Property Graphs

ANNA QUERALT

FACULTAT D'INFORMÀTICA DE BARCELONA

Graph Operations

Basic graph operations

- Adjacency
- Path queries
  - Path existence (reachability)
  - Regular path queries (RPQs)
- Graph patterns
  - Basic Graph Patterns (BGPs)
  - Navigational Graph Patterns (NGPs)
- *Graph metrics*

Adjacency

Formal definition:

$$\text{Adjacency}(n) = \bar{N}$$
$$n_i \in \bar{N} \iff \exists e_1 \mid \rho(e_1) = (n_i, n) \lor \rho(e_1) = (n, n_i)$$

Computational cost: linear cost on the number of edges to visit

Examples:

- Find all friends of a person
- Airports with a direct connection
- Movies watched by a person
- Products bought by a customer
- …

Path Queries

Based on the general concept of reachability in graph theory

- A node *n1* can reach a node *n2* (*n2* is reachable from *n1*) if there exists a sequence of adjacent nodes which starts with *n1* and ends with *n2*

Path query:

$$x \xrightarrow{\alpha} y$$

- *x*, *y* are nodes
- α is a regular expression over *Lab* that specifies conditions on the path
  - α =* denotes path existence without any further constraints (reachability)

Regular Path Queries (RPQs)

Path query:

$$P = x \xrightarrow{\alpha} y$$

- *x*, *y* are nodes
- α is a regular expression over *Lab* that specifies conditions on the path

Regular expression operators

*	Kleene star
+	Kleene plus
·	Concatenation
|	Union
-	Inverse

… and combinations of them

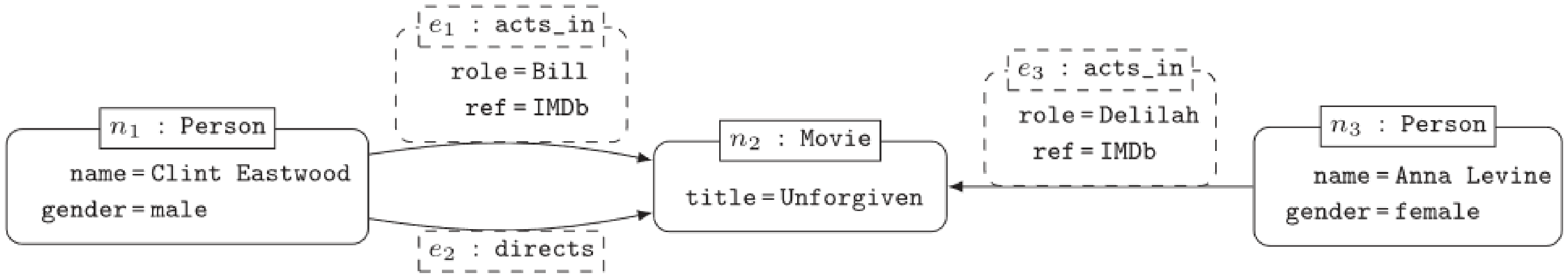
Examples:

$$P := x \xrightarrow{\texttt{knows}^{+}} y \qquad P' := x \xrightarrow{\texttt{knows}^{+} \cdot \texttt{likes}} y \qquad P'' := x \xrightarrow{\texttt{knows}^{+} \cdot (\texttt{likes} \,|\, \texttt{dislikes})} y$$

Activity

Assume a graph containing relationships and nodes like the ones shown below

- Define a RPQ including expressions from the previous slide to find *all co-actors of all actors*
  - Which are the solutions you will get?
- Define a RPQ to retrieve *all actors you can reach by following the co-actoring path at least once*
- Define a RPQ to find *all persons that participate in the same movie*



Pattern Matching

Based on the subgraph isomorphism problem in graph theory

- Input: property graph **G**, and a graph pattern **P**
- Output: all sub-graphs of **G** that are isomorphic to **P**

Computational cost: hard to compute, in general, NP-complete

Examples:

- Group of cities all of them directly connected by flights
- People who have ordered the same item
- ...

Pattern Matching

Based on *basic graph patterns* (BGPs)

- Equivalent to conjunctive queries

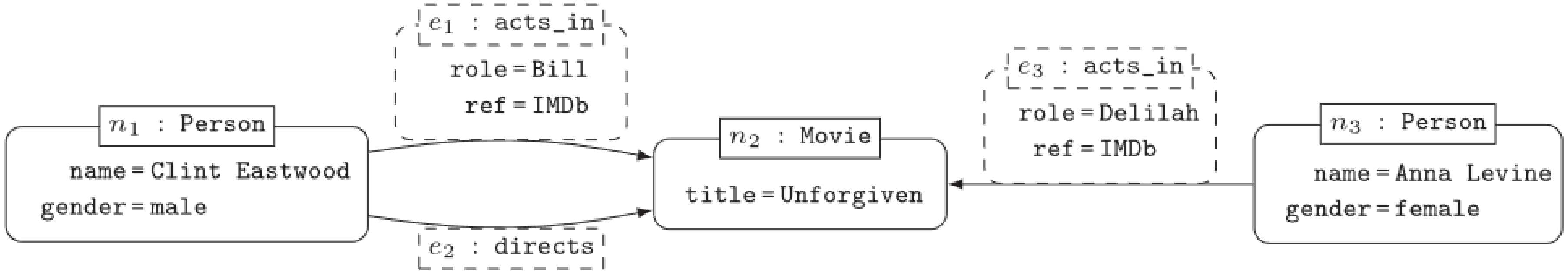
A *BGP* for querying property graphs is a property graph where variables can appear in place of any constant (ids/labels/properties)

A **match** for a *BGP* is a mapping from variables to constants such that when the mapping is applied to the *BGP*, the result is *contained* within the original graph

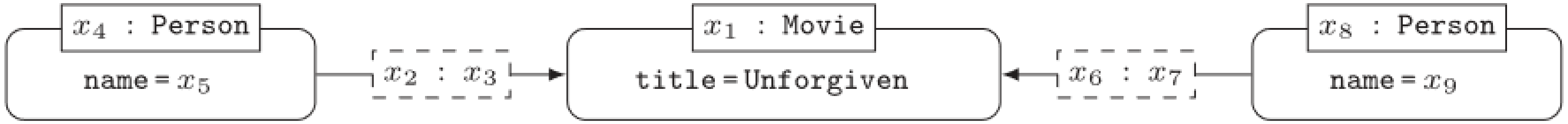
The **results** for a *BGP* are then **all mappings** from variables in the query to constants that comprise a match

Example of Graph Pattern

Graph:



BGP:



Evaluating Graph Patterns

This is completely different from SQL and Relational Algebra!

Evaluating a *bgp* **Q** against a graph database **G** corresponds to listing **all** possible matches of **Q** with respect to **G**

Formally:

Definition 3.5 (Match). Given an edge-labelled graph $G = (V, E)$ and a bgp $Q = (V', E')$, a *match* h of Q in G is a mapping from $Const \cup Var$ to $Const$ such that:

(1) for each constant $a \in Const$, it is the case that $h(a) = a$; that is, the mapping maps constants to themselves; and

(2) for each edge $(b, l, c) \in E'$, it holds that $(h(b), h(l), h(c)) \in E$; this condition imposes that (a) each edge of Q is mapped to an edge of G, and (b) the structure of Q is preserved in its image under h in G (that is, when h is applied to all the terms in Q, the result is a sub-graph of G).

Extracted from: R. Angles et al. Foundations of Modern Query Languages for Graph Databases

Semantics of a Match

Homomorphism-based semantics:

- The previous definition corresponds to a homomorphism from **Q** to **G**
  - Multiple variables in **Q** can map to the same term in **G** (within a match)

Isomorphism-based semantics: adds constraints to the mapping function

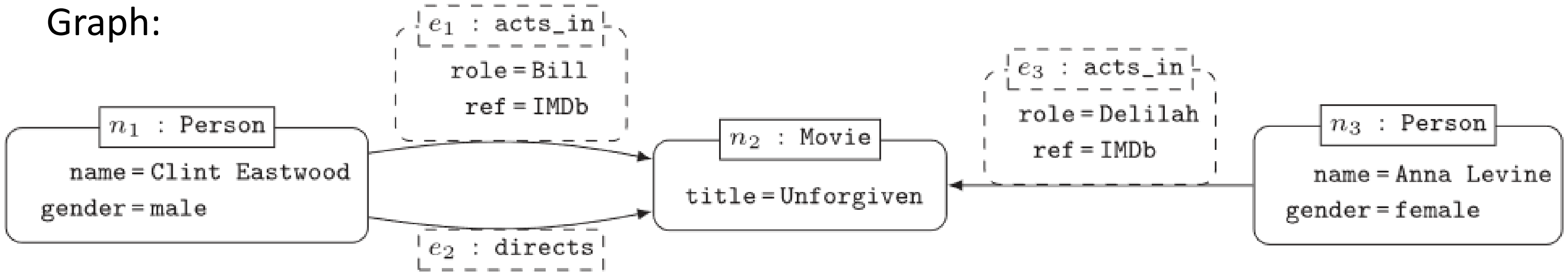
- Strict isomorphism semantics (no-repeated-anything)
  - Each variable in **Q** maps to a different term in **G** (within a match)
- No repeated-node semantics
  - Each variable representing a node in **Q** maps to a different node in **G** (within a match)
- No repeated-edge semantics:
  - Each variable representing an edge in **Q** maps to a different edge in **G** (within a match)

Activity

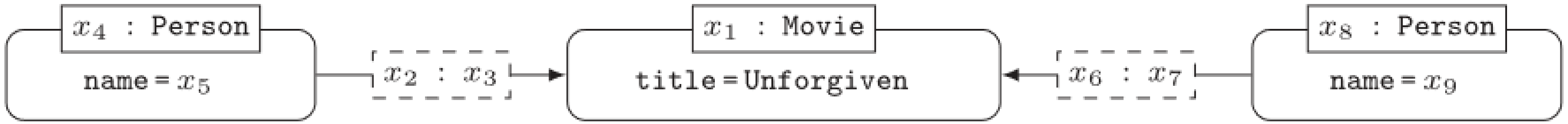
Objective: Understand the differences between isomorphism-based and homomorphism-based semantics in pattern matching

- *Given the following graph, bgp and potential results…*

Graph:



BGP:



Activity

Objective: Understand the differences between isomorphism-based and homomorphism-based semantics in pattern matching

- *Given the following graph, bgp and potential results…*

Results:

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
1	n_2	e_2	directs	n_1	Clint Eastwood	e_3	acts_in	n_3	Anna Levine
2	n_2	e_3	acts_in	n_3	Anna Levine	e_2	directs	n_1	Clint Eastwood
3	n_2	e_1	acts_in	n_1	Clint Eastwood	e_3	acts_in	n_3	Anna Levine
4	n_2	e_3	acts_in	n_3	Anna Levine	e_1	acts_in	n_1	Clint Eastwood
5	n_2	e_2	directs	n_1	Clint Eastwood	e_1	acts_in	n_1	Clint Eastwood
6	n_2	e_1	acts_in	n_1	Clint Eastwood	e_2	directs	n_1	Clint Eastwood
7	n_2	e_1	acts_in	n_1	Clint Eastwood	e_1	acts_in	n_1	Clint Eastwood
8	n_2	e_2	directs	n_1	Clint Eastwood	e_2	directs	n_1	Clint Eastwood
9	n_2	e_3	acts_in	n_1	Anna Levine	e_3	acts_in	n_1	Anna Levine

Which results would be obtained when applying **isomorphism-based** semantics? And **homomorphism-based** semantics?

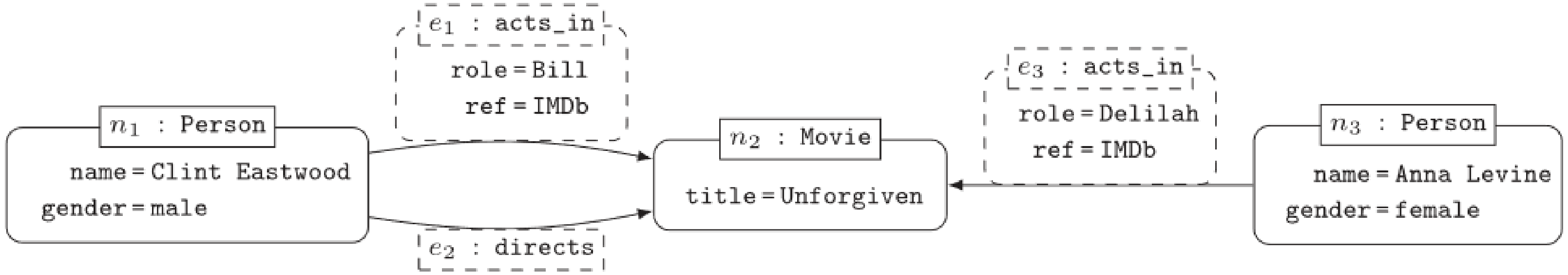
- For isomorphism, **distinguish the three isomorphism semantics** presented

Activity

Objective: Understand the relationship between RPQs and BGPs

Assume a graph containing relationships and nodes like the ones shown below

- Define a BGP to find *all co-actors of all actors*
- Define a BGP to retrieve *all actors you can reach by following the co-actoring path at least once*
- Define a BGP to find *all persons that participate in the same movie*

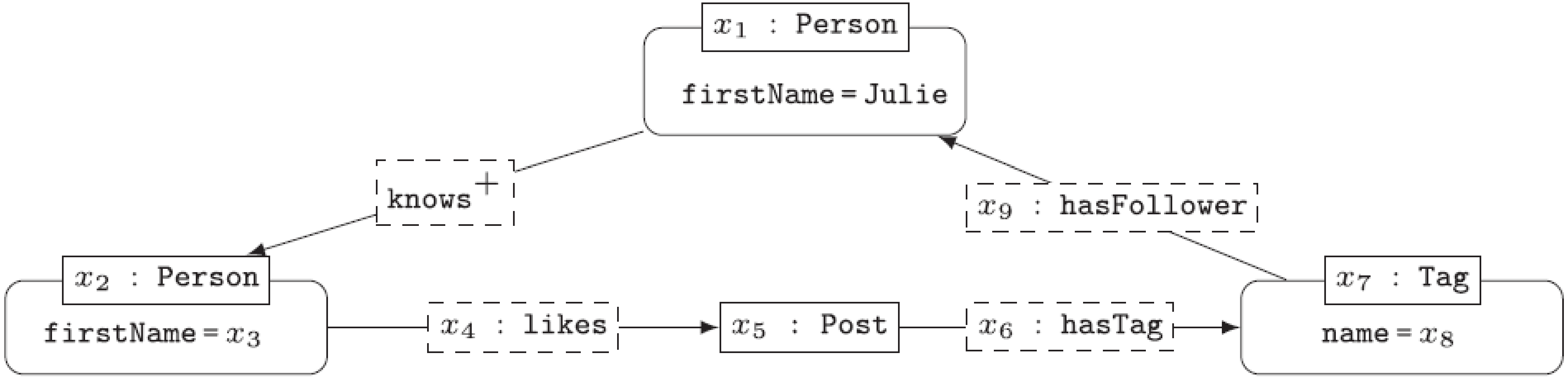


Navigational Graph Patterns (NGPs)

NGPs are a combination of Pattern Matching and Path Queries

- BGPs where edge labels can be RPQs

Example:



Graph Metrics

Take into account the graph topology only

They can be defined as combinations of adjacency, reachability, pattern matching

- Given their relevance, they are typically provided as built-in functions

Examples:

- the min / max degree in the graph
- the graph diameter
- the graph density / sparsity
- betwenness of a node
- the pageRank of a node
- …

Summary

The basic operations on graphs are:

- Adjacency
- Path queries
- Graph patterns

The result of a Pattern Matching query depends on the semantics assumed:

- Homomorphism
- Strict Isomorphism
- No-repeated-node Isomorphism
- No-repeated-edge Isomorphism

Thanks! Any Question?