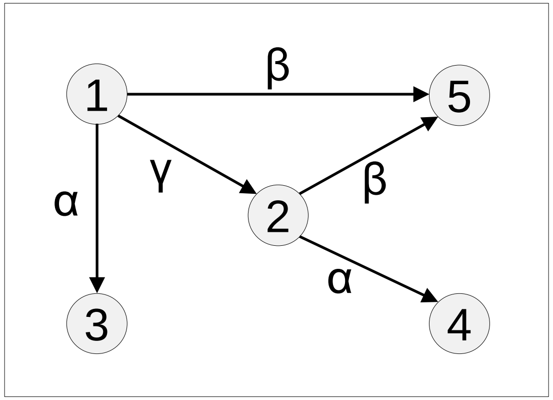
Graph Databases

Graph Databases and Applications

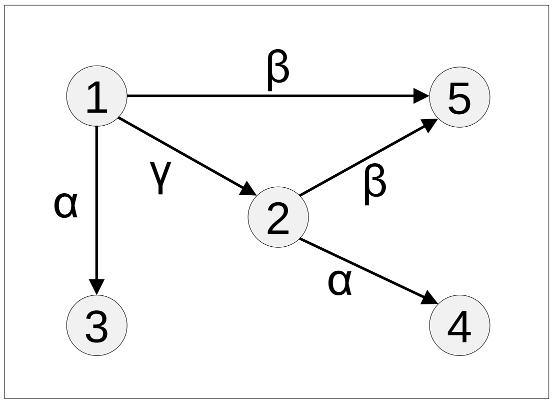
- Graph databases are crucial when topology is as important as the data

- 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 task 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 $v \xrightarrow{\alpha} u$

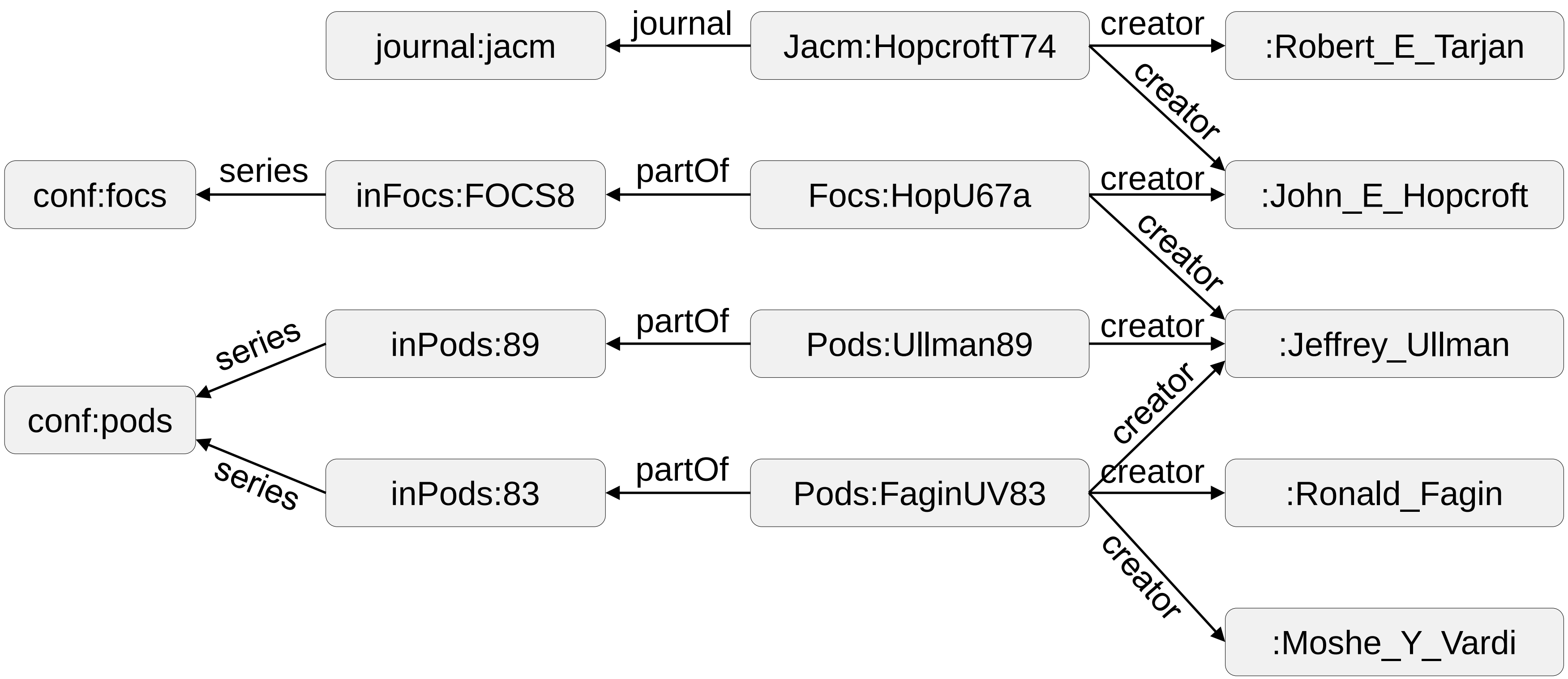
where $u,v \in V$ and $\alpha \in \Lambda$

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

$Q(G) = \{(v,u) \mid v,u \in V$ and

there is a path π from v to u such that $\lambda(\pi) \in L(Q)\}$

RPQs With Inverses (2RPQs)

Extension of RPQs with inverses – two-way RPQs

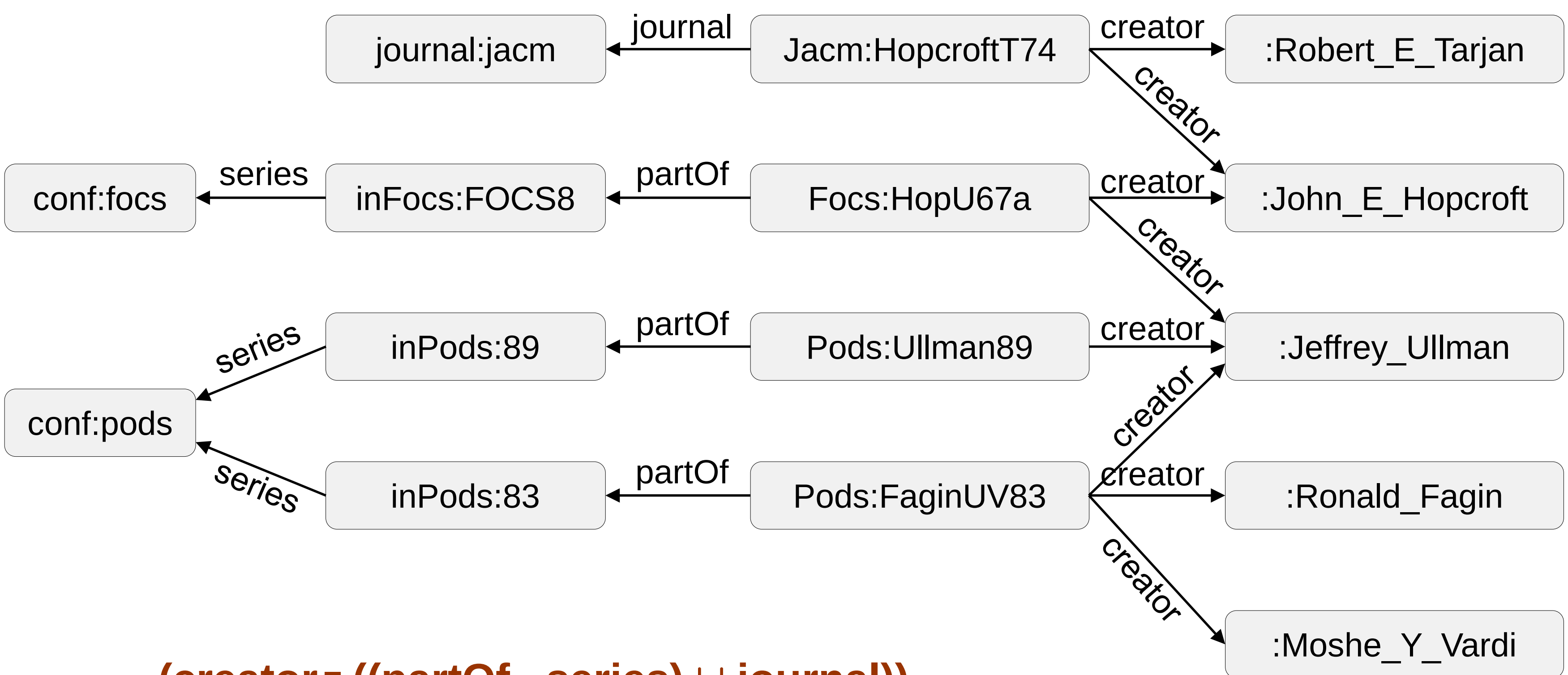
- First studied in 2000
- 2RPQs over Λ = RPQs over $\Lambda^{\pm} = \Lambda \cup \{\alpha^{-} \mid \alpha \in \Lambda\}$
- Given a graph database $G = (V,E)$ over Λ and 2RPQ Q over Λ

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

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

Querying Graph Database

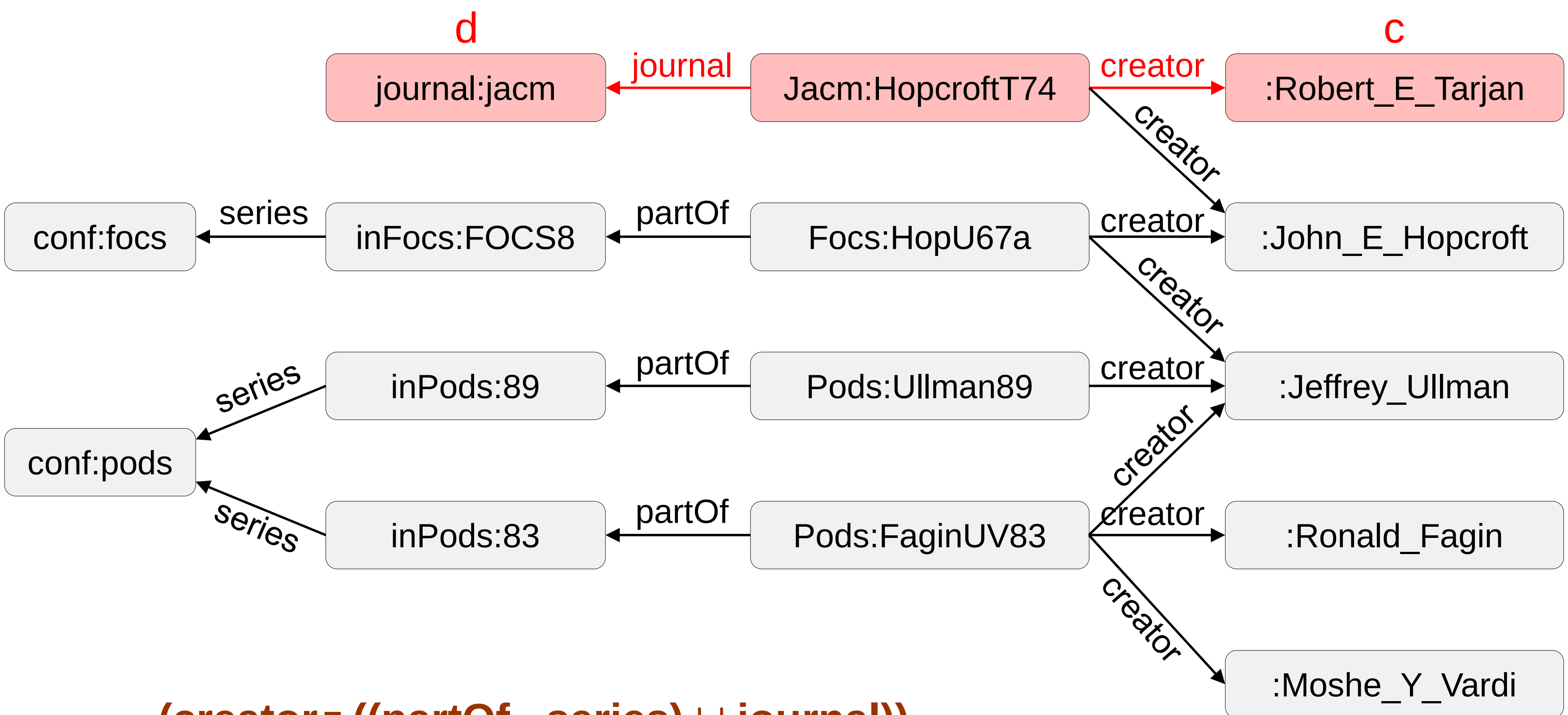
Compute the pairs (c,d) such that author c has published in conference or journal d



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

Querying Graph Database

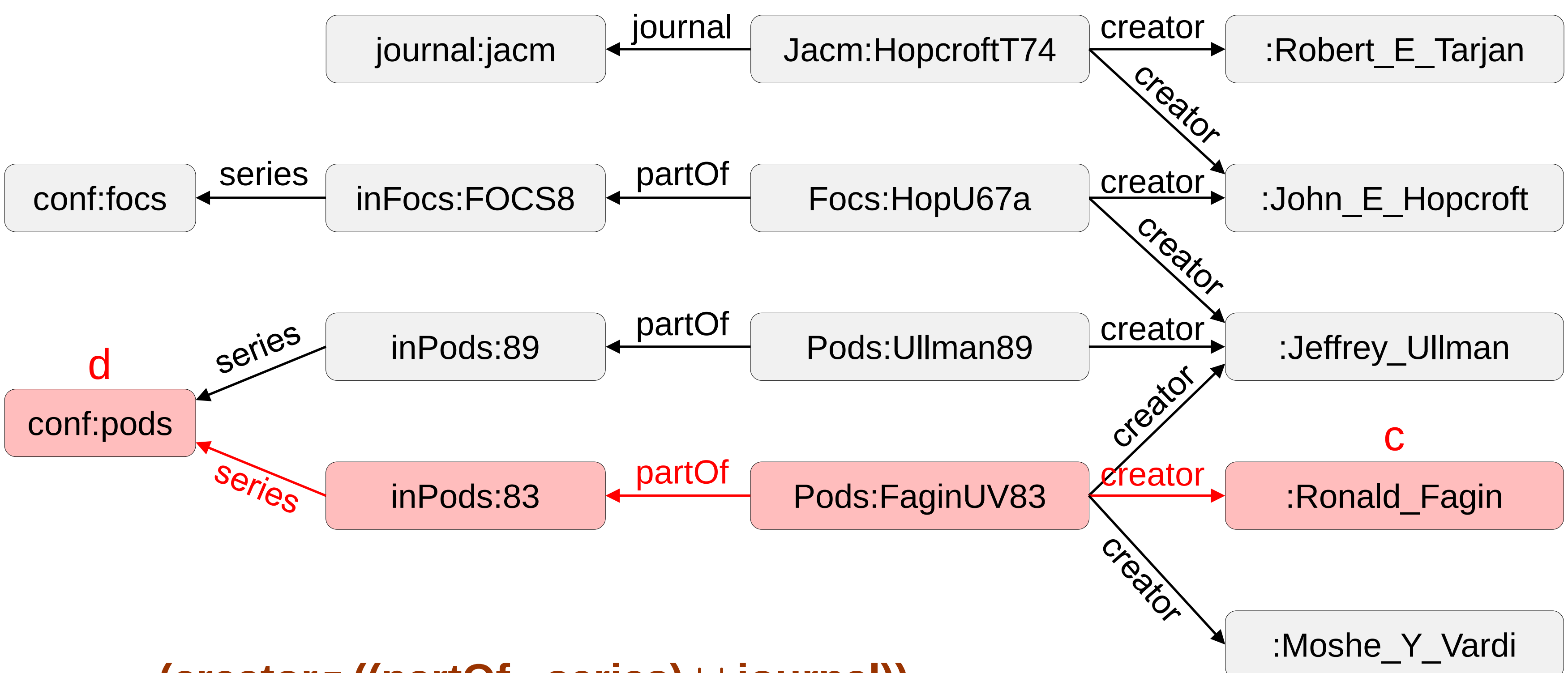
Compute the pairs (c,d) such that author c has published in conference or journal d



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

Querying Graph Database

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



(creator^{-} ((partOf · series) $\cup$ 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)

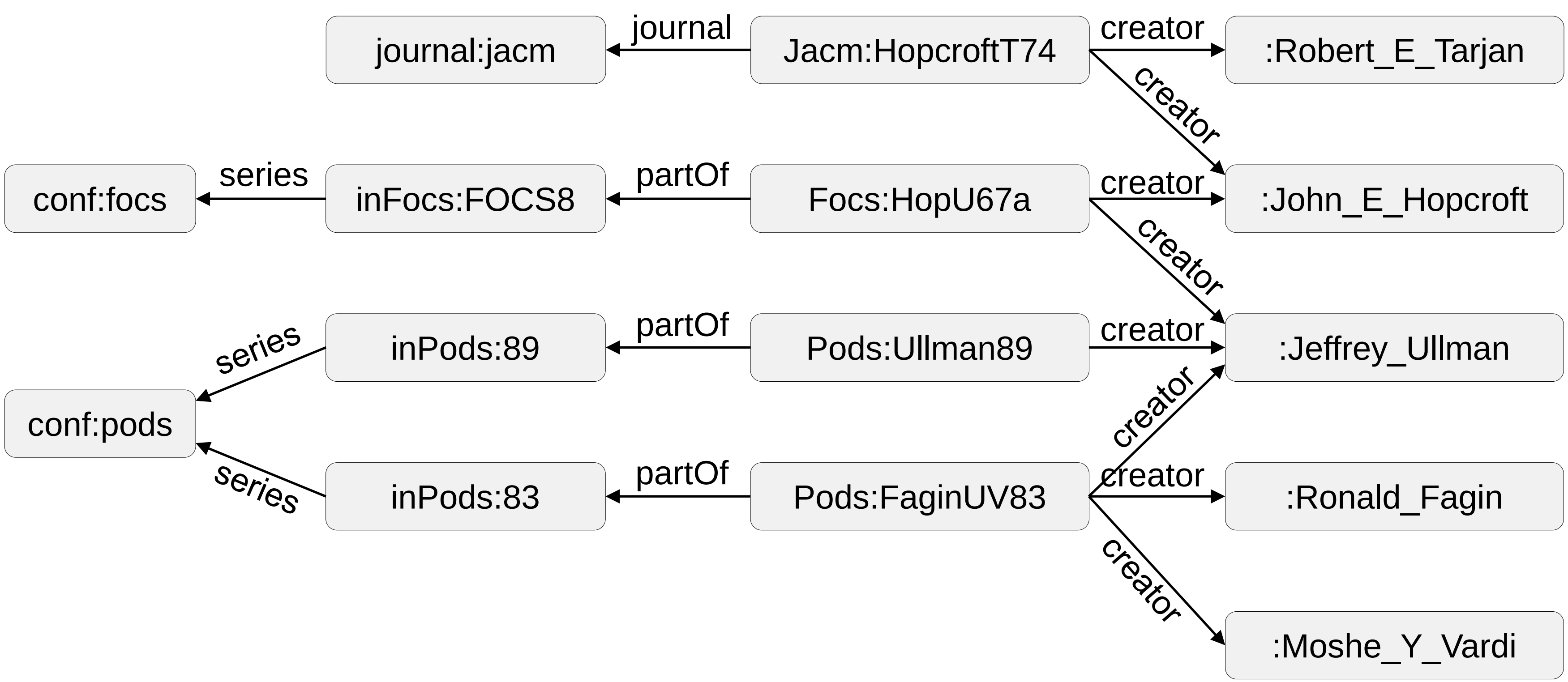
Limitation of RPQs

- 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

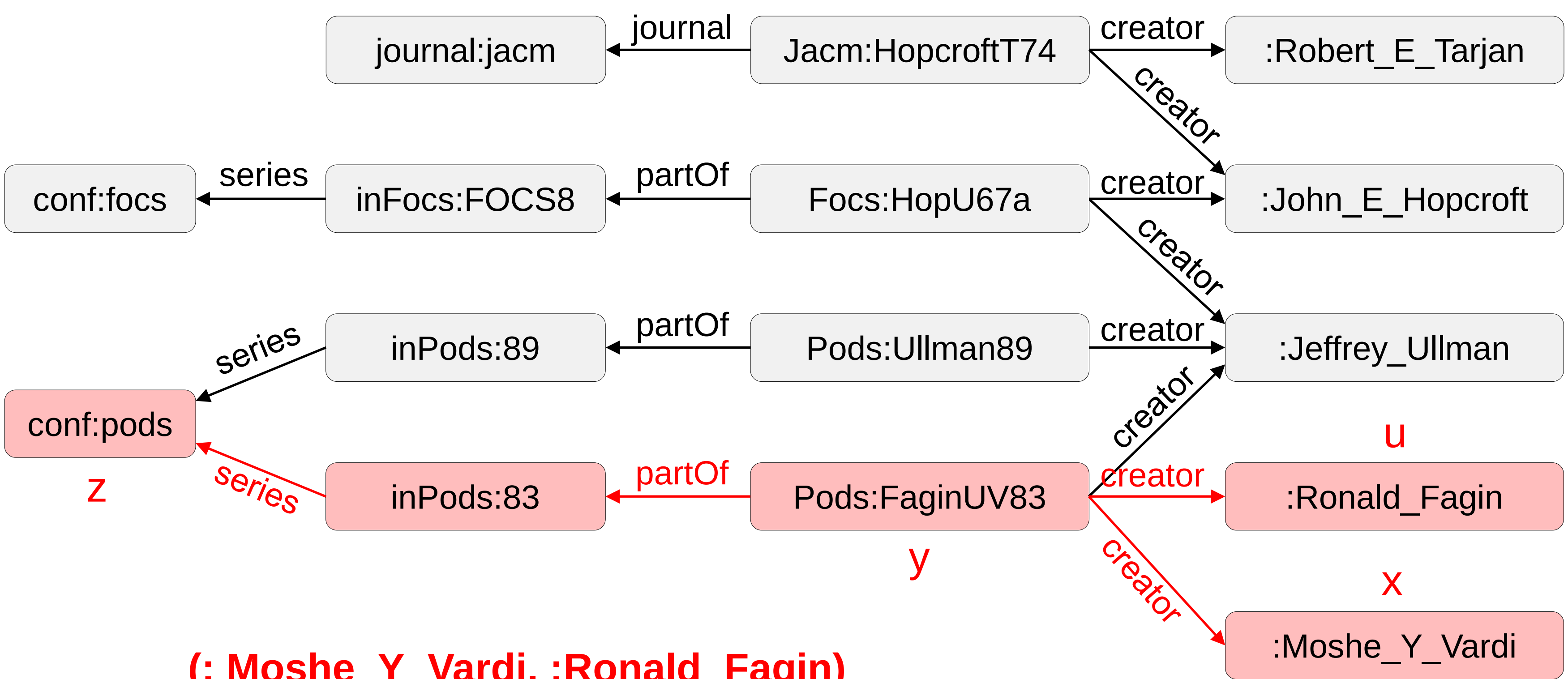
Compute the pairs (c,d) that are coauthors of a conference paper



C2RPQs: Example

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

$Q(x,u)$:- $(x, \text{creator}^{-}, y), (y, \text{partOf} \cdot \text{series}, z), (y, \text{creator}, u)$



(: 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):

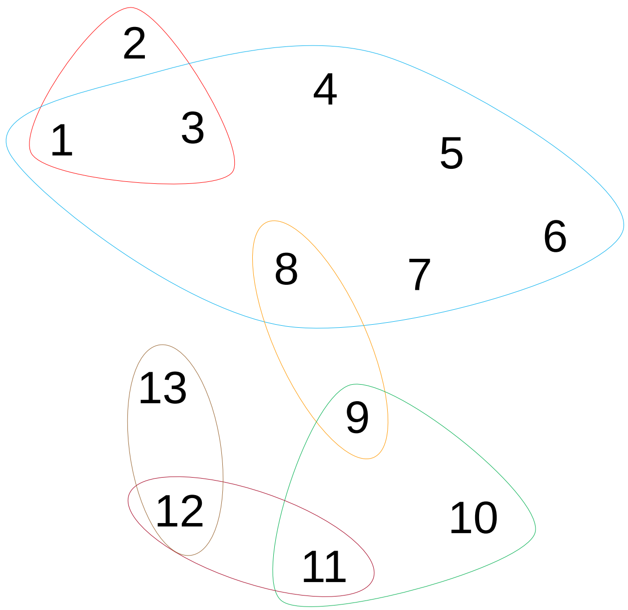
Theorem: EVAL_{Q}(**C2RPQ**) is in NLOGSPACE

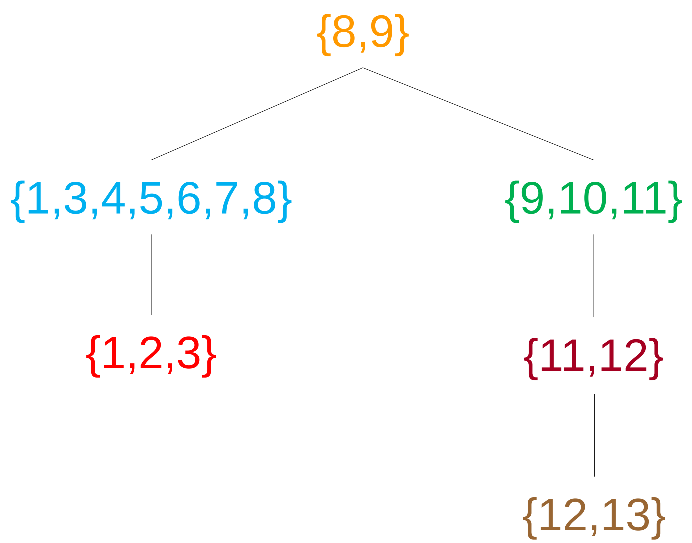
Basic Graph Query Languages: Recap

- 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

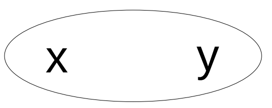




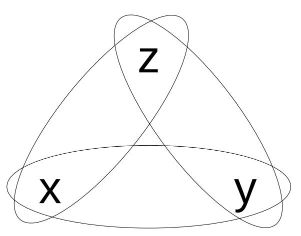
Acyclic C2RPQs

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** $\mid Q$ 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)

- RegularSimplePath$_{(0 \cdot 0)^*}$
- 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

Limitations of CRPQs

Compute the pairs (c,d) that are linked by a path labeled in $\{\alpha^n\beta^n \mid n \geq 0\}$

$$v \xrightarrow{\pi_1} w \xrightarrow{\pi_2} u$$

such that $\lambda(\pi_1) \in L(\alpha^*)$ and $\lambda(\pi_2) \in L(\beta^*)$ and $|\lambda(\pi_1)| = |\lambda(\pi_2)|$

Not expressible using CRPQs. We need:

- To define complex relationships among labels of paths
- To include paths in the output of a query

Comparing Paths With Regular Relations

- Regular languages for n-ary relations
- n-ary regular relations: set of n-tuples $(w_1,\ldots,w_n)$ of words over an alphabet Λ
- Accepted by a synchronous automaton over Λ^n
  - The input strings are written in the n-tapes
  - Shorter strings are padded with the symbol # not in Λ
  - At each step, the automaton simultaneously reads the next symbol on each tape, terminating when it reads # on each tape

Synchronous Automata

w_1	=	α	α	β	...	α	β	γ
w_2	=	α	β	α	...	α		
w_3	=	β	β		...			
...								
w_n	=	α	β	β	...	α	γ	

Synchronous Automata

w_1	=	α	α	β	...	α	β	γ
w_2	=	α	β	α	...	α	#	#
w_3	=	β	β	#	...	#	#	#
...								
w_n	=	α	β	β	...	α	γ	#

Synchronous Automata

w_1	=	α	α	β	...	α	β	γ
w_2	=	α	β	α	...	α	#	#
w_3	=	β	β	#	...	#	#	#
...								
w_n	=	α	β	β	...	α	γ	#

Synchronous Automata

w_1	=	α	α	β	...	α	β	γ
w_2	=	α	β	α	...	α	#	#
w_3	=	β	β	#	...	#	#	#
...								
w_n	=	α	β	β	...	α	γ	#

Synchronous Automata

w_1	=	α	α	β	...	α	β	γ
w_2	=	α	β	α	...	α	#	#
w_3	=	β	β	#	...	#	#	#
...								
w_n	=	α	β	β	...	α	γ	#

Regular Relations: Examples

- All regular languages – regular relations of arity 1
- Path equality: $w_1 = w_2$
- Length comparison: $|w_1| = |w_2|$, $|w_1| < |w_2|$, $|w_1| \leq |w_2|$
- Prefix: w_1 is a prefix of w_2

Extended CRPQs With Regular Relations (REG)

An ECRPQ(REG) is a rule obtained from a CRPQ as follows

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

annotate each pair (x_i,y_i) with a path variable π_i

$$Q(\mathbf{z}) \coloneq (x_1, \pi_1, y_1), \ldots, (x_n, \pi_n, y_n)$$

compare labels of paths in $\boldsymbol{\pi}_j$ w.r.t. $S_j \in$ REG

$$Q(\mathbf{z}) \coloneq (x_1, \pi_1, y_1), \ldots, (x_n, \pi_n, y_n), \wedge_j S_j(\boldsymbol{\pi}_j)$$

output some of π_i's as a tuple $\boldsymbol{\pi}$ in the output

$$\boxed{Q(\mathbf{z},\boldsymbol{\pi}) \coloneq (x_1, \pi_1, y_1), \ldots, (x_n, \pi_n, y_n), \wedge_j S_j(\boldsymbol{\pi}_j)}$$

Evaluation of EC2RPQ(REG)

$Q(\mathbf{z},\boldsymbol{\pi})$:- $(x_1, \pi_1, y_1), \ldots, (x_n, \pi_n, y_n), \wedge_j S_j(\boldsymbol{\pi}_j)$

Same as CRPQs, but

- Each π_i is mapped to a path ρ_i in the graph database
- For each j, if $\boldsymbol{\pi}_j = (\pi_{j1},\ldots,\pi_{jk}) \Rightarrow (\lambda(\rho_{j1}),\ldots,\lambda(\rho_{jk})) \in S_j$

Example of ECRPQ(REG)

Compute the pairs (c,d) that are linked by a path labeled in $\{\alpha^n\beta^n \mid n \geq 0\}$

$$v \xrightarrow{\pi_1} w \xrightarrow{\pi_2} u$$

such that $\lambda(\pi_1) \in L(\alpha^*)$ and $\lambda(\pi_2) \in L(\beta^*)$ and $|\lambda(\pi_1)| = |\lambda(\pi_2)|$

$Q(x,y)$:- (x, π_1, z), (z, π_2, y), $\alpha^*(\pi_1)$, $\beta^*(\pi_2)$, Equal_Length(π_1,π_2)

ECRPQ(REG) vs. CRPQs

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

$$\equiv$$

$Q(\mathbf{z})$:- $(x_1, \pi_1, y_1), \ldots, (x_n, \pi_n, y_n), Q_1(\pi_1), \ldots, Q_n(\pi_n)$

Complexity of EC2RPQ(REG)

Theorem: It holds that

- EVAL(**ECRPQ(REG)**) is PSPACE-complete
- EVAL$_{Q}$(**ECRPQ(REG)**) is in NLOGSPACE (data complexity)
- CONT(**ECRPQ(REG)**) is undecidable

Beyond Regular Relations

- Subsequences – w_1 is a subsequence of w_2, i.e., w_1 can be obtained from w_2 by deleting some letters

- Subword: $w_3 \cdot w_1 \cdot w_4 = w_2$

…we can exploit rational relations (RAT) - **ECRPQ(RAT)**

Path Query Languages: Recap

- CRPQs do not allow to compare labels of paths and export paths
- This has led to the introduction of ECRPQ(REG)
  - Preserves data tractability
  - But containment becomes undecidable
- We can go beyond REG – ECRPQ(RAT)
  - Undecidability of query evaluation
  - We obtain data tractability if we restrict the syntax

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)

Associated Papers

- Mariano P. Consens, Alberto O. Mendelzon: Low Complexity Aggregation in GraphLog and Datalog. Theor. Comput. Sci. 116(1): 95-116 (1993)

  One of the papers introducing (C)RPQs

- Pablo Barcelo: Querying graph databases. PODS 2013: 175-188
- Renzo Angles, Claudio Gutierrez: Survey of graph database models. ACM Comput. Surv. 40(1) (2008)

  Two surveys of graph languages, two are more theoretical, one more practical

Associated Papers

- Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Moshe Y. Vardi: Rewriting of Regular Expressions and Regular Path Queries. J. Comput. Syst. Sci. 64(3): 443-465 (2002)

  Introducing two-way queries

- Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Moshe Y. Vardi: Reasoning on regular path queries. SIGMOD Record 32(4): 83-92 (2003)

- Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Moshe Y. Vardi: Containment of Conjunctive Regular Path Queries with Inverse. KR 2000: 176-185

  Static analysis of regular path queries

- Leonid Libkin, Wim Martens, Domagoj Vrgoc: Querying graph databases with XPath. ICDT 2013: 129-140

  Adding data values to (C)RPQs

Associated Papers

- Pablo Barcelo, Leonid Libkin, Anthony Widjaja Lin, Peter T. Wood: Expressive Languages for Path Queries over Graph-Structured Data. ACM Trans. Database Syst. 37(4): 31 (2012)

  Extending RPQs with regular relations

- Pablo Barcelo, Diego Figueira, Leonid Libkin: Graph Logics with Rational Relations. Logical Methods in Computer Science 9(3) (2013)

  Extending RPQs with rational relations

- Dominik D. Freydenberger, Nicole Schweikardt: Expressiveness and Static Analysis of Extended Conjunctive Regular Path Queries. AMW 2011

  Resolving some of the questions on the containment of path queries

- Jelle Hellings, Bart Kuijpers, Jan Van den Bussche, Xiaowang Zhang: Walk logic as a framework for path query languages on graph databases. ICDT 2013: 117-128

  A different approach to expanding the power of path languages

Associated Papers

- Pablo Barcelo, Leonid Libkin, Juan L. Reutter: Querying Regular Graph Patterns. Journal of the ACM 61(1): 8:1-8:54 (2014)

  Incomplete information in graph databases and querying it

- Wenfei Fan, Xin Wang, Yinghui Wu: Querying big graphs within bounded resources. SIGMOD Conference 2014: 301-312

- Wenfei Fan: Graph pattern matching revised for social network analysis. ICDT 2012: 8-21

  Two papers on making graph queries scalable