Outline

Navigation and Patterns

Rule-based languages

Moving to RDF

Outline

Datalog

Rule-based languages

Containment of Queries

Regular Queries

In the lasts years we have seen renewed efforts
to find expressive graph languages
that are algebraically closed

The majority are based on Datalog

Datalog for graphs

A (binary) Datalog program is a set of rules

$$R(z_1, z_1) \leftarrow P_1(x_1, y_1), \ldots, P_n(x_n, y_n)$$

Datalog for graphs

A (binary) Datalog program is a set of rules

$$R(z_1, z_1) \leftarrow P_1(x_1, y_1), \ldots, P_n(x_n, y_n)$$

$z_1, z_2 \in \{x_1, y_1, \ldots, x_n, y_n\}$

$P_1, \ldots, P_n$ are either graph labels or predicates

R is a predicate

Datalog for graphs

A (binary) Datalog program is a set of rules

$$R(z_1, z_1) \leftarrow P_1(x_1, y_1), \ldots, P_n(x_n, y_n)$$

$z_1, z_2 \in \{x_1, y_1, \ldots, x_n, y_n\}$

$P_1, \ldots, P_n$ are either graph labels or predicates

R is a predicate

Datalog programs use their own, new predicates

Datalog for graphs

A (binary) Datalog program is a set of rules

$$R(z_1, z_1) \leftarrow P_1(x_1, y_1), \ldots, P_n(x_n, y_n)$$

$z_1, z_2 \in \{x_1, y_1, \ldots, x_n, y_n\}$

$P_1, \ldots, P_n$ are either graph labels or predicates

R is a predicate

Datalog programs use their own, new predicates

(in general Datalog predicates can have any arity)

Datalog: example

(knows | helps)+

Find nodes connected by a sequence of knows edges or helps edges

Datalog: example

(knows | helps)+

Find nodes connected by a sequence of knows edges or helps edges

$$
\begin{aligned}
R(x,y) &\leftarrow \text{knows}(x,y) \\
R(x,y) &\leftarrow \text{helps}(x,y) \\
R(x,y) &\leftarrow R(x,z), R(z,y) \\
\text{Ans(x,y)} &\leftarrow R(x,y)
\end{aligned}
$$

Datalog: example

(knows | helps)+

Find nodes connected by a sequence of knows edges or helps edges

$$
\begin{aligned}
R(x,y) &\leftarrow \text{knows}(x,y) \\
R(x,y) &\leftarrow \text{helps}(x,y) \\
R(x,y) &\leftarrow R(x,z), R(z,y) \\
\text{Ans(x,y)} &\leftarrow R(x,y)
\end{aligned}
$$

We use a special predicate Ans to collect answers (in this case pairs of nodes)

Datalog: example

(knows | helps)+

Find nodes connected by a sequence of knows edges or helps edges

$$
\begin{aligned}
R(x,y) &\leftarrow \text{knows}(x,y) \\
R(x,y) &\leftarrow \text{helps}(x,y) \\
R(x,y) &\leftarrow R(x,z), R(z,y) \\
\text{Ans(x,y)} &\leftarrow R(x,y)
\end{aligned}
$$

To evaluate over a graph:

apply all rules until we reach a fixed point.

Datalog: example

(knows | helps)+

Find nodes connected by a sequence of knows edges or helps edges

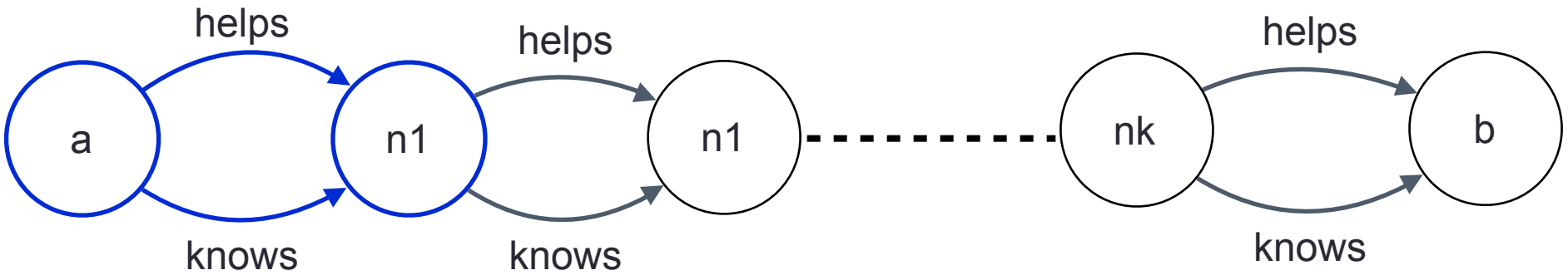
$$
\begin{aligned}
R(x,y) &\leftarrow \text{knows}(x,y) \\
R(x,y) &\leftarrow \text{helps}(x,y) \\
R(x,y) &\leftarrow R(x,z), R(z,y) \\
\text{Ans(x,y)} &\leftarrow R(x,y)
\end{aligned}
$$

To evaluate over a graph:

apply all rules until we reach a fixed point.

The evaluation is everything in Ans.

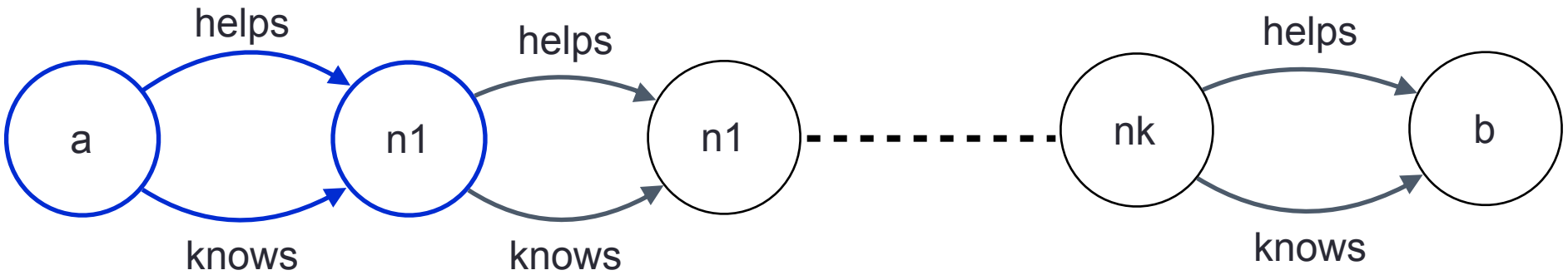
Datalog: example



Say u is a friend of v if
u knows v and u helps v

Find all nodes connected
by a chain of friends

Datalog: example



Say u is a friend of v if
u knows v and u helps v

Find all nodes connected
by a chain of friends

$$\text{Friend}(x, y) \leftarrow \text{knows}(x, y), \text{helps}(x, y)$$
$$\text{Friend}(x, y) \leftarrow \text{Friend}(x, z), \text{Friend}(z, y)$$
$$\text{Ans(x,y)} \leftarrow \text{Friend}(x, y)$$

Datalog: example

Say u is a friend of v if
u knows v and u helps v

$$\text{Friend}(x, y) \leftarrow \text{knows}(x, y), \text{helps}(x, y)$$
$$\text{Friend}(x, y) \leftarrow \text{Friend}(x, z), \text{Friend}(z, y)$$
$$\text{Ans(x,y)} \leftarrow \text{Friend}(x, y)$$

Datalog: example

Find all nodes connected by a chain of friends

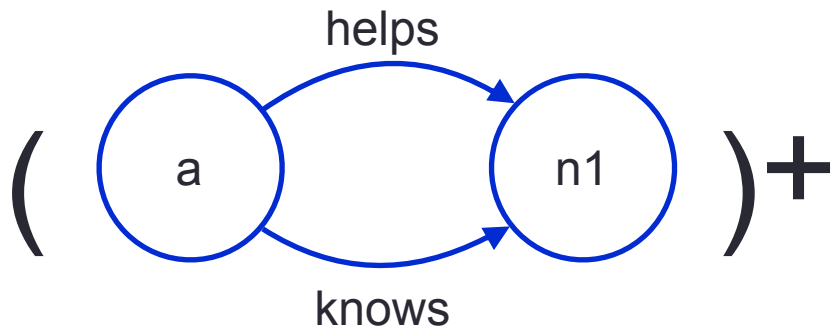
$$\begin{aligned}
\text{Friend}(x, y) &\leftarrow \text{knows}(x, y), \text{helps}(x, y) \\
\text{Friend}(x, y) &\leftarrow \text{Friend}(x, z), \text{Friend}(z, y) \\
\text{Ans(x,y)} &\leftarrow \text{Friend}(x, y)
\end{aligned}$$

Datalog: example

Find all nodes connected by a chain of friends

$$\begin{aligned} \text{Friend}(x,y) &\leftarrow \text{knows}(x,y), \text{helps}(x,y) \\ \text{Friend}(x,y) &\leftarrow \text{Friend}(x,z), \text{Friend}(z,y) \\ \text{Ans}(x,y) &\leftarrow \text{Friend}(x,y) \end{aligned}$$

Kleene star!



Expressive power

Datalog can express any navigational pattern (and much more)

Expressive power

Datalog can express any navigational pattern (and much more)

The problem is that Datalog is too expressive!
Many problems associated to Datalog are hard

Evaluation

$$\begin{aligned} \text{Friend}(x,y) &\leftarrow \text{knows}(x,y), \text{helps}(x,y) \\ \text{Friend}(x,y) &\leftarrow \text{Friend}(x,z), \text{Friend}(z,y) \\ \text{Ans(x,y)} &\leftarrow \text{Friend}(x,y) \end{aligned}$$

Does a pair (a,b) belong to Ans,
when computed over a graph G?

Evaluation

$$\begin{aligned} \text{Friend}(x,y) &\leftarrow \text{knows}(x,y), \text{helps}(x,y) \\ \text{Friend}(x,y) &\leftarrow \text{Friend}(x,z), \text{Friend}(z,y) \\ \text{Ans(x,y)} &\leftarrow \text{Friend}(x,y) \end{aligned}$$

Does a pair (a,b) belong to Ans,
when computed over a graph G?

EXPTIME-complete in general
PTIME-complete if the program is fixed

Evaluation

$$\begin{aligned} \text{Friend}(x,y) &\leftarrow \text{knows}(x,y), \text{helps}(x,y) \\ \text{Friend}(x,y) &\leftarrow \text{Friend}(x,z), \text{Friend}(z,y) \\ \text{Ans(x,y)} &\leftarrow \text{Friend}(x,y) \end{aligned}$$

Does a pair (a,b) belong to Ans,
when computed over a graph G?

EXPTIME-complete in general
PTIME-complete if the program is fixed

Other database problems are harder for datalog programs

Outline

Datalog

Rule-based languages

Containment of Queries

Regular Queries

Containment

A query Q is contained in a query Q' if

$Q(G) \subseteq Q'(G)$, for every graph database G

Containment

A query Q is contained in a query Q' if

$Q(G) \subseteq Q'(G)$, for every graph database G

Examples

$(aa)^+$ is contained in a^+

Containment

A query Q is contained in a query Q' if

$Q(G) \subseteq Q'(G)$, for every graph database G

Examples

$(aa)^+$ is contained in a^+

$Q(x) = \exists y\big(a^+(x,y) \wedge a^-(y,x)\big)$

is contained in $Q'(x) = \exists y\big(a^+(x,y)\big)$

Containment

A query Q is contained in a query Q' if

$Q(G) \subseteq Q'(G)$, for every graph database G

Examples

$$\begin{aligned} \text{follow}(x, y) &\leftarrow \text{knows}(x, y) \\ \text{follow}(x, y) &\leftarrow \text{popular}(y), \text{follow}(x, z) \end{aligned}$$

is contained in

$$\begin{aligned} \text{follow}(x, y) &\leftarrow \text{knows}(x, y) \\ \text{follow}(x, y) &\leftarrow \text{popular}(y), \text{knows}(x, z) \end{aligned}$$

Containment problem is the theoretical backbone of many other data-related tasks

Containment problem is the theoretical backbone of many other data-related tasks

Containment problem is the theoretical backbone of many other data-related tasks

Containment problem is the theoretical backbone of many other data-related tasks

- query optimisation

Containment problem is the theoretical backbone of many other data-related tasks

- query optimisation
- query answering using views or access patterns

Containment problem is the theoretical backbone of many other data-related tasks

- query optimisation
- query answering using views or access patterns
- data integration, ontology-based query answering

Complexity of Containment: Primitives

Input: Q, Q'
Question: Is Q contained in Q'?

Containment is: [Calvanese, De Giacomo, Lenzerini, Vardi 00'; R. 13]

PSPACE-hard for RPQs
in PSPACE for 2RPQs and NPQs

Complexity of Containment: Primitives

Input: Q, Q'
Question: Is Q contained in Q'?

Containment is: [Calvanese, De Giacomo, Lenzerini, Vardi 00'; R. 13]

PSPACE-complete for path queries

Complexity of Containment: Primitives

Input: Q, Q'
Question: Is Q contained in Q'?

Containment is: [Calvanese, De Giacomo, Lenzerini, Vardi 00; Barceló, Pérez, R. 13]

EXPSPACE-hard for CRPQs
in EXPSPACE for C2RPQs and CNPQs

Complexity of Containment: Primitives

Input: Q, Q'
Question: Is Q contained in Q'?

Containment is: [Calvanese, De Giacomo, Lenzerini, Vardi 00; Barceló, Pérez, R. 13]

EXPSPACE-complete for navigational patterns

Complexity of Containment: Primitives

Input: Q, Q'
Question: Is Q contained in Q'?

Containment is: [Chandra, Merlin 77; Shmueli 87]

NP-complete for CQs

Undecidable for general Datalog programs

Complexity of Containment: Primitives

Input: Q, Q'
Question: Is Q contained in Q'?

Containment is: [Chandra, Merlin 77; Shmueli 87]

NP-complete for CQs

Undecidable for general Datalog programs

(Undecidable for Relational Algebra expressions,
but Datalog has no negation!)

Complexity of Containment: Primitives

Input: Q, Q'
Question: Is Q contained in Q'?

Containment is: [Chaudhuri, Vardi 92]

2EXPSPACE-complete for a Datalog program
in a non-recursive Datalog program

Complexity of Containment: Primitives

Input: Q, Q'
Question: Is Q contained in Q'?

Containment is: [Calvanese, De Giacomo, Vardi 05]

2EXPSPACE-complete for a Datalog program
in a C2RPQ (or even unions of…)

New goal

Algebraically closed graph query language such that:

New goal

Algebraically closed graph query language such that:

New goal

Algebraically closed graph query language such that:

- can express navigational patterns (C2RPQs, CNPQs)

New goal

Algebraically closed graph query language such that:

- can express navigational patterns (C2RPQs, CNPQs)
- evaluation is “easy” (NP combined, NLogSpace data)

New goal

Algebraically closed graph query language such that:

- can express navigational patterns (C2RPQs, CNPQs)
- evaluation is “easy” (NP combined, NLogSpace data)
- containment is “easy” (decidable in an elemental class)

Idea

Datalog can express very intricate recursive queries

$$
\begin{aligned}
R(x,z) &\leftarrow P(x,y), Q(y,z) \\
P(x,z) &\leftarrow Q(x,y), R(y,z) \\
Q(x,z) &\leftarrow R(x,y), P(y,z)
\end{aligned}
$$

Idea

Datalog can express very intricate recursive queries

$$
\begin{aligned}
R(x,z) &\leftarrow P(x,y), Q(y,z) \\
P(x,z) &\leftarrow Q(x,y), R(y,z) \\
Q(x,z) &\leftarrow R(x,y), P(y,z)
\end{aligned}
$$

We don't need this.

We just need a way of applying kleene star to patterns

Outline

Datalog

Rule-based languages

Containment of Queries

Regular Queries

Regular Queries (by example)

(knows | helps)+

Find nodes connected by a sequence of knows edges or helps edges

Regular Queries (by example)

(knows | helps)+

Find nodes connected by a sequence of knows edges or helps edges

$$
\begin{aligned}
R(x, y) &\leftarrow \text{knows}(x, y) \\
R(x, y) &\leftarrow \text{helps}(x, y) \\
\text{Ans}(x, y) &\leftarrow R^{+}(x, y)
\end{aligned}
$$

Regular Queries (by example)

(knows | helps)+

Find nodes connected by a sequence of knows edges or helps edges

$$
\begin{aligned}
R(x, y) &\leftarrow \text{knows}(x, y) \\
R(x, y) &\leftarrow \text{helps}(x, y) \\
\text{Ans}(x, y) &\leftarrow R^{+}(x, y)
\end{aligned}
$$

To evaluate: same as datalog,
the + operator is transitive closure

Regular Queries (by example)

$$
\begin{aligned}
R(x, y) &\leftarrow \text{knows}(x, y) \\
R(x, y) &\leftarrow \text{helps}(x, y) \\
\text{Ans}(x, y) &\leftarrow R^{+}(x, y)
\end{aligned}
$$

To evaluate: same as datalog,
the + operator is transitive closure

Regular Queries (by example)

$$
\begin{aligned}
R(x,y) &\leftarrow \text{knows}(x,y) \\
R(x,y) &\leftarrow \text{helps}(x,y) \\
\text{Ans}(x,y) &\leftarrow R^{+}(x,y)
\end{aligned}
$$

To evaluate: same as datalog,
the + operator is transitive closure

$$
\begin{aligned}
R^{+} = \; & R \cup \\
& R \circ R \cup \\
& R \circ R \circ R \cup \\
& \vdots
\end{aligned}
$$

($R \circ R$ is the composition
of binary relation R)

Regular Queries (by example)

$$
\begin{aligned}
R(x,y) &\leftarrow \text{knows}(x,y) \\
R(x,y) &\leftarrow \text{helps}(x,y) \\
\text{Ans}(x,y) &\leftarrow R^+(x,y)
\end{aligned}
$$

To evaluate: same as datalog,
the + operator is transitive closure

$$
\begin{aligned}
R^+ = \; & R \cup \\
& R \circ R \cup \\
& R \circ R \circ R \cup \\
& \vdots
\end{aligned}
$$

$$
\begin{aligned}
R^+(x,y) &\leftarrow R(x,y) \\
R^+(x,z) &\leftarrow R^+(x,y), R^+(y,z)
\end{aligned}
$$

($R \circ R$ is the composition
of binary relation R)

Regular Queries

Extended datalog rules are of the form

$$S(z_1, z_2) \leftarrow R_1(x_1, y_1), \ldots, R_n(x_n, y_n)$$

Regular Queries

Extended datalog rules are of the form

$$S(z_1, z_2) \leftarrow R_1(x_1, y_1), \ldots, R_n(x_n, y_n)$$

$z_1, z_2 \in \{x_1, y_1, \ldots, x_n, y_n\}$

$R_1 \ldots, R_n$ are either graph labels,
predicates, or
expressions P^+ (for predicate or label P)

S is a fresh predicate

Regular Queries

Extended datalog rules are of the form

$$S(z_1, z_2) \leftarrow R_1(x_1, y_1), \ldots, R_n(x_n, y_n)$$

A Regular Query is a set of
extended datalog rules that is non-recursive

Non recursive

Ordering on predicates,
left hand side predicate is greater than everything on the right

$$
\begin{aligned}
R(x,y) &\leftarrow P(x,x), P(x,y) \\
R(x,y) &\leftarrow Q(x,y), P(y,x) \\
P^{+}(x,z) &\leftarrow R(x,z), Q(z,z) \\
R^{+}(x,y) &\leftarrow P^{+}(x,u), R(u,v), Q(v,y)
\end{aligned}
$$

Non recursive

Ordering on predicates,
left hand side predicate is greater than everything on the right

$$
\begin{aligned}
R(x,y) &\leftarrow P(x,x), P(x,y) \\
R(x,y) &\leftarrow Q(x,y), P(y,x) \\
P^+(x,z) &\leftarrow R(x,z), Q(z,z) \\
R^+(x,y) &\leftarrow P^+(x,u), R(u,v), Q(v,y)
\end{aligned}
$$

$$Q \prec P \prec R \prec P^+ \prec R^+$$

non-recursive

Non recursive

Ordering on predicates,
left hand side predicate is greater than everything on the right

$$
\begin{aligned}
R(x,y) &\leftarrow P(x,x), P(x,y) \\
R(x,y) &\leftarrow Q(x,y), P(y,x) \\
P^+(x,z) &\leftarrow R(x,z), Q(z,z) \\
R^+(x,y) &\leftarrow P^+(x,u), R(u,v), Q(v,y)
\end{aligned}
$$

$$Q \prec P \prec R \prec P^+ \prec R^+$$

non-recursive

recursive

$$
\begin{aligned}
R(x,y) &\leftarrow P(x,y), Q(x,y) \\
P(x,y) &\leftarrow R(x,y), Q(y,x)
\end{aligned}
$$

Regular Queries (RQ)

Extended datalog rules are of the form

$$S(z_1, z_2) \leftarrow R_1(x_1, y_1), \ldots, R_n(x_n, y_n)$$

A regular query is a set of extended datalog rules that is non-recursive

Regular Queries (RQ)

Extended datalog rules are of the form

$$S(z_1, z_2) \leftarrow R_1(x_1, y_1), \ldots, R_n(x_n, y_n)$$

A regular query is a set of extended datalog rules that is non-recursive

Basically, the only recursion we allow is the + (can be simulated using recursive datalog):

Regular Queries (RQ)

Extended datalog rules are of the form

$$S(z_1, z_2) \leftarrow R_1(x_1, y_1), \ldots, R_n(x_n, y_n)$$

A regular query is a set of extended datalog rules that is non-recursive

Basically, the only recursion we allow is the + (can be simulated using recursive datalog):

$$\begin{aligned} R^+(x, y) &\leftarrow R(x, y) \\ R^+(x, z) &\leftarrow R^+(x, y), R^+(y, z) \end{aligned}$$

Regular Queries $\subsetneq$ Datalog

But, regular queries can express all navigational patterns

Regular Queries $\subsetneq$ Datalog

But, regular queries can express all navigational patterns

Path queries:

Regular Queries $\subsetneq$ Datalog

But, regular queries can express all navigational patterns

(knows | helps)+

Path queries:

$$
\begin{aligned}
R(x,y) &\leftarrow \text{knows}(x,y) \\
R(x,y) &\leftarrow \text{helps}(x,y) \\
\text{Ans}(x,y) &\leftarrow R^{+}(x,y)
\end{aligned}
$$

Regular Queries $\subsetneq$ Datalog

But, regular queries can express all navigational patterns

(knows | helps)+

Path queries:

$$
\begin{aligned}
R(x,y) &\leftarrow \text{knows}(x,y) \\
R(x,y) &\leftarrow \text{helps}(x,y) \\
\text{Ans}(x,y) &\leftarrow R^{+}(x,y)
\end{aligned}
$$

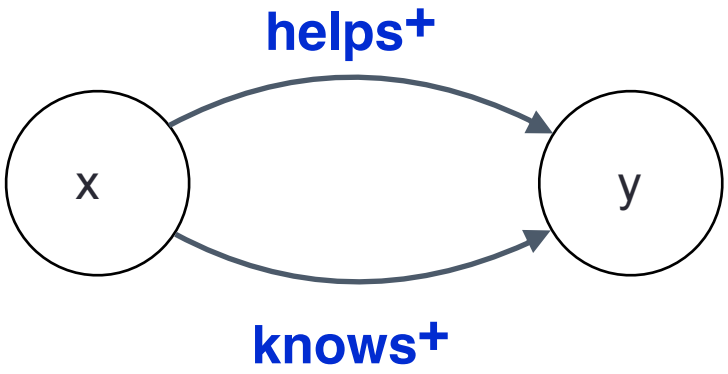
(knows knows [helps])+

$$
\begin{aligned}
R(x,y) &\leftarrow \text{knows}(x,u), \text{knows}(u,y), \text{helps}(y,z) \\
\text{Ans}(x,y) &\leftarrow R^{+}(x,y)
\end{aligned}
$$

Regular Queries $\subsetneq$ Datalog

But, regular queries can express all navigational patterns

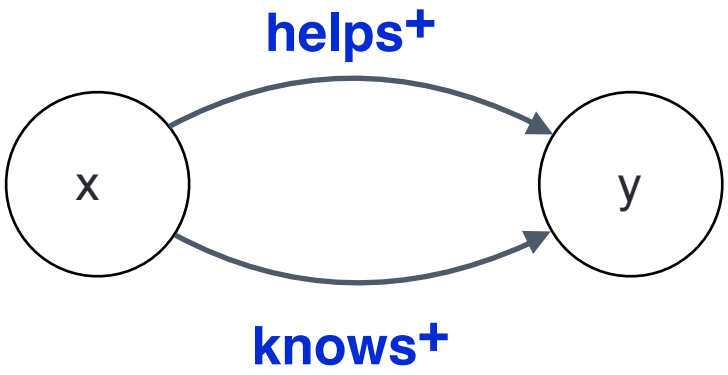
$$Q(x, y) = (x, \text{helps}^+, y) \wedge (x, \text{knows}^+, y)$$



Regular Queries $\subsetneq$ Datalog

But, regular queries can express all navigational patterns

$$Q(x, y) = (x, \text{helps}^+, y) \wedge (x, \text{knows}^+, y)$$



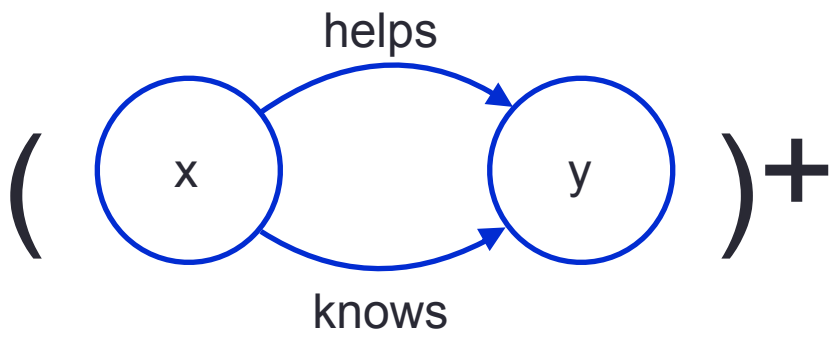
$$\begin{aligned} H(u, v) &\leftarrow \text{helps}(u, v) \\ K(u, v) &\leftarrow \text{knows}(u, v) \\ \text{Ans}(x, y) &\leftarrow H^+(x, y), K^+(x, y) \end{aligned}$$

Regular queries can express much more!

Regular queries can express much more!

Say u is a friend of v if
u knows v and u helps v

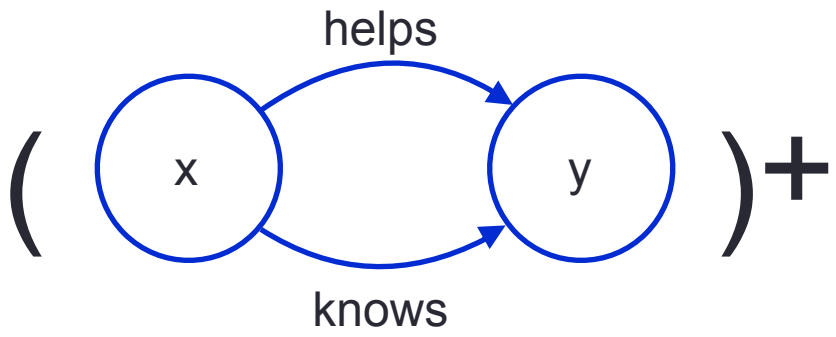
Find all nodes connected
by a chain of friends



Regular queries can express much more!

Say u is a friend of v if
u knows v and u helps v

Find all nodes connected
by a chain of friends



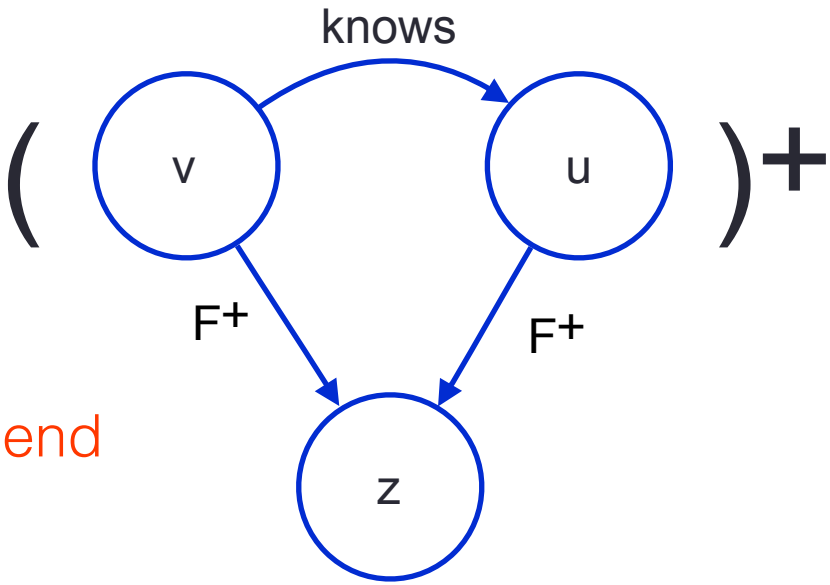
$$F(u,v) \leftarrow \text{helps}(u,v), \text{knows}(u,v)$$
$$\text{Ans}(x,y) \leftarrow F^{+}(x,y)$$

Regular queries can express much more!

Regular queries can express much more!

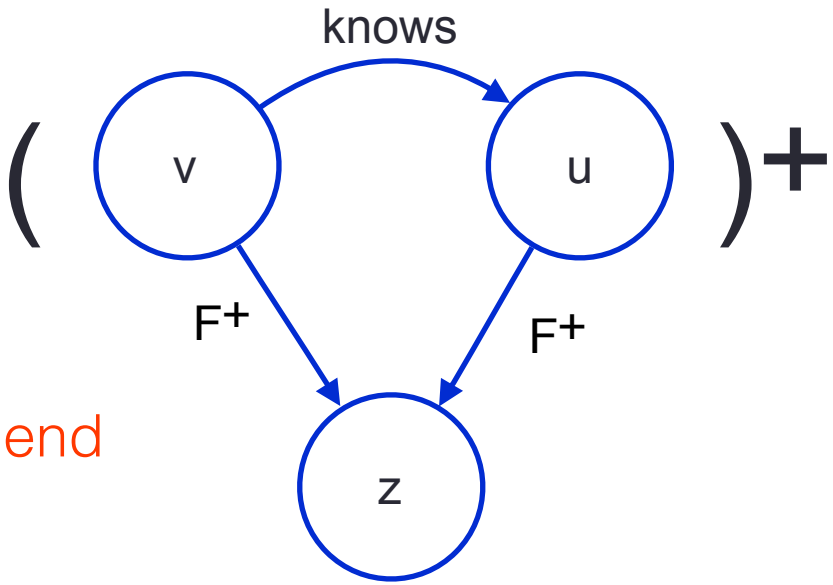
Find nodes connected by a chain of acquaintances:

u is an acquaintance of v if
- v knows u
- v and u have an indirect common friend



Regular queries can express much more!

Find nodes connected by a chain of acquaintances:



u is an acquaintance of v if
- v knows u
- v and u have an indirect common friend

$$F(u,v) \leftarrow \text{helps}(u,v), \text{knows}(u,v)$$
$$A(u,v) \leftarrow \text{knows}(v,u), F^{+}(v,z), F^{+}(u,z)$$
$$\text{Ans}(x,y) \leftarrow A^{+}(x,y)$$

Evaluation problem (Regular Queries):

Given graph G, nodes u,v from G, RQ Q.
Is (u,v) in the evaluation of Q over G?

Evaluation of Regular Queries is NP-complete

(already NP-hard because of CQs)

Evaluation in NP (Regular Queries):

Evaluation in NP (Regular Queries):

1.- Maximum number of tuples in a predicate
is polynomial, in $O(G^2)$
(just binary predicates)

Evaluation in NP (Regular Queries):

1.- Maximum number of tuples in a predicate
is polynomial, in $O(G^2)$
(just binary predicates)

2.- Do this for every predicate $O(G^2 \cdot |Q|)$

Evaluation in NP (Regular Queries):

1.- Maximum number of tuples in a predicate
is polynomial, in $O(G^2)$
(just binary predicates)

2.- Do this for every predicate $O(G^2 \cdot |Q|)$

3.- Guess a polynomial witness for every tuple
in every predicate that needs to be computed

Evaluation problem (Regular Queries):

Given graph G, nodes u,v from G, RQ Q.
Is (u,v) in the evaluation of Q over G?

Evaluation of Regular queries is

NLogSpace-complete (data complexity)
already NLogSpace-hard because of RPQs

Proof by merging
- CQ evaluation in NLogSpace
- Evaluating transitive closures is in NLogSpace

Containment problem (Regular Queries):

Given RQs Q and Q'.
Is Q' contained in Q?

Containment problem (Regular Queries):

Given RQs Q and Q'.
Is Q' contained in Q?

Containment is [R., Romero, Vardi 15]

2EXPSPACE-complete for Regular Queries

Containment problem (Regular Queries):

Given RQs Q and Q'.
Is Q' contained in Q?

Containment is [R., Romero, Vardi 15]

2EXPSPACE-complete for Regular Queries

Can we find a language with the same bounds as navigational queries?

Flat nested UC2RPQs

Collection of rules of the form

$$S(z_1, z_2) \leftarrow R_1(x_1, y_1), \ldots, R_n(x_n, y_n)$$

Flat nested UC2RPQs

Collection of rules of the form

$$S(z_1, z_2) \leftarrow R_1(x_1, y_1), \ldots, R_n(x_n, y_n)$$

$z_1, z_2 \in \{x_1, y_1, \ldots, x_n, y_n\}$

$R_1 \ldots, R_n$ are either 2RPQs (over labels and their inverses), or

expressions P^+ (for predicate P)

S is a fresh predicate

Flat nested UC2RPQs

Collection of rules of the form

$$S(z_1, z_2) \leftarrow R_1(x_1, y_1), \ldots, R_n(x_n, y_n)$$

$z_1, z_2 \in \{x_1, y_1, \ldots, x_n, y_n\}$

$R_1 \ldots, R_n$ are either 2RPQs (over labels and their inverses), or

expressions P^+ (for predicate P)

S is a fresh predicate

Each new predicate P appears only once on the left hand side of any rule

Flat nested UC2RPQs

Collection of rules of the form

$$S(z_1, z_2) \leftarrow R_1(x_1, y_1), \ldots, R_n(x_n, y_n)$$

$z_1, z_2 \in \{x_1, y_1, \ldots, x_n, y_n\}$

$R_1 \ldots, R_n$ are either 2RPQs (over labels and their inverses), or

expressions P^+ (for predicate P)

S is a fresh predicate

Every time we define a new predicate P we can use it only once, only in the form of a transitive closure P^+

Containment (Flat nested UC2RPQs):

Some results: [R., Romero, Vardi 15]

Flat nested UC2RPQs are exponentially more succinct than regular queries.

Containment of flat nested UC2RPQs is EXPSPACE-complete
(hardness from navigational patterns).

Recap

Recap

Recap

Datalog programs are good candidates for graph queries,
but maybe more expressive than what is needed

Recap

Datalog programs are good candidates for graph queries,
but maybe more expressive than what is needed

Regular Queries:

Non-recursive datalog + transitive closure of predicates

Recap

Datalog programs are good candidates for graph queries,
but maybe more expressive than what is needed

Regular Queries:

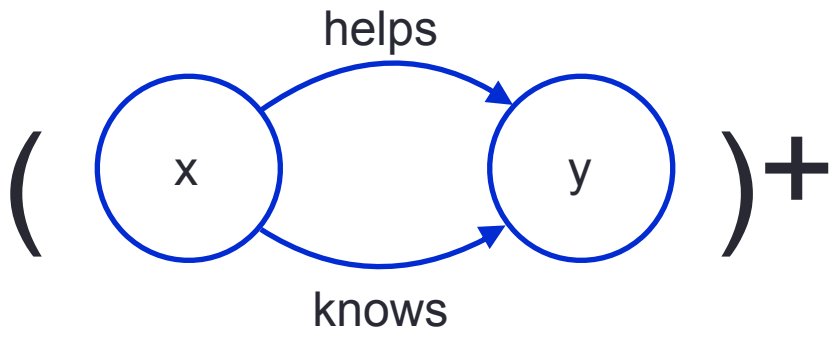
Non-recursive datalog + transitive closure of predicates

- Expresses navigational graph patterns
- Algebraically closed
- Complexity of evaluation similar to patterns
- Complexity of containment elemental,
  same as containment of datalog in non-rec datalog

Regular Queries (example)

Say u is a friend of v if
u knows v and u helps v

Find all nodes connected
by a chain of friends



$$F(u, v) \leftarrow \text{helps}(u, v), \text{knows}(u, v)$$
$$\text{Ans}(x, y) \leftarrow F^{+}(x, y)$$

Expressive power

Datalog

adds recursive rules

Regular Queries

Flat nUC2RPQs

adds transitive closure

unions of navigational patterns

adds path queries

(binary) non recursive Datalog

Arbitrary Arity?

$$F(u, v) \leftarrow \text{helps}(u, v), \text{knows}(u, v)$$
$$\text{Ans}(x, y) \leftarrow F^{+}(x, y)$$

Arbitrary Arity?

$$F(u, v) \leftarrow \text{helps}(u, v), \text{knows}(u, v)$$
$$\text{Ans}(x, y) \leftarrow F^{+}(x, y)$$

Regular queries use only binary predicates.
Can we use predicates of greater arity?

Arbitrary Arity?

$$F(u, v) \leftarrow \text{helps}(u, v), \text{knows}(u, v)$$
$$\text{Ans}(x, y) \leftarrow F^{+}(x, y)$$

Regular queries use only binary predicates.
Can we use predicates of greater arity?

For the transitive closure predicates P^{+} :

Not easy (not even clear how to define it).

Arbitrary Arity?

$$F(u, v) \leftarrow \text{helps}(u, v), \text{knows}(u, v)$$
$$\text{Ans}(x, y) \leftarrow F^{+}(x, y)$$

Regular queries use only binary predicates.
Can we use predicates of greater arity?

For other predicates: Yes! [R., Romero, Vardi 16]

- Evaluation becomes PSPACE-complete
  (same as non-recursive Datalog)
- Containment remains 2EXPSPACE-complete

Other approaches

Regular Queries restrict recursion to the P^+ operator.
But there are other alternatives!

Lifting Monadic Datalog

Monadic Datalog: Only Unary predicates in left side of rules

Lifting Monadic Datalog

Monadic Datalog: Only Unary predicates in left side of rules

Monadically defined queries [Bourhis, Krötzsch, Rudolph]:

Add two special constants λ_1, λ_2

$$
\begin{aligned}
U(y) &\leftarrow P(\lambda_1, y) \\
U(z) &\leftarrow P(y, z), U(y) \\
\text{hit} &\leftarrow U(\lambda_2)
\end{aligned}
$$

A pair (a, b) is in the answer if this program hits when $a = \lambda_1$ and $b = \lambda_2$

Lifting Monadic Datalog

Monadic Datalog: Only Unary predicates in left side of rules

Monadically defined queries [Bourhis, Krötzsch, Rudolph]:

Add two special constants λ_1, λ_2

$$
\begin{aligned}
U(y) &\leftarrow P(\lambda_1, y) \\
U(z) &\leftarrow P(y, z), U(y) \\
\text{hit} &\leftarrow U(\lambda_2)
\end{aligned}
$$

A pair (a, b) is in the answer if this program hits when $a = \lambda_1$ and $b = \lambda_2$

Transitive closure of P!

Lifting Monadic Datalog

Monadic Datalog: Only Unary predicates in left side of rules

Monadically defined queries [Bourhis, Krötzsch, Rudolph]:

Can find algebraically closed fragment
with good evaluation and containment bounds

More recursion but guarding programs

TriQL [Arenas, Gottlob, Pieris]:

Allow more recursion,
Constraint rules to have guards
(problematic variables in rules need to appear in certain ways)

More recursion but guarding programs

TriQL [Arenas, Gottlob, Pieris]:

Allow more recursion,
Constraint rules to have guards
(problematic variables in rules need to appear in certain ways)

Can generate much more expressive languages,
some of them algebraically closed