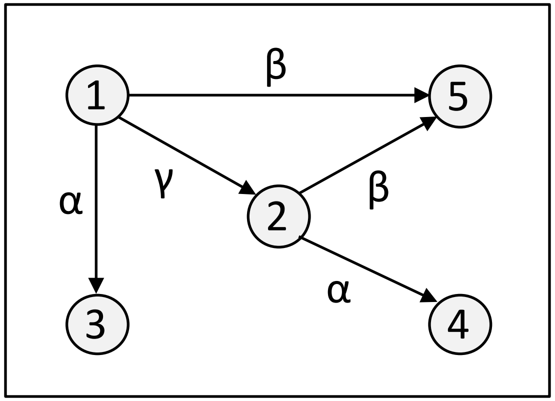
Graph-structured Data

Graph Databases and Applications

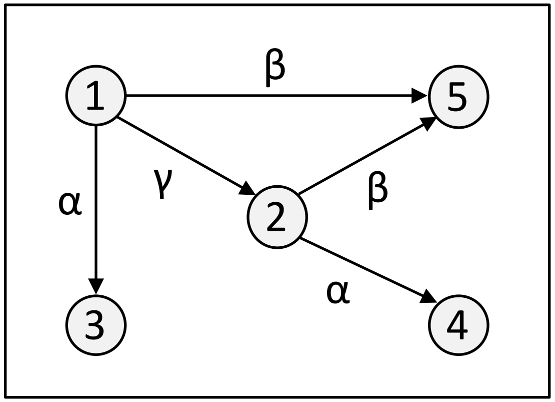
- Graph databases are crucial when topology is as important as the data

- Several modern applications
  - Semantic Web and RDF
  - Social networks
  - Knowledge graphs
  - etc.



Graph Databases vs. Relational Databases

- Simply use standard relational databases



Graph	id_o	label	id_t
	1	α	3
	1	β	5
	1	γ	2
	2	β	5
	2	α	4

- Problems:
  - We need to navigate the graph - recursion is needed
  - We can use Datalog - performance issues (complexity mismatch, basic static analysis tasks are undecidable)

Graph Data Model

- Different applications gave rise to different graph data models

- But, the essence is the same

finite, directed, edge labeled graphs

Graph Data Model

An graph database G over a finite alphabet Λ is a pair

(V, E)

finite set of node ids

set of edges of the form $v \xrightarrow{\alpha} u$

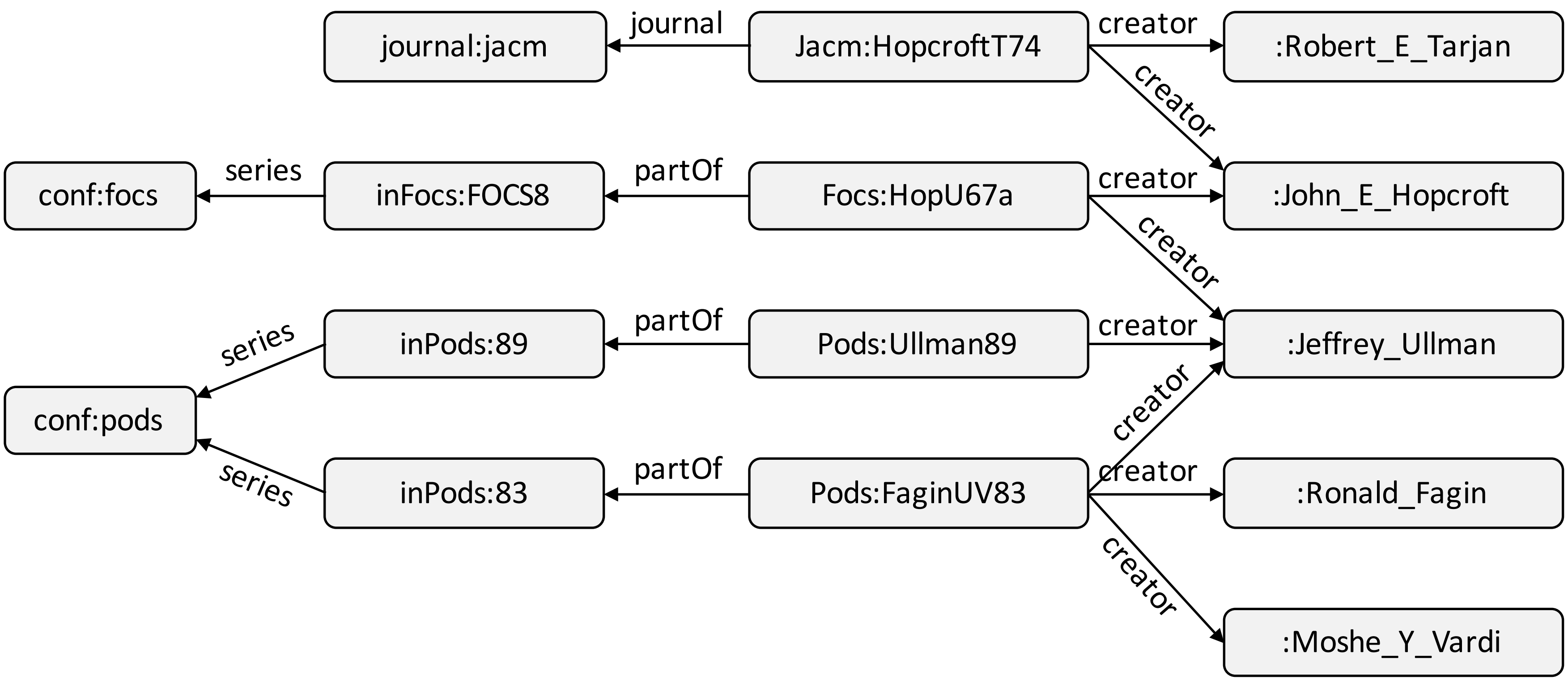
where u,v ∈ V and α ∈ Λ

Path in G: $\pi = v_1 \xrightarrow{\alpha_1} v_2 \xrightarrow{\alpha_2} v_3 \;\cdots\; v_k \xrightarrow{\alpha_k} v_{k+1}$

The label of π is $\lambda(\pi) = \alpha_1\alpha_2\alpha_3\ldots\alpha_k \in \Lambda^*$

Graph Database: Example

A graph database representation of a fragment of DBLP



Regular Path Queries (RPQs)

Basic building block of graph queries

- First studied in 1989
- An RPQ is a regular expression over a finite alphabet Λ
- Given a graph database $G = (V,E)$ over Λ and RPQ Q over Λ

$$Q(G) = \{(v,u) \mid v,u \in V \text{ and there is a path } \pi \text{ from } v \text{ to } u \text{ such that } \lambda(\pi) \in L(Q)\}$$

RPQs With Inverses (2RPQs)

Extension of RPQs with inverses - two-way RPQs

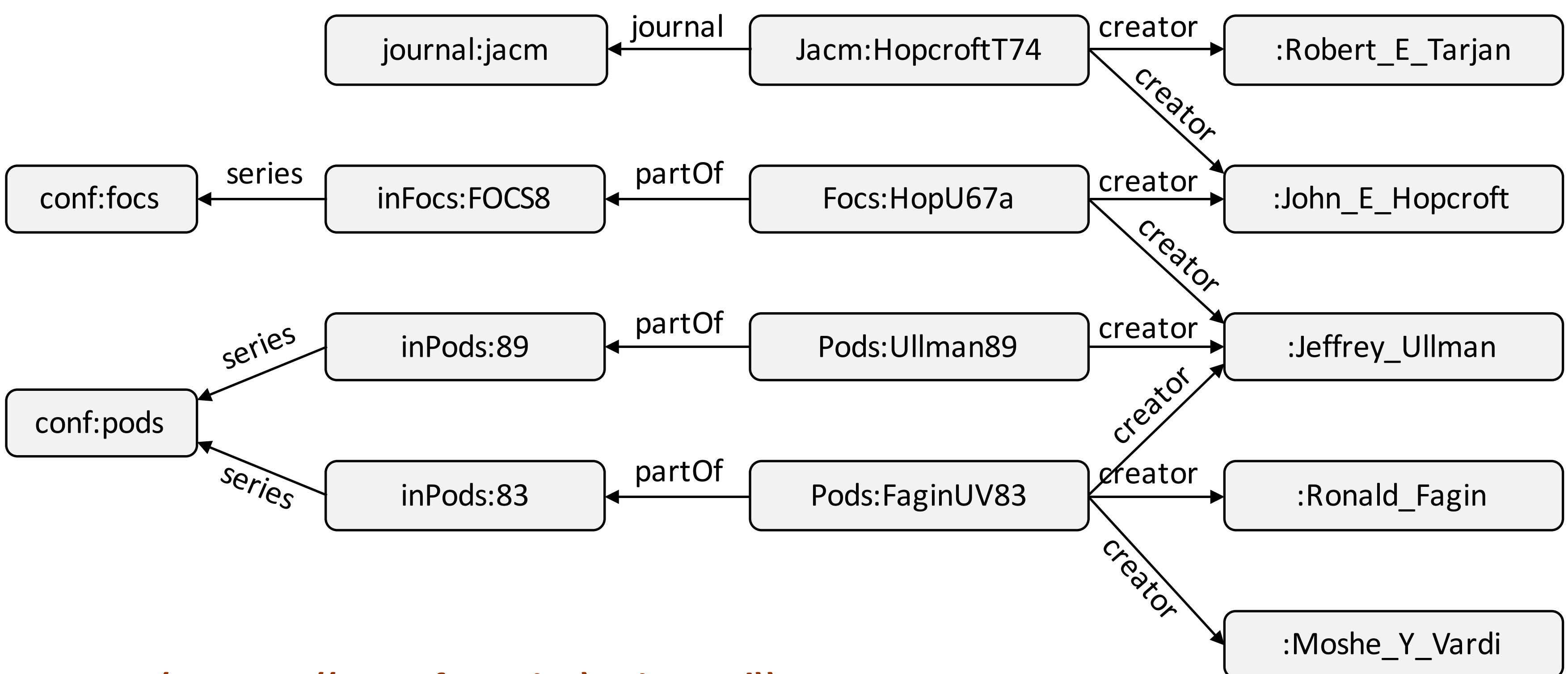
- First studied in 2000
- 2RPQs over Λ = RPQs over $\Lambda^{\pm} = \Lambda \cup \{\alpha^{-} \mid \alpha \in \Lambda\}$
- Given a graph database $G = (V,E)$ over Λ and 2RPQ Q over Λ

$$Q(G) = Q(G^{\pm})$$

obtained from G by adding $u \xrightarrow{\alpha^{-}} v$ for each $v \xrightarrow{\alpha} u$

Querying Graph Database

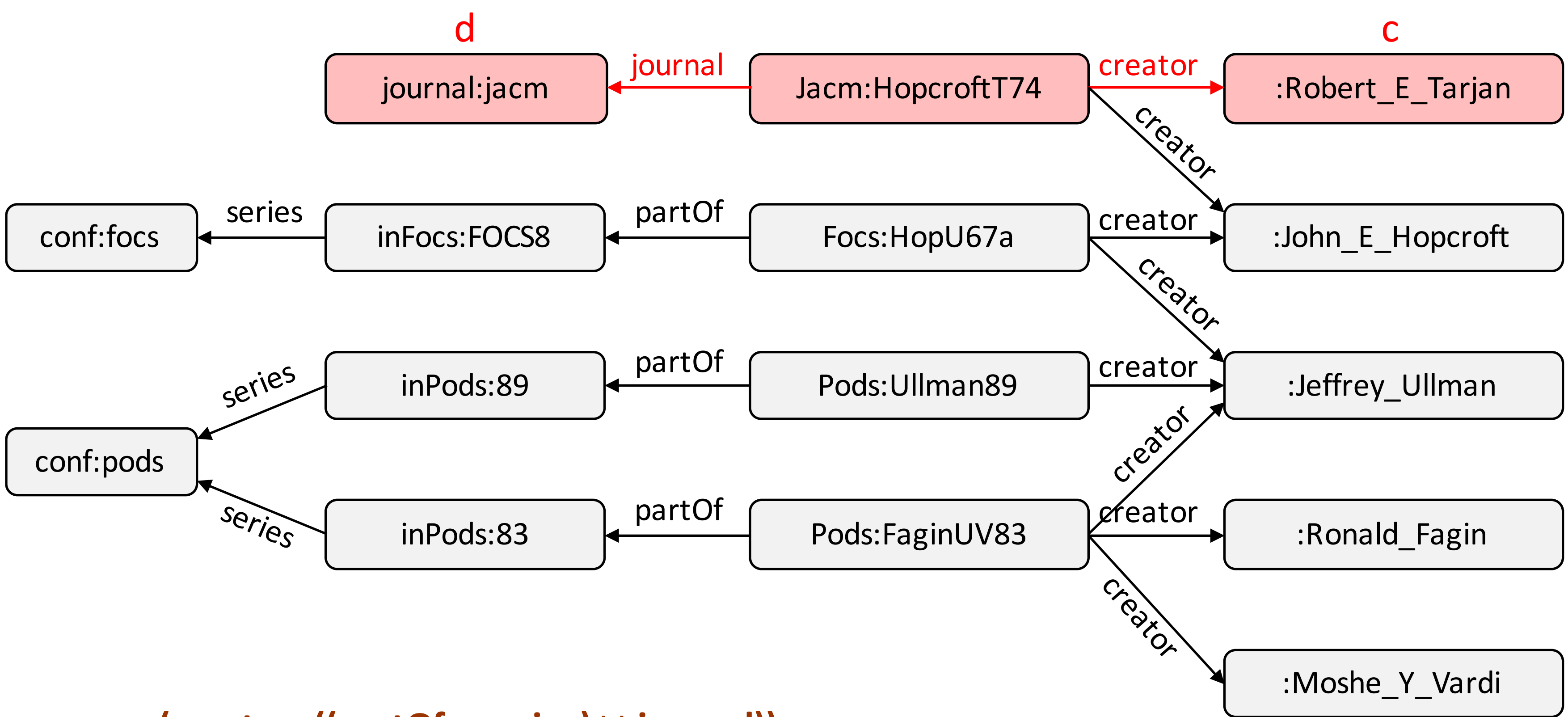
Compute the pairs (c,d) such that author c has published in conference or journal d



(creator⁻ ((partOf · series) ∪ journal))

Querying Graph Database

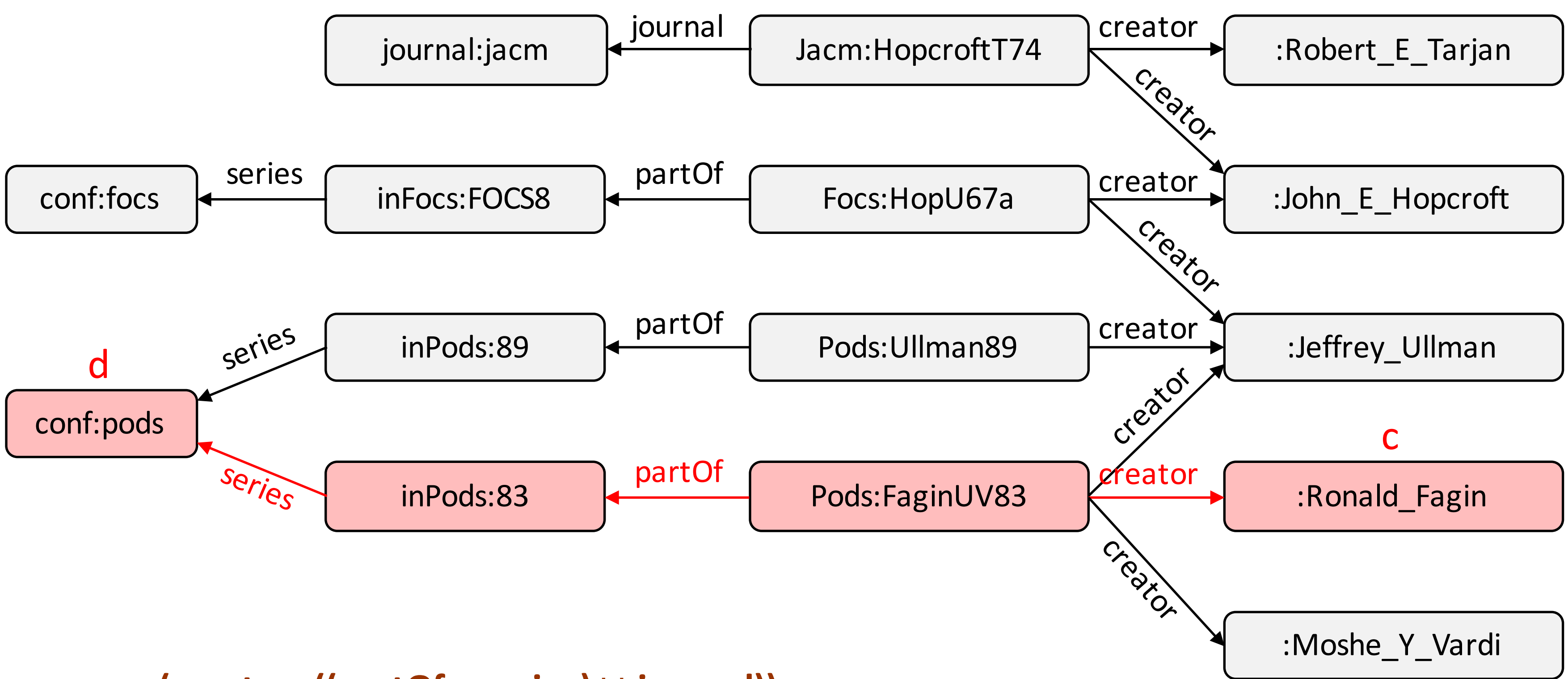
Compute the pairs (c,d) such that author c has published in conference or journal d



$(\text{creator}^{-}\ ((\text{partOf}\ \cdot\ \text{series}) \cup \text{journal}))$

Querying Graph Database

Compute the pairs (c,d) such that author c has published in conference or journal d



$$(\mathbf{creator}^{-}\ ((\mathbf{partOf} \cdot \mathbf{series}) \cup \mathbf{journal}))$$

Evaluation of 2RPQs

EVAL(**2RPQ**)

Input: a graph database G, a 2RPQ Q, two nodes v,u of G

Question: $(v,u) \in Q(G)$?

It boils down to the problem:

RegularPath

Input: a graph database G over Λ, a regular expression Q over $\Lambda^{\pm}$, two nodes v,u of G

Question: is there a path π from v to u in $G^{\pm}$ such that $\lambda(\pi) \in L(Q)$

Complexity of RegularPath

Theorem: RegularPath can be solved in time $O(|G| \cdot |Q|)$

Proof Idea: by exploiting nondeterministic finite automata (NFA)

- Compute in linear time from Q an equivalent NFA A_Q
- Compute in linear time an NFA A_G obtained from $G^{\pm}$ by setting v and u as initial and finite states, respectively
- There is a path π from v to u in $G^{\pm}$ such that $\lambda(\pi) \in L(Q)$ iff $L(A_G) \cap L(A_Q)$ is non-empty
- Non-emptiness can be checked in time $O(|A_G| \cdot |A_Q|) = O(|G| \cdot |Q|)$

A graph database can be naturally seen as an NFA

- nodes are states
- edges are transitions

Complexity of 2RPQs

We immediately get that:

Theorem: EVAL(**2RPQ**) can be solved in time $O(|G| \cdot |Q|)$

Regarding the data complexity (i.e., Q is fixed):

Theorem: EVAL[Q] (**2RPQ**) is in NLOGSPACE

(by exploiting the previous automata construction)

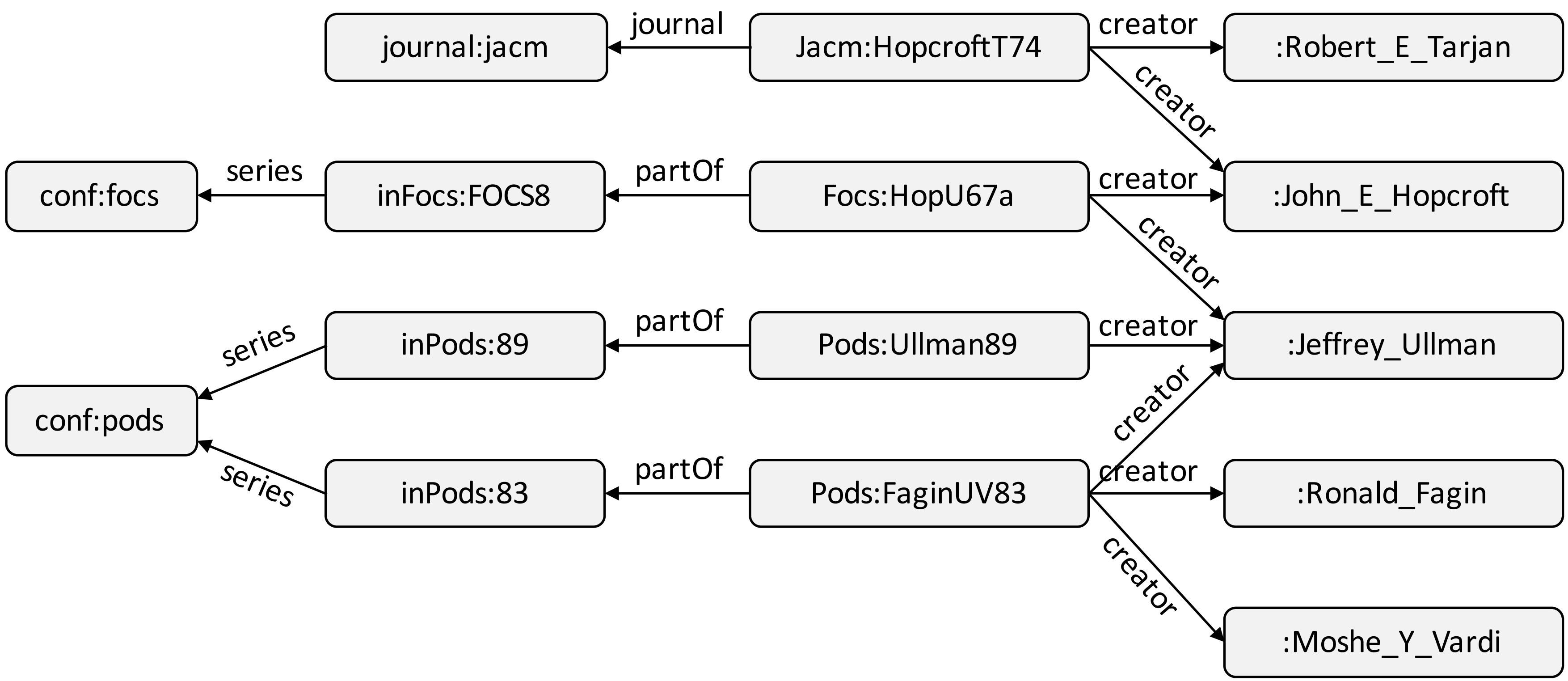
Limitation of RPQs

- RPQs are not able to express arbitrary patterns over graph databases (e.g., compute the pairs (c,d) that are coauthors of a conference paper)

- We need to enrich RPQs with joins and projections
  - Conjunctive regular path queries (CRPQs)
  - C2RPQs if we add inverses

C2RPQs: Example

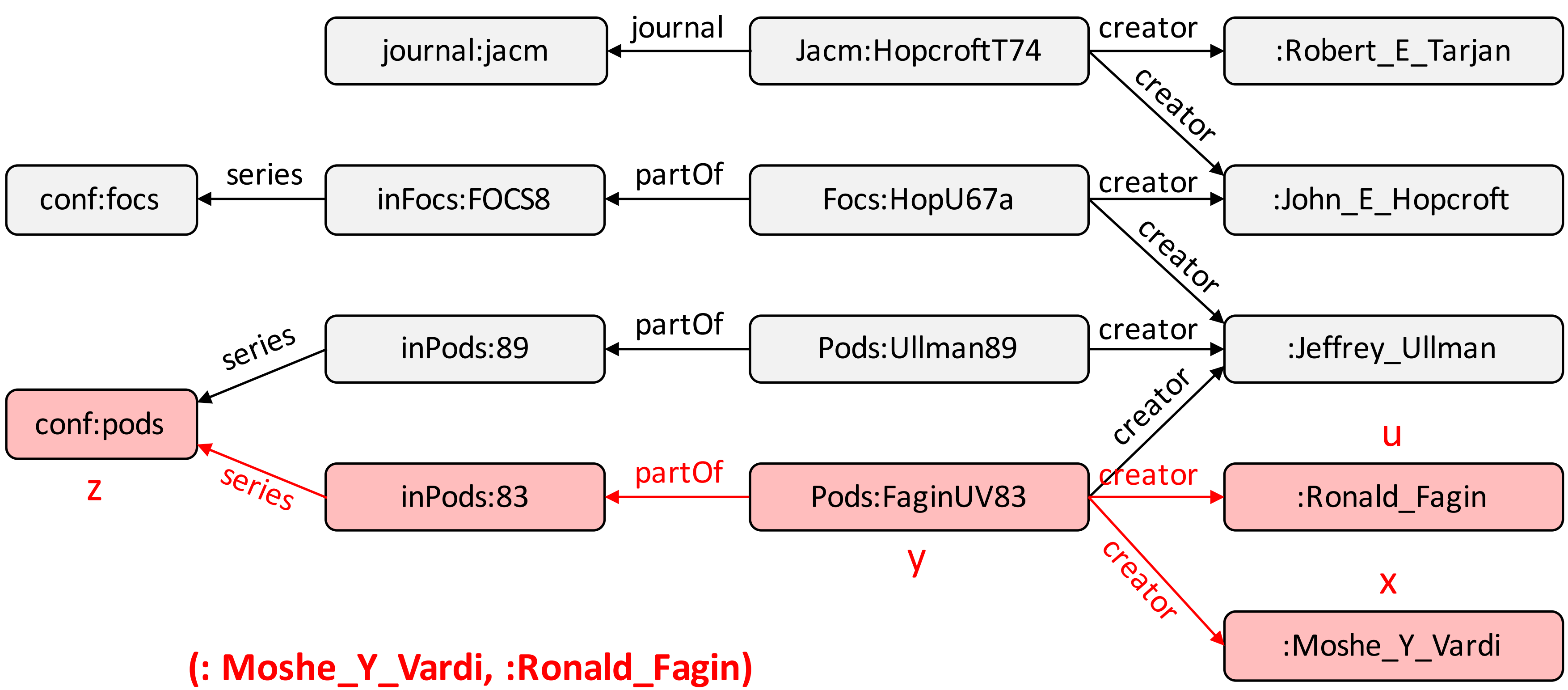
Compute the pairs (c,d) that are coauthors of a conference paper



C2RPQs: Example

Compute the pairs (c,d) that are coauthors of a conference paper

Q(x,u) :- (x, creator⁻, y), (y, partOf · series, z), (y, creator , u)



(: Moshe_Y_Vardi, :Ronald_Fagin)

C2RPQs: Formal Definition

A C2RPQ over an alphabet Λ is a rule of the form

$$Q(\mathbf{z}) \text{ :- } (x_1, Q_1, y_1), \ldots, (x_n, Q_n, y_n)$$

where x_i, y_i are variables,

Q_i is a 2RPQ over Λ,

$\mathbf{z}$ are the output variables from $\{x_1, y_1, \ldots, x_n, y_n\}$

Remark: C2RPQs are more expressive than 2RPQs (previous example)

Evaluation of C2RPQs

To evaluate a C2RPQ of the form

$$Q(\mathbf{z}) \text{ :- } (x_1, Q_1, y_1), \ldots, (x_n, Q_n, y_n)$$

we simply need to evaluate the conjunctive query

$$Q(\mathbf{z}) \text{ :- } Q_1(x_1, y_1), \ldots, Q_n(x_n, y_n)$$

where each Q_i stores the result of evaluating the 2RPQ Q_i

Complexity of C2RPQs

Theorem: EVAL(**C2RPQ**) is NP-complete

Proof Hints:

- **Upper bound:** polynomial time reduction to EVAL(**CQ**)
- **Lower bound:** inherited from CQs over graphs

Regarding the data complexity (i.e., Q is fixed):

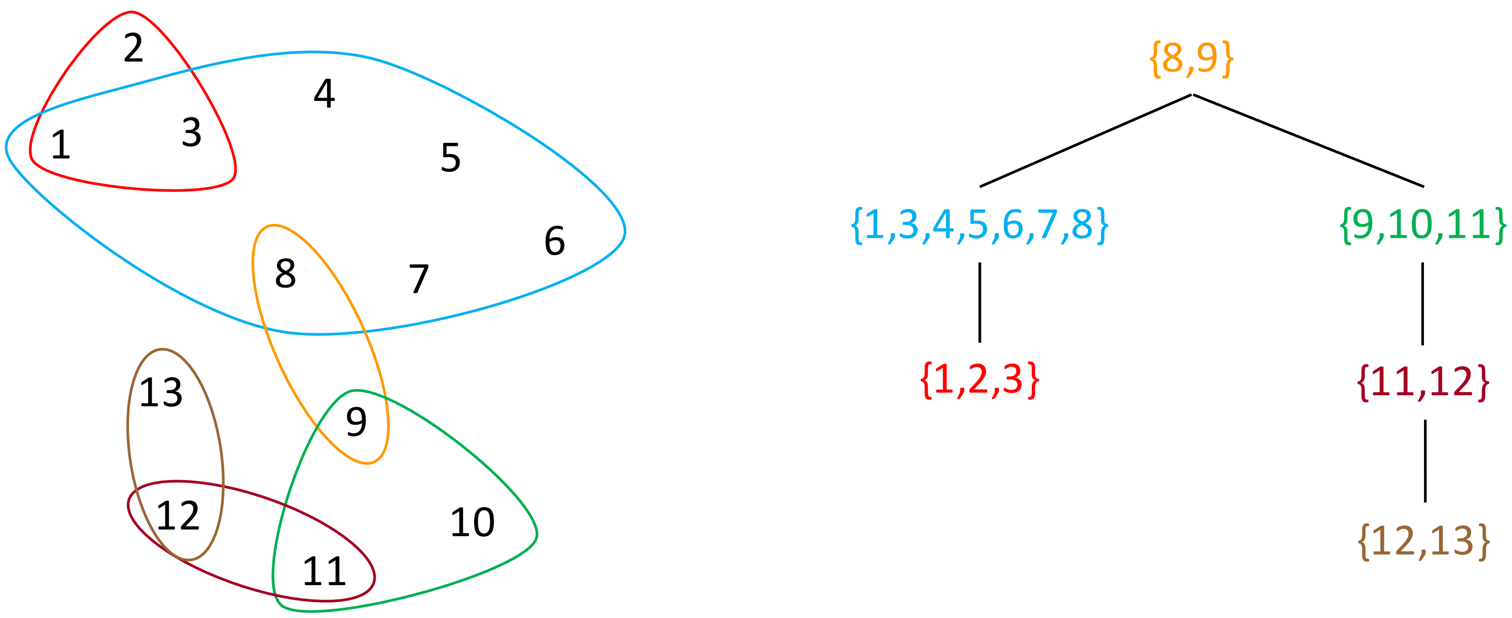
Theorem: EVAL[Q](**C2RPQ**) is in NLOGSPACE

Basic Graph Query Languages: Recap

- Two-way regular path queries (2RPQs)
  - Can be evaluated in linear time in combined complexity, and in NLOGSPACE in data complexity
- Conjunctive 2RPQs (C2RPQs)
  - Evaluation is NP-complete in combined complexity, and in NLOGSPACE in data complexity

Towards Tractable C2RPQs

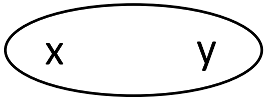
Recall acyclic conjunctive queries



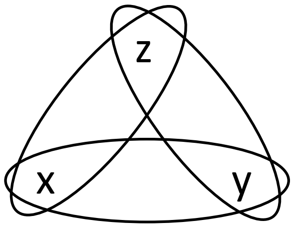
Acyclic C2RPQs

A C2RPQ is acyclic if its underlying CQ is acyclic

Q :- (x, Q_1, x), (x, Q_2, y), (y, Q_3, x)



Q :- (x, Q_1, y), (y, Q_2, z), (z, Q_3, x)



Equivalently, the underlying graph does not contain cycles of length ≥ 3

Complexity of Acyclic C2RPQs

Theorem: EVAL(**AC2RPQ**) can be solved in time $O(|G|^2 \cdot |Q|^2)$

{$Q \in$ **C2RPQ** | Q is acyclic}

Proof Idea: recall that we can reduce EVAL(**C2RPQ**) to EVAL(**CQ**)

Simple Path Semantics

Simple Path: No node is repeated

In this case, EVAL(**2RPQ**) boils down to the problem:

RegularSimplePath

Input: a graph database G over Λ, a regular expression Q over $\Lambda^{\pm}$,

two nodes v,u of G

Question: is there a simple path π from v to u in $G^{\pm}$ such that $\lambda(\pi) \in L(Q)$

Simple Path Semantics

Theorem: RegularSimplePath is NP-complete

Theorem: RegularSimplePath[Q] is NP-complete (data complexity)

- $\text{RegularSimplePath}_{(0.0)^*}$
- Is there a simple directed path of even length? NP-complete
- NP-complete data complexity means impractical

Containment of Graph Queries

CONT(**L**)

Input: two queries $Q_1 \in \mathbf{L}$ and $Q_2 \in \mathbf{L}$

Question: $Q_1 \subseteq Q_2$? (i.e., $Q_1(G) \subseteq Q_2(G)$ for every graph database G?)

Containment of Graph Queries

Theorem: CONT(**RPQ**) is PSPACE-complete

Proof Hint: exploit containment of regular expressions

Theorem: CONT(**2RPQ**) is PSPACE-complete

Proof Hint: exploit containment of two-way automata, while the lower bound is inherited from RPQs

Theorem: CONT(**C2RPQ**) is EXPSPACE-complete

Proof Hint: exploit containment of two-way automata, while the lower bound is by reduction from a tiling problem

Querying Graphs With Data

- So far queries talk about the topology of the data

- However, graph databases contain data - data graphs

- We have query languages that can talk about data paths

  (obtained by replacing each node in a path by its value)