{ccc}
A & B & C \\
1 & 3 & x \\
3 & y & 7 \\
1 & x & z \\
\end{array}
\]

\[
\begin{array}{c}
A(1) \\
A(x) \\
B(x) \\
A(y) \\
B(1) \\
B(2) \\
\end{array}
\]

\[
\begin{array}{c}
r \rightarrow * \\
r \rightarrow A(1) \\
r \rightarrow A(x) B(x) \\
r \rightarrow A(y) B(1) B(2) \\
\end{array}
\]

\[
\begin{array}{c}
r \rightarrow * \\
r \rightarrow A(1) \\
r \rightarrow A(x) B(x) \\
r \rightarrow A(y) B(1) B(2) \\
\end{array}
\]
Representing partially defined data: Graph DB’s

Next: features that need to be addressed in the study of querying graph patterns.

▷ Examples from social networks
Key features of graph patterns

For graph patterns, incomplete specification may arise in 3 ways:
Key features of graph patterns

For graph patterns, incomplete specification may arise in 3 ways:

- Node Variables
Key features of graph patterns

For graph patterns, incomplete specification may arise in 3 ways:

- Node Variables
Key features of graph patterns

For graph patterns, incomplete specification may arise in 3 ways:

- Node Variables
- Label Variables
Key features of graph patterns

For graph patterns, incomplete specification may arise in 3 ways:

- Node Variables
- Label Variables

For graph patterns, incomplete specification may arise in 3 ways:

- Node Variables
- Label Variables

\[ \Sigma^* \cdot \text{emp} \cdot \Sigma^* \cdot \text{emp} \cdot \Sigma^* \]
Key features of graph patterns

For graph patterns, incomplete specification may arise in 3 ways:

- Node Variables
- Label Variables
- Regular Expressions
We study how to query partially defined graph data
Outline

Motivation

Graph Patterns

Querying Graph Patterns

Tractable cases

Queries with path output
Outline

Motivation

Graph Patterns

Querying Graph Patterns

Tractable cases

Queries with path output
Graph Databases
Path from $n_1$ to $n_4$
Path from $n_1$ to $n_4$

The *label* of the path is $ac$
Graph patterns
Graph patterns

Node Variables
Graph patterns

- Node Variables
- Regular expressions
Graph patterns

- Node Variables
- Regular expressions
- Label Variables
Semantics of graph patterns: by homomorphisms

Graph homomorphism maps
Semantics of graph patterns: by homomorphisms

Graph homomorphism maps

- Nodes to Nodes
- Edge label variables to $\Sigma$
- Edge expressions are witnessed by paths
Semantics of graph patterns: by homomorphisms

Graph homomorphism maps

- Nodes to Nodes
- Edge label variables to $\Sigma$
- Edge expressions are witnessed by paths
Semantics of graph patterns: by homomorphisms

Graph homomorphism maps

- Nodes to Nodes
- Edge label variables to $\Sigma$
- Edge expressions are witnessed by paths
Semantics of graph patterns: by homomorphisms

Graph homomorphism maps
- Nodes to Nodes
- Edge label variables to $\Sigma$
- Edge expressions are witnessed by paths
Semantics of graph patterns: by homomorphisms

Graph homomorphism maps

- Nodes to Nodes
- Edge label variables to $\Sigma$
- Edge expressions are witnessed by paths
Semantics of graph patterns: by homomorphisms

Graph homomorphism maps

- Nodes to Nodes
- Edge label variables to $\Sigma$
- Edge expressions are witnessed by paths
Semantics of graph patterns: by homomorphisms

Graph homomorphism maps

- Nodes to Nodes
- Edge label variables to $\Sigma$
- Edge expressions are witnessed by paths
Semantics of graph patterns: by homomorphisms

Graph homomorphism maps
- Nodes to Nodes
- Edge label variables to $\Sigma$
- Edge expressions are witnessed by paths
Semantics of graph patterns: by homomorphisms

Graph homomorphism maps
- Nodes to Nodes
- Edge label variables to $\Sigma$
- Edge expressions are witnessed by paths
Each pattern $\pi$ represents an infinite set $\llbracket \pi \rrbracket$ of graph databases:

$$\llbracket \pi \rrbracket = \{ G \mid \text{there is a homomorphism from } \pi \text{ to } G \}$$
A graph query $Q$ consists of:
- a graph pattern
- a tuple of output variables

Intuitively,

Select all tuples of nodes that *realize* the pattern on a graph.
Using Graph Patterns to query databases

A graph query $Q$ consists of:

- a graph pattern
- a tuple of output variables

Intuitively,

Select all tuples of nodes that realize the pattern on a graph.

$\left(n_6, n_7, n_6\right) \in Q(G)$
Using Graph Patterns to query databases

A graph query $Q$ consists of:

- a graph pattern
- a tuple of output variables

Intuitively,

Select all tuples of nodes that realize the pattern on a graph.

$\{(n_6, n_7, n_6) \in Q(G)\}$
Defining **Classes of patterns** according to their features

<table>
<thead>
<tr>
<th>Features</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>node variables</td>
<td>(nv)</td>
</tr>
<tr>
<td>label variables</td>
<td>(lv)</td>
</tr>
<tr>
<td>regular expressions in the edges</td>
<td>(re)</td>
</tr>
</tbody>
</table>

We define $\mathcal{P}^\sigma$, for $\sigma \subseteq \{nv, lv, re\}$
## Defining Classes of patterns according to their features

<table>
<thead>
<tr>
<th>Features:</th>
<th>node variables (nv)</th>
<th>label variables (lv)</th>
<th>regular expressions in the edges (re)</th>
</tr>
</thead>
</table>

We define $\mathcal{P}^\sigma$, for $\sigma \subseteq \{nv, lv, re\}$

- $\mathcal{P}$: subgraph isomorphism
- $\mathcal{P}^{nv, re}$: Essentially CRPQ queries
- $\mathcal{P}^{nv, lv}$: Can only use variables, but no expression in the edges
We classify classes of patterns in terms of the sets that they can represent.
We classify classes of patterns in terms of the sets that they can represent.
Outline

Motivation

Graph Patterns

Querying Graph Patterns

Tractable cases

Queries with path output
Querying Graph Patterns

By means of *certain answers*:

\[
\text{certain}(Q, \pi) = \bigcap \{Q(G) \mid G \in [\pi]\}.
\]
Querying Graph Patterns

By means of *certain answers*:

\[
\text{certain}(Q, \pi) = \bigcap \{Q(G) \mid G \in [\pi]\}.
\]

- Combined and data complexity
By means of \textit{certain answers}:

\[ \text{certain}(Q, \pi) = \bigcap \{ Q(G) \mid G \in [\pi] \}. \]

- Combined and data complexity
- Full classification for CRPQs
- Upper bounds are maintained for the \textit{most general case}
Combined complexity

For CRPQ's:
Combined complexity

For CRPQ’s:

\[ \mathcal{P}^{nv,lv,re} : \text{EXPSPACE-c.} \]

\[ \mathcal{P}^{nv,lv} : \Pi_2^p-c. \quad \mathcal{P}^{nv,re} : \text{EXPSPACE-c.} \quad \mathcal{P}^{lv,re} : \text{EXPSPACE-c.} \]

\[ \mathcal{P}^{nv} : \text{NP-c.} \quad \mathcal{P}^{lv} : \Pi_2^p-c. \quad \mathcal{P}^{re} : \text{EXPSPACE-c.} \]

\[ \mathcal{P} : \text{NP-c.} \]

General upper bound:

The combined complexity of arbitrary graph queries on arbitrary patterns is in \text{EXPSPACE}.
For CRPQ's:

$$\mathcal{P}_{nv}^{l_v, re} : \text{coNP-c.}$$

$$\mathcal{P}_{nv}^{l_v} : \text{coNP-c.}$$

$$\mathcal{P}^{l_v, re} : \text{coNP-c.}$$

$$\mathcal{P}_{nv}^{l_v} : \text{NLOGSPACE-c.}$$

$$\mathcal{P}^{l_v} : \text{coNP-c.}$$

$$\mathcal{P}^{re} : \text{coNP-c.}$$

$$\mathcal{P} : \text{NLOGSPACE-c.}$$
Data Complexity

For CRPQ's:

\[ \mathcal{P}^{\text{nv},\text{lv},\text{re}} : \text{coNP-c.} \]

\[ \mathcal{P}^{\text{nv},\text{lv}} : \text{coNP-c.} \quad \mathcal{P}^{\text{nv},\text{re}} : \text{coNP-c.} \quad \mathcal{P}^{\text{lv},\text{re}} : \text{coNP-c.} \]

\[ \mathcal{P}^{\text{nv}} : \text{NLogspace-c.} \quad \mathcal{P}^{\text{lv}} : \text{coNP-c.} \quad \mathcal{P}^{\text{re}} : \text{coNP-c.} \]

\[ \mathcal{P} : \text{NLogspace-c.} \]

General upper bound:

Data complexity of arbitrary graph queries over arbitrary graph patterns is in \text{coNP}.
Outline

Motivation

Graph Patterns

Querying Graph Patterns

Tractable cases

Queries with path output
Finding Tractable Classes (data complexity)

- $P^{nv}$: naive tables
Finding Tractable Classes (data complexity)

- $\mathcal{P}^{nv}$: naive tables
- Intractable for $\mathcal{P}^{lv}$ or $\mathcal{P}^{re}$
Finding Tractable Clases (data complexity)

- $\mathcal{P}^{nv}$: naive tables
- Intractable for $\mathcal{P}^{lv}$ or $\mathcal{P}^{re}$

Restrictions:
- *Structure*: underlying graphs
- Codd patterns

A pattern is in $\mathcal{P}^{nv,lv}_{Codd}$ if every variable occurs at most once in it
Finding Tractable Classes (data complexity)

Attempts to put restriction in the structure are not very good:
Finding Tractable Classes (data complexity)

Attempts to put restriction in the structure are not very good:

\[
\begin{align*}
\text{coNP-hard for paths in } \mathcal{P}^{lv}. \\
\text{coNP-hard for } \text{DAGs in } \mathcal{P}^{re} \text{ or } \text{DAGs in } \mathcal{P}^{lv}_{\text{Codd}}.
\end{align*}
\]
Finding Tractable Classes (data complexity)

Attempts to put restriction in the structure are not very good:

\[-\text{coNP-hard for paths in } \mathcal{P}^{lv}.
\]
\[-\text{coNP-hard for } \text{DAGs in } \mathcal{P}^{\text{re}} \text{ or } \text{DAGs in } \mathcal{P}^{lv}_{\text{Codd}}.
\]

The only possibility appears to be patterns in $\mathcal{P}^{nv,lv}_{\text{Codd}}$ or $\mathcal{P}^{nv,\text{re}}$ with nice underlying graphs.
Bounded treewidth gives us tractability (data complexity)

- The treewidth of a pattern $\pi$ is the treewidth of its underlying graph.
Bounded treewidth gives us tractability (data complexity)

- The treewidth of a pattern $\pi$ is the treewidth of its underlying graph.

Theorem:

Query answering is in $\text{PTIME}$ for CRPQ's over classes of patterns in $\mathcal{P}^{\text{nv,lv}}_{\text{Codd}}$ or $\mathcal{P}^{\text{nv,re}}_{\text{Codd}}$ with bounded treewidth.
Outline

Motivation

Graph Patterns

Querying Graph Patterns

Tractable cases

Queries with path output
Queries with path output

Certain paths:
- Words that label a path in all graphs \textit{represented} by \( \pi \)
Queries with path output

Certain paths:

- Words that label a path in all graphs \textit{represented} by \( \pi \)
Queries with path output

Certain paths:

- Words that label a path in all graphs \textit{represented} by $\pi$
Queries with path output

Certain paths:

- Words that label a path in all graphs \textit{represented} by $\pi$
Queries with path output

Certain paths:

- Words that label a path in all graphs represented by $\pi$

- Certain paths contain all words of length $m$

- The size of each certain path is greater than $2^m$
We introduce Incomplete automata

In essence, a graph pattern with distinguished initial and final nodes
We introduce Incomplete automata

In essence, a graph pattern with distinguished initial and final nodes

Some results:

► The *smallest* NFA representing the certain paths can be doubly-exponential in the size of $\pi$. 
We introduce Incomplete automata

In essence, a graph pattern with distinguished initial and final nodes

Some results:

- The smallest NFA representing the certain paths can be doubly-exponential in the size of $\pi$.
- Checking whether there exists a certain path is $\text{ExpSPACE}$-complete.
- Checking if a word is a certain path is $\text{coNP}$-complete.
We study how to query graph patterns

How *features* of patterns affect query answering:

- Each feature strictly increases the expressivity of patterns
- Features have to be carefully chosen to guarantee tractability
- We identify classes with PTIME query answering
- And others for which we hope to find good heuristics
We study how to query graph patterns

How *features* of patterns affect query answering:
- Each feature strictly increases the expressivity of patterns
- Features have to be carefully chosen to guarantee tractability
- We identify classes with PTIME query answering
- And others for which we hope to find good heuristics

Starting point for research in *schema mappings, exchanging* and *integrating* graph data.
Querying Graph Patterns

Pablo Barceló  
U. de Chile

Leonid Libkin  
U. of Edinburgh

Juan Reutter  
U. of Edinburgh