Eliminating recursion from monadic datalog on trees

Filip Mazowiecki

University of Warwick

(joint work with Filip Murlak, Joanna Ochremiak and Adam Witkowski)

Seminar 2016
Oxford

Introduction

(datalog: examples, problems, restrictions)

Unions of conjunctive queries (UCQ)

No negation

Unions of conjunctive queries (UCQ)

No negation

Conjunctive queries (CQ):

Unions of conjunctive queries (UCQ)

No negation

Conjunctive queries (CQ):

$$\exists x_1 \dots \exists x_k \varphi,$$

φ is a conjunction of atomic formulas

Unions of conjunctive queries (UCQ)

No negation

Conjunctive queries (CQ):

$$\exists x_1 \ldots \exists x_k \varphi,$$

φ is a conjunction of atomic formulas

e.g., there exists a triangle:

$$\exists x_1, x_2, x_3 \ E(x_1, x_2) \land E(x_2, x_3) \land E(x_3, x_1)$$

Unions of conjunctive queries (UCQ)

No negation

Conjunctive queries (CQ):

$$\exists x_1 \ldots \exists x_k \varphi,$$

φ is a conjunction of atomic formulas

e.g., there exists a triangle:

$$\exists x_1, x_2, x_3 \; E(x_1, x_2) \wedge E(x_2, x_3) \wedge E(x_3, x_1)$$

UCQ are finite disjunctions of CQ

Unions of conjunctive queries (UCQ)

No negation

Conjunctive queries (CQ):

$$\exists x_1 \ldots \exists x_k \varphi,$$

φ is a conjunction of atomic formulas

e.g., there exists a triangle:

$$\exists x_1, x_2, x_3 \; E(x_1, x_2) \wedge E(x_2, x_3) \wedge E(x_3, x_1)$$

UCQ are finite disjunctions of CQ

Satisfiability is not interesting (always YES)

Unions of conjunctive queries (UCQ)

No negation

Conjunctive queries (CQ):

$$\exists x_1 \ldots \exists x_k \varphi,$$

φ is a conjunction of atomic formulas

e.g., there exists a triangle:

$$\exists x_1, x_2, x_3 \; E(x_1, x_2) \land E(x_2, x_3) \land E(x_3, x_1)$$

UCQ are finite disjunctions of CQ

Satisfiability is not interesting (always YES)

Theorem (Chandra and Merlin)

Containment for CQ and UCQ is NP-complete.

Datalog on finite structures

Datalog on finite structures

A datalog program is a set of **rules**

Datalog on finite structures

A datalog program is a set of **rules**

$$buys(X, Y) \leftarrow likes(X, Y)$$
$$buys(X, Y) \leftarrow trendy(X), buys(Z, Y)$$

Datalog on finite structures

A datalog program is a set of **rules**

$$buys(X,Y) \leftarrow likes(X,Y)$$
$$\underbrace{buys(X,Y)}_{\text{head}} \leftarrow trendy(X), buys(Z,Y)$$

Datalog on finite structures

A datalog program is a set of **rules**

$$buys(X,Y) \leftarrow likes(X,Y)$$
$$\underbrace{buys(X,Y)}_{\text{head}} \leftarrow \underbrace{trendy(X), buys(Z,Y)}_{\text{body}}$$

Datalog on finite structures

A datalog program is a set of **rules**

$$\begin{aligned} buys(X,Y) &\leftarrow likes(X,Y) \\ \underbrace{buys(X,Y)}_{\text{head}} &\leftarrow \underbrace{trendy(X), buys(Z,Y)}_{\text{body}} \end{aligned}$$

- **extensional** predicates ($likes, trendy$)

Datalog on finite structures

A datalog program is a set of **rules**

$$\begin{aligned} buys(X,Y) &\leftarrow likes(X,Y) \\ \underbrace{buys(X,Y)}_{\text{head}} &\leftarrow \underbrace{trendy(X), buys(Z,Y)}_{\text{body}} \end{aligned}$$

- **extensional** predicates ($likes, trendy$)
- **intensional** predicates ($buys$)

Datalog on finite structures

A datalog program is a set of **rules**

$$\begin{aligned} buys(X,Y) &\leftarrow likes(X,Y) \\ \underbrace{buys(X,Y)}_{\text{head}} &\leftarrow \underbrace{trendy(X), buys(Z,Y)}_{\text{body}} \end{aligned}$$

- **extensional** predicates ($likes, trendy$)
- **intensional** predicates ($buys$)
- one designated **goal** predicate ($buys$)

Datalog evaluation

UCQ with fixpoint

Datalog evaluation

UCQ with fixpoint

Example:

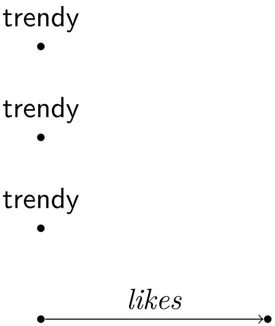
$$\mathcal{P}_1: \begin{aligned} & buys(X,Y) \leftarrow likes(X,Y) \\ & buys(X,Y) \leftarrow trendy(X), buys(Z,Y) \end{aligned}$$

Datalog evaluation

UCQ with fixpoint

Example:

$$\mathcal{P}_1: buys(X,Y) \leftarrow likes(X,Y)$$
$$buys(X,Y) \leftarrow trendy(X), buys(Z,Y)$$

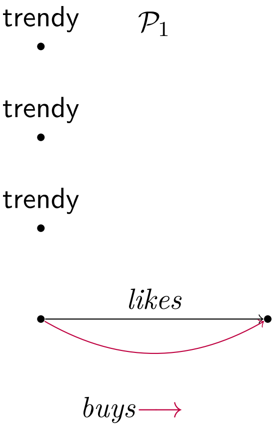


Datalog evaluation

UCQ with fixpoint

Example:

$$\mathcal{P}_1: \; buys(X,Y) \leftarrow likes(X,Y)$$
$$buys(X,Y) \leftarrow trendy(X), buys(Z,Y)$$

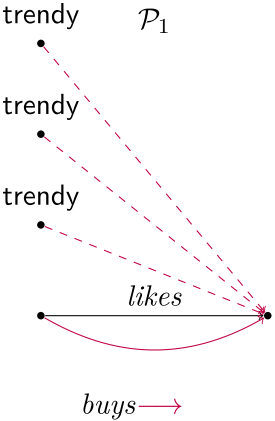


Datalog evaluation

UCQ with fixpoint

Example:

$$\mathcal{P}_1: \; buys(X,Y) \leftarrow likes(X,Y)$$
$$buys(X,Y) \leftarrow trendy(X), buys(Z,Y)$$



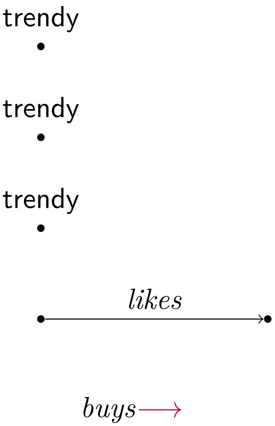
Datalog evaluation

UCQ with fixpoint

Example:

$$\mathcal{P}_1: buys(X,Y) \leftarrow likes(X,Y)$$
$$buys(X,Y) \leftarrow trendy(X), buys(Z,Y)$$

$$\mathcal{P}'_1: buys(X,Y) \leftarrow likes(X,Y)$$
$$buys(X,Y) \leftarrow trendy(X), likes(Z,Y)$$



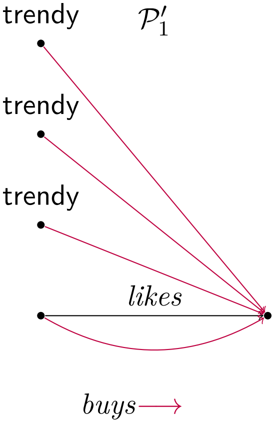
Datalog evaluation

UCQ with fixpoint

Example:

$$\mathcal{P}_1 : buys(X,Y) \leftarrow likes(X,Y)$$
$$buys(X,Y) \leftarrow trendy(X), buys(Z,Y)$$

$$\mathcal{P}_1' : buys(X,Y) \leftarrow likes(X,Y)$$
$$buys(X,Y) \leftarrow trendy(X), likes(Z,Y)$$



Datalog evaluation

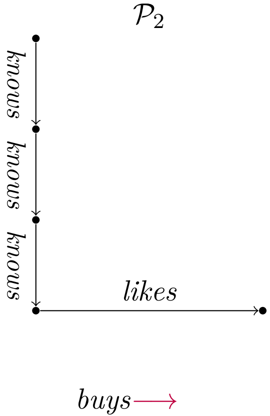
UCQ with fixpoint

Example:

$$\mathcal{P}_1\colon\ buys(X,Y) \leftarrow likes(X,Y)$$
$$buys(X,Y) \leftarrow trendy(X), buys(Z,Y)$$

$$\mathcal{P}_1'\colon\ buys(X,Y) \leftarrow likes(X,Y)$$
$$buys(X,Y) \leftarrow trendy(X), likes(Z,Y)$$

$$\mathcal{P}_2\colon\ buys(X,Y) \leftarrow likes(X,Y)$$
$$buys(X,Y) \leftarrow knows(X,Z), buys(Z,Y)$$



Datalog evaluation

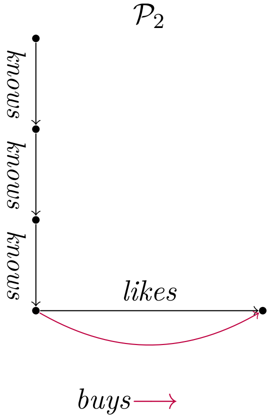
UCQ with fixpoint

Example:

$$\mathcal{P}_1\colon \mathit{buys}(X,Y) \leftarrow \mathit{likes}(X,Y)$$
$$\mathit{buys}(X,Y) \leftarrow \mathit{trendy}(X), \mathit{buys}(Z,Y)$$

$$\mathcal{P}_1'\colon \mathit{buys}(X,Y) \leftarrow \mathit{likes}(X,Y)$$
$$\mathit{buys}(X,Y) \leftarrow \mathit{trendy}(X), \mathit{likes}(Z,Y)$$

$$\mathcal{P}_2\colon \mathit{buys}(X,Y) \leftarrow \mathit{likes}(X,Y)$$
$$\mathit{buys}(X,Y) \leftarrow \mathit{knows}(X,Z), \mathit{buys}(Z,Y)$$



Datalog evaluation

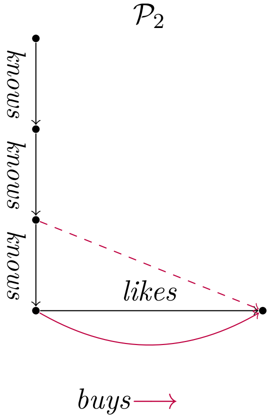
UCQ with fixpoint

Example:

$\mathcal{P}_1\colon$ $buys(X,Y) \leftarrow likes(X,Y)$
$buys(X,Y) \leftarrow trendy(X), buys(Z,Y)$

$\mathcal{P}_1'\colon$ $buys(X,Y) \leftarrow likes(X,Y)$
$buys(X,Y) \leftarrow trendy(X), likes(Z,Y)$

$\mathcal{P}_2\colon$ $buys(X,Y) \leftarrow likes(X,Y)$
$buys(X,Y) \leftarrow knows(X,Z), buys(Z,Y)$



Datalog evaluation

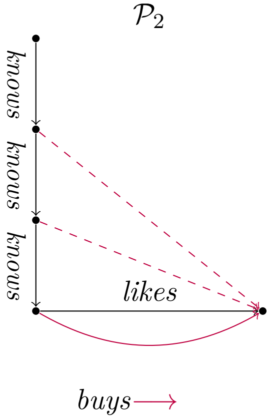
UCQ with fixpoint

Example:

$\mathcal{P}_1$: $buys(X,Y) \leftarrow likes(X,Y)$
$buys(X,Y) \leftarrow trendy(X), buys(Z,Y)$

$\mathcal{P}_1'$: $buys(X,Y) \leftarrow likes(X,Y)$
$buys(X,Y) \leftarrow trendy(X), likes(Z,Y)$

$\mathcal{P}_2$: $buys(X,Y) \leftarrow likes(X,Y)$
$buys(X,Y) \leftarrow knows(X,Z), buys(Z,Y)$



Datalog evaluation

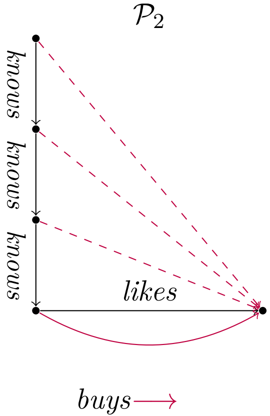
UCQ with fixpoint

Example:

$$\mathcal{P}_1\colon\ buys(X,Y) \leftarrow likes(X,Y)$$
$$buys(X,Y) \leftarrow trendy(X), buys(Z,Y)$$

$$\mathcal{P}_1'\colon\ buys(X,Y) \leftarrow likes(X,Y)$$
$$buys(X,Y) \leftarrow trendy(X), likes(Z,Y)$$

$$\mathcal{P}_2\colon\ buys(X,Y) \leftarrow likes(X,Y)$$
$$buys(X,Y) \leftarrow knows(X,Z), buys(Z,Y)$$



Datalog evaluation

UCQ with fixpoint

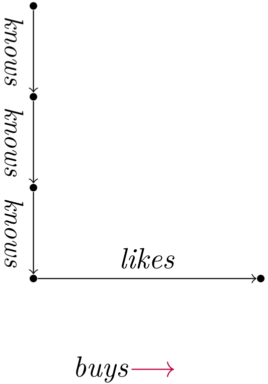
Example:

$\mathcal{P}_1$: $buys(X,Y) \leftarrow likes(X,Y)$
$buys(X,Y) \leftarrow trendy(X), buys(Z,Y)$

$\mathcal{P}'_1$: $buys(X,Y) \leftarrow likes(X,Y)$
$buys(X,Y) \leftarrow trendy(X), likes(Z,Y)$

$\mathcal{P}_2$: $buys(X,Y) \leftarrow likes(X,Y)$
$buys(X,Y) \leftarrow knows(X,Z), buys(Z,Y)$

$\mathcal{P}'_2$: $buys(X,Y) \leftarrow likes(X,Y)$
$buys(X,Y) \leftarrow knows(X,Z), likes(Z,Y)$



Datalog evaluation

UCQ with fixpoint

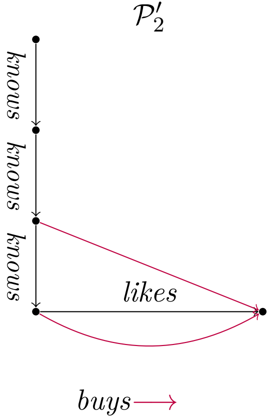
Example:

$$\mathcal{P}_1\colon\ buys(X,Y) \leftarrow likes(X,Y)$$
$$buys(X,Y) \leftarrow trendy(X), buys(Z,Y)$$

$$\mathcal{P}_1'\colon\ buys(X,Y) \leftarrow likes(X,Y)$$
$$buys(X,Y) \leftarrow trendy(X), likes(Z,Y)$$

$$\mathcal{P}_2\colon\ buys(X,Y) \leftarrow likes(X,Y)$$
$$buys(X,Y) \leftarrow knows(X,Z), buys(Z,Y)$$

$$\mathcal{P}_2'\colon\ buys(X,Y) \leftarrow likes(X,Y)$$
$$buys(X,Y) \leftarrow knows(X,Z), likes(Z,Y)$$



Datalog evaluation

UCQ with fixpoint

Example:

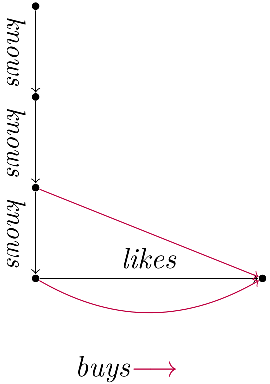
$$\mathcal{P}_1: \; buys(X,Y) \leftarrow likes(X,Y)$$
$$buys(X,Y) \leftarrow trendy(X), buys(Z,Y)$$

$$\mathcal{P}_1': \; buys(X,Y) \leftarrow likes(X,Y)$$
$$buys(X,Y) \leftarrow trendy(X), likes(Z,Y)$$

$$\mathcal{P}_2: \; buys(X,Y) \leftarrow likes(X,Y)$$
$$buys(X,Y) \leftarrow knows(X,Z), buys(Z,Y)$$

$$\mathcal{P}_2': \; buys(X,Y) \leftarrow likes(X,Y)$$
$$buys(X,Y) \leftarrow knows(X,Z), likes(Z,Y)$$

$\mathcal{P}_2'$



$\mathcal{P}_1 \equiv \mathcal{P}_1'$

Datalog evaluation

UCQ with fixpoint

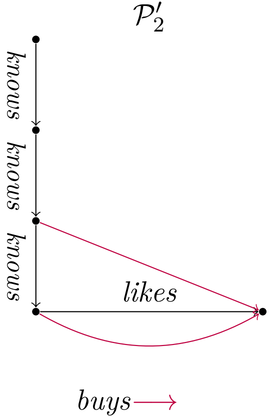
Example:

$\mathcal{P}_1$: $buys(X,Y) \leftarrow likes(X,Y)$
$buys(X,Y) \leftarrow trendy(X), buys(Z,Y)$

$\mathcal{P}_1'$: $buys(X,Y) \leftarrow likes(X,Y)$
$buys(X,Y) \leftarrow trendy(X), likes(Z,Y)$

$\mathcal{P}_2$: $buys(X,Y) \leftarrow likes(X,Y)$
$buys(X,Y) \leftarrow knows(X,Z), buys(Z,Y)$

$\mathcal{P}_2'$: $buys(X,Y) \leftarrow likes(X,Y)$
$buys(X,Y) \leftarrow knows(X,Z), likes(Z,Y)$



$\mathcal{P}_1 \equiv \mathcal{P}_1'$

$\mathcal{P}_2 \not\equiv \mathcal{P}_2'$

Datalog decision problems

Datalog decision problems

Equivalence

Datalog decision problems

Equivalence

1. Are two given programs equivalent?

Datalog decision problems

Equivalence

1. Are two given programs equivalent?
2. Is a program equivalent to **a given** nonrecursive program (UCQ)?

Datalog decision problems

Equivalence

1. Are two given programs equivalent?
2. Is a program equivalent to **a given** nonrecursive program (UCQ)?

Boundedness

Datalog decision problems

Equivalence

1. Are two given programs equivalent?
2. Is a program equivalent to **a given** nonrecursive program (UCQ)?

Boundedness

3. Does the program really use recursion?

Datalog decision problems

Equivalence

1. Are two given programs equivalent?
2. Is a program equivalent to **a given** nonrecursive program (UCQ)?

Boundedness

3. Does the program really use recursion?

3'. Is a program equivalent to **any** nonrecursive program (UCQ)?

Datalog decision problems

Equivalence

1. Are two given programs equivalent?
2. Is a program equivalent to **a given** nonrecursive program (UCQ)?

Boundedness

3. Does the program really use recursion?

3'. Is a program equivalent to **any** nonrecursive program (UCQ)?

(1) is undecidable [Shmueli]

Datalog decision problems

Equivalence

1. Are two given programs equivalent?
2. Is a program equivalent to **a given** nonrecursive program (UCQ)?

Boundedness

3. Does the program really use recursion?

3'. Is a program equivalent to **any** nonrecursive program (UCQ)?

(1) is undecidable [Shmueli]

(3), (3') are undecidable [Gaifman et al.]

Datalog decision problems

Equivalence

1. Are two given programs equivalent?
2. Is a program equivalent to **a given** nonrecursive program (UCQ)?

Boundedness

3. Does the program really use recursion?

3'. Is a program equivalent to **any** nonrecursive program (UCQ)?

(1) is undecidable [Shmueli]

(3), (3') are undecidable [Gaifman et al.]

Theorem (Chaudhuri,Vardi)

(2) is 2EXPTIME-complete.

Datalog restrictions

- Monadic datalog

Datalog restrictions

- Monadic datalog

e.g. $buys(X, Y)$ is binary, but $P(X)$ is monadic

Datalog restrictions

- Monadic datalog

e.g. $buys(X, Y)$ is binary, but $P(X)$ is monadic

Theorem (Cosmadakis et al.; Benedikt et al.)
Containment for monadic datalog is 2EXPTIME-complete. Boundedness is in 3EXPTIME.

Datalog restrictions

- Monadic datalog

e.g. $buys(X, Y)$ is binary, but $P(X)$ is monadic

Theorem (Cosmadakis et al.; Benedikt et al.)
Containment for monadic datalog is 2EXPTIME-complete. Boundedness is in 3EXPTIME.

- Linear datalog

e.g. $P(X) \leftarrow E(X, Y), P(Y)$ is linear

Datalog restrictions

- Monadic datalog

e.g. $buys(X, Y)$ is binary, but $P(X)$ is monadic

Theorem (Cosmadakis et al.; Benedikt et al.)
Containment for monadic datalog is 2EXPTIME-complete. Boundedness is in 3EXPTIME.

- Linear datalog

e.g. $P(X) \leftarrow E(X, Y), P(Y)$ is linear

but $P(X) \leftarrow E(X, Y), P(Y), E(X, Z), P(Z)$ is not linear

Datalog restrictions

- Monadic datalog

e.g. $buys(X, Y)$ is binary, but $P(X)$ is monadic

Theorem (Cosmadakis et al.; Benedikt et al.)

Containment for monadic datalog is 2EXPTIME-complete. Boundedness is in 3EXPTIME.

- Linear datalog

e.g. $P(X) \leftarrow E(X, Y), P(Y)$ is linear

but $P(X) \leftarrow E(X, Y), P(Y), E(X, Z), P(Z)$ is not linear

A rule of thumb

Linearity does not change decidability, but lowers complexities.

One more restriction

- Connected datalog

One more restriction

- Connected datalog

For every rule r there is a graph G_r

One more restriction

- Connected datalog

For every rule r there is a graph G_r

$$r: \quad buys(X,Y) \leftarrow knows(X,Z), buys(Z,Y)$$

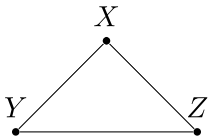
One more restriction

- Connected datalog

For every rule r there is a graph G_r

$r: \quad buys(X,Y) \leftarrow knows(X,Z), buys(Z,Y)$

G_r



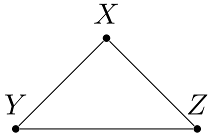
One more restriction

- Connected datalog

For every rule r there is a graph G_r

$r: \quad buys(X,Y) \leftarrow knows(X,Z), buys(Z,Y)$

G_r



A program is **connected** if all G_r are connected

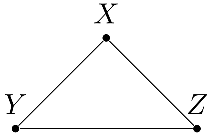
One more restriction

- Connected datalog

For every rule r there is a graph G_r

$r: \quad buys(X, Y) \leftarrow knows(X, Z), buys(Z, Y)$

G_r



A program is **connected** if all G_r are connected

A program is **downward** iff it is monadic

$+$ all G_r are trees with X in the root

Datalog on trees

(basic results on different models)

Finite trees

Finite trees

Extensional relations are of given interpretation

Finite trees

Extensional relations are of given interpretation

- Σ set of labels (finite or infinite)

Finite trees

Extensional relations are of given interpretation

- Σ set of labels (finite or infinite)
- navigational relations: $\downarrow, \downarrow_+$

Finite trees

Extensional relations are of given interpretation

- Σ set of labels (finite or infinite)
- navigational relations: $\downarrow, \downarrow_+$ (in this talk no siblings)

Finite trees

Extensional relations are of given interpretation

- Σ set of labels (finite or infinite)
- navigational relations: $\downarrow, \downarrow_+$ (in this talk no siblings)
- label equality $X \sim Y$

Finite trees

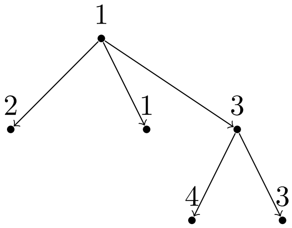
Extensional relations are of given interpretation

- Σ set of labels (finite or infinite)
- navigational relations: $\downarrow, \downarrow_+$ (in this talk no siblings)
- label equality $X \sim Y$
- no relations of arbitrary interpretation ($knows, likes$ etc.)

Finite trees

Extensional relations are of given interpretation

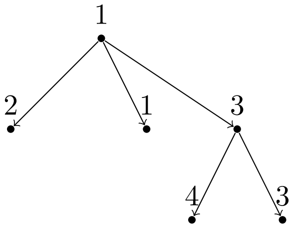
- Σ set of labels (finite or infinite)
- navigational relations: $\downarrow, \downarrow_+$ (in this talk no siblings)
- label equality $X \sim Y$
- no relations of arbitrary interpretation ($knows, likes$ etc.)



Finite trees

Extensional relations are of given interpretation

- Σ set of labels (finite or infinite)
- navigational relations: $\downarrow, \downarrow_+$ (in this talk no siblings)
- label equality $X \sim Y$
- no relations of arbitrary interpretation ($knows, likes$ etc.)



UCQ containment is decidable

Datalog on finite trees example

Datalog on finite trees example

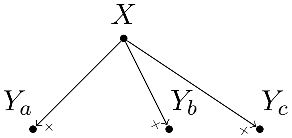
$$
\begin{aligned}
P(X) \leftarrow\ & \downarrow_{+}(X, Y_a), \downarrow_{+}(X, Y_b), \downarrow_{+}(X, Y_c), \\
& a(Y_a), b(Y_b), c(Y_c) \\
& \downarrow(Y_a, Z_a), \downarrow(Y_b, Z_b), \downarrow(Y_c, Z_c) \\
& \sim (X, Z_a), \sim (X, Z_b), \sim (X, Z_c)
\end{aligned}
$$

Datalog on finite trees example

$$
\begin{aligned}
P(X) \;\leftarrow\; & \downarrow_{+}(X, Y_a), \downarrow_{+}(X, Y_b), \downarrow_{+}(X, Y_c), \\
& a(Y_a), b(Y_b), c(Y_c) \\
& \downarrow(Y_a, Z_a), \downarrow(Y_b, Z_b), \downarrow(Y_c, Z_c) \\
& \sim (X, Z_a), \sim (X, Z_b), \sim (X, Z_c)
\end{aligned}
$$

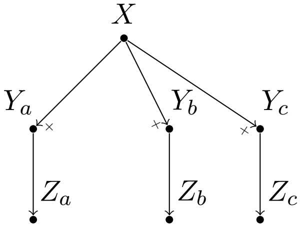
Datalog on finite trees example

$$
\begin{aligned}
P(X) \leftarrow\ & \downarrow_+(X, Y_a), \downarrow_+(X, Y_b), \downarrow_+(X, Y_c), \\
& a(Y_a), b(Y_b), c(Y_c) \\
& \downarrow(Y_a, Z_a), \downarrow(Y_b, Z_b), \downarrow(Y_c, Z_c) \\
& \sim(X, Z_a), \sim(X, Z_b), \sim(X, Z_c)
\end{aligned}
$$



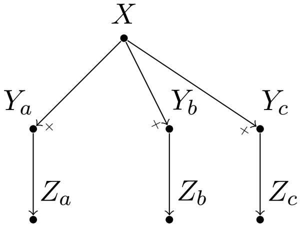
Datalog on finite trees example

$$
\begin{aligned}
P(X) \leftarrow\ & \downarrow_+(X, Y_a), \downarrow_+(X, Y_b), \downarrow_+(X, Y_c), \\
& a(Y_a), b(Y_b), c(Y_c) \\
& \downarrow(Y_a, Z_a), \downarrow(Y_b, Z_b), \downarrow(Y_c, Z_c) \\
& \sim (X, Z_a), \sim (X, Z_b), \sim (X, Z_c)
\end{aligned}
$$



Datalog on finite trees example

$$
\begin{aligned}
P(X) \leftarrow\ & \downarrow_+(X, Y_a), \downarrow_+(X, Y_b), \downarrow_+(X, Y_c), \\
& a(Y_a), b(Y_b), c(Y_c) \\
& \downarrow(Y_a, Z_a), \downarrow(Y_b, Z_b), \downarrow(Y_c, Z_c) \\
& \sim(X, Z_a), \sim(X, Z_b), \sim(X, Z_c)
\end{aligned}
$$



Not expressible in RegXPath

Datalog on finite trees

Datalog on finite trees

Even satisfiability is undecidable [Abiteboul et al.]

Datalog on finite trees

Even satisfiability is undecidable [Abiteboul et al.]

But we need at least binary datalog.

Datalog on finite trees

Even satisfiability is undecidable [Abiteboul et al.]

But we need at least binary datalog.

Finite alphabet?

Datalog on finite trees

Even satisfiability is undecidable [Abiteboul et al.]

But we need at least binary datalog.

Finite alphabet?

Theorem (Gottlob and Koch)

Over trees monadic datalog is equivalent to MSO

Datalog on finite trees

Even satisfiability is undecidable [Abiteboul et al.]

But we need at least binary datalog.

Finite alphabet?

Theorem (Gottlob and Koch)

Over trees monadic datalog is equivalent to MSO

Containment is EXPTIME-complete [Frochaux et al.]

Datalog on finite trees

Even satisfiability is undecidable [Abiteboul et al.]

But we need at least binary datalog.

Finite alphabet?

Theorem (Gottlob and Koch)

Over trees monadic datalog is equivalent to MSO

Containment is EXPTIME-complete [Frochaux et al.]

Infinite alphabet?

Datalog on finite trees

Even satisfiability is undecidable [Abiteboul et al.]

But we need at least binary datalog.

Finite alphabet?

Theorem (Gottlob and Koch)

Over trees monadic datalog is equivalent to MSO

Containment is EXPTIME-complete [Frochaux et al.]

Infinite alphabet?

Theorem (Abiteboul et al.)

Containment of linear connected monadic datalog in UCQ is undecidable.

Datalog on finite trees

Even satisfiability is undecidable [Abiteboul et al.]

But we need at least binary datalog.

Finite alphabet?

Theorem (Gottlob and Koch)

Over trees monadic datalog is equivalent to MSO

Containment is EXPTIME-complete [Frochaux et al.]

Infinite alphabet?

Theorem (Abiteboul et al.)

Containment of linear connected monadic datalog in UCQ is undecidable.

$+$ Some decidability results for bounded depth trees.

Our work

Our work

- models: trees, where Σ is infinite

Our work

- models: trees, where Σ is infinite
- ranked trees vs unranked trees

Our work

- models: trees, where Σ is infinite
- ranked trees vs unranked trees
- datalog: monadic (always), connected (always), linear, downward

Our work

- models: trees, where Σ is infinite
- ranked trees vs unranked trees
- datalog: monadic (always), connected (always), linear, downward
- problems:

Our work

- models: trees, where Σ is infinite
- ranked trees vs unranked trees
- datalog: monadic (always), connected (always), linear, downward
- problems:

equivalence/containment $\quad \mathcal{P} \subseteq \mathcal{Q}$

Our work

- models: trees, where Σ is infinite
- ranked trees vs unranked trees
- datalog: monadic (always), connected (always), linear, downward
- problems:

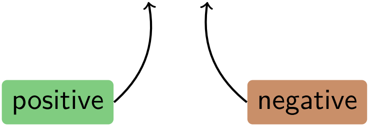
equivalence/containment $\mathcal{P} \subseteq \mathcal{Q}$

positive

Our work

- models: trees, where Σ is infinite
- ranked trees vs unranked trees
- datalog: monadic (always), connected (always), linear, downward
- problems:

equivalence/containment $\mathcal{P} \subseteq \mathcal{Q}$



Our work

- models: trees, where Σ is infinite
- ranked trees vs unranked trees
- datalog: monadic (always), connected (always), linear, downward
- problems:

equivalence/containment $\mathcal{P} \subseteq \mathcal{Q}$

positive

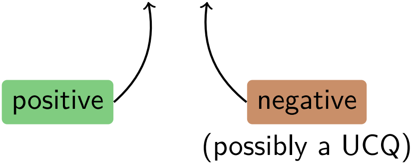
negative

(possibly a UCQ)

Our work

- models: trees, where Σ is infinite
- ranked trees vs unranked trees
- datalog: monadic (always), connected (always), linear, downward
- problems:

equivalence/containment $\mathcal{P} \subseteq \mathcal{Q}$



boundedness $\mathcal{P}$

Containment

(our results)

Containment in short

Containment of linear datalog in UCQ is undecidable.

Containment in short

Containment of linear datalog in UCQ is undecidable.

- needs $\downarrow_+$

Containment in short

Containment of linear datalog in UCQ is undecidable.

- needs $\downarrow_+$
- is on words

Containment in short

Containment of linear datalog in UCQ is undecidable.

- needs $\downarrow_+$
- is on words

child-only (no $\downarrow_+$)

Containment in short

Containment of linear datalog in UCQ is undecidable.

- needs $\downarrow_+$
- is on words

child-only (no $\downarrow_+$)

downward programs

Containment in short

Containment of linear datalog in UCQ is undecidable.

- needs $\downarrow_+$
- is on words

child-only (no $\downarrow_+$)

downward programs

Trees

Containment in short

Containment of linear datalog in UCQ is undecidable.

- needs $\downarrow_+$
- is on words

child-only (no $\downarrow_+$)

downward programs

Trees

ranked

Containment in short

Containment of linear datalog in UCQ is undecidable.

- needs $\downarrow_+$
- is on words

	child-only (no $\downarrow_+$)	downward programs
Trees		
ranked	2EXPTIME	

Containment in short

Containment of linear datalog in UCQ is undecidable.

- needs $\downarrow_+$
- is on words

Trees	child-only (no $\downarrow_+$)	downward programs
ranked	2EXPTIME	UNDEC.

Containment in short

Containment of linear datalog in UCQ is undecidable.

- needs $\downarrow_+$ → child-only (no $\downarrow_+$)
- is on words → downward programs

Trees	child-only (no $\downarrow_+$)	downward programs
ranked	2EXPTIME	UNDEC.
unranked		

Containment in short

Containment of linear datalog in UCQ is undecidable.

- needs $\downarrow_+$
- is on words

Trees	child-only (no $\downarrow_+$)	downward programs
ranked	2EXPTIME	UNDEC.
unranked		2EXPTIME

Containment in short

Containment of linear datalog in UCQ is undecidable.

- needs $\downarrow_+$
- is on words

Trees	child-only (no $\downarrow_+$)	downward programs
ranked	2EXPTIME	UNDEC.
unranked	UNDEC.	2EXPTIME

Containment on ranked trees

$\mathcal{P} \subseteq \mathcal{Q}$?

Containment on ranked trees

$\mathcal{P} \subseteq \mathcal{Q}$?

- Downward programs

Containment on ranked trees

$\mathcal{P} \subseteq \mathcal{Q}$?

- Downward programs

We can enforce a "main path"

Containment on ranked trees

$\mathcal{P} \subseteq \mathcal{Q}$?

- Downward programs

We can enforce a “main path”

$\implies$ undecidable

Containment on ranked trees

$\mathcal{P} \subseteq \mathcal{Q}$?

- Downward programs

We can enforce a "main path"

$\implies$ undecidable

- Child-only programs

Containment on ranked trees

$\mathcal{P} \subseteq \mathcal{Q}$?

- Downward programs

We can enforce a "main path"

$\implies$ undecidable

- Child-only programs

On ranked trees without descendant neighborhood is "small"

Containment on ranked trees

$\mathcal{P} \subseteq \mathcal{Q}$?

- Downward programs

We can enforce a "main path"

$\implies$ undecidable

- Child-only programs

On ranked trees without descendant neighborhood is "small"

We can restrict to a finite alphabet

Containment on ranked trees

$\mathcal{P} \subseteq \mathcal{Q}$?

- Downward programs

We can enforce a “main path”

$\implies$ undecidable

- Child-only programs

On ranked trees without descendant neighborhood is “small”

We can restrict to a finite alphabet

$\implies$ decidability by MSO translation,

Containment on ranked trees

$\mathcal{P} \subseteq \mathcal{Q}$?

- Downward programs

We can enforce a "main path"

$\implies$ undecidable

- Child-only programs

On ranked trees without descendant neighborhood is "small"

We can restrict to a finite alphabet

$\implies$ decidability by MSO translation,

automata construction gives 2EXPTIME

Containment on ranked trees

$\mathcal{P} \subseteq \mathcal{Q}$?

- Downward programs

We can enforce a "main path"

$\implies$ undecidable

- Child-only programs

On ranked trees without descendant neighborhood is "small"

We can restrict to a finite alphabet

$\implies$ decidability by MSO translation,

automata construction gives 2EXPTIME

- Linearity does not change anything (in both cases)

Containment on ranked trees

$\mathcal{P} \subseteq \mathcal{Q}$?

- Downward programs

We can enforce a “main path”

$\implies$ undecidable

- Child-only programs

On ranked trees without descendant neighborhood is “small”

We can restrict to a finite alphabet

$\implies$ decidability by MSO translation,

automata construction gives 2EXPTIME

- Linearity does not change anything (in both cases)
- $\mathcal{Q}$ is a UCQ does not change anything (in both cases)

Containment on unranked trees

$\mathcal{P} \subseteq \mathcal{Q}$?

Containment on unranked trees

$\mathcal{P} \subseteq \mathcal{Q}$?

- Downward programs

Containment on unranked trees

$\mathcal{P} \subseteq \mathcal{Q}$?

- Downward programs

Canonical models: use an alphabet as big as possible

Containment on unranked trees

$\mathcal{P} \subseteq \mathcal{Q}$?

- Downward programs

Canonical models: use an alphabet as big as possible

Complexity 2EXPTIME (EXPSPACE for linear programs)

Containment on unranked trees

$\mathcal{P} \subseteq \mathcal{Q}$?

- Downward programs

Canonical models: use an alphabet as big as possible

Complexity 2EXPTIME (EXPSPACE for linear programs)

$\mathcal{Q}$ is a UCQ does not change anything

Containment on unranked trees

$\mathcal{P} \subseteq \mathcal{Q}$?

- Downward programs

Canonical models: use an alphabet as big as possible

Complexity 2EXPTIME (EXPSPACE for linear programs)

$\mathcal{Q}$ is a UCQ does not change anything

- Child-only programs

Containment on unranked trees

$\mathcal{P} \subseteq \mathcal{Q}$?

- Downward programs

Canonical models: use an alphabet as big as possible

Complexity 2EXPTIME (EXPSPACE for linear programs)

$\mathcal{Q}$ is a UCQ does not change anything

- Child-only programs

We can simulate $\downarrow_+$

Containment on unranked trees

$\mathcal{P} \subseteq \mathcal{Q}$?

- Downward programs

Canonical models: use an alphabet as big as possible

Complexity 2EXPTIME (EXPSPACE for linear programs)

$\mathcal{Q}$ is a UCQ does not change anything

- Child-only programs

We can simulate $\downarrow_+$

$X {\downarrow_+} Y, Y \sim X, \mathcal{P}(Y)$

Containment on unranked trees

$\mathcal{P} \subseteq \mathcal{Q}$?

- Downward programs

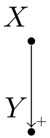
Canonical models: use an alphabet as big as possible

Complexity 2EXPTIME (EXPSPACE for linear programs)

$\mathcal{Q}$ is a UCQ does not change anything

- Child-only programs

We can simulate $\downarrow_+$



$X{\downarrow_+}Y, Y \sim X, \mathcal{P}(Y)$

Containment on unranked trees

$\mathcal{P} \subseteq \mathcal{Q}$?

- Downward programs

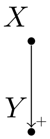
Canonical models: use an alphabet as big as possible

Complexity 2EXPTIME (EXPSPACE for linear programs)

$\mathcal{Q}$ is a UCQ does not change anything

- Child-only programs

We can simulate $\downarrow_+$



$X{\downarrow_+}Y, Y \sim X, \mathcal{P}(Y)$ $\qquad$ $X{\downarrow}Y, X{\downarrow}Z, \mathcal{P}(Y), Y \sim Z, \mathcal{R}(Z)$

Containment on unranked trees

$\mathcal{P} \subseteq \mathcal{Q}$?

- Downward programs

Canonical models: use an alphabet as big as possible

Complexity 2EXPTIME (EXPSPACE for linear programs)

$\mathcal{Q}$ is a UCQ does not change anything

- Child-only programs

We can simulate $\downarrow_+$

$X{\downarrow_+}Y, Y \sim X, \mathcal{P}(Y)$ $\quad$ $X{\downarrow}Y, X{\downarrow}Z, \mathcal{P}(Y), Y \sim Z, \mathcal{R}(Z)$

Containment on unranked trees

$\mathcal{P} \subseteq \mathcal{Q}$?

- Downward programs

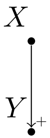
Canonical models: use an alphabet as big as possible

Complexity 2EXPTIME (EXPSPACE for linear programs)

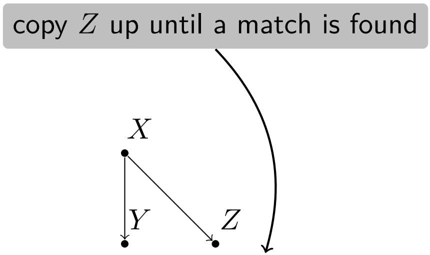
$\mathcal{Q}$ is a UCQ does not change anything

- Child-only programs

We can simulate $\downarrow_+$



$X{\downarrow_+}Y, Y \sim X, \mathcal{P}(Y)$



$X{\downarrow}Y, X{\downarrow}Z, \mathcal{P}(Y), Y \sim Z, \mathcal{R}(Z)$

Containment on unranked trees

$\mathcal{P} \subseteq \mathcal{Q}$?

- Downward programs

Canonical models: use an alphabet as big as possible

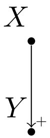
Complexity 2EXPTIME (EXPSPACE for linear programs)

$\mathcal{Q}$ is a UCQ does not change anything

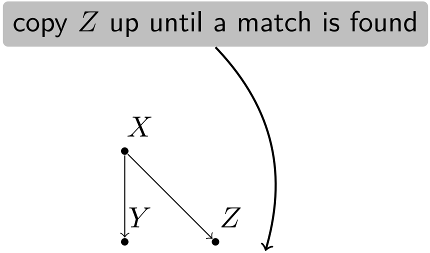
- Child-only programs

We can simulate $\downarrow_+$

$\implies$ undecidable



$X \downarrow_+ Y, Y \sim X, \mathcal{P}(Y)$



$X \downarrow Y, X \downarrow Z, \mathcal{P}(Y), Y \sim Z, \mathcal{R}(Z)$

Containment of child-only programs on unranked trees

$\mathcal{P} \subseteq \mathcal{Q}$?

Containment of child-only programs on unranked trees

$\mathcal{P} \subseteq \mathcal{Q}$?

- Child-only programs (continued)

Containment of child-only programs on unranked trees

$\mathcal{P} \subseteq \mathcal{Q}$?

- Child-only programs (continued)

$\mathcal{Q}$ is a UCQ does not change anything

Containment of child-only programs on unranked trees

$\mathcal{P} \subseteq \mathcal{Q}$?

- Child-only programs (continued)

$\mathcal{Q}$ is a UCQ does not change anything

But for linear programs we get decidability!

Containment of child-only programs on unranked trees

$\mathcal{P} \subseteq \mathcal{Q}$?

- Child-only programs (continued)

$\mathcal{Q}$ is a UCQ does not change anything

But for linear programs we get decidability!

Theorem

Containment of linear child-only programs is in 3EXPTIME,
in 2EXPTIME if $\mathcal{Q}$ is a UCQ

Containment of child-only programs on unranked trees

$\mathcal{P} \subseteq \mathcal{Q}$?

- Child-only programs (continued)

$\mathcal{Q}$ is a UCQ does not change anything

But for linear programs we get decidability!

Theorem

Containment of linear child-only programs is in 3EXPTIME, in 2EXPTIME if $\mathcal{Q}$ is a UCQ

Proof similar as for downward programs

Containment of child-only programs on unranked trees

$\mathcal{P} \subseteq \mathcal{Q}$?

- Child-only programs (continued)

$\mathcal{Q}$ is a UCQ does not change anything

But for linear programs we get decidability!

Theorem

Containment of linear child-only programs is in 3EXPTIME,
in 2EXPTIME if $\mathcal{Q}$ is a UCQ

Proof similar as for downward programs

Recently improved to 2EXPTIME-complete [Bojańczyk et al.]

Boundedness

(our results)

Boundedness

$\mathcal{P}$ bounded?

Boundedness

$\mathcal{P}$ bounded?

For linear datalog undecidable

Boundedness

$\mathcal{P}$ bounded?

For linear datalog undecidable

$\mathcal{P} = \bigcup_i C_i$

Boundedness

$\mathcal{P}$ bounded?

For linear datalog undecidable

$\mathcal{P} = \bigcup_i C_i$

Boundedness or UCQ definability?

Boundedness

$\mathcal{P}$ bounded?

For linear datalog undecidable

$\mathcal{P} = \bigcup_i C_i$

Boundedness or UCQ definability?

$$\begin{aligned} \mathcal{P}(X) &\leftarrow X{\downarrow}Y, \mathcal{P}(Y) \\ \mathcal{P}(X) &\leftarrow b(X) \end{aligned}$$

Boundedness

$\mathcal{P}$ bounded?

For linear datalog undecidable

$\mathcal{P} = \bigcup_i C_i$

Boundedness or UCQ definability?

$$\begin{aligned} \mathcal{P}(X) &\leftarrow X{\downarrow}Y, \mathcal{P}(Y) \\ \mathcal{P}(X) &\leftarrow b(X) \end{aligned}$$

$$\mathcal{P}'(X) \leftarrow X{\downarrow_{+}}Y, b(Y)$$

Boundedness

$\mathcal{P}$ bounded?

For linear datalog undecidable

$\mathcal{P} = \bigcup_i C_i$

Boundedness or UCQ definability?

$$\begin{aligned} \mathcal{P}(X) &\leftarrow X{\downarrow}Y, \mathcal{P}(Y) \\ \mathcal{P}(X) &\leftarrow b(X) \end{aligned}$$

$\mathcal{P} \equiv \mathcal{P}'$

$$\mathcal{P}'(X) \leftarrow X{\downarrow_+}Y, b(Y)$$

Boundedness

$\mathcal{P}$ bounded?

For linear datalog undecidable

$\mathcal{P} = \bigcup_i C_i$

Boundedness or UCQ definability?

$$\begin{aligned} \mathcal{P}(X) &\leftarrow X{\downarrow}Y, \mathcal{P}(Y) \\ \mathcal{P}(X) &\leftarrow b(X) \end{aligned}$$

$\mathcal{P} \equiv \mathcal{P}'$

$$\mathcal{P}'(X) \leftarrow X{\downarrow_+}Y, b(Y)$$

On arbitrary models, or without descendant:

boundedness = UCQ definability

Boundedness

$\mathcal{P}$ bounded?

For linear datalog undecidable

$\mathcal{P} = \bigcup_i C_i$

Boundedness or UCQ definability?

$$\begin{aligned} \mathcal{P}(X) &\leftarrow X{\downarrow}Y, \mathcal{P}(Y) \\ \mathcal{P}(X) &\leftarrow b(X) \end{aligned} \qquad \boxed{\mathcal{P} \equiv \mathcal{P}'}$$

$$\mathcal{P}'(X) \leftarrow X{\downarrow_+}Y, b(Y)$$

On arbitrary models, or without descendant:

boundedness = UCQ definability

We focus on the child-only fragment

Boundedness for child-only programs

$\mathcal{P}$ bounded?

Boundedness for child-only programs

$\mathcal{P}$ bounded?

- Ranked trees

Boundedness for child-only programs

$\mathcal{P}$ bounded?

- Ranked trees

Key Lemma

$\mathcal{P}$ is bounded iff every $\mathcal{P}(X)$ can be evaluated locally

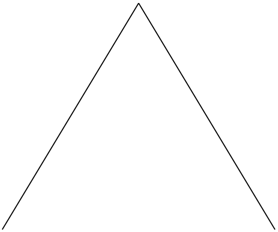
Boundedness for child-only programs

$\mathcal{P}$ bounded?

- Ranked trees

Key Lemma

$\mathcal{P}$ is bounded iff every $\mathcal{P}(X)$ can be evaluated locally



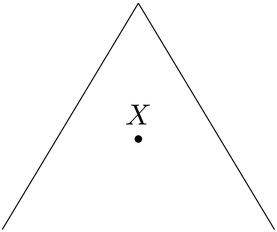
Boundedness for child-only programs

$\mathcal{P}$ bounded?

- Ranked trees

Key Lemma

$\mathcal{P}$ is bounded iff every $\mathcal{P}(X)$ can be evaluated locally



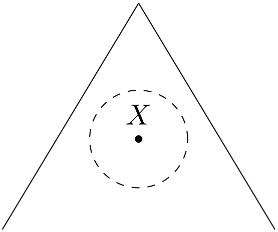
Boundedness for child-only programs

$\mathcal{P}$ bounded?

- Ranked trees

Key Lemma

$\mathcal{P}$ is bounded iff every $\mathcal{P}(X)$ can be evaluated locally



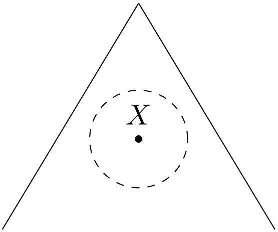
Boundedness for child-only programs

$\mathcal{P}$ bounded?

- Ranked trees

Key Lemma

$\mathcal{P}$ is bounded iff every $\mathcal{P}(X)$ can be evaluated locally



Using automata from containment we show a 2EXPTIME procedure

Boundedness for child-only programs (continued)

$\mathcal{P}$ bounded?

Boundedness for child-only programs (continued)

$\mathcal{P}$ bounded?

- Unranked trees

Boundedness for child-only programs (continued)

$\mathcal{P}$ bounded?

- Unranked trees

Problematic example

Boundedness for child-only programs (continued)

$\mathcal{P}$ bounded?

- Unranked trees

Problematic example

$$\begin{aligned}
\mathcal{P}(X_2) \;&\leftarrow R{\downarrow}X_1, X_1{\downarrow}X_2,\\
&\quad R{\downarrow}Y_1, Y_1{\downarrow}Y_2, X_2 \sim Y_1\\
\mathcal{P}(X_2) \;&\leftarrow R{\downarrow}X_1, X_1{\downarrow}X_2,\\
&\quad R{\downarrow}Y_1, Y_1{\downarrow}Y_2, a(Y_2)
\end{aligned}$$

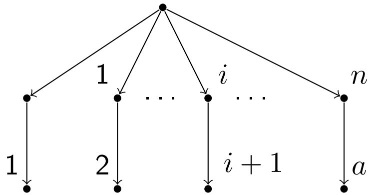
Boundedness for child-only programs (continued)

$\mathcal{P}$ bounded?

- Unranked trees

Problematic example

$$\begin{aligned}\mathcal{P}(X_2) &\leftarrow R{\downarrow}X_1, X_1{\downarrow}X_2,\\ &\quad R{\downarrow}Y_1, Y_1{\downarrow}Y_2, X_2 \sim Y_1\\ \mathcal{P}(X_2) &\leftarrow R{\downarrow}X_1, X_1{\downarrow}X_2,\\ &\quad R{\downarrow}Y_1, Y_1{\downarrow}Y_2, a(Y_2)\end{aligned}$$



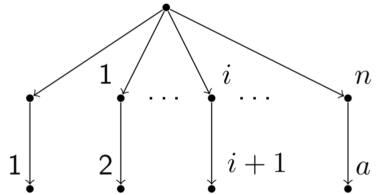
Boundedness for child-only programs (continued)

$\mathcal{P}$ bounded?

- Unranked trees

Problematic example

$$\begin{aligned}\mathcal{P}(X_2) &\leftarrow R{\downarrow}X_1, X_1{\downarrow}X_2,\\ &\quad R{\downarrow}Y_1, Y_1{\downarrow}Y_2, X_2 \sim Y_1\\ \mathcal{P}(X_2) &\leftarrow R{\downarrow}X_1, X_1{\downarrow}X_2,\\ &\quad R{\downarrow}Y_1, Y_1{\downarrow}Y_2, a(Y_2)\end{aligned}$$



Previous key lemma does not work

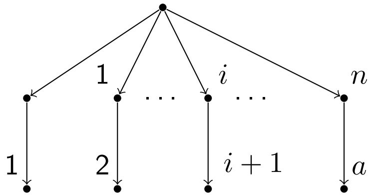
Boundedness for child-only programs (continued)

$\mathcal{P}$ bounded?

- Unranked trees

Problematic example

$$\begin{aligned}\mathcal{P}(X_2) &\leftarrow R{\downarrow}X_1, X_1{\downarrow}X_2,\\ &\quad R{\downarrow}Y_1, Y_1{\downarrow}Y_2, X_2 \sim Y_1\\ \mathcal{P}(X_2) &\leftarrow R{\downarrow}X_1, X_1{\downarrow}X_2,\\ &\quad R{\downarrow}Y_1, Y_1{\downarrow}Y_2, a(Y_2)\end{aligned}$$



Previous key lemma does not work

Work in progress . . .

Effective boundedness

Boundedness asks about existence of a UCQ

Effective boundedness

Boundedness asks about existence of a UCQ

Can you produce an equivalent UCQ?

Effective boundedness

Boundedness asks about existence of a UCQ

Can you produce an equivalent UCQ?

All decidability results here were effective.

Effective boundedness

Boundedness asks about existence of a UCQ

Can you produce an equivalent UCQ?

All decidability results here were effective.

if $|\mathcal{P}| = n$ then the size of equivalent UCQ is $\leq f(n)$,

Effective boundedness

Boundedness asks about existence of a UCQ

Can you produce an equivalent UCQ?

All decidability results here were effective.

if $|\mathcal{P}| = n$ then the size of equivalent UCQ is $\leq f(n)$,
where f is computable

Effective boundedness

Boundedness asks about existence of a UCQ

Can you produce an equivalent UCQ?

All decidability results here were effective.

if $|\mathcal{P}| = n$ then the size of equivalent UCQ is $\leq f(n)$,
where f is computable

For datalog boundedness decidability proofs essentially give f

Effective boundedness

Boundedness asks about existence of a UCQ

Can you produce an equivalent UCQ?

All decidability results here were effective.

if $|\mathcal{P}| = n$ then the size of equivalent UCQ is $\leq f(n)$,
where f is computable

For datalog boundedness decidability proofs essentially give f

Is boundedness and effective boundedness the same problem?

Boundedness and containment in a UCQ

Forget about trees, restrictions, etc.

Boundedness and containment in a UCQ

Forget about trees, restrictions, etc.

Suppose a computable f is given
(maximal bound for the equivalent UCQ)

Boundedness and containment in a UCQ

Forget about trees, restrictions, etc.

Suppose a computable f is given
(maximal bound for the equivalent UCQ)

Then decidability of containment $\implies$ decidability of boundedness

Boundedness and containment in a UCQ

Forget about trees, restrictions, etc.

Suppose a computable f is given
(maximal bound for the equivalent UCQ)

Then decidability of containment $\implies$ decidability of boundedness
because $\mathcal{P} = \bigcup_i C_i$, so we check $\mathcal{P} \subseteq C_i$ for $i \leq f(|\mathcal{P}|)$

Boundedness and containment in a UCQ

Forget about trees, restrictions, etc.

Suppose a computable f is given
(maximal bound for the equivalent UCQ)

Then decidability of containment $\implies$ decidability of boundedness
because $\mathcal{P} = \bigcup_i C_i$, so we check $\mathcal{P} \subseteq C_i$ for $i \leq f(|\mathcal{P}|)$

And decidability of boundedness $\implies$ decidability of containment

Boundedness and containment in a UCQ

Forget about trees, restrictions, etc.

Suppose a computable f is given
(maximal bound for the equivalent UCQ)

Then decidability of containment $\implies$ decidability of boundedness
because $\mathcal{P} = \bigcup_i C_i$, so we check $\mathcal{P} \subseteq C_i$ for $i \leq f(|\mathcal{P}|)$

And decidability of boundedness $\implies$ decidability of containment
Consider $\mathcal{P} \subseteq \mathcal{Q}$ $\quad$ ($\mathcal{Q}$ is a UCQ)

Boundedness and containment in a UCQ

Forget about trees, restrictions, etc.

Suppose a computable f is given
(maximal bound for the equivalent UCQ)

Then decidability of containment $\implies$ decidability of boundedness
because $\mathcal{P} = \bigcup_i C_i$, so we check $\mathcal{P} \subseteq C_i$ for $i \leq f(|\mathcal{P}|)$

And decidability of boundedness $\implies$ decidability of containment
Consider $\mathcal{P} \subseteq \mathcal{Q}$ ($\mathcal{Q}$ is a UCQ)
If $\mathcal{P}$ is bounded then reduces to deciding containment for UCQs

Boundedness and containment in a UCQ

Forget about trees, restrictions, etc.

Suppose a computable f is given
(maximal bound for the equivalent UCQ)

Then decidability of containment $\implies$ decidability of boundedness
because $\mathcal{P} = \bigcup_i C_i$, so we check $\mathcal{P} \subseteq C_i$ for $i \leq f(|\mathcal{P}|)$

And decidability of boundedness $\implies$ decidability of containment
Consider $\mathcal{P} \subseteq \mathcal{Q}$ $\quad$ ($\mathcal{Q}$ is a UCQ)
If $\mathcal{P}$ is bounded then reduces to deciding containment for UCQs
If not, $\mathcal{P}$ and $\mathcal{Q}$ still could be equivalent $(X{\downarrow_+}Y, b(Y))$

Boundedness and containment in a UCQ

Forget about trees, restrictions, etc.

Suppose a computable f is given
(maximal bound for the equivalent UCQ)

Then decidability of containment $\implies$ decidability of boundedness
because $\mathcal{P} = \bigcup_i C_i$, so we check $\mathcal{P} \subseteq C_i$ for $i \leq f(|\mathcal{P}|)$

And decidability of boundedness $\implies$ decidability of containment
Consider $\mathcal{P} \subseteq \mathcal{Q}$ ($\mathcal{Q}$ is a UCQ)
If $\mathcal{P}$ is bounded then reduces to deciding containment for UCQs
If not, $\mathcal{P}$ and $\mathcal{Q}$ still could be equivalent $(X{\downarrow_+}Y, b(Y))$
Let:

Boundedness and containment in a UCQ

Forget about trees, restrictions, etc.

Suppose a computable f is given
(maximal bound for the equivalent UCQ)

Then decidability of containment $\implies$ decidability of boundedness
because $\mathcal{P} = \bigcup_i C_i$, so we check $\mathcal{P} \subseteq C_i$ for $i \leq f(|\mathcal{P}|)$

And decidability of boundedness $\implies$ decidability of containment
Consider $\mathcal{P} \subseteq \mathcal{Q}$ ($\mathcal{Q}$ is a UCQ)
If $\mathcal{P}$ is bounded then reduces to deciding containment for UCQs
If not, $\mathcal{P}$ and $\mathcal{Q}$ still could be equivalent $(X{\downarrow_+}Y, b(Y))$
Let:

$$\mathcal{R}(X) \leftarrow \mathcal{P}(X)$$
$$\mathcal{R}(X) \leftarrow \mathcal{Q}(X)$$

Boundedness and containment in a UCQ

Forget about trees, restrictions, etc.

Suppose a computable f is given
(maximal bound for the equivalent UCQ)

Then decidability of containment $\implies$ decidability of boundedness
because $\mathcal{P} = \bigcup_i C_i$, so we check $\mathcal{P} \subseteq C_i$ for $i \leq f(|\mathcal{P}|)$

And decidability of boundedness $\implies$ decidability of containment
Consider $\mathcal{P} \subseteq \mathcal{Q}$ ($\mathcal{Q}$ is a UCQ)
If $\mathcal{P}$ is bounded then reduces to deciding containment for UCQs
If not, $\mathcal{P}$ and $\mathcal{Q}$ still could be equivalent $(X{\downarrow_+}Y, b(Y))$
Let:

$\mathcal{R}(X) \leftarrow \mathcal{P}(X)$ If $\mathcal{R}$ is not bounded then $\mathcal{P} \not\subseteq \mathcal{Q}$
$\mathcal{R}(X) \leftarrow \mathcal{Q}(X)$

Boundedness and containment in a UCQ

Forget about trees, restrictions, etc.

Suppose a computable f is given
(maximal bound for the equivalent UCQ)

Then decidability of containment $\implies$ decidability of boundedness
because $\mathcal{P} = \bigcup_i C_i$, so we check $\mathcal{P} \subseteq C_i$ for $i \leq f(|\mathcal{P}|)$

And decidability of boundedness $\implies$ decidability of containment
Consider $\mathcal{P} \subseteq \mathcal{Q}$ ($\mathcal{Q}$ is a UCQ)
If $\mathcal{P}$ is bounded then reduces to deciding containment for UCQs
If not, $\mathcal{P}$ and $\mathcal{Q}$ still could be equivalent $(X{\downarrow_+}Y, b(Y))$
Let:

$\mathcal{R}(X) \leftarrow \mathcal{P}(X)$ If $\mathcal{R}$ is not bounded then $\mathcal{P} \not\subseteq \mathcal{Q}$
$\mathcal{R}(X) \leftarrow \mathcal{Q}(X)$ Otherwise reduces to UCQ containment ($\mathcal{R} \subseteq \mathcal{Q}$)

Conclusions

On the border between decidability and undecidability.

Conclusions

On the border between decidability and undecidability.

Boundedness on unranked trees?

Conclusions

On the border between decidability and undecidability.

Boundedness on unranked trees?

UCQ definability?

Conclusions

On the border between decidability and undecidability.

Boundedness on unranked trees?

UCQ definability?

Datalog with siblings?

Conclusions

On the border between decidability and undecidability.

Boundedness on unranked trees?

UCQ definability?

Datalog with siblings?
(child, next-sibling only)

Conclusions

On the border between decidability and undecidability.

Boundedness on unranked trees?

UCQ definability?

Datalog with siblings?
(child, next-sibling only)

Boundedness vs effective boundedness?

Conclusions

On the border between decidability and undecidability.

Boundedness on unranked trees?

UCQ definability?

Datalog with siblings?
(child, next-sibling only)

Boundedness vs effective boundedness?
vs containment?