

# Determining Relevance of Accesses at Runtime

Michael Benedikt  
Computing Laboratory  
Oxford University  
Oxford OX1 3QD, UK  
michael.benedikt@comlab.ox.ac.uk

Georg Gottlob  
Computing Laboratory  
& Oxford Man Institute  
Oxford University  
georg.gottlob@comlab.ox.ac.uk

Pierre Senellart  
Institut Télécom; Télécom ParisTech  
CNRS LTCI, 46 rue Barrault  
75634 Paris, France  
pierre.senellart@telecom-paristech.fr

## ABSTRACT

Consider the situation where a query is to be answered using Web sources that restrict the accesses that can be made on backend relational data by requiring some attributes to be given as input of the service. The accesses provide lookups on the collection of attributes values that match the binding. They can differ in whether or not they require arguments to be generated from prior accesses. Prior work has focused on the question of whether a query can be answered using a set of data sources, and in developing static access plans (e.g., Datalog programs) that implement query answering. We are interested in dynamic aspects of the query answering problem: given partial information about the data, which accesses could provide relevant data for answering a given query? We consider immediate and long-term notions of “relevant accesses”, and ascertain the complexity of query relevance, for both conjunctive queries and arbitrary positive queries. In the process, we relate dynamic relevance of an access to query containment under access limitations and characterize the complexity of this problem; we produce several complexity results about containment that are of interest by themselves.

## Categories and Subject Descriptors

H.2.3 [Database Management]: Logical Design, Languages—*data models, query languages*; F.2.0 [Analysis of Algorithms and Problem Complexity]: General

## General Terms

Algorithms, Theory

## Keywords

Access patterns, binding patterns, deep Web, relevance

## 1. INTRODUCTION

*Relevance under access limitations.* A large part of the information on the World Wide Web is not available through

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

*PODS'11*, June 13–15, 2011, Athens, Greece.

Copyright 2011 ACM 978-1-4503-0660-7/11/06 ...\$10.00.

the *surface Web*, the set of Web pages reachable by following hyperlinks, but lies in the *deep Web* (or *hidden Web*), that provides entry points to databases accessible via HTML forms or Web services. Hundreds of thousands of such deep-Web sources exist [16]. Even when the information is available on the surface Web, it can be more effective to access it through a (possibly elaborate) Web form query. Each source of the deep Web has one or several interfaces that limit the kind of accesses that can be performed, e.g., some fields of the forms have to be filled in before submission.

A number of works, e.g. [25, 13], have dealt with the problems of answering queries using views in the presence of such access restrictions but the focus is usually on obtaining a static query plan (e.g., a rewriting of an original conjunctive query, or a Datalog program). We consider a dynamic approach to query answering and study the following problem: given some existing knowledge about the data, knowledge that is bound to evolve as we access sources, is making this particular access relevant to a query? In other words, can this particular access give, immediately or after some other accesses, some knowledge that will yield an answer to the query?

Let us consider the following example. A user wishes to get information about the loan capabilities of a large bank. A relation schema for this can be visualized as:

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

**Employee** stores information about employees, including their title and office. **Office** stores information about offices, including the state in which they are located. **Approval** tells which kinds of loans a bank is approved to make in each state, while **Manager** stores which employee manages which other employee.

Data from a number of distinct Web data sources (or distinct query interfaces from the same source) can be used to answer the query:

- a form **EmpOffAcc** where an **EmpId** can be entered, which returns office records for that employee;
- a form **EmpManAcc** where an **EmpId** can be entered, and the identifiers of their managers are returned;
- a form **OfficeInfoAcc** that allows one to enter an **OffId**, and returns all the office information;
- a form **StateApprAcc** that allows one to enter a state, and returns the approval information for that state.

A user wishes to know if there is some loan officer for their bank located in Illinois, and also whether the company is

authorized to perform 30-year mortgages in Illinois. This can be phrased as a Boolean query  $Q$ , expressible in SQL as follows:

```
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='30yr'
```

A federated query engine that tries to answer this query will require querying the distinct interfaces with concrete values. At a certain stage the query engine may know about a particular set of employee names, and also know certain facts – either via querying or via an additional knowledge base. Which interfaces should it use to answer the query? In particular: Is an access to the `EmpManAcc` form with `EmpId` “12345” useful for answering  $Q$ ? There are actually a number of subtleties in this question, that we discuss now.

*The relevance of an access depends on the existing knowledge base.* At the beginning of the process, when no other information is known about the data, the access might be useful to get some other `EmpId` which may in turn be used in the `EmpOffAcc` interface to find an Illinoisan loan officer. But if we already know that the company has a loan officer located in Illinois, then clearly such an access is unnecessary. We call this existing knowledge base the *configuration* in which the access is made.

*The relevance depends on how closely linked the Web forms are.* Clearly the interface is irrelevant to the query if the query engine is free to enter `EmpId` values “at random” into the `EmpOffAcc` interface. But if such values are widely dispersed and there is no way to guess them, an efficient tactic might be to take `EmpId`’s that we know about, query for their managers using the `EmpManAcc` interface, and then use the resulting offices in the `OfficeInfoAcc` interface. In this work we will thus distinguish between accesses that require a value that is already in the knowledge base of the engine (*dependent* accesses), from those that allow a “free guess”. Note that in the case of *static* query answering plans, the notion of a “free access” trivializes the questions.

*The relevance depends on whether one is interested in immediate or long-term impact.* Without any initial knowledge, there is no way an access to `EmpOffAcc` may directly provide a witness for the query. On the other hand, as discussed above, the result to an access may be used in some cases to gain some information that will ultimately satisfy the query. In this work, we consider both *immediate relevance* and *long-term relevance* of a particular access.

**Main questions studied.** In this article, we are interested in the following problems:

- (i) How to define a model for querying under access restrictions that takes into account the history of accesses?
- (ii) What is the complexity of relevance?
- (iii) Calì and Martinenghi have studied in [5] the complexity of *containment under access constraints*, motivated by query optimization. How does relevance relate to containment? Are these notions at all related, and if so, can the respective decision problems be transformed into one another?
- (iv) What is the complexity of containment under access constraints?

- (v) If problems are hard, can we identify the source of this complexity?

One particular reason why these problems are challenging is that they do not deal with a concrete database, but a virtual database of which we have a partial view, a view that evolves as we access it. The notion of relevance of accesses has not been investigated in the literature; the closest work, on containment under access constraints [5], only provides an upper bound of `coNEXPTIME`, for a restricted query language (conjunctive queries, with only limited use of constants). Determining a lower bound for containment was left as an open problem. Hardness results are difficult to obtain, because the access model that we present is quite simple and does not offer obvious clues of how to encode known hard problems to get lower bounds.

**Results.** We emphasize the following contributions of our work.

We provide the first formal definition of dynamic relevance of accesses for a query  $Q$ , using a simple and powerful model, answering thus item (i).

We give a combined complexity characterization of the relevance problem in all combinations of cases (immediate or long-term relevance, independent or dependent accesses, conjunctive or positive queries), inside the polynomial and exponential hierarchy of complexity classes; for long-term relevance, we mostly focus on accesses without any input, extension to arbitrary accesses is left for future work. This gives a satisfactory answer to question (ii). For several of our hardness results, we invented sophisticated coding techniques to enforce database accesses to produce grids that would then allow us to encode tiling problems. One particular hurdle to overcome was the limited “coding power” of conjunctive queries. We therefore had to use and extend techniques for encoding disjunctions into a conjunctive query.

We exhibit reductions in both directions between dynamic relevance and containment under access constraints. By these results, we succeed in elucidating the relationship between containment and long term relevance, thus providing an exhaustive answer to item (iii).

We generalize the `coNEXPTIME` upper bound to a stronger notion of containment, and provide a matching lower bound, solving thus item (iv). This `coNEXPTIME` upper bound for containment, and the associated `NEXPTIME` upper bound for relevance are rather surprising and not at all obvious. In fact, the more immediate upper bounds, that we show for positive queries, are `co2NEXPTIME` and `2NEXPTIME`, respectively.

We highlight specific cases of interest where the complexity of relevance is lower, e.g., conjunctive queries with a single occurrence of a relation, or conjunctive queries with small arity. We also show that all problems are polynomial-time in data complexity (for the independent case,  $AC^0$ ), suggesting the feasibility of the relevance analysis. These two points together bring a first answer to item (v).

A summary of complexity results is shown in Table 1.

**Organization.** We start with formal definitions of the problem and the various degrees of relevance of a query to an access in Section 2. We next establish (Section 3) the connection between relevance and the topic of containment that was studied in [5]. In Section 4, we study the case of independent accesses (accesses that do not require the input

**Table 1: Summary of combined complexity results**

	Immediate relevance	Long-term relevance (Boolean access)	Containment
Independent accesses (CQs)	DP-complete	$\Sigma_2^P$ -complete	$\Pi_2^P$ -complete
Independent accesses (PQs)	DP-complete	$\Sigma_2^P$ -complete	$\Pi_2^P$ -complete
Dependent accesses (CQs)	DP-complete	NEXPTIME-complete	coNEXPTIME-complete
Dependent accesses (PQs)	DP-complete	2NEXPTIME-complete	co2NEXPTIME-complete

value having been generated by a previous access). Here the access patterns play quite a small role, but relevance is still a non-trivial notion – the issues revolve around reasoning about a very restricted form of query containment. In Section 5 we turn to dependent accesses, where the notion of containment is of primary interest. We extend techniques of [5] to isolate the complexity of containment under access patterns for both conjunctive queries and positive queries; in the process we give the complexity of relevance for both these classes. Related work is discussed in Section 6. Due to space constraints, many proofs are omitted; they can be found in the full paper [3], together with the study of the specific case of relations of small arity, where the complexity of relevance is lower.

## 2. PRELIMINARIES

We use bold face (e.g.,  $\mathbf{a}$ ) to denote sets of attributes or tuples of constants.

*Modeling data sources.* We have a schema  $\text{Sch}$  consisting of a set of relations  $\text{Tables}(\text{Sch}) = \{S_1 \dots S_n\}$ , each  $S_i$  having a set of attributes  $\text{ATT}(S_i)$ . Following [19, 5], we assume each attribute  $a_{ij}$  of relation  $S_i$  has an *abstract domain*  $\text{Dom}(a_{ij})$  chosen in some countable set of abstract domains. Two attributes may share the same domain and different domains may overlap. In the dependent case, domains are used to constrain some input values to come from constants of the appropriate type.

Given a source instance  $\mathbf{l}$  for  $\text{Sch}$ , a *configuration* for  $\mathbf{l}$ , (with respect to  $\text{Sch}$ , when not understood from context) is a subset  $\text{Conf}$  of  $\mathbf{l}$ , that is, for each  $S_i$ , a subset  $\text{Conf}(S_i)$  of the tuples in  $\mathbf{l}(S_i)$  (the content of relation  $S_i$  in  $\mathbf{l}$ ). By a *configuration* we mean any  $\text{Conf}$  that is a configuration for some instance  $\mathbf{l}$ . We then say that a configuration  $\text{Conf}$  is *consistent* with  $\mathbf{l}$  if  $\text{Conf} \subseteq \mathbf{l}$ . Note that a configuration will generally be consistent with many instances (in particular, the empty configuration is consistent with all instances).

We have a set of *access methods*  $\text{ACS} = \{\text{AcM}_1 \dots \text{AcM}_m\}$  with each  $\text{AcM}_i$  consisting of a source relation  $\text{Rel}(\text{AcM}_i)$  and a set  $\text{InputAtt}(\text{AcM}_i)$  of input attributes from the set of attributes of  $\text{Rel}(\text{AcM}_i)$ ; implicitly, each access method allows one to put in a tuple of values for  $\text{InputAtt}(\text{AcM}_i)$  and get as a result a set of matching tuples. If a relation does not have any access methods, no new facts can be learned about this relation: its content is fixed as that of the initial configuration.

Access methods are of two different varieties, based on the values that can be entered into them. An access method may be either *dependent* or *independent*. In a dependent access, one can only use as an input bindings values that have appeared in the configuration in the appropriate domain. An independent access can make use of any value.

A combination of an access method and a binding to the input places of the accessed relation will be referred to as an *access*. We will often write an access by adding “?” to the non-input places, omitting the exact method: e.g.  $R(3, ?)$  is an access (via some method) to  $R$  with the first place bound to 3. If  $R$  does not have any output attributes, we say that it is a *Boolean* access, and we write for instance  $R(3)?$  for an access that checks whether  $3 \in R$ . If  $R$  does not have any input attributes, we say that it is a *free* access. We do not assume access methods to be *exact*, i.e., to return all tuples that are compatible with the binding. They are only assumed to be *sound*, i.e., they can return any sound subset of the data, and possibly a different subset on each use.

Given a set of attributes  $\mathbf{a}$  of a relation  $S_i$ , a database instance  $\mathbf{l}$ , and a binding  $\text{Bind}$  of each attribute in  $\mathbf{a}$  to a value from  $\text{Dom}(\mathbf{a})$ , we let  $\mathbf{l}(\text{Bind}, S_i)$  to be the set of tuples in  $\mathbf{l}$  whose projection onto  $\mathbf{a}$  agrees with  $\text{Bind}$ . For a configuration  $\text{Conf}$ , its *active domain*  $\text{Adom}(\text{Conf}) = \{(c, \mathcal{C})\}$  is the set of constants that appear in a  $\text{Conf}(S_i)$  for some  $i$ , together with their abstract domains: for instance, if  $(c, d) \in \text{Conf}(S)$  and  $\text{Dom}(\text{ATT}(S)) = (\mathcal{C}, \mathcal{D})$ , both  $(c, \mathcal{C})$  and  $(d, \mathcal{D})$  belong to  $\text{Adom}(\text{Conf})$ .

Given a configuration  $\text{Conf}$ , a *well-formed access* consists of an access method  $\text{AcM}$  and an assignment  $\text{Bind}$  of values to the attributes of  $\text{InputAtt}(\text{AcM})$  such that either a)  $\text{AcM}$  is independent; or b)  $\text{AcM}$  is dependent and all values in  $\text{Bind}$ , together with corresponding domains of the input attributes, are in  $\text{Adom}(\text{Conf})$ . A well-formed access  $(\text{AcM}, \text{Bind})$  at configuration  $\text{Conf}$  on instance  $\mathbf{l}$  leads, possibly non-deterministically, to any new configuration  $\text{Conf}'$  in which:

- (i)  $\text{Conf}(\text{Rel}(\text{AcM})) \subseteq \text{Conf}'(\text{Rel}(\text{AcM}))$ ;
- (ii)  $\text{Conf}'(\text{Rel}(\text{AcM})) \subseteq \text{Conf}(\text{Rel}(\text{AcM})) \cup \mathbf{l}(\text{Bind}, \text{Rel}(\text{AcM}))$ ;
- (iii)  $\text{Conf}(S_i) = \text{Conf}'(S_i)$  for all  $S_i \neq \text{Rel}(\text{AcM})$ .

That is, the tuples seen in  $S_i = \text{Rel}(\text{AcM})$  can increase by adding some tuples consistent with the access, the access  $(\text{AcM}, \text{Bind})$  is now completed and every other access stays the same in terms of completion. Note that the new configuration is still consistent with the instance.

In general, there can be many successor configurations. We sometimes write  $\text{Conf} + (\text{AcM}, \text{Bind}, \text{Resp})$  to denote an arbitrary such “response configuration”.

A configuration  $\text{Conf}'$  is *reachable* from another configuration  $\text{Conf}$  (w.r.t. an instance) if there is some sequence of well-formed accesses that can lead from  $\text{Conf}$  to  $\text{Conf}'$ .

*Queries.* We will consider conjunctive queries (CQs), i.e., conjunctions of atomic facts, and positive existential queries, or just positive queries (PQs) for short, i.e., first-order formulas without universal quantifiers or negation. PQs have the inconvenient of being *unsafe* [2] query languages; however, as discussed at the end of this section, we focus on Boolean

queries in this work, for which the problem does not occur. We recall some basic facts about the complexity of these languages: query evaluation over CQs or PQs is NP-complete in combined complexity (membership in NP holds for any existentially quantified first-order query, NP-hardness is a classical result [8]), while the data complexity of evaluating an arbitrary first-order query is  $AC^0$  [2]. On the other hand, the query containment problem is NP-complete for CQs [8], but it is  $\Pi_2^P$ -complete for PQs [26]. We require that variables shared across subgoals of a query are consistent with domain restrictions: if the same variables  $x$  occur in attribute  $a$  of  $R$  and attribute  $a'$  of  $R'$  then  $Dom(a) = Dom(a')$ . The *output domain* of a query  $Q$  is the tuple of domains of the output variables of the query. We also assume that all constants appearing in the query are present in the configuration; in this way, constants from the query can be used in dependent accesses.

The fundamental question we ask in this work is: given a configuration  $Conf$ , which well-formed accesses for that configuration can contribute to answering the query  $Q$ ?

**Immediate relevance.** We begin with analyzing whether a given access can have immediate impact on a query – whether the result of the access can impact the information we have about a query output.

We recall the notion of certain answers, which capture the notion of “information” precisely. Given a configuration  $Conf$  and a tuple  $\mathbf{t}$  of constants from  $Conf$  with the same domain as the output domain of a query  $Q$ , we say that  $\mathbf{t}$  is a *certain answer for  $Q$  at  $Conf$*  if for every instance  $I$  consistent with  $Conf$  we have  $\mathbf{t} \in Q(I)$ . If the query  $Q$  is Boolean (i.e., with no free variables), we say that it is certain (or simply true) in a configuration  $Conf$  if for every instance  $I$  consistent with  $Conf$ ,  $Q(I)$  is true.

We now consider the impact of a new well-formed access  $(AcM, Bind)$  on source  $S$  in a configuration  $Conf$ . The result of this is some new set of tuples  $Resp$  for  $S$ .

Let  $Conf + (AcM, Bind, Resp)$  be a response configuration for the access  $(AcM, Bind)$ . We say the configuration (or even the response  $Resp$ , seen as a collection of tuples) is an *increasing response* for  $Q$  to  $(AcM, Bind)$  in  $Conf$  if there exists a tuple  $\mathbf{t}$  such that  $\mathbf{t}$  is not a certain answer for  $Q$  at  $Conf$  while  $\mathbf{t}$  is a certain answer for  $Q$  at  $Conf + (AcM, Bind, Resp)$ .

An access  $(AcM, Bind)$  is *immediately relevant* for the query  $Q$  (IR in short) in a configuration  $Conf$  if there is some increasing response to the access.

**Long-term impact.** We formalize the notion of an access that can *eventually* yield information.

Given an access  $(AcM, Bind)$ , a *path* from  $(AcM, Bind)$  starting from configuration  $Conf$  on database instance  $I$  is a sequence of configurations and accesses

$$Conf_1, (AcM_1, Bind_1), \dots, (AcM_{n-1}, Bind_{n-1}), Conf_n$$

where  $Conf_1 = Conf$ ,  $(AcM_1, Bind_1) = (AcM, Bind)$ , and  $Conf_{i+1}$  is a successor configuration for access  $(AcM_i, Bind_i)$  on  $Conf_i$ .

By “the certain answers to  $Q$  after  $p$ ” we mean the certain answers to  $Q$  on  $Conf_n$ , where  $p$  terminates in configuration  $Conf_n$ .

Given a path  $p$ , the *truncated path* of  $p$  is the maximal

subpath

$$Conf_1, (AcM_2, Bind_2), Conf'_2, \dots, (AcM_i, Bind_i), Conf'_i$$

such that each  $(AcM_j, Bind_j) : 2 \leq j \leq i$  is a well-formed access at  $Conf'_{j-1}$  (with  $Conf'_1 = Conf_1$ ). That is, we eliminate the initial access in  $p$ , and then find the longest subpath of  $p$  that did not depend on this initial access.

We say that an access  $(AcM, Bind)$  is *long-term relevant (LTR)* for  $Q$  at configuration  $Conf$  if for some instance  $I$  consistent with  $Conf$ , for some path  $p$  beginning with  $(AcM, Bind)$  the certain answers to  $Q$  at  $p$  are different from those at the truncated path of  $p$ .

**EXAMPLE 2.1.** Suppose that we have a schema with relations  $S, T$ , and a query  $Q = S \bowtie T$ . Suppose we have a configuration  $Conf$  in which  $S$  and  $T$  have not yet been accessed, and there is a dependent access method on  $T$ . Now consider an access  $(AcM, Bind)$  on  $S$ . It is long-term relevant for  $Q$ , since it is possible that  $(AcM, Bind)$  returns some new values, and using these values to access  $T$  could yield some new tuples for  $Q$ .  $\square$

When we speak about the problem of “determining whether an access is relevant”, we mean either of the problems IR or LTR.

**The complexity of relevance.** We make a few general observations about the complexity of determining if an access is relevant for a query.

First, we observe that there is a tight relation between the general question of relevance and the special case of Boolean queries. For a number  $k$ , let  $IR(k)$  be the problem of determining whether an access in a given configuration is immediately relevant for a query with output arity  $k$ , relative to a schema, and similarly for LTR. Let  $\mathbf{c}_k$  be a tuple of  $k$  new constant symbols. For any fixed  $k$  we can solve  $IR(k)$  by considering every tuple of items that come either from the configuration or from  $\mathbf{c}_k$  substituting them in for the head of the query and then determining whether the access is IR on the configuration for the Boolean query thus created. This shows:

**PROPOSITION 2.2.** *Let  $k$  be any number. There is a polynomial time reduction from  $IR(k)$  to  $IR(0)$ , and from  $LTR(k)$  to  $LTR(0)$ .*

We will thus focus on the Boolean case  $k = 0$  in this work.

Second, note that checking that an access is relevant, for any of the notion of relevance we have defined, requires that we know that the query is not already satisfied in the configuration, which is  $coNP$ -hard.

### 3. RELEVANCE AND CONTAINMENT

In this section, we introduce the notion of *containment of queries under access limitations* and show how it is strongly related to long-term relevance. We will use this connection to ascertain the complexity of relevance in some cases.

**Containment under access limitations.** Query containment under access limitations was shown to be decidable by Li and Chang [20], and further investigated by Cali and Martinenghi in [5]. We adapt here the definition to our setting, and show further in Proposition 3.6 that the definition of [5] is essentially a special case of ours. We give the definition for queries of arbitrary arity, but as we explained we will focus on the Boolean case further on.

DEFINITION 3.1. Let  $\text{Sch}$  be a schema and  $\text{ACS}$  a set of access methods over  $\text{Sch}$ . Let  $Q_1$  and  $Q_2$  be two queries defined over  $\text{ACS}$  and  $\text{Conf}$  a configuration over  $\text{Sch}$ . We assume  $Q_1$  and  $Q_2$  have the same arity. We say that  $Q_1$  is contained in  $Q_2$  under  $\text{ACS}$  starting from  $\text{Conf}$ , denoted  $Q_1 \sqsubseteq_{\text{ACS}, \text{Conf}} Q_2$ , if for every configuration  $\text{Conf}'$  reachable from  $\text{Conf}$ ,  $Q_1(\text{Conf}') \subseteq Q_2(\text{Conf}')$ . We simply say that  $Q_1$  is contained in  $Q_2$  under  $\text{ACS}$  and note  $Q_1 \sqsubseteq_{\text{ACS}} Q_2$  if  $\text{Conf}$  is the empty configuration.  $\square$

As noted in [5], in the presence of dependent accesses, the notion of query containment under access limitations is strictly weaker than the usual notion of query containment:

EXAMPLE 3.2. Let  $R$  and  $S$  be two unary relations with the same domain, each one with a single dependent access method: Boolean for  $R$ , and free for  $S$ . Consider queries  $Q_1 = \exists x R(x)$  and  $Q_2 = \exists x S(x)$ . Starting from the empty configuration, the only well-formed access paths that make  $Q_1$  true, i.e., produce an  $R(x)$  atom, must first access  $S$  and produce  $S(x)$ . This means that  $Q_1 \sqsubseteq_{\text{ACS}} Q_2$  while, obviously,  $Q_1 \not\sqsubseteq Q_2$ .  $\square$

More generally, many classical results that hold for the classical notion of query containment are not true any more in the presence of access constraints: for instance, query containment of conjunctive queries cannot be tested by the existence of a homomorphism, and query containment of unions of conjunctive queries does not mean that all disjuncts of the first query are contained in some disjunct of the second query, as is true without access constraints [26].

*Relating containment to relevance.* Query containment under access limitations is of interest in its own right, but also for the connection to long-term relevance. We begin by showing that containment under access limitations can be reduced to the complement of long-term relevance.

PROPOSITION 3.3. *There is a polynomial-time many-one reduction from the problem of query containment of Boolean CQs (resp., PQs) under access limitations, starting from a given configuration, to determining whether an access is not long-term relevant to a Boolean CQ (resp., PQ), in another configuration. If the query is a PQ, the configuration can be chosen to be the same.*

PROOF. We give an overview of the main idea, details are in [3]. For positive queries, the proof works by “coding two queries as one disjunction” – we create a query  $\tau(Q, Q')$  and access such that if the access returns successfully, then the query is equivalent to  $Q$ , and otherwise to  $Q'$ . Disjunction can be eliminated by the idea of “coding Boolean operations in relations”, which will be used often in this paper.  $\square$

We can thus prove lower bounds for relevance using lower bounds for containment. As an example, query containment under access limitations obviously covers the classical notion of query containment (just make all access methods free). This immediately entails that long-term relevance is  $\text{coNP}$ -hard for CQs and  $\Sigma_2^P$ -hard for PQs, even if all variables are from the same abstract domain. Conjunctive query containment in the presence of datatype restrictions and fixed relations is  $\Pi_2^P$ -hard (this follows from [26]) and hence containment under access in our setting is  $\Pi_2^P$ -hard. We will show that this latter lower bound actually already holds for conjunctive queries even in very restricted settings (cf. Proposition 4.5).

In the other direction, from relevance to containment, we also have a polynomial-time many-one reduction, but only for positive queries and only for Boolean accesses.

PROPOSITION 3.4. *There is a polynomial-time many-one reduction from the problem of long-term relevance of a Boolean access for a Boolean positive query in a given configuration, to the complement of query containment of Boolean positive queries under access limitations, starting from another configuration.*

PROOF. We assume given a schema  $\text{Sch}$  and a set of access methods  $\text{ACS}$ . Let  $\text{Conf}$  and  $Q$  be, respectively, a configuration and a positive query over  $\text{Sch}$ . We consider an access  $(\text{AcM}, \text{Bind})$  with  $\text{AcM} \in \text{ACS}$ . Let  $R = \text{Rel}(\text{AcM})$ . To simplify the presentation we assume that input attributes of  $\text{AcM}$  come before output attributes.

We add to  $\text{Sch}$  a relation  $\text{IsBind}$  with same arity  $k$  and domain as  $\text{Bind}$ , without any access. We add to  $\text{Conf}$  the single fact  $\text{IsBind}(\text{Bind})$  and let  $\text{Conf}'$  denote the new configuration. We rewrite  $Q$  as  $Q'$  by replacing every occurrence of  $R(i_1 \dots i_k, o_1 \dots o_p)$  with

$$R(i_1 \dots i_k, o_1 \dots o_p) \vee \text{IsBind}(i_1 \dots i_k).$$

Then  $(\text{AcM}, \text{Bind})$  is LTR for  $Q$  in  $\text{Conf}$  if and only if  $Q' \not\sqsubseteq_{\text{ACS}, \text{Conf}'} Q$ .

Assume  $(\text{AcM}, \text{Bind})$  is LTR for  $Q$  in  $\text{Conf}$ . There is a well-formed path  $p$  starting with the access, with truncated path  $p'$ , such that  $Q$  is true in  $\text{Conf} + p$  and false in  $\text{Conf} + p'$  (and, since  $\text{IsBind}$  does not occur in  $Q$ , also false in  $\text{Conf}' + p'$ ). For every subgoal  $R(i_1 \dots i_k, o_1 \dots o_p)$  of  $Q$  that is witnessed by the first access of  $p$ ,  $\text{IsBind}(i_1 \dots i_k)$  is true in  $\text{Conf}'$  and thus a witness that  $Q$  is true in  $\text{Conf} + p$  yields a witness that  $Q'$  is true in  $\text{Conf}' + p'$ .

Conversely, assume there is a path  $p'$  such that  $Q'$  is true and  $Q$  is false in  $\text{Conf}' + p'$ . For every  $R(i_1 \dots i_k, o_1 \dots o_p)$  that is false in  $Q$  while the corresponding disjunction is true in  $Q'$ , we construct a ground fact  $R(\text{Bind}, c_1 \dots c_p)$  where  $(c_1 \dots c_p)$  are the constants that  $(o_1 \dots o_p)$  are mapped to in a witness of  $Q'$ . Then we build a new path  $p$  by prepending to  $p'$  all these ground facts, returned by  $(\text{AcM}, \text{Bind})$ . The path  $p$  witnesses that  $(\text{AcM}, \text{Bind})$  is LTR for  $Q$  in  $\text{Conf}$ .  $\square$

Finally, for conjunctive queries, we prove similarly a different form of reduction, a Turing reduction in nondeterministic polynomial time:

PROPOSITION 3.5. *Long-term relevance of a Boolean access for a CQ can be decided with a nondeterministic polynomial-time algorithm with access to an oracle for query containment of CQs under access limitations.*

Both reductions will be used to show that upper bound results can be lifted from containment to relevance. Even though the latter reduction seems weak, it will be enough for our purpose (see Section 5).

*Containment and containment.* Our notion of query containment under access limitations starting from a given configuration differs in some ways from the notion introduced by Cali and Martinenghi in [5]. In this part of the paper, to emphasize the distinction, we refer to the former as *config-containment* and to the latter as *CM-containment*. The differences are as follows: (i) In *CM-containment*, there is always exactly one access method per regular relation, whereas in *config-containment* there may be zero or several access

methods per relations; (ii) CM-containment is defined with respect to a set of existing constants (of the various abstract domains) that can be used in access paths, while config-containment, as its name implies, uses a more general notion of pre-existing configuration, with constants as well as ground facts; (iii) Access methods in CM-containment are always exact, they return the complete collection of facts compatible with the binding that are present in the database instance; we do not make such an assumption for config-containment and merely assume accesses are sound; (iv) In addition to regular relations, CM-containment also supports *artificial relations* which are unary (monadic) relations “whose content is accessible and amounts only to” some constant value. To the best of our understanding, they correspond in our setting to relations without any access methods, except that config-containment allows them to have arbitrary arity. It would be interesting to see if the result of this paper (and of [5]) can be extended to the case where all relations have an access method.

Among these four, the significant difference is the forbidding of multiple accesses per relation, which means CM-containment is a special case of config-containment:

**PROPOSITION 3.6.** *There are polynomial-time reductions in both directions between CM-containment and the special case of config-containment when relations have at most one access method and relations without an access method have arity bounded by a constant  $K$ . The query language (CQs, PQs) is preserved by the reductions.*

**PROOF.** The argument from CM-containment to config-containment is simple, since config-containment allows a richer initial condition.

The reduction the other way requires us to code the configuration in the contained query.  $\square$

As can be verified, all hardness proofs for containment that we present in this paper make use of no relation with multiple access methods, and the arity the relations with no accesses is bounded by 3. This means all lower bounds for config-containment obtained in this paper yield identical lower bounds for CM-containment.

Calì and Martinenghi give in [5] a proof that the CM-containment problem is decidable in  $\text{coNEXPTIME}$  for conjunctive queries, without any lower bound. We give in Section 5 the same upper bound of  $\text{coNEXPTIME}$  for config-containment, as well as a matching lower bound.

By having the possibility of several access methods per relation, and of some fixed base knowledge given by relations without accesses, we allow modeling of a more realistic setting, where multiple sources of the deep Web may share the same schema, and where we want to ask queries over these sources as well as over some fully accessible local knowledge.

## 4. INDEPENDENT ACCESSES

We establish in this section complexity bounds for the problem of determining whether an access is relevant to a query, when all access methods are *independent*. Our upper bounds will be fairly immediate – the main work involved is in the lower bounds.

In the case of independent accesses, we have some immediate facts: (i) An access to a relation that is not mentioned in the query can not be relevant in either of our senses. (ii) A path witnessing the fact that an access is long-term relevant

can always be pruned to include only subgoals of the query, with each subgoal occurring at most once. This gives a bound of  $\Sigma_2^P$  in combined complexity for checking long-term relevance, since checking that the truncation of the path does not satisfy the query is in  $\text{coNP}$  for the considered query languages. (iii) Since constants can be guessed at will, abstract domain constraints do not have any impact on relevance for independent accesses and we can assume all attributes to share the same domain.

We study the complexity of whether an access is relevant, for immediate and long-term relevance.

**Immediate relevance.** The following result characterizes the combined and data complexity of IR for independent access methods.

**PROPOSITION 4.1.** *We assume all access methods to be independent. Given a Boolean positive query  $Q$ , configuration  $\text{Conf}$ , and access  $(\text{AcM}, \text{Bind})$ , determining whether  $(\text{AcM}, \text{Bind})$  is immediately relevant for the query  $Q$  in  $\text{Conf}$  is a DP-complete problem. If we know the query is not certain in  $\text{Conf}$ , then the problem is NP-complete. Lower bounds hold even if the query is conjunctive.*

*If the query is fixed, the problem is in  $\text{AC}^0$ .*

**PROOF.** Membership in DP works via guessing a witness configuration and verifying it. One can show that the witness need not be large, and verifying it requires checking a conjunctive query and a negation of a conjunctive query.

Hardness uses a coding of satisfiable/unsatisfiable pairs of queries.  $\square$

**Long-term relevance.** We now move to LTR for independent methods.

**EXAMPLE 4.2.** Consider  $Q = R(x, 5) \wedge S(5, z)$ , a configuration in which only  $R(3, 5)$  holds, and an access method to  $R$  on the second component. Clearly, an access  $R(?, 5)$  is not long-term relevant, since any witness  $x$  discovered in the response to this access could be replaced by 3. On the other hand, if the configuration had  $R(3, 6)$  then an access  $R(?, 5)$  would be long-term relevant.  $\square$

Let us first consider a simple case: that of conjunctive queries where the accessed relation only occurs once. In this particular case, it is possible to decide LTR by checking whether the subgoal containing the accessed relation is not in a connected component of the query that is already certain.

More formally, let  $R = \text{Rel}(\text{AcM})$ . Let  $h$  be the (necessarily unique) partial mapping substituting the binding for the corresponding elements of the subgoal of the conjunctive query  $Q$  containing  $R$ , and  $Q_h$  be the query obtained by applying  $h$  to the variables of  $Q$ . If no such  $h$  exists (since the subgoal conflicts with the binding), then clearly the access is not LTR. Otherwise, let  $G(Q_h)$  be the graph whose vertices are the subgoals of  $Q_h$  with an edge between subgoals if and only if they share a variable in  $Q_h$ . Let  $Q_h - \text{Sat}(\text{Conf})$  be the query obtained from  $Q_h$  by removing any subgoal that lies in a component of  $G(Q_h)$  that is satisfied in  $\text{Conf}$ . If  $Q_h - \text{Sat}(\text{Conf})$  contains the  $R$  subgoal, we return true, otherwise we return false.

If this algorithm returns true, then let  $g$  be the subgoal containing  $R$ ,  $h$  the homomorphism, and  $C$  be the component of  $g$  in  $G(Q_h)$ . If  $C$  contains only  $g$ , then clearly the access is long-term relevant, since we can consider any path that

begins with the access and then continues through every subgoal. If  $C$  contains other subgoals, then there is some variable shared between  $g$  and other subgoals; again we take a path beginning at  $R$ , accessing each additional subgoal in turn (in an arbitrary order) and using new elements as inputs while returning elements not in  $\text{Conf}$  for all variables in the subgoals. Note that we do not have to access the other subgoals using the shared variables – we use an arbitrary access method for other relations (we know at least one exists), and choose a response such that the results match the subgoals. This witnesses that the access is LTR. Conversely, suppose  $R$  does not occur in  $Q_h - \text{Sat}(\text{Conf})$  and that we have a path  $p$  witnessing that the access is long-term relevant. Let  $h'$  be a homomorphism from  $Q$  into the  $\text{Conf} + p$ , and let  $h''$  be formed from  $h'$  by replacing all elements in  $C$  by witnesses in  $\text{Conf}$ . The existence of  $h''$  proves that the path is not a witness of the fact the access is LTR.

We have thus the following complexity result in this particular case of conjunctive queries with only one occurrence of the accessed relation (hardness is again shown via a coding argument).

**PROPOSITION 4.3.** *We assume all access methods to be independent. Given a Boolean conjunctive query  $Q$ , a configuration  $\text{Conf}$ , and an access  $(\text{AcM}, \text{Bind})$ , such that  $\text{Rel}(\text{AcM})$  only occurs once in  $Q$ , determining whether  $(\text{AcM}, \text{Bind})$  is long-term relevant for the query  $Q$  in  $\text{Conf}$  is a coNP-complete problem.*

In the case where a relation is repeated, however, simply looking at satisfied components is not sufficient.

**EXAMPLE 4.4.** Consider  $Q = R(x, y) \wedge R(x, 5)$ , an empty configuration, and an access to  $R$  with second component 3 (i.e.,  $R(?, 3)$ ). Clearly this access is not long-term relevant in the empty configuration, since in fact  $Q$  is equivalent to the existential closure of  $R(x, 5)$ , and the access can reveal nothing about such a query. But no subgoals are realized in the configuration.  $\square$

In the general case (repeated relations, positive queries), we can fall back on the  $\Sigma_2^P$  algorithm described at the beginning of the section. Surprisingly, this is the best we can do even in very limited situations:

**PROPOSITION 4.5.** *We assume all access methods to be independent. Given a Boolean positive query  $Q$ , a configuration  $\text{Conf}$ , and an access  $(\text{AcM}, \text{Bind})$ , determining whether  $(\text{AcM}, \text{Bind})$  is long-term relevant for the query  $Q$  in  $\text{Conf}$  is  $\Sigma_2^P$ -complete. The lower bound holds even if  $Q$  is conjunctive and known not to be certain in  $\text{Conf}$ , and even if all relations are freely accessible and all variables have the same infinite datatype.*

*If the query is fixed, the problem is  $\text{AC}^0$ .*

**PROOF.** The lower bound follows from results of Miklau and Suciu [21]; we explain the connection to their notion of *criticality*, which is closely related to relevance. For a query  $Q$  on a single relation  $R$  and a finite domain  $D$  (i.e., a finite set of constants), a tuple  $\mathbf{t}$  (with same arity as  $R$ ) is *critical* for  $Q$  if there exists an instance  $I$  of  $R$  with values in  $D$  such that deleting  $\mathbf{t}$  from  $I$  changes the value of  $Q$ .

**THEOREM 4.6** (4.10 OF [21]). *The problem of deciding, for conjunctive query  $Q$  and set  $D$  whether a tuple  $\mathbf{t}$  is not critical is  $\Pi_2^P$ -hard, even for fixed  $D$  and  $\mathbf{t}$ .*

It is easy to see that  $\mathbf{t}$  is critical iff the Boolean access  $R(\mathbf{t})$  is LTR in the empty configuration (or, more precisely,

in a configuration that only contains constants of the queries but no facts for  $R$ ): this holds since LTR is easily seen to be equivalent to the existence of some instance of size at most  $|Q|$  where adding the tuple given by the access changes the truth value of  $Q$  to true. Hence the theorem above shows that LTR is  $\Sigma_2^P$ -hard even for a fixed configuration.

The combined complexity upper bound has already been discussed. We now discuss data complexity. We can assume without loss of generality  $Q$  is in disjunctive normal form (i.e., a union of conjunctive queries). Let us present the  $\Sigma_2^P$  algorithm slightly differently. We make a guess for each subgoal  $G$  of  $Q$  of the following nondeterministic choices:

1.  $G$  is not witnessed;
2.  $G$  is witnessed by the configuration;
3.  $G$  is witnessed by the first access;
4.  $G$  is witnessed by a further access.

For such a guess  $h$ , we write  $\mathcal{G}_1^h, \mathcal{G}_2^h, \mathcal{G}_3^h, \mathcal{G}_4^h$  the corresponding partition of the set of subgoals. We restrict valid guesses to those where (i) at least one of the disjuncts of  $Q$  has all its subgoals witnessed (i.e., in  $\mathcal{G}_2^h \cup \mathcal{G}_3^h \cup \mathcal{G}_4^h$ ); (ii) all subgoals in  $\mathcal{G}_3^h$  are compatible with  $(\text{AcM}, \text{Bind})$  (if we want the access to also be part of the input, we can easily encode this condition into the constructed formula).

Let  $\mathcal{H}$  be the set of all valid guesses (there is a fixed number of them once the query is fixed). For a given  $h \in \mathcal{H}$ , we rewrite  $Q$  into two queries  $Q'_h$  and  $Q''_h$ .  $Q'_h$  is obtained from  $Q$  by replacing every variable mapped to an input attribute of subgoals in  $\mathcal{G}_3^h$  with the binding, by replacing every subgoal of  $\mathcal{G}_4^h$  with *true*, and by dropping every disjunct of  $\mathcal{H}$  that has a subgoal in  $\mathcal{G}_1^h$ .  $Q''_h$  is obtained from  $Q'_h$  by replacing every subgoal in  $\mathcal{G}_3^h$  with *true*. Then  $(\text{AcM}, \text{Bind})$  is LTR for  $Q$  in  $\text{Conf}$  if and only if the following first-order query evaluates to *true* on  $\text{Conf}$ :  $\bigvee_{h \in \mathcal{H}} Q'_h \wedge \neg Q''_h$ . This query is exponential in the size of  $Q$ .  $\square$

As shown, long-term relevance, even in the independent case and for the relatively simple language of conjunctive queries, is already at the second level of the polynomial hierarchy in combined complexity. Introducing dependent accesses will move the problem into the exponential hierarchy.

## 5. DEPENDENT ACCESSSES

We now turn to the case when some of the accesses are dependent. The results for IR are clearly the same as in the independent case, since this only considers the impact of a single access. We thus only discuss long-term relevance.

In this abstract, we deal *only with long-term relevance for boolean accesses*. We strongly rely for establishing the upper bound results of this section on the connection between relevance and containment that was established in Section 3. Lower bound arguments will be based on constraining paths to be exponentially or doubly exponentially long, using reductions from tiling.

The upper bounds will make use of methods due to Cali and Martinenghi, which are related to those of Chaudhuri and Vardi [9, 10] for Datalog containment. In [5], Cali and Martinenghi show that we can assume that counterexamples to containment of  $Q$  in  $Q'$  (or witnesses to long-term relevance) are *tree-like*. In the containment setting, this means that every element outside the initial configuration and the image of  $Q$  under a homomorphism occurs in at most two accesses, one as an output and possibly one as an input. Each element outside of the configuration can be associated

with an atom – the atom that generated that element as output. For elements  $n_1, n_2$  neither in  $\text{Conf}$  or  $h(Q)$ , say that  $n_1 \prec n_2$  if  $n_2$  is generated by an access to  $n_1$ , and let  $\prec^*$  be the transitive closure of  $\prec$ . Then the tree-like requirement corresponds to the fact that  $\prec^*$  is a tree. This is exactly what Calì and Martinenghi refer to as a *crayfish* chase database, and in [5] they give the “unfolding” construction that shows that such counterexample models always exist – we use this often throughout this section. By exploiting the limited structure of a tree-like database further we will be able to extend the upper bounds of [5], taking into account configuration constants and multiple accesses.

In contrast with what happens for the independent case, results are radically different for conjunctive and positive queries. We thus study the complexity of long-term relevance and query containment in turn for both query languages.

## 5.1 Conjunctive Queries

For conjunctive queries, we show we have  $\text{coNEXPTIME}$ -completeness of containment and  $\text{NEXPTIME}$ -completeness of the LTR problem. We first establish the hardness through a reduction of a tiling problem of an exponential-size corridor that yields an exponential-size path. Recall that the lower bound for query containment directly implies the lower bound for relevance, thanks to Proposition 3.3.

**THEOREM 5.1.** *Boolean conjunctive query containment under access limitations is  $\text{coNEXPTIME}$ -hard. Consequently, long-term relevance of an access for a conjunctive query is  $\text{NEXPTIME}$ -hard.*

**PROOF.** We show a reduction from the  $\text{NEXPTIME}$ -complete problem consisting in tiling a  $2^n \times 2^n$  corridor, where  $n$  is given in unary, under horizontal and vertical constraints (see Section 3.2 of [17]). A tile will be represented by an access atom  $\text{Tile}(t, \mathbf{b}, \mathbf{c}, x, y)$ , where  $\mathbf{b}$  is the vertical position (i.e., the row),  $\mathbf{c}$  the horizontal position (i.e., the column),  $t$  the tile type, and  $x, y$  are values that link one tile to the next generated tile in the witness path, as will become clear later. The  $n$  bit binary representation of the decimal number  $d$  is denoted by  $[d]$ .

For a given tiling problem with tile types  $t_1, \dots, t_k$ , horizontal relation  $H$ , vertical relation  $V$ , and initial tiles of respective type  $t_{i_0}, \dots, t_{i_{m-1}}$ , we construct the following containment problem. The schema has the following relations with their respective arities as superscripts:  $\text{Bool}^1$ ,  $\text{TileType}^1$ ,  $\text{SameTile}^3$ ,  $\text{Horiz}^3$ ,  $\text{Vert}^3$ ,  $\text{And}^3$ ,  $\text{Or}^3$ ,  $\text{Eq}^3$ , all of them having no access methods, and  $\text{Tile}^{2n+3}$ , with a single access method whose input arguments are all but the last. We generate the following configuration  $\text{Conf}$ :

$$\begin{aligned} & \text{Bool}(0), \text{Bool}(1); \\ & \text{TileType}(t_1), \dots, \text{TileType}(t_k); \\ & \text{SameTile}(t_i, t_i, 1) \text{ for } 1 \leq i \leq m, \\ & \text{SameTile}(t_i, t_j, 0) \text{ for } i \neq j, 1 \leq i, j \leq m; \\ & \text{Horiz}(t_i, t_j, 1) \text{ for all } \langle t_i, t_j \rangle \in H, \\ & \text{Horiz}(t_i, t_j, 0) \text{ for all } \langle t_i, t_j \rangle \notin H; \\ & \text{Vert}(t_i, t_j, 1) \text{ for all } \langle t_i, t_j \rangle \in V, \\ & \text{Vert}(t_i, t_j, 0) \text{ for all } \langle t_i, t_j \rangle \notin V; \\ & \text{And}(0, 0, 0), \text{And}(1, 0, 0), \text{And}(0, 1, 0), \text{And}(1, 1, 1); \\ & \text{Or}(0, 0, 0), \text{Or}(0, 1, 1), \text{Or}(1, 0, 1), \text{Or}(1, 1, 1); \\ & \text{Eq}(0, 0, 1), \text{Eq}(1, 0, 0), \text{Eq}(0, 1, 0), \text{Eq}(1, 1, 1); \\ & \text{Tile}(t_{i_0}, [0], [0], c_0, c_1), \text{Tile}(t_{i_1}, [0], [1], c_1, c_2). \end{aligned}$$

We use three domains:  $\mathcal{B}$  (used for Booleans),  $\mathcal{T}$  (used for tile types),  $\mathcal{C}$  (used for chaining up tiles). They are assigned

as follows:  $\text{Bool}$ ,  $\text{And}$ ,  $\text{Or}$ ,  $\text{Eq}$  have all their argument in  $\mathcal{B}$ ; the argument of  $\text{TileType}$  has domain  $\mathcal{T}$ ;  $\text{SameTile}$ ,  $\text{Horiz}$ , and  $\text{Vert}$  have their first two arguments of domain  $\mathcal{T}$  and the third of domain  $\mathcal{B}$ ; finally, the first argument of  $\text{Tile}$  has domain  $\mathcal{T}$  and the remaining ones domain  $\mathcal{B}$ , except for the last two, that are in  $\mathcal{C}$ .

We reduce to the complement of query containment of  $Q_1$  into  $Q_2$  where  $Q_1$  is the atom  $\text{Tile}(u, [2^n - 1], [2^n - 1], x, y)$  and  $Q_2$  consists of the following conjunction of atoms:

$$\begin{aligned} & \text{Tile}(u, \mathbf{b}, \mathbf{c}, x, y) \wedge \text{Tile}(v, \mathbf{d}, \mathbf{e}, y, z) \wedge \text{Tile}(w, \mathbf{f}, \mathbf{g}, y', z') \\ & \wedge \text{Tile}(q, \mathbf{v}, \mathbf{h}, y', z'') \wedge \text{BOOLCONS}, \end{aligned}$$

where  $u, v, w, q$  are variables intended for tile types, each of  $\mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{e}, \mathbf{f}, \mathbf{g}, \mathbf{h}, \mathbf{v}$  is a tuple of  $n$  variables intended for Booleans, and  $x, y, y', z, z', z''$  are variables intended for linking elements, and where  $\text{BOOLCONS}$  consists of a conjunction of  $\text{And}$ ,  $\text{Or}$ , and  $\text{Eq}$  atoms imposing a number of Boolean constraints on the bit-vectors  $\mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{e}, \mathbf{f}, \mathbf{g}, \mathbf{h}, \mathbf{v}$ . Since we want  $\text{BOOLCONS}$  to remain conjunctive, its construction is a bit intricate, but in essence it states that there is “something wrong” with the tiling. More precisely, it consists of a conjunction of the following subformulas:

1. A subformula  $\text{SUB}_1(i_1)$  such that  $\text{SUB}_1(0)$  holds true iff the functional dependency from the next-to-last argument of the  $\text{Tile}$  relation to the  $2n$  bit-valued attributes in the same relation is violated for some tuple. To express this, we use the last two of the four  $\text{Tile}$  atoms in the above formula, because they have both the same variable  $y'$  in their next-to-last position. We require that some  $f_i$  differs from the corresponding  $v_i$  or some  $g_j$  differs from its corresponding  $h_j$ . We can assert this as:  $\text{Eq}(f_1, v_1, a_1) \wedge \text{Eq}(f_2, v_2, a_i) \wedge \dots \wedge \text{Eq}(f_n, v_n, a_n) \wedge \text{Eq}(g_1, h_1, a_{n+1}) \wedge \text{Eq}(g_2, h_2, a_{n+i}) \wedge \dots \wedge \text{Eq}(g_n, h_n, a_{2n}) \wedge \text{And}(a_1, a_2, r_1) \wedge \text{And}(r_1, a_3, r_2) \wedge \dots \wedge \text{And}(r_{2n-2}, a_{2n}, i_1)$ .

2. A subformula  $\text{SUB}_2(i_2)$  such that  $\text{SUB}_2(0)$  holds true iff two accesses  $A_1$  and  $A_2$  on the  $\text{Tile}$  relation, where the output value of  $A_1$  is equal to the value of the penultimate argument of  $A_2$  are such that their bit-vectors are in a wrong relationship. The latter just means that the concatenated two bit-vectors of  $A_1$  are *not* a predecessor of the concatenated two bit-vectors of  $A_2$ . To express this, we use the first two atoms of  $Q_2$ . Indeed, they are already linked via variable  $y$ . It is now just necessary to express that their bit-vectors are wrong. To do this, we design a conjunction of atoms  $\text{SUCC}(\mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{e}, s)$  for which  $s = 1$  iff vector  $\langle \mathbf{b}, \mathbf{c} \rangle$  is a numeric predecessor of  $\langle \mathbf{d}, \mathbf{e} \rangle$ , and  $s = 0$  otherwise. Then, let  $\text{SUB}_2(i_2) = \text{SUCC}(\mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{e}, i_2)$ . The  $\text{SUCC}(\mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{e}, s)$  subformula can be easily constructed by using purely Boolean atoms only. Briefly, we first define a  $\text{SUCC}_i(\mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{e}, s_i)$  formula for  $1 \leq i \leq 2n$  such that  $s_i = 1$  iff the leading  $i - 1$  bits of both vectors coincide. The  $i$ th bit of  $\langle \mathbf{b}, \mathbf{c} \rangle$  is 0, while the  $i$ th bit of  $\langle \mathbf{d}, \mathbf{e} \rangle$  is 1, and all bits in positions above  $i$  are 1 in  $\langle \mathbf{b}, \mathbf{c} \rangle$  and 0 in  $\langle \mathbf{d}, \mathbf{e} \rangle$ . All this is easily done with  $\text{And}$  and  $\text{Eq}$  atoms. Finally,  $\text{SUCC}$  is constructed by taking all  $\text{SUCC}_i$  and or-ing their  $s_i$ -values:  $\text{Or}(s_1, s_2, k_1) \wedge \text{Or}(k_1, s_3, k_2) \wedge \dots \wedge \text{Or}(k_{2n-2}, s_{2n}, s)$ . Note that we only need a polynomial number of atoms.

3. A subformula  $\text{SUB}_3(i_3)$  such that  $\text{SUB}_3(0)$  holds true iff some vertical or horizontal constraints are violated or if the initial  $m$  tiles are of wrong tile type. Informally, we thus need to assert in this subformula that there exist two tiles, such that some tiling constraint is violated. Here we can, for example, use the second and third atoms of  $Q_2$ .



For the horizontal constraints, we define in the obvious way subformulas that check that  $\mathbf{d}$  is the predecessor of  $\mathbf{f}$ , that  $\mathbf{e}$  and  $\mathbf{g}$  are equal, and  $\text{Horiz}(v, w, 0)$ . The resulting truth value is or-red with a violation of the vertical constraints which is encoded in a similar way. To all this, we also connect disjunctively (conjoining *Or* atoms) all possible violations of the correct tile-type of the  $m$  initial tiles. We can use the third atom of  $Q_2$  for this. Such a violation arises if in  $\text{Tile}(w, \mathbf{f}, \mathbf{g}, y', z')$   $\mathbf{f} = [0]$ ,  $\mathbf{g} = [i] < [m]$ , and  $w$  is any tile type but the correct one. This can be easily expressed using the *SameTile* predicate and Boolean operators.

4. Finally, we add to the conjunction so far  $\text{SUB}_1(i_1) \wedge \text{SUB}_2(i_2) \wedge \text{SUB}_3(i_3)$  a subformula  $\text{SUB}_4$  expressing that at least one of the bits  $i_1, i_2, i_3$  must be zero:  $\text{SUB}_4 = \text{And}(i_1, i_2, j) \wedge \text{And}(j, i_3, 0)$ . This concludes the construction of  $Q_2$ .

We claim that the grid is tiled iff there is an access path  $p$  based on conf *Conf* that falsifies  $Q_2$  and satisfies  $Q_1$  (see [3] for the proof of this fact).  $\square$

We now deal with upper bounds. Chang and Li noted that for every conjunctive query (or UCQ)  $Q$ , and any set of access patterns, one can write a Monadic Datalog query  $Q_{acc}$  that represents the answers to  $Q$  that can be obtained according to the access patterns – the intentional predicates represent the accessible elements of each datatype, which can be axiomatized via recursive rules corresponding to each access method. One can thus show that containment of  $Q$  in  $Q'$  under access patterns is reduced to containment of the Monadic Datalog query  $Q_{acc}$  in  $Q'$ . Although containment between Datalog queries is undecidable, containment of Monadic Datalog queries is decidable (in 2EXPTIME [11]) and containment of Datalog queries in UCQs is decidable (in 3EXPTIME [10]), this does not give tight bounds for our problem. Chaudhuri and Vardi [10] have shown that containment of Monadic Datalog queries in *connected* UCQs is in coNEXPTIME. The queries considered there have a head predicate with one free variable, and the connectedness requirement is that the graph connecting atoms when they share a variable is connected – thus the head atom is connected to every other variable. Connectedness is a strong condition – it implies that in a tree-like model one need only look for homomorphisms that lie close to the root.

We now show a coNEXPTIME upper bound, matching our lower bound and extending the prior results above. From the nondeterministic polynomial-time Turing reduction from containment to relevance (Proposition 3.3), we deduce NEXPTIME membership for long-term relevance of Boolean accesses.

**THEOREM 5.2.** *Boolean conjunctive query containment under access limitations is in coNEXPTIME. Long-term relevance of a Boolean access for a conjunctive query is in NEXPTIME.*

We now outline the proof of coNEXPTIME-membership for containment under access patterns. Consider queries  $Q, Q'$ , and configuration *Conf*.

An element  $n$  in an instance  $\mathsf{I}$  is *fresh* if it is not in the initial configuration *Conf*. Call a homomorphism  $h$  of a subquery  $Q''$  of  $Q'$  into an instance *freshly-connected* if the following graph  $G''$  is connected: The vertices of  $G''$  are the atoms of  $Q''$  and there is an edge between two such atoms if they overlap in a variable mapped to the same fresh value by

$h$ . Given an element  $n$ , a homomorphism of query  $Q'(x)$  is *rooted at  $n$*  if it maps the distinguished variable  $x$  to  $n$  and includes only one atom containing  $x$ .

In this proof, we assume for convenience that all values of enumerated types mentioned in the queries are in the initial configuration – and thus fresh values are always of a non-enumerated type. This hypothesis can be removed, since the part of any witness to non-containment that involves enumerated types can always be guessed, staying within the required bounds.

In order to get an exponential-sized witness to non-containment, we would like to abstract an element of an instance by *all* subqueries of  $Q'$  that it satisfies, or even all the freshly-connected homomorphisms. However, this would require looking at many queries, giving a doubly exponential bound (as in Theorem 5.6).

The following key lemma states that it suffices to look at just one freshly-connected homomorphism.

**CLAIM 5.3.** *For each tree-like instance  $\mathsf{I}$  and element  $n$  in  $\mathsf{I}$ , and for each query atom  $A$  that maps into an  $\mathsf{I}$ -atom containing  $n$ , there is a unique maximal freshly-connected partial homomorphism rooted in  $n$  that includes atom  $A$  in its domain.*

**PROOF.** Consider a function  $h$  mapping variables of the atom  $A$  into  $\mathsf{I}$  such that  $h(x) = n$ . We claim that there is only one way to extend  $h$  to a freshly connected homomorphism including other atoms. Clearly, to satisfy the freshness requirement, any other atom  $A'$  must include at least one other variable in common with  $A$ , say  $y$ , mapped to  $n_2$  by  $h$ . To satisfy rootedness, this variable must not contain  $x$ . But in a tree-like model there can be only one other fact  $F$  in  $\mathsf{I}$  that contains  $y$ , and hence for every position  $p$  of  $A'$  we can only map a variable in that position to the corresponding argument of  $F$ .  $\square$

For elements  $n_1$  and  $n_2$  and atom  $A$ , let  $h_1^A$  and  $h_2^A$  be the maximal freshly-connected partial homomorphisms given by the claim above for  $n_1$  and  $n_2$ , respectively. We say two elements  $n_1$  and  $n_2$  in  $\mathsf{I}$  are *maxfresh-equivalent* if for every  $A$  in  $Q'$  there is an isomorphism  $r$  of  $\mathsf{I}$  that preserves the initial configuration, and such that  $h_1^A \circ r = h_2^A$ .

We say that fresh elements  $n_1$  and  $n_2$  are *similar* if they are maxfresh-equivalent, and if  $n_1$  occurs as input variable  $i$  in atom  $A$ , then the same is true for  $n_2$  – recall that fresh elements in tree-like instances occur as the inputs to at most one atom.

We now show that the maxfresh-equivalence classes of subtrees can be determined compositionally.

**LEMMA 5.4.** *Suppose  $A$  is an atom satisfied by fresh elements  $n_0, n_1 \dots n_k$  along with configuration elements  $\mathbf{c}$  in a tree-like instance  $\mathsf{I}$ , with each  $n_i$  in the subtree of  $n_0$ . Suppose that the same is true for fresh  $n'_0, n'_1 \dots n'_k$  and  $\mathbf{c}$ , with  $\mathbf{c}$  in the same arguments of  $A$ . If each  $n_i$  is maxfresh-equivalent to  $n'_i$ , then  $n_0$  is maxfresh-equivalent to  $n'_0$ .*

**PROOF.** We show that the subtrees of  $n_0$  and  $n'_0$  satisfy the same subqueries of  $Q'$ . Suppose a freshly-connected subquery  $Q''$  were to hold in  $n_0$  with  $h$  a witness. Let  $Q'(x)$  be the query where  $x$  is made free, and let  $h_i$  be the restriction of  $h$  to the variables that are connected to  $x$  and map within subtrees of the  $n_i$ . This is a freshly-connected homomorphism, so must also be realized in the subtree of  $n'_i$  in  $\mathsf{I}$ . But then we can extend the homomorphism to the

subtree of  $n'_0$  by mapping  $A$  according to the fact holding in common within  $n'_0$  and  $n_0$ .  $\square$

We now show that one representative of each similarity class suffices for a counterexample model.

LEMMA 5.5. *If  $Q$  is not contained in  $Q'$  under access patterns, then there is a witness instance  $I$  in which no two elements are similar.*

Proof: Consider an arbitrary tree-like instance  $I$  satisfying the access patterns, such that  $I$  satisfies  $Q$  and not  $Q'$ . Given distinct  $n_1$  and  $n_2$  that are similar, we show that they can be identified, possibly shrinking the model. By Lemma 5.4 this identification preserves the similarity type, so assuming we have proven this, we can get a single representative by induction.

Consider the case where  $n_1$  is an ancestor of  $n_2$  (the case where the two are incomparable is similar). Consider the model  $I'$  obtained by identifying  $n_1$  and  $n_2$ , removing all items that lie below  $n_1$  and which do not lie below  $n_2$  in the dependency graph. Let  $n'$  denote the image of  $n_1$  under the identification in  $I'$ .  $I'$  is still generated by a well-formed access path, while  $Q$  is still satisfied, since the homomorphic image of  $Q$  was not modified. If  $Q'$  were satisfied in  $I'$ , let  $h'$  be a homomorphism witnessing this. Clearly the image of  $h'$  must include  $n'$ . Let  $h_{n'}$  be the maximal connected subquery of  $Q'$  that maps to a contiguous subtree rooted at  $n'$ . Then up to isomorphism  $h_{n'}$  is the same as the maximal fresh homomorphism rooted at  $n_2$ ; and since  $n_1$  and  $n_2$  are max-fresh equivalent, this means that  $h_{n'}$  is (modulo composition with an isomorphism) the same as  $h_{n_1}$  the maximal fresh homomorphism rooted at  $n_1$ .

Let  $IM$  be the image of  $h'$ . We define a mapping  $f$  taking elements of  $IM$  to  $I$  as follows: nodes in the image of  $h_{n'}$  go to their isomorphic image in the image of  $h_{n_1}$ ; other nodes lying in both  $IM$  and the subtree of  $n'$  go to their isomorphic image in the subtree of  $n_2$ , while nodes lying outside the subtree of  $n'$  are mapped to the identity. We argue that the composition of  $h'$  with  $f$  gives a homomorphism of  $Q'$  into  $I$ . Atoms all of whose elements are in the domain of  $h_{n'}$  are preserved since  $n'$  and  $n_1$  are max-fresh equivalent. The properties of maximal equivalence classes guarantee that for all other atoms of  $Q'$  either: a) all variables that are mapped by  $h'$  to fresh elements are mapped to elements in the subtree of  $n'$  outside of  $h_{n'}$ ; hence such atoms are preserved by  $f$ ; or b) all variables that are mapped by  $h'$  to fresh elements are mapped to elements either in  $h_{n'}$  or above it; such atoms are preserved, since  $f$  is an isomorphism on these elements. Thus  $Q'$  holds in  $I$ , a contradiction.

From Lemma 5.5 we see that whenever there is a counterexample to containment, there is a DAG-shaped model of size the number of similarity classes – since this is exponential in the input, an access path that generates it can be guessed by a NEXPTIME algorithm, and the verification that it is a well-formed access path and witnesses  $Q \wedge \neg Q'$  can be done in polynomial time in the size of the path (albeit in  $D^P$  in the size of the queries).

## 5.2 Positive Queries

We now turn to the case where queries can use nesting of  $\vee$  and  $\wedge$ , but no negation. Here we will see that the complexity of the problems becomes exponentially harder. The upper bounds will be via a type-abstraction mechanism; the lower bounds will again go via tiling problems, although of a more involved sort.

THEOREM 5.6. *The problem of determining whether query  $Q$  is contained in  $Q'$  under access constraints is complete for co2NEXPTIME, while determining whether a Boolean access is LTR for a positive query  $Q$  is complete for 2NEXPTIME.*

PROOF. Again, results about relevance are derived by the reductions of Section 3.

The upper bound holds by showing that if there is a counterexample to containment then there is one of doubly-exponential size – this is in turn shown by seeing that we can identify two elements if they satisfy the same queries of size at most the maximum of  $|Q|, |Q'|$ .

Hardness follows from reducing to the problem of tiling a corridor of width and height doubly-exponential in  $n$ . By choosing  $Q'$  appropriately, we can force the model to consist of a sequence of linked elements each of which lies at the root of a tree of polynomial size. Such a tree can spell out a string of exponentially many bits, and we can further ensure that successive occurrences trees encode a description of a doubly-exponential sized tiling.  $\square$

We note that in terms of data complexity, we still have tractability, even in this very general case:

PROPOSITION 5.7. *When the queries are fixed, the complexity of containment is in polynomial time. Similarly, the complexity of LTR is in polynomial time once the query is fixed.*

PROOF. Again we give the argument only for containment and use Proposition 3.3 to conclude for relevance (note the remark in Proposition 3.3 about configurations being the same). In the co2NEXPTIME membership argument for containment in Theorem 5.6 we have shown a witness instance in which the elements consist of  $k(Q, Q')$  elements of each type, where the number of types is  $l(Q, Q')$ , hence with a constant number of elements outside the configuration once  $Q$  and  $Q'$  are fixed. The number of possible access sequences is thus polynomial in the configuration, and verifying that a sequence is well-formed and satisfies  $Q \wedge \neg Q'$  can be done in polynomial time since  $Q$  and  $Q'$  are fixed.  $\square$

## 6. RELATED WORK

We overview the existing literature on answering queries in the presence of limited access patterns, highlighting the differentiators of our approach using the vocabulary of Section 2. The problem of querying under access restrictions was originally motivated by access to built-in-predicates with infinitely many tuples (such as  $<$ ) – which is only reasonable if all variables are bound – and by the desire to access relations only on their indexed attributes (see [27], chapter 12). More recently, the rise of data-integration systems with sources whose content are accessible through interfaces with limited capabilities [25] has been the main driver of interest in the subject; the most well-known example is the querying of deep Web sources [16] accessible through Web forms. Research efforts on deep Web exploration [16] and on the use of Web services to complement extensional knowledge bases [24] are practical settings where the notion of dynamic relevance of a Web source is fundamental.

With a few exceptions that will be noted, most of the existing work focuses on static analysis for dependent accesses. By static we mean that they seek a means to answer the query *ab initio* that does not consider the configuration or

adapt to it. By dependent we mean, as in the second part of this paper, that it is impossible to guess a constant to be used in a bound access. Typically accesses are also supposed to be exact and not merely sound. This last limitation is unrealistic in the case of deep Web sources, where a given source will often have only partial knowledge over some data collection. In contrast, our results give new bounds on static problems, but also consider dynamic relevance. We allow a collection of sound accesses that can be dependent or independent. Queries considered in the literature are usually conjunctive, but work on query answerability and rewriting has also considered richer query languages, such as unions of conjunctive queries (UCQs) [18], UCQs with negations [23, 12], or even first-order logic [22]. In a few cases [25, 13, 12], existing work assumes that both queries and sources are expressed as *views* over a global schema, and the limited access problem is combined to that of answering queries using views [15]. In the other cases, the query is supposed to be directly expressed in terms of the source relations, as we have done in this work.

**Static analysis.** The first study of query answering when sources have limited patterns is by Rajaraman, Sagiv, and Ullman [25]. Given a conjunctive query over a global schema and a set of views over the same schema, with exact dependent access patterns, they show that determining whether there exists a conjunctive query plan over the views that is equivalent to the original query and respects the access patterns is NP-complete. This is based on the observation that the size of a query plan can be bounded by the size of the query: one can just keep in the plan subgoals that either are mapped to one of the subgoal of the query, or provide an initial binding for one of the variables of the queries.

Duschka, Genesereth, and Levy study in [13] the general problem of answering queries using views. They solve this by constructing a Datalog query plan formed of *inverse rules* obtained from the source descriptions, a plan computing the maximally contained answer to a query. Although this approach was geared towards data integration without limitations, the same work extends it to incorporate limited access patterns on sources in a straightforward manner.

Li and Chang [19] propose a static query planning framework based on Datalog for getting the maximally contained answer to a query with exact dependent access pattern; the query language considered is a proper subset of UCQs, with only natural joins allowed. In the follow-up work [18], the query language is lifted to UCQs, and Li shows that testing the existence of an exact query rewriting is NP-complete, which can be seen as an extension of the result from [25], from CQs to UCQs (though views are not considered in [18]). That article also proposes a dynamic approach when an exact query rewriting does not exist, which we discuss further.

Nash and Ludäscher [23] extend the results of [18] to the case of UCQs with negations. The complexity of the exact rewriting problem is reduced to standard query containment and thus becomes in this case  $\Pi_2^P$ -complete. Other query languages are also considered in [22], up to first-order-logic, for which the rewriting problem becomes undecidable. Deutsch, Ludäscher, and Nash add in [12] views and constraints (in the form of weakly acyclic tuple-generating dependencies) to the setting of dependent exact patterns. Using the chase procedure, they show that the exact rewriting problem remains  $\Pi_2^P$ -complete in the presence of a large class of integrity con-

straints, and they provide algorithms for obtaining maximally contained answer and minimally containing static plan.

Finally, still in the case of dependent exact accesses and for conjunctive queries, Cali and Martinenghi [6] build on the query planning framework of [19] and show how to obtain a query plan for maximally contained answer that is minimal in terms of the number of accesses made to the sources.

**Dynamic computation of maximal answers.** Some works referenced in the previous paragraphs also consider dynamic, runtime, aspects of the problem, i.e., taking into account the current configuration. Thus, [18] provides an algorithm that finds the complete answer to a query under dependent exact accesses whenever possible, even if an exact query rewriting plan cannot be obtained. This is based on a recursive, exhaustive, enumeration of all constants that can be retrieved from sources, using the techniques of the inverse rule algorithm [13]. The algorithm has no optimality guarantee, since no check is made for the relevance of an access to the query, for any notion of relevance. An extension to UCQs with negation is proposed in [23], with a very similar approach.

**Dynamic relevance.** To our knowledge, the only work to consider the dynamic relevance of a set of accesses with binding patterns is by Cali, Calvanese, and Martinenghi [4]. They define an access with a binding as *dynamically relevant* under a set of constraints (functional dependencies and a very restricted version of inclusion dependencies) for a given configuration if this access can produce new tuples. They show dynamic relevance can be decided in polynomial time. The fact that a source has limited access patterns does not play any role here since one only considers a given binding and disregards all other accesses, and there is no query involved.

**Related work on query containment.** We want to conclude this section by mentioning a few other works that are not dealing with answering queries under binding patterns *per se* but are still pertinent to the problem studied in this paper. We have already compared in detail in Section 3 our work with the complexity analysis [5] of query containment under access limitations – in brief, our results generalize the upper bounds to a richer model, provide matching lower bounds, and give bounds for larger collections of queries. However the arguments used in proofs of our upper bound results of this paper rely heavily on the crayfish chase procedure described in this work.

Chaudhuri and Vardi [9, 10] consider the problem of containment of Datalog queries in unions of conjunctive queries – it is easily seen (e.g. from the Datalog-based approaches to limited access patterns mentioned earlier) that this problem subsumes containment under access patterns. Indeed, containment under access patterns is subsumed by containment of Monadic Datalog in UCQs, a problem shown by [9] to be in **coNEXPTIME** in special cases (e.g. connected queries without constants): the upper-bounds rely on the ability to make models tree-like, as ours do. In the case of binary accesses, containment under access limitations can be reduced to containment of a *path query* in a conjunctive query. This problem is studied in [14, 7] which together give an **PSPACE**-bound for containment of path queries – as our results show, these results give neither tight bounds for the binary case or the best lower bound for the general case.

**Other related work.** Finally, Abiteboul, Bourhis, and Mari-

noiu [1] consider dynamic relevance of a service call to a query in the framework of ActiveXML (XML documents with service calls). A service call is dynamically relevant if it can produce new parts of the tree that will eventually change the query result. They show in particular that non-relevance is in  $\Sigma_2^P$  (an unpublished extension of their work shows  $\Sigma_2^P$ -hardness also by reduction from the critical tuple problem [21]). Though the framework is different, their notion of relevance is close in spirit to our notion of long-term relevance.

## 7. CONCLUSION

We investigated here the problems of analyzing the access paths that can originate from a particular data access. When accesses are not tightly coupled, the problem is closely-related to reasoning whether a tuple is “critical” to a given query result. Here we have fairly tight complexity bounds, although admittedly no algorithms that are promising from a practical perspective as yet.

In the setting where accesses are dependent on one another, we have shown a tight connection between relevance problems and containment under access limitations for Boolean accesses. We have shown new bounds for both the relevance and containment problems for conjunctive and positive queries. However, there are still many open issues regarding complexity. For low arity (see [3]) we do not have tight bounds on LTR. For arbitrary arity we have tight bounds for CQs and for positive queries, but we do not know if our lower bounds for positive queries also hold for positive queries of restricted forms (e.g., UCQs). We believe that all of our results for containment can be extended to relevance of non-Boolean accesses, using the same proofs, but we leave this for future work.

## 8. ACKNOWLEDGMENTS

M. Benedikt is supported in part by EP/G004021/1 and EP/H017690/1 of the Engineering and Physical Sciences Research Council UK and in part by EC FP7-ICT-233599. G. Gottlob’s and P. Senellart’s research leading to these results has received funding from the European Research Council under the European Community’s Seventh Framework Programme (FP7/2007-2013) / ERC grant agreements 246858 *DIADEM* and 226513 *Webdam*, respectively. We thank Pierre Bourhis for great assistance with an earlier draft of the paper.

## 9. REFERENCES

- [1] S. Abiteboul, P. Bourhis, and B. Marinhoi. Satisfiability and relevance for queries over active documents. In *PODS*, 2009.
- [2] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
- [3] M. Benedikt, G. Gottlob, and P. Senellart. Determining relevance of accesses at runtime (extended version). *CoRR*, 2011.
- [4] A. Cali, D. Calvanese, and D. Martinenghi. Dynamic query optimization under access limitations and dependencies. *J. UCS*, 15(1):33–62, 2009.
- [5] A. Cali and D. Martinenghi. Conjunctive query containment under access limitations. In *ER*, 2008.
- [6] A. Cali and D. Martinenghi. Querying data under access limitations. In *ICDE*, 2008.
- [7] D. Calvanese, G. D. Giacomo, M. Lenzerini, and M. Y. Vardi. Containment of conjunctive regular path queries with inverse. In *KR*, 2000.
- [8] A. K. Chandra and P. M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *STOC*, Boulder, USA, 1977.
- [9] S. Chaudhuri and M. Y. Vardi. On the complexity of equivalence between recursive and nonrecursive Datalog programs. In *PODS*, 1994.
- [10] S. Chaudhuri and M. Y. Vardi. On the equivalence of recursive and nonrecursive Datalog programs. *JCSS*, 54(1):61–78, 1997.
- [11] S. S. Cosmadakis, H. Gaifman, P. C. Kanellakis, and M. Y. Vardi. Decidable optimization problems for database logic programs. In *STOC*, 1988.
- [12] A. Deutsch, B. Ludäscher, and A. Nash. Rewriting queries using views with access patterns under integrity constraints. *Theor. Comput. Sci.*, 371(3):200–226, 2007.
- [13] O. M. Duschka and A. Y. Levy. Recursive plans for information gathering. In *IJCAI*, 1997.
- [14] D. Florescu, A. Levy, and D. Suci. Query containment for conjunctive queries with regular expressions. In *PODS*, 1998.
- [15] A. Y. Halevy. Answering queries using views: A survey. *VLDB Journal*, 10:270–294, 2001.
- [16] B. He, M. Patel, Z. Zhang, and K. C.-C. Chang. Accessing the deep Web: A survey. *Communications of the ACM*, 50(2):94–101, 2007.
- [17] D. S. Johnson. A catalog of complexity classes. In *Handbook of Theoretical Computer Science*. MIT Press, 1990.
- [18] C. Li. Computing complete answers to queries in the presence of limited access patterns. *VLDB J.*, 12(3):211–227, 2003.
- [19] C. Li and E. Y. Chang. Answering queries with useful bindings. *ACM Trans. Database Syst.*, 26(3):313–343, 2001.
- [20] C. Li and E. Y. Chang. On answering queries in the presence of limited access patterns. In *ICDT*, 2001.
- [21] G. Miklau and D. Suci. A formal analysis of information disclosure in data exchange. *JCSS*, 73(3):507–534, 2007.
- [22] A. Nash and B. Ludäscher. Processing first-order queries under limited access patterns. In *PODS*, 2004.
- [23] A. Nash and B. Ludäscher. Processing union of conjunctive queries with negation under limited access patterns. In *EDBT*, 2004.
- [24] N. Preda, G. Kasneci, F. M. Suchanek, T. Neumann, W. Yuan, and G. Weikum. Active knowledge: dynamically enriching RDF knowledge bases by Web services. In *SIGMOD*, 2010.
- [25] A. Rajaraman, Y. Sagiv, and J. D. Ullman. Answering queries using templates with binding patterns. In *PODS*, 1995.
- [26] Y. Sagiv and M. Yannakakis. Equivalences among relational expressions with the union and difference operators. *J. ACM*, 27(4):633–655, 1980.
- [27] J. D. Ullman. *Principles of Database and Knowledge-Base Systems*, volume 2. Computer Science Press, 1989.