Determining Relevance of Accesses at Runtime

Michael Benedikt, Georg Gottlob, Pierre Senellart
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

When is an access relevant?

Consider:

- a schema $S$, with access methods for schema relations
- a query $Q$ over $S$
- some pre-existing knowledge Conf of the content of relations of $S$
- an access method over a base relation $R \in S$, 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 $Q$. 
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?
The relevance of \( a = \text{Manager}(12345,?) \) 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

Relevance of an Access

Formal Definitions

Relevance and Query Containment

Independent Accesses

Dependent Accesses

Conclusion
Framework

We assume given:

- a relational schema $S = \{S_1 \ldots S_n\}$ (each attribute has an abstract domain);
- a set of access methods $A = \{A_1 \ldots A_m\}$ where each $A_i$ is the given of:
  1. one relation $S_i$ of $S$
  2. a subset of the attributes of $S_i$ that are input attributes
  3. either of the dependent or independent types
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. $Conf \subseteq Conf'$
2. $Conf'$ — $Conf$ only contains tuples of the accessed relation, and all these tuples agree with the binding
Configuration paths

A well-formed path between configurations $\text{Conf}$ and $\text{Conf}'$ is a sequence of configurations

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

such that for all $i \geq 1$, $a_i$ is a well-formed access that leads from $\text{Conf}_{i-1}$ to $\text{Conf}_i$. We say $\text{Conf}'$ is reachable from $\text{Conf}$.

The truncation of this path is the path

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

with $k$ maximum such that the path is still well-formed, and $\text{Conf}'_i$ contains all facts of $\text{Conf}_i$ except those produced by $a_1$. 
Queries

- Only Boolean queries
- Two query languages, subsets of the relational calculus:
  
  \begin{align*}
  \text{Conjunctive queries (CQs)} & \exists, \land \\
  \text{Positive queries (PQs)} & \exists, \land, \lor
  \end{align*}

- Queries should be consistent with attribute domains
- Constants in the query are assumed to also be part of the configuration
- We note $\text{Conf} \models Q$ 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 $Conf'$
  - $Conf \not|= Q$
  - $Conf' |= Q$

- $a$ 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:
  - $Conf' |= Q$
  - $Conf'' \not|= Q$
Example

\[ Q = R(x, y) \land S(y, z). \ Conf = \emptyset. \ a = R(?, ?). \ Access \ method \ on \ S. \]

- \( a \) is \textbf{not IR} for \( Q \) in \( Conf \).
- \( a \) is \textbf{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 $\text{Conf} \nmid Q$, which is coNP-hard for CQs.
Outline

Relevance of an Access

Formal Definitions

Relevance and Query Containment

Independent Accesses

Dependent Accesses

Conclusion
Schema $S$, set of access methods $A$, configuration $Conf$.

**Definition**

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

$$Conf' \models Q_1 \Rightarrow Conf' \models Q_2.$$ 

Notion introduced (in a restricted form) in [Calì and Martinenghi, 2008a], shown to be coNEXPTIME for conjunctive queries. No lower bound given.
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 \( Q_1 \not\sqsubseteq Q_2 \).
Let \( Q \) be one of CQs, PQs.

**Proposition**

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

Immediate application: LTR is \( \Sigma^P_2 \)-hard for PQs.
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. It will be enough to show matching complexity results for containment and LTR.
Outline

Relevance of an Access

Formal Definitions

Relevance and Query Containment

Independent Accesses

Dependent Accesses

Conclusion
Proposition

The combined complexity of IR for CQs or PQs is DP-complete. 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

The combined complexity of IR for CQs or PQs is $\text{DP}$-complete. If the query is fixed, the problem is in $\text{AC}^0$.

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.
Proposition

In the absence of dependent accesses, the combined complexity of LTR for CQs or PQs is $\Sigma_2^P$-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 the hardness of determining whether a tuple is critical for a query in a relational database [Miklau and Suciu, 2007].
Proposition

In the absence of dependent accesses, the combined complexity of LTR for CQs or PQs is $\Sigma_2^P$-complete. If the query is fixed, the problem is in $\mathcal{AC}^0$.

Proof sketch.

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

Relevance of an Access

Formal Definitions

Relevance and Query Containment

Independent Accesses

Dependent Accesses

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

- The combined complexity of LTR for CQs is $\text{NEXPTIME}$-complete.
- The combined complexity of LTR for PQs is $2\text{NEXPTIME}$-complete.
- LTR for PQs is polynomial-time if the query is fixed.
Outline

Relevance of an Access

Formal Definitions

Relevance and Query Containment

Independent Accesses

Dependent Accesses

Conclusion
In brief

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

- Combined complexity:

<table>
<thead>
<tr>
<th></th>
<th>IR</th>
<th>LTR</th>
<th>Containment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indep. accesses (CQs)</td>
<td>DP-c</td>
<td>$\Sigma^p_2$-c</td>
<td>coNP-c</td>
</tr>
<tr>
<td>Indep. accesses (PQs)</td>
<td>DP-c</td>
<td>$\Sigma^p_2$-c</td>
<td>$\Pi^p_2$-c</td>
</tr>
<tr>
<td>Dep. accesses (CQs)</td>
<td>DP-c</td>
<td>NEXPTIME-c</td>
<td>coNEXPTIME-c</td>
</tr>
<tr>
<td>Dep. accesses (PQs)</td>
<td>DP-c</td>
<td>2NEXPTIME-c</td>
<td>co2NEXPTIME-c</td>
</tr>
</tbody>
</table>

- Data complexity: everything is in $P$ ($AC^0$ 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**: \( \exists \) an instance, \( \exists \) a path, such that the query is true after the path and not after the truncation of the path
  - \( \exists \) an instance, \( \forall \) paths such the query is true after the path, it is not after the truncation of the path
  - \( \forall \) instances, \( \exists \) a path, such that the query is true after the path and not after the truncation of the path
Merci.


