Determining Relevance of Accesses at Runtime

Michael Benedikt, Georg Gottlob,
Pierre Senellart

Free University of Bozen–Bolzano, 27 January 2011

The deep Web

Definition (Deep Web, Hidden Web, Invisible Web)

All the content on the Web that is not directly accessible through hyperlinks. In particular: HTML forms, Web services.

Size estimate: 500 times more content than on the surface Web! [Bergman, 2001]. Hundreds of thousands of deep Web databases [He et al., 2007]

Querying the deep Web

- A large part of deep Web data (phone directories, library catalogs, etc.) is essentially relational
- Access to deep Web necessary goes through restricted query interfaces, named here access methods
- Typically: for a given form interface to relational data, some input attributes must be bound, other attributes are free
- Given a query (say, conjunctive) over base relations, answering it using restricted interfaces may 1) not be possible 2) require an unbounded number of calls to query interfaces
- Large body of work on the computation of static query plans under access limitations [Rajaraman et al., 1995, Duschka and Levy, 1997, Li, 2003, Nash and Ludäscher, 2004, Calì and Martinenghi, 2008b]: not our concern here

When is an access relevant?

Consider:

- a schema $\mathcal{S}$, with access methods for schema relations
- a query Q over $\mathcal{S}$
- some pre-existing knowledge Conf of the content of relations of $\mathcal{S}$
- an access method over a base relation $R \in \mathcal{S}$, and a binding $\vec{b}$ of the input attributes to constants; the corresponding access is denoted $R(\vec{b}, ? \ldots ?)$ (or $R(\vec{b})?$ if there are only input attributes)

We want to know if $R(\vec{b}, ? \ldots ?)$ is relevant to Q in Conf, i.e., if it may bring us knowledge on the truth value of Q.

Simplifying assumption: no views, accesses are directly on base relations

Motivating example

Schema (input attributes in blue)

```
Employee(EmpId, Title, LastName, FirstName, OffId)
Office(OffId, StreetAddress, State, Phone)
Approval(State, Offering)
Manager(EmpId, EmpId)
```

Query

```
SELECT DISTINCT 1 FROM Employee E, Office O, Approval A
WHERE E.Title='loan officer' AND E.OffId=O.OffId
  AND O.State='Illinois' AND A.State='Illinois' AND A.Offering='30'
```

Is the access "Manager(12345,?)" relevant to the query?

Motivating example

Schema (input attributes in blue)

```
Employee(EmpId, Title, LastName, FirstName, OffId)
Office(OffId, StreetAddress, State, Phone)
Approval(State, Offering)
Manager(EmpId, EmpId)
```

Query

```
SELECT DISTINCT 1 FROM Employee E, Office O, Approval A
WHERE E.Title='loan officer' AND E.OffId=O.OffId
  AND O.State='Illinois' AND A.State='Illinois' AND A.Offering='30'
```

Is the access “Manager(12345,?)” relevant to the query?

Different notions of relevance

The relevance of a =“Manager(12345,?)” depends on several factors:

Initial configuration If we already know of a loan officer in Illinois, a is not relevant. Otherwise, it might be.

Dependence of accesses If it is possible to “guess” employee ids at random (independent accesses), a is not relevant. If all employee ids used must appear as the result of a previous access (dependent accesses), a may be relevant.

Immediate and long-term relevance By itself, a cannot make the query true if it was not true already: it is not immediately relevant. But it may provide employee ids that will be used to build a witness to the query, i.e., it is long-term relevant.

Problem studied

Algorithms for, complexity of determining if an access is relevant to a query in a given configuration:

- independent vs dependent case
- immediate relevance vs long-term relevance
- query language

We focus on combined complexity, but we also present data complexity results.

Outline

Framework

We assume given:

- a relational schema $\mathcal{S} = \{S_1 \dots S_n\}$ (each attribute has an abstract domain);
- a set of access methods $\mathcal{A} = \{A_1 \dots A_m\}$ where each A_i is the given of:
  1. one relation S_i of $\mathcal{S}$
  2. a subset of the attributes of S_i that are input attributes
  3. either of the dependent or independent types

Configurations and accesses

- A configuration Conf is an instance of the relational schema.
- Given a configuration Conf, a well-formed access a is the given of:
  - an access method A_k
  - an assignment of input attributes of A_k to constants such that either:
    - A_k is independent
    - or all values of the binding are constants of Conf of the proper domain
- A configuration Conf and a well-formed access a leads (non-deterministically) to a new configuration Conf′ with:
  1. $\mathsf{Conf} \subseteq \mathsf{Conf}'$
  2. $\mathsf{Conf}' - \mathsf{Conf}$ only contains tuples of the accessed relation, and all these tuples agree with the binding

Configuration paths

A well-formed path between configurations Conf and Conf′ is a sequence of configurations

$$(\mathsf{Conf} =)\mathsf{Conf}_0 \rightarrow^{a_1} \mathsf{Conf}_1 \rightarrow^{a_2} \dots \quad \mathsf{Conf}_{n-1} \rightarrow^{a_n} \mathsf{Conf}_n (= \mathsf{Conf}')$$

such that for all $i \geqslant 1$, a_i is a well-formed access that leads from Conf_{i-1} to Conf_i. We say Conf′ is reachable from Conf.

The truncation of this path is the path

$$(\mathsf{Conf} =)\mathsf{Conf}_0 \rightarrow^{a_2} \mathsf{Conf}'_2 \rightarrow^{a_3} \dots \quad \mathsf{Conf}_{n-1} \rightarrow^{a_k} \mathsf{Conf}'_k$$

with k maximum such that the path is still well-formed, and Conf'_i contains all facts of Conf_i except those produced by a_1.

Queries

- Only Boolean queries
- Two query languages, subsets of the relational calculus:

  Conjunctive queries (CQs) $\exists, \wedge$

  Positive queries (PQs) $\exists, \wedge, \vee$

- Queries should be consistent with attribute domains
- Constants in the query are assumed to also be part of the configuration
- We note $\mathsf{Conf} \models Q$ when Q is true in Conf

Immediate and long-term relevance

Query Q, configuration Conf, access a.

- a is immediately relevant (IR) for Q in Conf if there exists a configuration Conf$'$ such that:
  - a may lead from Conf to Conf$'$
  - Conf $\not\models Q$
  - Conf$' \models Q$
- a is long-term relevant (LTR) for Q in Conf if there exists a well-formed path p starting with a and leading to some Conf$'$, whose truncation p' leads from Conf to Conf$''$ such that:
  - Conf$' \models Q$
  - Conf$'' \not\models Q$

Simple example

Example

$Q = R(x, y) \wedge S(y, z)$. Conf $= \varnothing$. $a = R(?, ?)$. Access method on S.

- a is not IR for Q in Conf.
- a is LTR for Q in Conf.

First observations

- For a fixed arity k, relevance for a query of output arity k reduces to relevance for Boolean queries.
- Determining relevance for Q in Conf requires checking that $\text{Conf} \not\models Q$, which is coNP-hard for CQs.

Outline

Containment under access limitations

Schema $\mathcal{S}$, set of access methods $\mathcal{A}$, configuration Conf.

Definition

Query Q_1 is contained in Q_2 under $\mathcal{A}$ starting from Conf, denoted $Q_1 \sqsubseteq_{\mathcal{A},\mathsf{Conf}} Q_2$ if for every configuration Conf′ reachable from Conf,

$$\mathsf{Conf}' \models Q_1 \Rightarrow \mathsf{Conf}' \models Q_2.$$

Notion introduced (in a restricted form) in [Calì and Martinenghi, 2008a], shown to be coNEXPTIME for conjunctive queries. No lower bound given.

Access limitations impact containment

Example ([Calì and Martinenghi, 2008a])

- $Q_1 = R(x),\ Q_2 = S(x)$
- Dependent access methods $\mathcal{A} = \{R(\cdot)?, S(?)\}$
- $Q_1 \sqsubseteq_{\mathcal{A},\varnothing} Q_2$ while $Q_1 \not\sqsubseteq Q_2$.

From containment to relevance

Let $\mathcal{Q}$ be one of CQs, PQs.

Proposition

There is a polynomial-time many-one reduction from query containment of queries in $\mathcal{Q}$ under access limitations to the complement of LTR of dependent accesses for queries in $\mathcal{Q}$.

Immediate application: LTR is Σ_2^{P}-hard for PQs.

From relevance to containment

Proposition

There is a reduction from LTR of dependent accesses to the complement of query containment, which is:

- *a polynomial-time many-one reduction for PQs;*
- *a nondeterministic polynomial-time Turing reduction for CQs.*

The weaker form of reduction comes from the need for disjunction.
Enough to show matching complexity results for containment and LTR in most cases.

Outline

Immediate relevance

Proposition

IR for CQs or PQs is DP-complete in combined complexity. If the query is fixed, the problem is in AC^0.

Proof sketch.

Upper bound: the problem is shown to be in NP (by a short-witness argument) as soon as the query is known not to be true.

Lower bound: coding of satisfiability/unsatisfiability pair as a single query.

Data complexity: the algorithm can be implemented as a first-order formula.

Immediate relevance

Proposition

IR for CQs or PQs is DP-complete in combined complexity. If the query is fixed, the problem is in AC^0.

Proof sketch.

Upper bound: the problem is shown to be in NP (by a short-witness argument) as soon as the query is known not to be true.

Lower bound: coding of satisfiability/unsatisfiability pair as a single query.

Data complexity: the algorithm can be implemented as a first-order formula.

Long-term relevance

Proposition

In the absence of dependent accesses, the combined complexity of LTR for CQs or PQs is Σ_2^P*-complete. If the query is fixed, the problem is in* AC^0*.*

Proof sketch.

The upper bound is straightforward. The lower bound is a consequence of a the hardness of determining whether a tuple is critical for a query in a relational database [Miklau and Suciu, 2007]. □

Long-term relevance

Proposition

In the absence of dependent accesses, the combined complexity of LTR for CQs or PQs is Σ_2^P-complete. If the query is fixed, the problem is in AC^0.

Proof sketch.

The upper bound is straightforward. The lower bound is a consequence of a the hardness of determining whether a tuple is critical for a query in a relational database [Miklau and Suciu, 2007]. □

Outline

Immediate relevance

Only one access involved in immediate relevance: dependence does not play a role. We are still DP-complete in combined complexity, and AC^0 in data complexity.

Long-term relevance

- Naïve idea: a witness path can be shortened by generating all needed constants in the initial access. Fails for accesses with only input attributes, or in the presence of domain constraints.
- Upper bound arguments: show that the access path must be tree-like [Calì and Martinenghi, 2008a, Chaudhuri and Vardi, 1997] (non trivial)
- Lower bound arguments: reduction from corridor tiling [Johnson, 1990] (non trivial either!)
- For conjunctive queries: additional trick needed to code Boolean operation with their truth values

Complexity results

Theorem

- *LTR for CQs is NEXPTIME-complete in combined complexity.*
- *LTR for PQs is 2NEXPTIME-complete in combined complexity.*
- *LTR for PQs is PTIME if the query is fixed.*

Outline

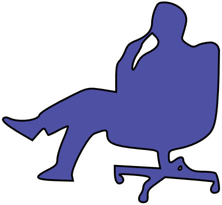
In brief

- Strong connection between long-term relevance and containment under access limitations
- Combined complexity:

	IR	LTR	Containment
Indep. accesses (CQs)	DP-c	Σ_2^P-c	coNP-c
Indep. accesses (PQs)	DP-c	Σ_2^P-c	Π_2^P-c
Dep. accesses (CQs)	DP-c	NEXPTIME-c	coNEXPTIME-c
Dep. accesses (PQs)	DP-c	2NEXPTIME-c	co2NEXPTIME-c

- Data complexity: everything is in PTIME (AC^0 for independent accesses).
- Not presented: a number of simpler cases (low arity, non-repeated relations, etc.)

Perspectives



- Extension to other query languages, especially UCQs
- Adding views, integrity constraints, and exactness constraints to the setting
- Other notions of relevance:
  - LTR: ∃ an instance, ∃ a path, such that the query is true after the path and not after the truncation of the path
  - ∃ an instance, ∀ paths such the query is true after the path, it is not after the truncation of the path
  - ∀ instances, ∃ a path, such that the query is true after the path and not after the truncation of the path

Merci.

Michael K. Bergman. The deep Web: Surfacing hidden value. *J. Electronic Publishing*, 7, 2001.

Andrea Calì and Davide Martinenghi. Conjunctive query containment under access limitations. In *ER*, 2008a.

Andrea Calì and Davide Martinenghi. Querying data under access limitations. In *ICDE*, 2008b.

Surajit Chaudhuri and Moshe Y. Vardi. On the equivalence of recursive and nonrecursive Datalog programs. *JCSS*, 54(1):61–78, 1997.

Oliver M. Duschka and Alon Y. Levy. Recursive plans for information gathering. In *IJCAI*, 1997.

Bin He, Mitesh Patel, Zhen Zhang, and Kevin Chen-Chuan Chang. Accessing the deep Web: A survey. *Communications of the ACM*, 50(2):94–101, 2007.

David S. Johnson. A catalog of complexity classes. In *Handbook of Theoretical Computer Science*. MIT Press, 1990.

Chen Li. Computing complete answers to queries in the presence of limited access patterns. *VLDB J.*, 12(3):211–227, 2003.

Gerome Miklau and Dan Suciu. A formal analysis of information disclosure in data exchange. *JCSS*, 73(3):507–534, 2007.

Alan Nash and Bertram Ludäscher. Processing union of conjunctive queries with negation under limited access patterns. In *EDBT*, 2004.

Anand Rajaraman, Yehoshua Sagiv, and Jeffrey D. Ullman. Answering queries using templates with binding patterns. In *PODS*, 1995.