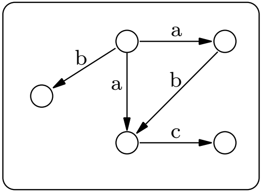
Querying Graph Patterns

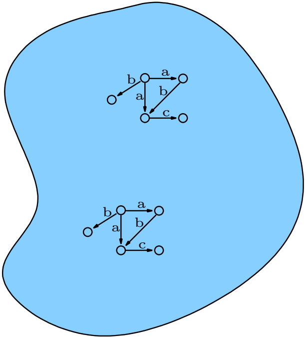
Pablo Barceló	Leonid Libkin	Juan Reutter
U. de Chile	U. of Edinburgh	U. of Edinburgh

Graph Patterns

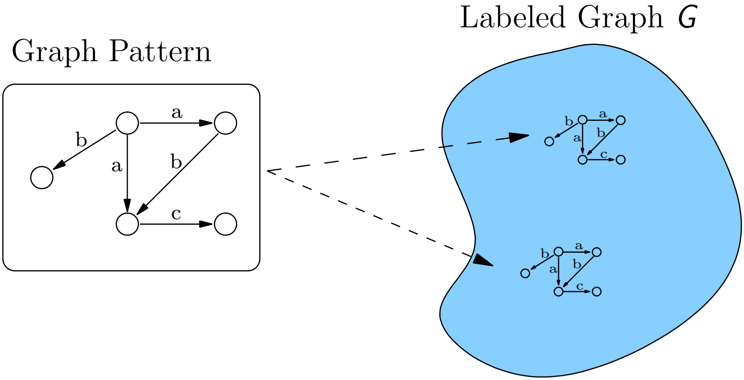
Graph Pattern



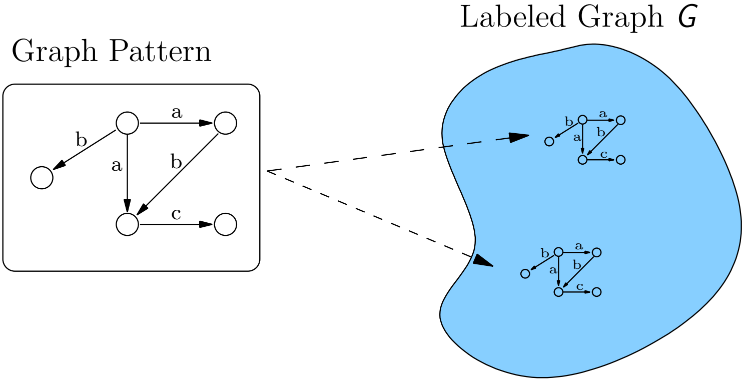
Labeled Graph G



Graph Patterns

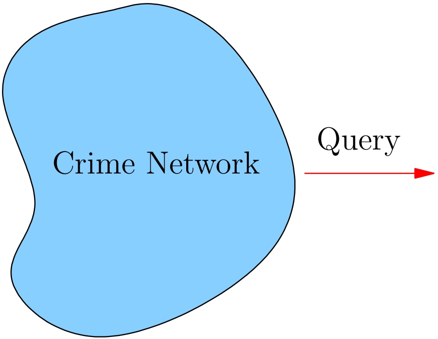


Graph Patterns

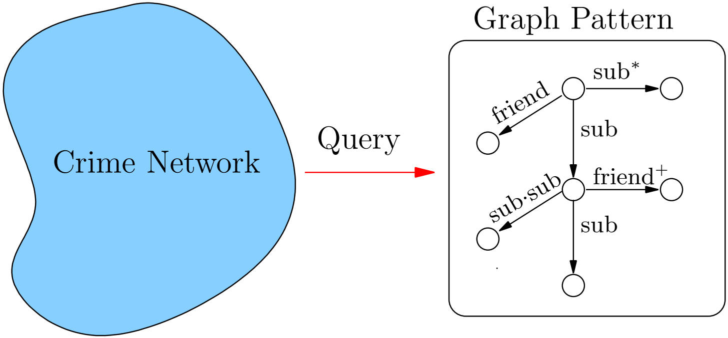


Applications in biology, social networks,
semantic web, and many others.

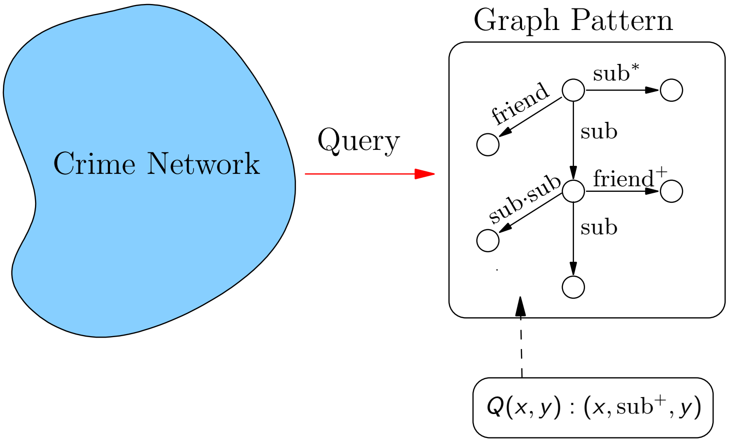
We often end up querying patterns rather than graphs



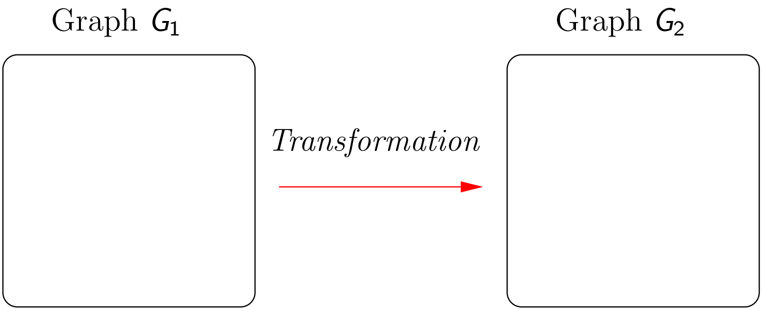
We often end up querying patterns rather than graphs



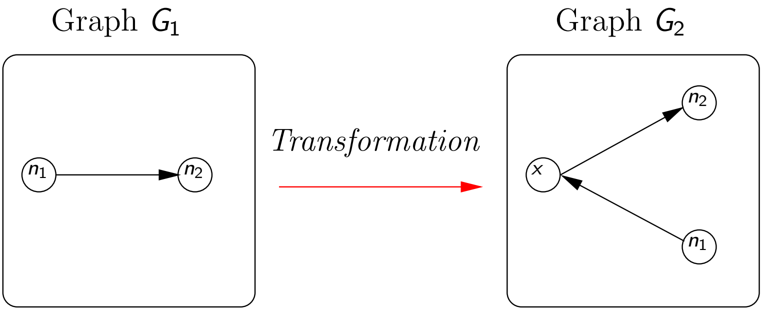
We often end up querying patterns rather than graphs



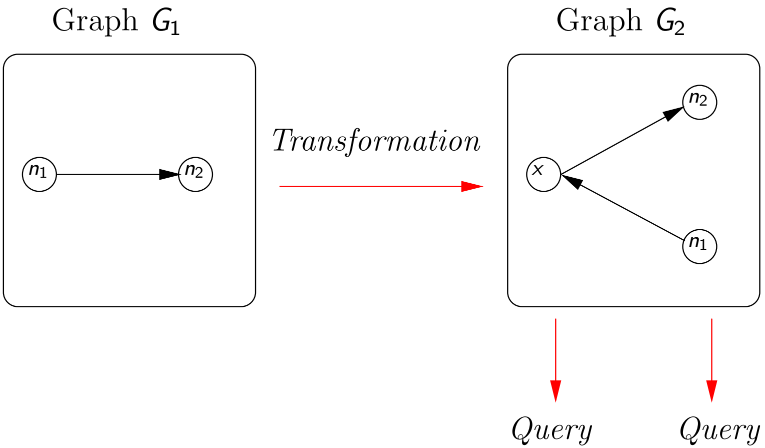
We often end up querying patterns rather than graphs



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

S

D	E
4	5
z	8
y	y

R

A	B	C
1	3	x
3	y	7
1	x	z

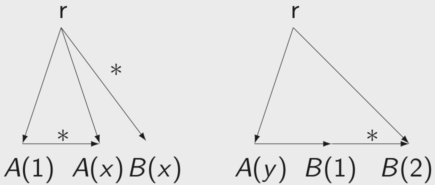
Representing partially defined data: XML

S

D	E
4	5
z	8
y	y

R

A	B	C
1	3	x
3	y	7
1	x	z



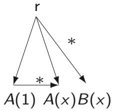
Representing partially defined data: Graph DB's

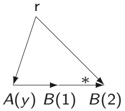
S

D	E
4	5
z	8
y	y

R

A	B	C
1	3	x
3	y	7
1	x	z



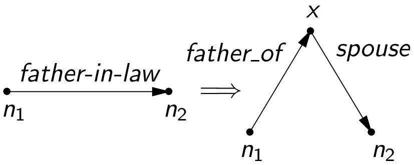


Next: features that need to be addressed in the study of querying graph patterns.

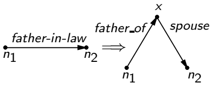
- Examples from social networks

Key features of graph patterns

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



Key features of graph patterns

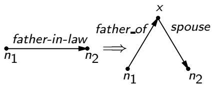


Node Variables

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

- Node Variables

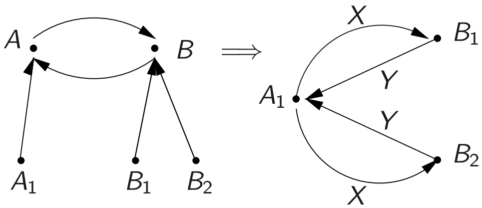
Key features of graph patterns



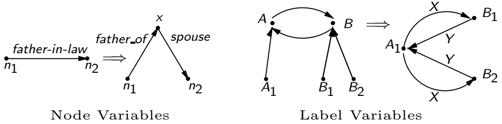
Node Variables

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

- Node Variables



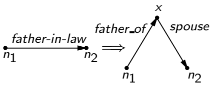
Key features of graph patterns



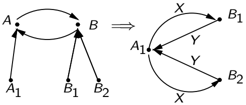
For graph patterns, incomplete specification may arise in 3 ways:

- Node Variables
- Label Variables

Key features of graph patterns



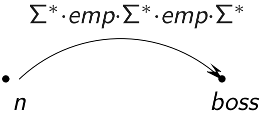
Node Variables



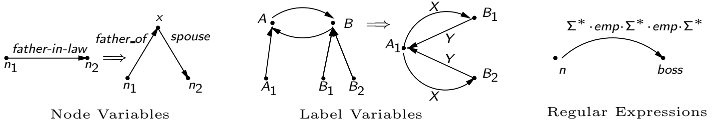
Label Variables

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:

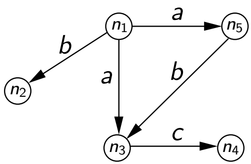
- Node Variables
- Label Variables
- Regular Expressions

We study how to query partially defined graph data

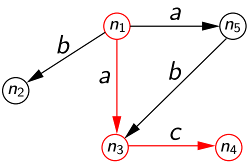
Outline

Outline

Graph Databases

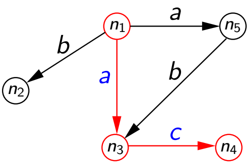


Graph Databases



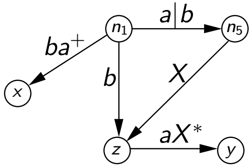
- Path from n_1 to n_4

Graph Databases

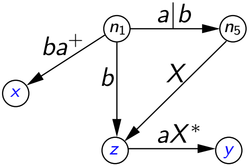


- Path from n_1 to n_4
- The *label* of the path is ac

Graph patterns

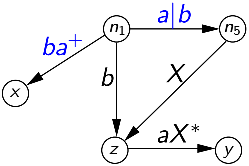


Graph patterns



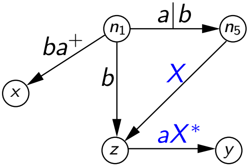
- Node Variables

Graph patterns



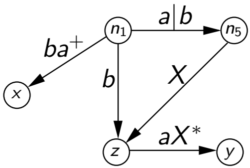
- Node Variables
- Regular expressions

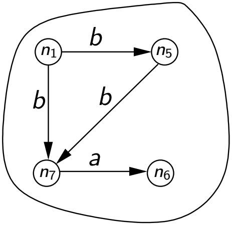
Graph patterns



- Node Variables
- Regular expressions
- Label Variables

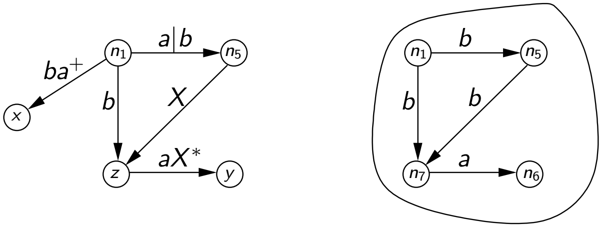
Semantics of graph patterns: by homomorphisms





Graph homomorphism maps

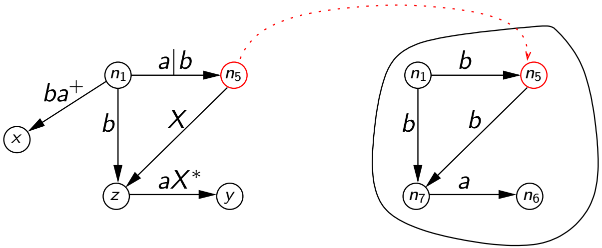
Semantics of graph patterns: by homomorphisms



Graph homomorphism maps

- Nodes to Nodes
- Edge label variables to Σ
- Edge expressions are witnessed by paths

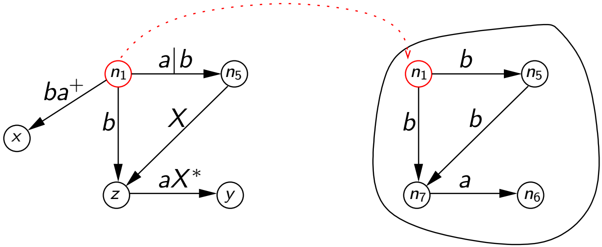
Semantics of graph patterns: by homomorphisms



Graph homomorphism maps

- Nodes to Nodes
- Edge label variables to Σ
- Edge expressions are witnessed by paths

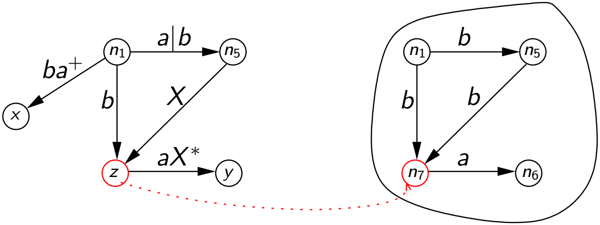
Semantics of graph patterns: by homomorphisms



Graph homomorphism maps

- Nodes to Nodes
- Edge label variables to Σ
- Edge expressions are witnessed by paths

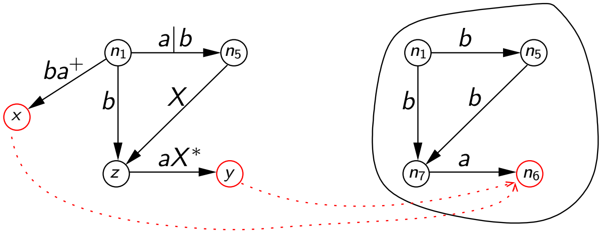
Semantics of graph patterns: by homomorphisms



Graph homomorphism maps

- Nodes to Nodes
- Edge label variables to Σ
- Edge expressions are witnessed by paths

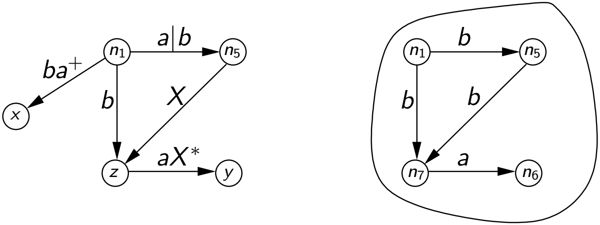
Semantics of graph patterns: by homomorphisms



Graph homomorphism maps

- Nodes to Nodes
- Edge label variables to Σ
- Edge expressions are witnessed by paths

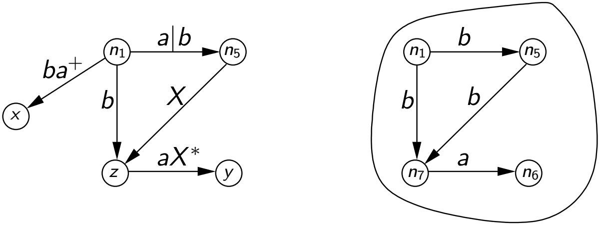
Semantics of graph patterns: by homomorphisms



Graph homomorphism maps

- Nodes to Nodes
- Edge label variables to Σ
- Edge expressions are witnessed by paths

Semantics of graph patterns: by homomorphisms



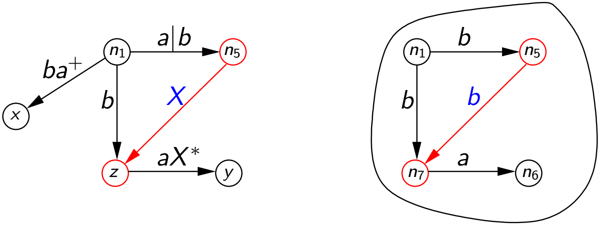
$X \longrightarrow b$

$aX^* \longrightarrow ab^*$

Graph homomorphism maps

- Nodes to Nodes
- Edge label variables to Σ
- Edge expressions are witnessed by paths

Semantics of graph patterns: by homomorphisms



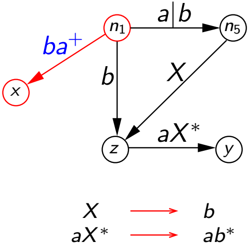
$X \longrightarrow b$

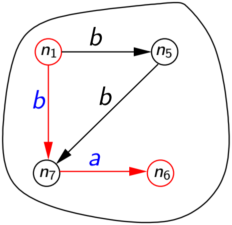
$aX^* \longrightarrow ab^*$

Graph homomorphism maps

- Nodes to Nodes
- Edge label variables to Σ
- Edge expressions are witnessed by paths

Semantics of graph patterns: by homomorphisms





Graph homomorphism maps

- Nodes to Nodes
- Edge label variables to Σ
- Edge expressions are witnessed by paths

Each pattern π represents an infinite set $[\![\pi]\!]$ of graph databases:

$$[\![\pi]\!] = \{G \mid \text{there is a homomorphism from } \pi \text{ to } G\}$$

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.

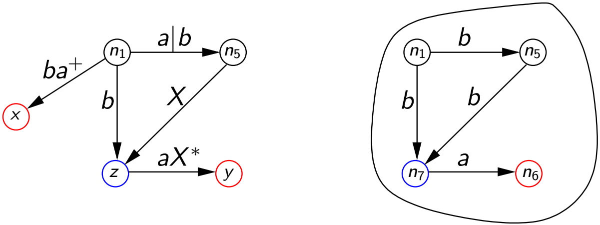
Using Graph Patterns to query databases

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.



$$(n_6, n_7, n_6) \in Q(G)$$

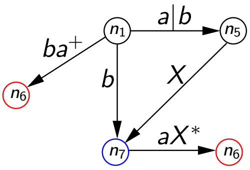
Using Graph Patterns to query databases

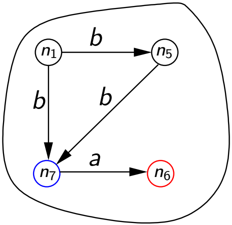
A graph query Q consists of:

- 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

Features:	node variables (nv) label variables (lv) regular expressions in the edges (re)

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

Defining Classes of patterns according to their features

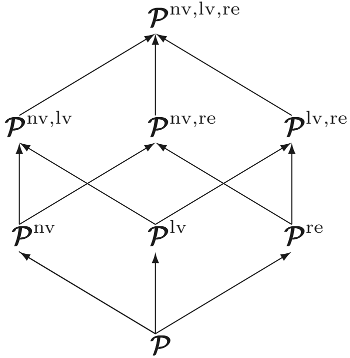
Features:	node variables (nv) label variables (lv) regular expressions in the edges (re)

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

- $\mathcal{P}$: subgraph isomorphism
- $\mathcal{P}^{\mathrm{nv},\mathit{re}}$: Essentially CRPQ queries
- $\mathcal{P}^{\mathrm{nv},\mathrm{lv}}$: 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

Querying Graph Patterns

By means of *certain answers*:

$$\mathrm{certain}(Q, \pi) \; = \; \bigcap \{ Q(G) \mid G \in [\![\pi]\!] \}.$$

Querying Graph Patterns

By means of *certain answers*:

$$\mathrm{certain}(Q, \pi) \; = \; \bigcap \{ Q(G) \mid G \in [\![\pi]\!] \}.$$

- Combined and data complexity

Querying Graph Patterns

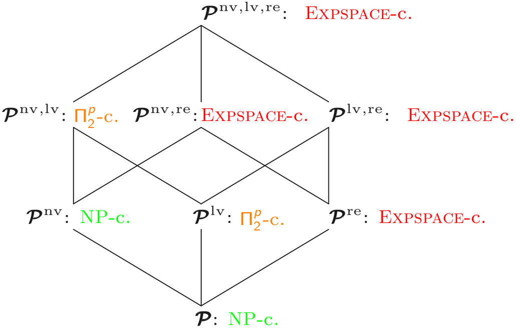
By means of *certain answers*:

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

- Combined and data complexity
- Full classification for CRPQs
- Upper bounds are maintained for the *most general case*

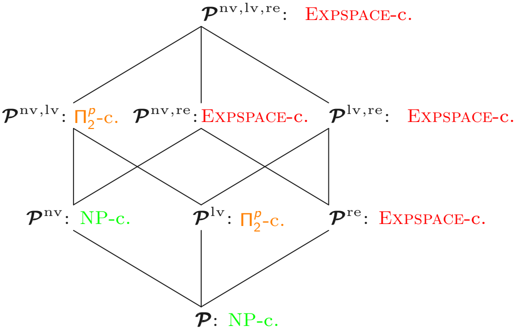
Combined complexity

For CRPQ's:



Combined complexity

For CRPQ's:

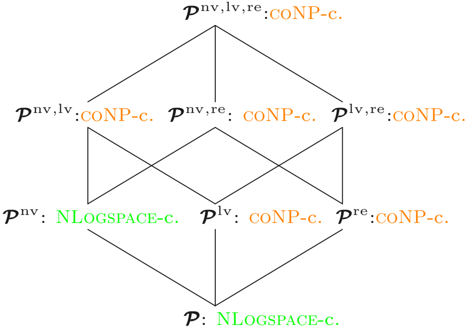


General upper bound:

The combined complexity of arbitrary graph queries on arbitrary patterns is in EXPSPACE.

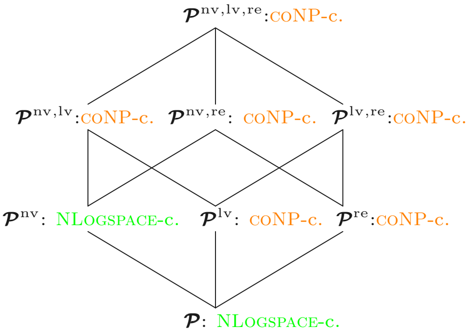
Data Complexity

For CRPQ's:



Data Complexity

For CRPQ's:



General upper bound:

Data complexity of arbitrary graph queries over arbitrary graph patterns is in coNP.

Outline

Finding Tractable Clases (data complexity)

- $\mathcal{P}^{\mathrm{nv}}$: naive tables

Finding Tractable Clases (data complexity)

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

Finding Tractable Clases (data complexity)

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

Restrictions:

- *Structure:* underlying graphs
- Codd patterns

A pattern is in $\mathcal{P}^{\mathrm{nv,lv}}_{\mathrm{Codd}}$ if every variable occurs at most once in it

Finding Tractable Clases (data complexity)

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

Finding Tractable Clases (data complexity)

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

coNP-hard for *paths* in $\mathcal{P}^{\mathrm{lv}}$.

coNP-hard for *DAGs* in $\mathcal{P}^{\mathrm{re}}$ or *DAGs* in $\mathcal{P}^{\mathrm{lv}}_{\mathrm{Codd}}$.

Finding Tractable Clases (data complexity)

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

coNP-hard for *paths* in $\mathcal{P}^{\mathrm{lv}}$.

coNP-hard for *DAGs* in $\mathcal{P}^{\mathrm{re}}$ or *DAGs* in $\mathcal{P}^{\mathrm{lv}}_{\mathrm{Codd}}$.

- The only possibility appears to be patterns in $\mathcal{P}^{\mathrm{nv,lv}}_{\mathrm{Codd}}$ or $\mathcal{P}^{\mathrm{nv,re}}$ with *nice underlying graphs*.

Bounded treewidth gives us tractability (data complexity)

- The treewidth of a pattern π is the treewitdth of its underlying graph.

Bounded treewidth gives us tractability (data complexity)

- The treewidth of a pattern π is the treewitdth of its underlying graph.

Theorem:

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

Outline

Queries with path output

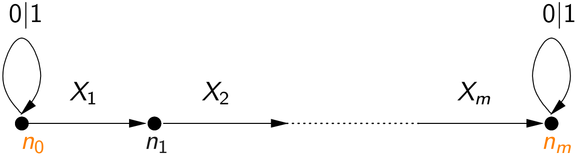
Certain paths:

- Words that label a path in all graphs *represented* by π

Queries with path output

Certain paths:

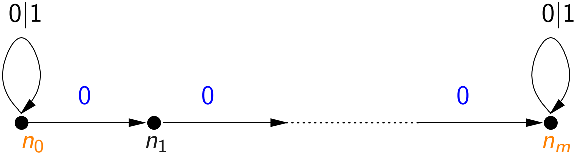
- Words that label a path in all graphs *represented* by π



Queries with path output

Certain paths:

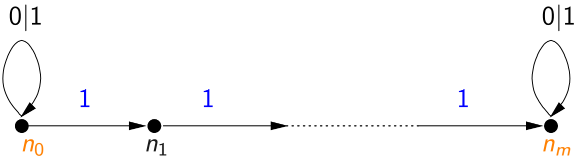
- Words that label a path in all graphs *represented* by π



Queries with path output

Certain paths:

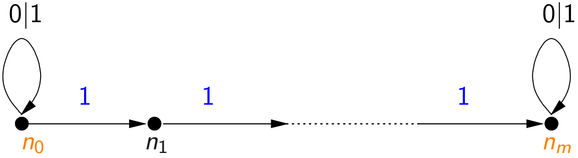
- Words that label a path in all graphs *represented* by π



Queries with path output

Certain paths:

- Words that label a path in all graphs *represented* by π



- 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 π.

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 π.
- Checking wether there exists a certain path is EXPSPACE-complete.
- Checking if a word is a certain path is 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