Conjunctive Query Containment under Constraints and Access Limitations: Two Chase Techniques

Andrea Calì

Dept. of Information Systems and Computing, Brunel University, UK
Oxford-Man Institute of Quantitative Finance, University of Oxford, UK

Politecnico di Bari
30th August 2010

What is query containment?

Definition

Given two queries q_1 and q_2 we say q_1 is contained in q_2, denoted $q_1 \subseteq q_2$, if for every database D we have

$$q_1(D) \subseteq q_2(D)$$

$q(D)$: result of evaluation of q over D

Query containment

- Fundamental issue in query optimisation
- We consider conjunctive queries over schemata schemata with constraints
- The presence of constraints makes query containment checking difficult
- Need for reasoning on constraints imposed by the database schema
- q_1 contained in q_2 under Σ, denoted $q_1 \subseteq_\Sigma q_2$, if for every database D that satisfies Σ we have $q_1(D) \subseteq q_2(D)$

Conjunctive query containment: algorithm

1. **freeze** body(q_1) and head(q_1) by turning each variable into a distinct (fresh) constant
2. evaluate q_2 over the frozen body of q_1
3. $q_1 \subseteq q_2$ iff the evaluation returns the frozen head of q_1

Testing containment amounts to checking the existence of a **query homomorphism** from q_2 to q_1 [Chandra & Merlin 1977].

Example

From [Ullman 1997]

$$\begin{array}{lcl} q_1: & p(X,Z) & \leftarrow & a(X,Y), a(Y,Z) \\ q_2: & p(X,Z) & \leftarrow & a(X,U), a(V,Z) \end{array}$$

Example

From [Ullman 1997]

$$\begin{array}{lll} q_1: & p(X,Z) & \leftarrow \quad a(X,Y), a(Y,Z) \\ q_2: & p(X,Z) & \leftarrow \quad a(X,U), a(V,Z) \end{array}$$

Frozen body(q_1):

$$\begin{array}{l} a(0,1) \leftarrow \\ a(1,2) \leftarrow \end{array}$$

Example

From [Ullman 1997]

$$\begin{array}{lll} q_1: & p(X,Z) & \leftarrow \quad a(X,Y), a(Y,Z) \\ q_2: & p(X,Z) & \leftarrow \quad a(X,U), a(V,Z) \end{array}$$

Frozen body(q_1):

$$a(0,1) \leftarrow$$
$$a(1,2) \leftarrow$$

Frozen head(q_1): $p(0,2) \leftarrow$

Example (contd.)

Applying q_2 to the frozen body(q_1), we find a substitution:

$$X \rightarrow 0,\ U \rightarrow 1,\ V \rightarrow 1,\ Z \rightarrow 2$$

that yields $p(0, 2)$ which is the frozen head of q_1.

Example (contd.)

Applying q_2 to the frozen body(q_1), we find a substitution:

$$X \rightarrow 0,\ U \rightarrow 1,\ V \rightarrow 1,\ Z \rightarrow 2$$

that yields $p(0, 2)$ which is the frozen head of q_1.
Therefore $q_1 \subseteq q_2$.

Example (contd.)

Applying q_2 to the frozen body(q_1), we find a substitution:

$$X \rightarrow 0,\ U \rightarrow 1,\ V \rightarrow 1,\ Z \rightarrow 2$$

that yields $p(0, 2)$ which is the frozen head of q_1.
Therefore $q_1 \subseteq q_2$.

Note

The frozen body of q_1 is a representative of (a piece of) all databases that provide an answer to q_1

Outline

1 Introduction

2 Query containment under database constraints

3 QC with the chase

4 The Deep Web

5 Query Containment – recall

6 The Crayfish-Chase

7 Complexity of QC under access limitations

8 Conclusions

Query containment under constraints

Definition

Given a set Σ of database dependencies, we say q_1 **is contained in** q_2 under Σ, denoted $q_1 \subseteq_\Sigma q_2$, if **for every database** D such that $D \models \Sigma$ we have

$$q_1(D) \subseteq q_2(D)$$

Our setting

Queries

- conjunctive queries (CQs)

Dependencies

1. key dependencies (KDs)
   $key(r) = \{A_1, \ldots, A_k\}$
2. inclusion dependencies (IDs) (generalisation of foreign key dependencies)
   $r_1[A_1, \ldots, A_m] \subseteq r_2[B_1, \ldots, B_m]$

QC under constraints: example

Schema

$$\text{employee}(Emp_id, Salary, Dept)$$
$$\text{department}(Dept, Location)$$

with constraint employee[3] $\subseteq$ department[1].

QC under constraints: example

Schema

$$\text{employee}(Emp_id, Salary, Dept)$$
$$\text{department}(Dept, Location)$$

with constraint employee[3] $\subseteq$ department[1].

Queries

$$\begin{array}{lll} q_1: & p(X) & \leftarrow \quad \text{employee}(X, Y, Z), \text{department}(Z, W) \\ q_2: & p(X) & \leftarrow \quad \text{employee}(X, Y, Z) \end{array}$$

QC under constraints: example

Schema

$$\text{employee}(Emp_id, Salary, Dept)$$
$$\text{department}(Dept, Location)$$

with constraint employee[3] $\subseteq$ department[1].

Queries

$$\begin{array}{lll} q_1: & p(X) & \leftarrow \quad \text{employee}(X, Y, Z), \text{department}(Z, W) \\ q_2: & p(X) & \leftarrow \quad \text{employee}(X, Y, Z) \end{array}$$

$q_1 \subseteq q_2$ and $q_2 \not\subseteq q_1$, but notice that $q_2 \subseteq_\Sigma q_1$ (queries are equivalent under Σ).

Checking QC under database dependencies

Intuition

- Once we freeze q_1 we are constructing a generic database that provides an answer to q_1
- When we freeze, we must construct a database that satisfies Σ
- We do that by constructing the chase of the frozen query

Chasing. . .

"My heart's in the Highlands, my heart is not here,
My heart's in the Highlands a-chasing the deer.
A-chasing the wild deer, and following the roe;
My heart's in the Highlands, wherever I go."

Chasing. . .

"My heart's in the Highlands, my heart is not here,
My heart's in the Highlands a-chasing the deer.
A-chasing the wild deer, and following the roe;
My heart's in the Highlands, wherever I go."

Robert Burns

Chasing. . .

"My heart's in the Highlands, my heart is not here,
My heart's in the Highlands a-chasing the deer.
A-chasing the wild deer, and following the roe;
My heart's in the Highlands, wherever I go."

Robert Burns

For much worse Scottish poetry, see William Mcgonagall (acclaimed as the worst British poet ever)

The chase

The chase "repairs" the frozen body of q_1 in two ways:

1. "collapsing" pairs of facts that violate a KD (**KD chase rule**)
2. adding facts when an ID is violated (**ID chase rule**)

Containment test under dependencies

Theorem [Johnson & Klug 1982]

$q_2 \subseteq_\Sigma q_1$ iff there is a homomorphism that maps body(q_2) on the chase of the frozen body(q_1), and the head of q_2 to the frozen head of q_1

Note

The chase is a representative for all databases that satisfy Σ and provide an answer for q_1.

Infinite chase: example

Relations $r/2$, $s/2$

Dependencies

$$\begin{array}{lll} \sigma_1: & r[1] & \subseteq s[1] \\ \sigma_2: & s[2] & \subseteq r[1] \\ \sigma_3: & s[2] & \subseteq s[1] \\ \gamma 1: & key(r) & = \{1\} \end{array}$$

Infinite chase: example

Relations $r/2$, $s/2$

Dependencies

$$\begin{array}{lll} \sigma_1: & r[1] & \subseteq s[1] \\ \sigma_2: & s[2] & \subseteq r[1] \\ \sigma_3: & s[2] & \subseteq s[1] \\ \gamma 1: & key(r) & = \{1\} \end{array}$$

Initial database (frozen body)

$$r(a, \alpha_0) \leftarrow$$
$$r(a, b) \leftarrow$$

Greek letters denote (frozen) non-distinguished variables (NDVs)

Infinite chase: example (contd.)

KD chase rule

We collapse the first two facts (due to γ_1) by forcing $\alpha_0 = b$.

Infinite chase: example (contd.)

KD chase rule

We collapse the first two facts (due to γ_1) by forcing $\alpha_0 = b$.

ID chase rule

Added facts due to IDs:

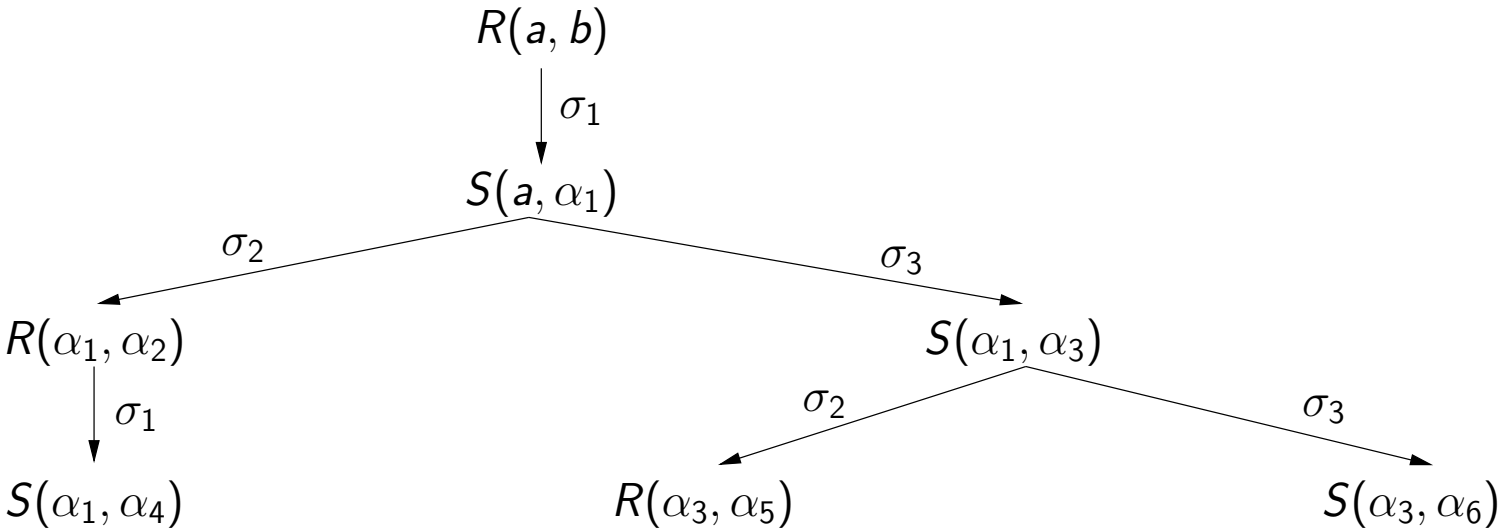
$$
\begin{aligned}
&s(a, \alpha_1) \leftarrow \\
&r(\alpha_1, \alpha_2) \leftarrow \\
&s(\alpha_1, \alpha_3) \leftarrow \\
&s(\alpha_1, \alpha_4) \leftarrow \\
&r(\alpha_3, \alpha_5) \leftarrow \\
&s(\alpha_3, \alpha_6) \leftarrow \\
&\ldots
\end{aligned}
$$

(...ad infinitum!)

Chase graph

- Facts in the frozen body of q_1 have level 0. A fact derived from the ID chase rule from another fact that is at level k has level $k + 1$
- If fact f_2 is derived from fact f_1 by an ID σ, there is an arc (f_1, f_2) labelled with σ

Example: chase graph



Undecidability of QC under KDs and IDs

Known result

QC under general functional dep. and IDs is undecidable [Chandra & Vardi 1982]

Undecidability of QC under KDs and IDs

Known result

QC under general functional dep. and IDs is undecidable [Chandra & Vardi 1982]

Theorem

QC under general KDs and IDs is undecidable

Proof sketch: Reduction from implication of KDs and IDs. Consider r/n, s/m, a set of dep. Σ and a constraint $\sigma: \; r[1,\ldots,k] \subseteq s[1,\ldots,k]$.

$$\begin{array}{lcl} q_1: \; q() & \leftarrow & r(X_1,\ldots,X_k,\ldots,X_n) \\ q_2: \; q() & \leftarrow & r(X_1,\ldots,X_k,\ldots,X_n), \\ & & s(X_1,\ldots,X_k,Y_1,\ldots,Y_{m-k}) \end{array}$$

it is easy to see that $q_1 \subseteq_\Sigma q_2$ iff $\Sigma \models \sigma$.

QC under IDs alone

Theorem [Johnson & Klug JCSS 1984]

Containment is decidable in PSPACE.

Proof sketch:

- only a finite portion of the chase is necessary
- notion of **equivalent conjuncts** (agree on non-fresh constants)
- given a fact, an equivalent conjunct is found within $\delta = |\Sigma| \cdot (W+1)^W$, W maximum "width" of IDs in Σ
- Taking into account joins in q_2: the necessary depth is $|q_2| \cdot \delta$
- A nontrivial guess shows memberships in PSPACE
- PSPACE-hardness is also proved (like undecidability)

QC under KDs and IDs: decidable cases

- key-based IDs [Johnson & Klug 1984]; limited class, but is more general than forign keys
- **non-key-conflicting IDs** [Calì & al. PODS 2003]; more general class than key-based IDs

Non-key-conflicting IDs

Definition

Non-key-conflicting IDs (NKCIDs) are of the form

$$r_1[\mathbf{A}_1] \subseteq r_2[\mathbf{A}_2]$$

where either:

1. no KD is defined over r_2
2. $\mathbf{A}_2$ is **not** a strict superset of $key(r_2)$

Separation Theorem

Theorem

Given q_1, q_2, Σ (NKCIDs), with

$$\Sigma = \Sigma_K \cup \Sigma_I (\text{KDs and IDs})$$

if the chase w.r.t. Σ does not fail in the first applications of the FD chase rule:

$$q_1 \subseteq_\Sigma q_2 \text{ iff } q_1 \subseteq_{\Sigma_I} q_2$$

Proof

Based on the fact that if KDs are not violated in the first step of the chase, they are never violated

Summary of complexity results

KDs	IDs	complexity
no	GEN	PSPACE
yes	no	NP
yes	FK	PSPACE
yes	NKC	PSPACE
yes	1KC	undecidable
yes	GEN	undecidable

The Deep Web

Data behind forms

- data accessible through Web forms
  - phone directories
  - auctions
  - stores
- heterogeneous sources can be integrated in a Web information system

Modelling access limitations

Example: white pages

- No possibility of asking for all entries (filling in no fields)
- At least one field must be filled in
- The result is a table

Modelling access limitations

Example: white pages

- No possibility of asking for all entries (filling in no fields)
- At least one field must be filled in
- The result is a table

Modelling

- We model each source as a table
- Filling in a field in the form corresponds to querying with a selection only

Example: whitepages.com – query

Example: whitepages.com – query

```
SELECT *
FROM whitepages
WHERE firstname='joseph'
 AND lastname='noto'
 AND stateprov='NJ'
```

Example: whitepages.com – results

Joseph A Noto

60 Inman Ave
Colonia, NJ 07067-1802
(732) 381-7723

Listing Details

Age: 45-49
Household: Jennie, Pamela

SPONSORED LINKS
Lookup Background Info for Joseph Noto.
Find School Information on Joseph Noto
View Joseph Noto Profile

HOUSEVALUES.COM
See What Joseph Noto's Home is Worth

beta **Make it easy for people to contact you on WhitePages.com** Learn more **Add your listing**

Joseph C Noto Jr

81 E Grove St
Bogota, NJ 07603-1107
(201) 488-6593

Listing Details

Age: 55-59

SPONSORED LINKS
Lookup Background Info for Joseph Noto.
Find School Information on Joseph Noto
View Joseph Noto Profile

HOUSEVALUES.COM
See What Joseph Noto's Home is Worth

Joseph C Noto

135 E McClellan Ave, Apt 13
Livingston, NJ 07039-1346
phone number unavailable

Listing Details

SPONSORED LINKS
Lookup Background Info for Joseph Noto.
Find School Information on Joseph Noto
View Joseph Noto Profile

HOUSEVALUES.COM
See What Joseph Noto's Home is Worth

Modelling query answering

Modelling

- We consider conjunctive queries in a relational setting
- We model each data source requiring a certain selection on attributes as a relation

Modelling query answering

Modelling

- We consider conjunctive queries in a relational setting
- We model each data source requiring a certain selection on attributes as a relation

Observations

- Limitations restrict the answers we can retrieve
- we are interested in maximal answers (w.r.t. set inclusion)

Outline

1 Introduction

2 Query containment under database constraints

3 QC with the chase

4 The Deep Web

5 Query Containment – recall

6 The Crayfish-Chase

7 Complexity of QC under access limitations

8 Conclusions

Example

Superscripts denote input and output attributes

Schema: concerts and artists

$$r_1^{oio}(Title, City, Artist)$$
$$r_2^{ioo}(Artist, Nation, City)$$

Example

Superscripts denote input and output attributes

Schema: concerts and artists

$$r_1^{oio}(Title, City, Artist)$$
$$r_2^{ioo}(Artist, Nation, City)$$

Query

$$q(A) \leftarrow r_2(A, italy, modena)$$

Example

Superscripts denote input and output attributes

Schema: concerts and artists

$r_1^{oio}(Title, City, Artist)$
$r_2^{ioo}(Artist, Nation, City)$

Query

$q(A) \leftarrow r_2(A, italy, modena)$

Best answering: q cannot be executed directly!

- Starting from the constant *modena*, we can access r_1
- then we can obtain tuples with new *Artist* constants
- with such values we can access r_2 and start over
- We consider abstract domains (*Year*, *Artist* etc.)

⋆ We assume never to enumerate domain values

Example (cont'd)

Relation r_1

azzurro	*modena*	*conte*

Relation r_2

Answer tuples

Example (cont'd)

Relation r_1

azzurro	*modena*	*conte*

Relation r_2

conte	*italy*	*genoa*

Answer tuples

Example (cont'd)

Relation r_1

azzurro	*modena*	*conte*
volare	*genoa*	*modugno*
K551	*genoa*	*kissin*

Relation r_2

conte	*italy*	*genoa*

Answer tuples

Example (cont'd)

Relation r_1

azzurro	*modena*	*conte*
volare	*genoa*	*modugno*
K551	*genoa*	*kissin*

Relation r_2

conte	*italy*	*genoa*
kissin	*russia*	*moscow*
modugno	*italy*	*polignano*

Answer tuples

Example (cont'd)

Relation r_1

azzurro	*modena*	*conte*
volare	*genoa*	*modugno*
K551	*genoa*	*kissin*
sole_mio	*moscow*	*pavarotti*

Relation r_2

conte	*italy*	*genoa*
kissin	*russia*	*moscow*
modugno	*italy*	*polignano*

Answer tuples

Example (cont'd)

Relation r_1

azzurro	*modena*	*conte*
volare	*genoa*	*modugno*
K551	*genoa*	*kissin*
sole_mio	*moscow*	*pavarotti*

Relation r_2

conte	*italy*	*genoa*
kissin	*russia*	*moscow*
modugno	*italy*	*polignano*
pavarotti	*italy*	*modena*

Answer tuples

Example (cont'd)

Relation r_1

azzurro	*modena*	*conte*
volare	*genoa*	*modugno*
K551	*genoa*	*kissin*
sole_mio	*moscow*	*pavarotti*

Relation r_2

conte	*italy*	*genoa*
kissin	*russia*	*moscow*
modugno	*italy*	*polignano*
pavarotti	*italy*	*modena*

Answer tuples

$\langle pavarotti \rangle$

Providing maximal answers

- Basic technique in [Millstein et al. 2000]
- Answering is inherently recursive
- Need for a set of initial constants (usually those in the query)
- Encoding in positive Datalog

Naive program for previous example

$$
\begin{array}{lrcl}
\rho_1: & q(A) & \leftarrow & \hat{r}_2(A, italian, modena) \\
\rho_2: & \hat{r}_1(T, C, A) & \leftarrow & r_1(T, C, A), dom_C(C) \\
\rho_3: & \hat{r}_2(A, N, C) & \leftarrow & r_2(A, N, C), dom_A(A) \\
\rho_4: & dom_T(T) & \leftarrow & \hat{r}_1(T, C, A) \\
\rho_5: & dom_C(C) & \leftarrow & \hat{r}_1(T, C, A) \\
\rho_6: & dom_A(A) & \leftarrow & \hat{r}_1(T, C, A) \\
\rho_7: & dom_A(A) & \leftarrow & \hat{r}_2(A, N, C) \\
\rho_8: & dom_N(N) & \leftarrow & \hat{r}_2(A, N, C) \\
\rho_9: & dom_C(C) & \leftarrow & \hat{r}_2(A, N, C) \\
\rho_{10}: & dom_N(italian) & & \\
\rho_{11}: & dom_C(modena) & &
\end{array}
$$

The containment problem

Notation

- Conjunctive queries q_1, q_2
- Relational schema $\mathcal{S}$ with limitations Λ
- Initial constants $I \supseteq \text{const}(q_1) \cup \text{const}(q_2)$
- $\text{ans}(q, \mathcal{S}, D, I)$: maximal answer to q evaluated on a schema $\mathcal{S}$ under limitations Λ using initial constants I on database D.

The containment problem

Notation

- Conjunctive queries q_1, q_2
- Relational schema $\mathcal{S}$ with limitations Λ
- Initial constants $I \supseteq \text{const}(q_1) \cup \text{const}(q_2)$
- $\text{ans}(q, \mathcal{S}, D, I)$: maximal answer to q evaluated on a schema $\mathcal{S}$ under limitations Λ using initial constants I on database D.

Containment

Containment $q_1 \subseteq_{\Lambda,I} q_2$ under limitations holds if for every database D for $\mathcal{S}$ we have

$$\text{ans}(q_1, \mathcal{S}, D, I) \subseteq \text{ans}(q_2, \mathcal{S}, D, I)$$

The containment problem (cont'd)

Containment is useful for:

- query minimisation
- optimisation of query execution

The containment problem (cont'd)

Containment is useful for:

- query minimisation
- optimisation of query execution

- Checking containment amounts to checking containment between two Datalog programs
  - ⋆ answering is inherently recursive
  - ⋆ ...due to cyclic reuse of constants according to their abstract domains
- Datalog query containment is undecidable; however, programs have a special form

The chase – recall

Used for:

- implication of relational dependencies
- query containment [Johnson & Klug 1984]
- querying incomplete data
- data integration and data exchange

We have a different version...

The crayfish-chase

The crayfish-chase

The crayfish-chase

Features

- Constructed starting from a query q and a set of initial constants
- It is a set of databases, denoted $\text{cchase}(q, \mathcal{S}, I)$
- Every database represents one way of "extracting" an answer tuple
- Thus, the chase serves as a tool for containment

The crayfish-chase: example

Schema

$r_1^{iio}(A, B, A)$
$r_2^{io}(A, B)$
$r_3^{o}(A)$

Query

$q(X_2) \leftarrow r_1(a, X_1, X_2)$

- W.l.o.g., we use constant-free queries
- First the query is frozen, then "expanded" by chasing
  - ⋆ Special constants ζ_i denote unknown values (labelled nulls)
- Chase seen as a forest, layered in levels

The crayfish-chase: example (cont'd)

Query with constants eliminated

$$q(X_2) \leftarrow r_1(X_a, X_1, X_2), \ell_a(X_a)$$

- ℓ_a is an aux. predicate with extension $\{\langle a \rangle\}$
- the chase starts from the frozen body

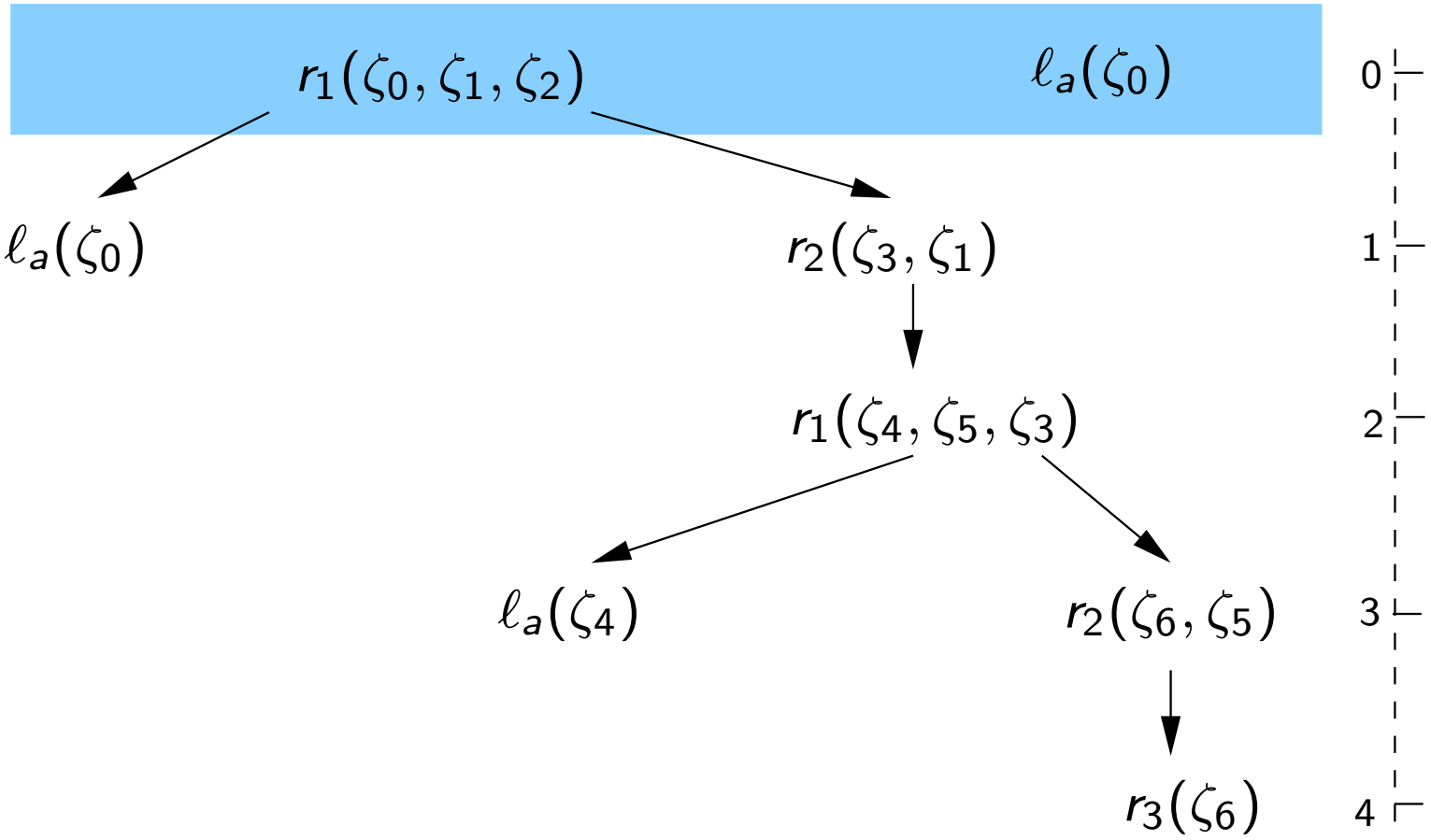
Frozen head

$$q(\zeta_2)$$

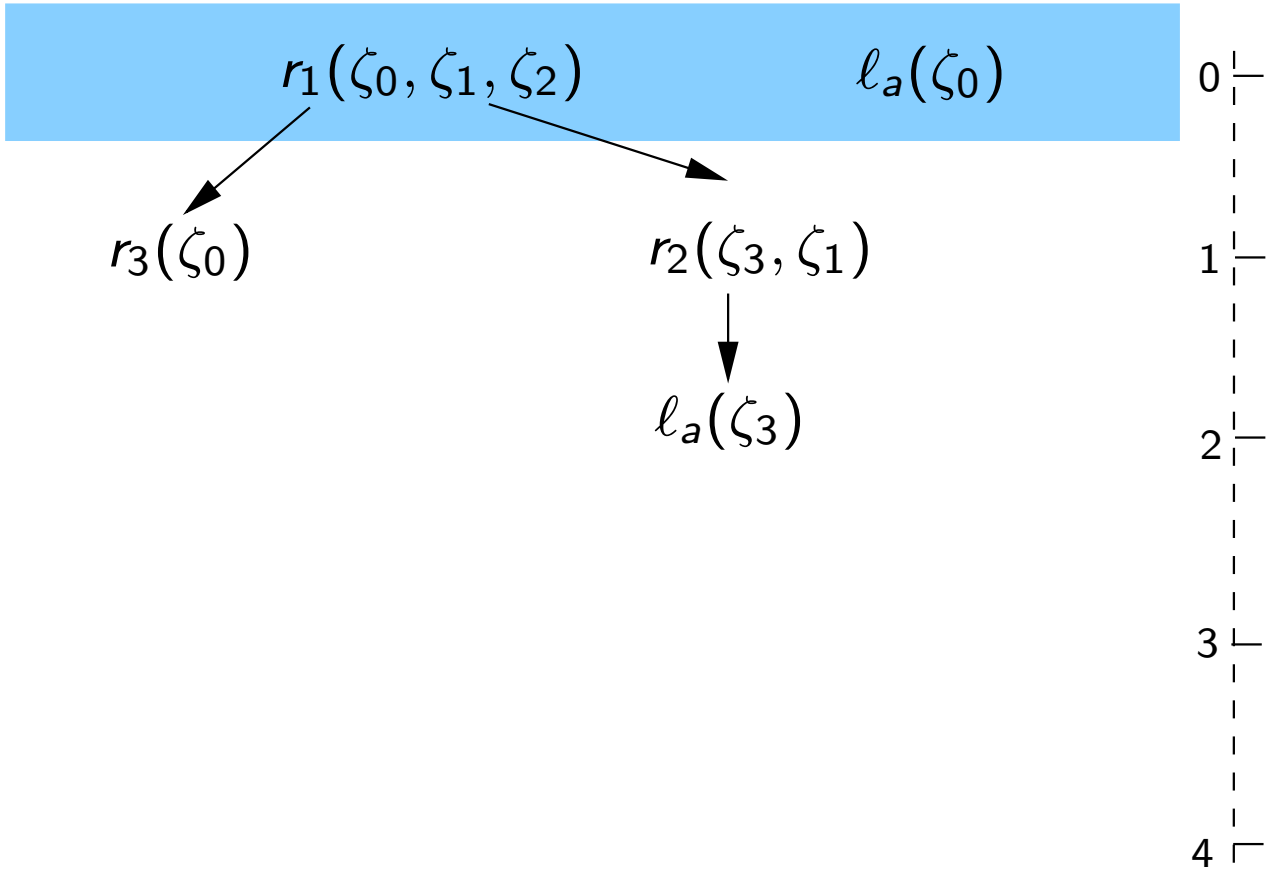
Frozen body

$$r_1(\zeta_0, \zeta_1, \zeta_2), \ell_a(\zeta_0)$$

Example of instance in the crayfish-chase



Example: another instance in the crayfish-chase



Main property of the crayfish-chase

Theorem

$q_1 \subseteq_{\Lambda,I} q_2$ if and only if for every database $C \in \text{cchase}(q_1, \mathcal{S}, I)$

$$\text{frozen_head}(q_1) \in q_2(C)$$

(frozen_head(q_1) is the same in every DB in the chase)

Main property of the crayfish-chase

Theorem

$q_1 \subseteq_{\Lambda,I} q_2$ if and only if for every database $C \in \text{cchase}(q_1, \mathcal{S}, I)$

$$\text{frozen_head}(q_1) \in q_2(C)$$

(frozen_head(q_1) is the same in every DB in the chase)

Warning

No indication of a strategy for deciding containment!

- it is possible that there is an infinite number of databases in a chase
- there is no bound on the size of databases in the chase

Decidability

Theorem

IF there exists a finite database $C \in \text{cchase}(q_1, \mathcal{S}, I)$ such that $q_1(C) \not\subseteq q_2(C)$,

THEN there exists another finite database $C' \in \text{cchase}(q_1, \mathcal{S}, I)$ such that

1. $q_1(C') \not\subseteq q_2(C')$, and
2. C' has maximum level $\delta = 2 \cdot |\mathcal{S}| + |q_2| - 3$

Decidability

Theorem

IF there exists a finite database $C \in \text{cchase}(q_1, \mathcal{S}, I)$ such that $q_1(C) \not\subseteq q_2(C)$,

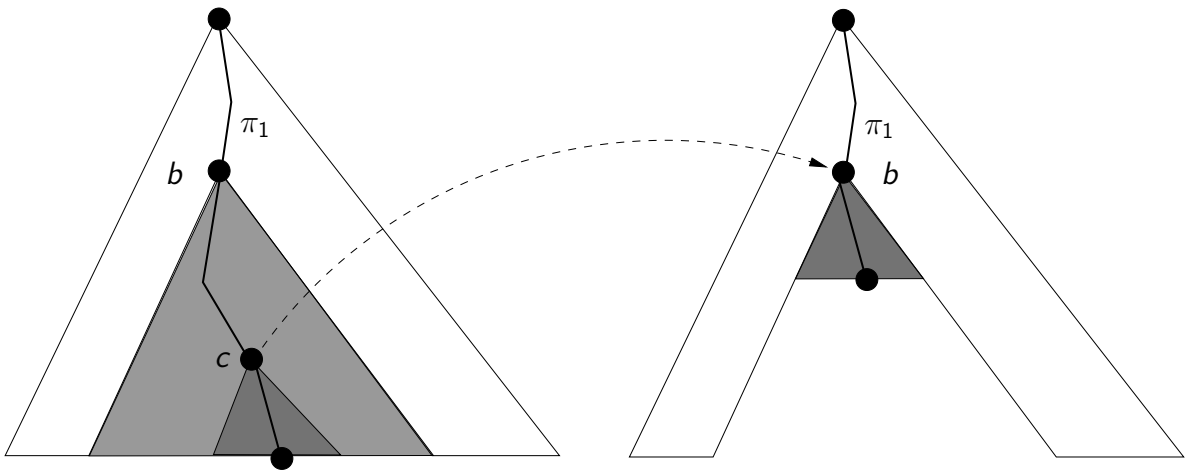
THEN there exists another finite database $C' \in \text{cchase}(q_1, \mathcal{S}, I)$ such that

1. $q_1(C') \not\subseteq q_2(C')$, and
2. C' has maximum level $\delta = 2 \cdot |\mathcal{S}| + |q_2| - 3$

Consequence

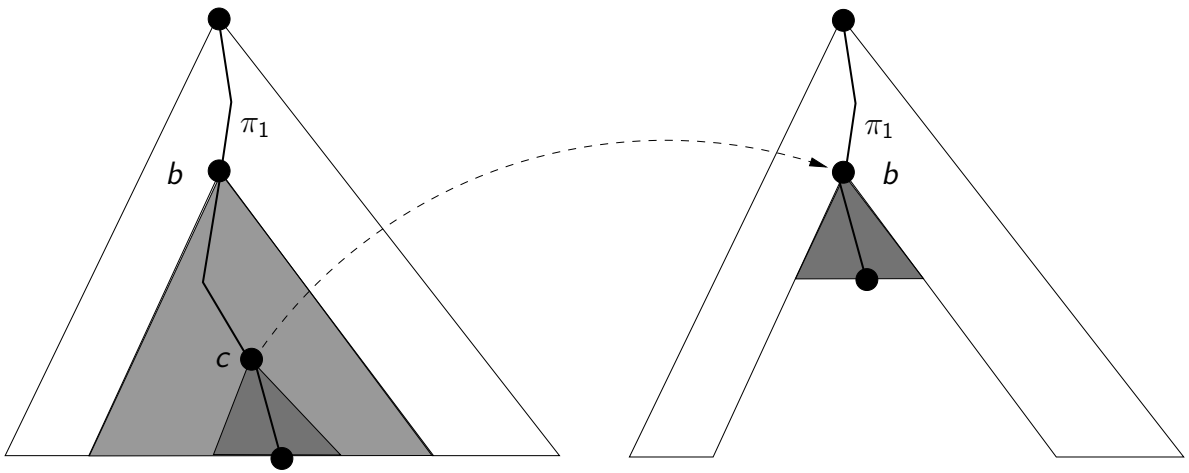
It is sufficient to check all databases in the chase up to a certain number of levels

Idea of the proof: iterative subtree replacement



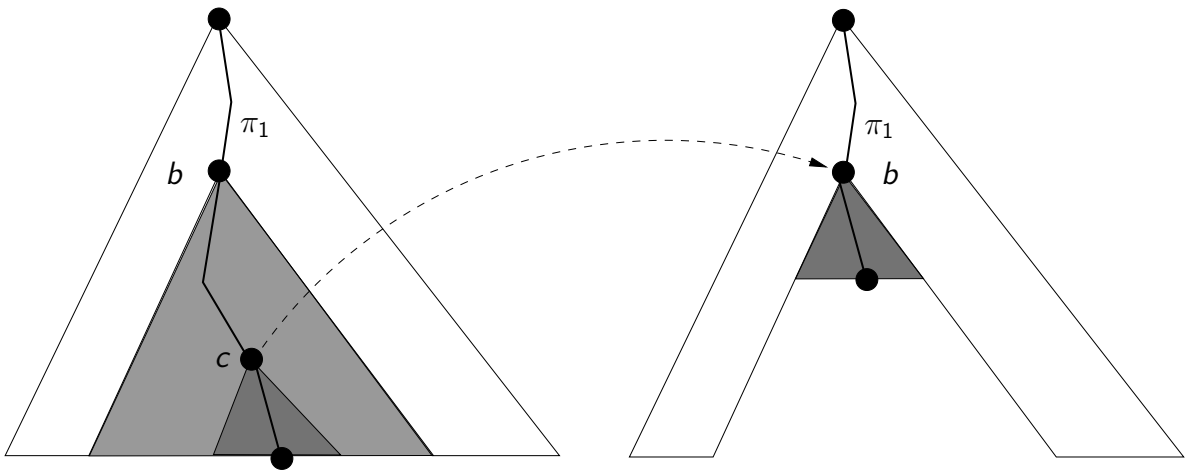
- Take a counterexample C in $\text{cchase}(q_1, \mathcal{S}, I)$

Idea of the proof: iterative subtree replacement



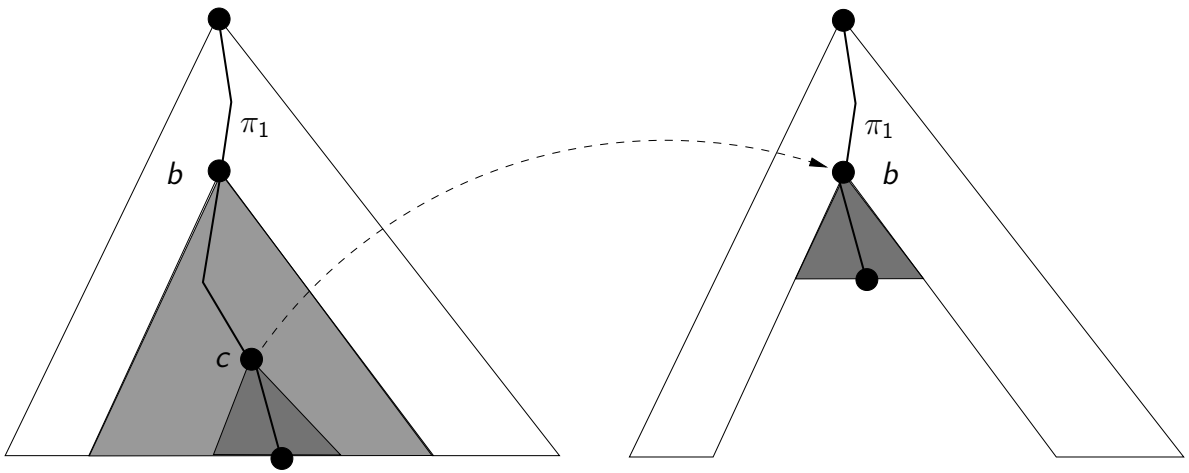
- Take a counterexample C in cchase$(q_1, \mathcal{S}, I)$
- if C exceeds the level δ, "shorten" it

Idea of the proof: iterative subtree replacement



- Take a counterexample C in cchase$(q_1, \mathcal{S}, I)$
- if C exceeds the level δ, "shorten" it
- C is "shortened" by tree replacement

Idea of the proof: iterative subtree replacement



- Take a counterexample C in cchase$(q_1, \mathcal{S}, I)$
- if C exceeds the level δ, "shorten" it
- C is "shortened" by tree replacement
- at each replacement, we get another counterexample

Complexity

Theorem

The complexity of checking containment of conjunctive queries under access limitations is in co-NEXPTIME.

Complexity

Theorem

The complexity of checking containment of conjunctive queries under access limitations is in co-NEXPTIME.

Proof sketch

1. guess $C \in \text{cchase}(q_1, \mathcal{S}, I)$ of depth less than the sufficient one; size W^δ (W: max. arity)
2. evaluate q_2 over C: feasible in polynomial time in C and det. exp. time in q_2
3. if no counterexample to containment is found, then containment holds (otherwise containment does not hold)

Conclusions

- Query Containment under relational constraints
  - decidable cases (relevant to practice)
  - tight complexity bounds
  - applications to answering queries on incomplete data
- Conjunctive query containment under access limitations
  - containment of special Datalog queries
  - notion of crayfish-chase
  - complexity (upper bound)

Conclusions

- Query Containment under relational constraints
  - decidable cases (relevant to practice)
  - tight complexity bounds
  - applications to answering queries on incomplete data
- Conjunctive query containment under access limitations
  - containment of special Datalog queries
  - notion of crayfish-chase
  - complexity (upper bound)

Acknowledgments

Thanks to:

- Georg Gottlob
- Davide Martinenghi
- ESPRC Data Exchange project (UK)
- ...

Acknowledgments

Thanks to:

- Georg Gottlob
- Davide Martinenghi
- ESPRC Data Exchange project (UK)
- ...