Principles of Guarded Structural Indexing

François Picalausa
Université Libre de Bruxelles (ULB)
Belgium
fpicalau@ulb.ac.be

George H. L. Fletcher
Eindhoven University of Technology
The Netherlands
g.h.l.fletcher@tue.nl

Jan Hidders
Delft University of Technology
The Netherlands
a.j.h.hidders@tudelft.nl

Stijn Vansummeren
Université Libre de Bruxelles (ULB)
Belgium
svsummer@ulb.ac.be

ABSTRACT

We present a new structural characterization of the expressive power of the acyclic conjunctive queries in terms of guarded simulations, and give a finite preservation theorem for the guarded simulation invariant fragment of first order logic.

We discuss the relevance of these results as a formal basis for constructing so-called guarded structural indexes. Structural indexes were first proposed in the context of semi-structured query languages and later successfully applied as an XML indexation mechanism for XPath-like queries on trees and graphs. Guarded structural indexes provide a generalization of structural indexes from graph databases to relational databases.

Categories and Subject Descriptors

F.4.1 [**Mathematical Logic and Formal Languages**]: Mathematical Logic; H.2.3 [**Database Management**]: Languages—*Query languages*; H.2.4 [**Database Management**]: Systems—*Query processing*; H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing—*Indexing Methods*

General Terms

Design, Languages, Theory

Keywords

Acyclicity, conjunctive queries, guarded simulation, fact simulation, finite preservation theorems, hypergraph

1. INTRODUCTION

Recent years have witnessed an increased interest in tree-based and graph-based data formats, first with the advent of XML and more recently with the wide adoption of RDF and JSON to store graphs arising in social networks and the Semantic Web, among other uses. This interest has led to the creation of many specialized graph storage and query processing engines. Typically, these engines use clever techniques that exploit the graph topology to provide the required performance on large input graphs (e.g., [14, 21, 31]). So-called *structural indexes* that provide succinct graph summaries constitute one important technique in this respect [5, 10, 11, 18, 22, 27, 30].

The key idea behind structural indexing is that for many practical graph query languages $\mathcal{Q}$ (reachability queries [10], XPath queries [11, 16], modal or temporal logic queries [4], ...) it is possible to group together the nodes of input graph G to obtain a more compact representation, called the *structural index* for G (with respect to $\mathcal{Q}$). The grouping is done in such a way that any query $Q \in \mathcal{Q}$ can be answered either directly on the structural index of G instead of on G itself, or can be answered more efficiently on G after obtaining pruning information from the index [5, 10, 11, 18, 22, 27, 30]. Since the index is typically (much) smaller than G itself, this way of processing Q can be significantly faster than evaluating Q directly over G.

Example 1. *To illustrate, Figure 1 shows a graph G and a structural index I for G. Observe that each node of I is actually a* set *of nodes in G. There is an edge between sets V and W in I if there is an edge between some $v \in V$ and some $w \in W$ in G. Clearly, I has fewer nodes and edges than G. Further observe that to evaluate a query Q such as "select all professors that advised someone who is currently a professor who is advising a PhD student", it suffices to evaluate Q on I: the resulting node $\{2, 3\}$ is exactly the set of nodes resulting from evaluating Q on G.*

To obtain a useful structural index, the grouping of nodes must obviously be done intelligently, in such a way that the right information can be retrieved from the index when processing a query. The notions of *simulation* and of *bisimulation* [29] are fundamental for this purpose. Essentially, bisimulation characterizes when two nodes in a graph share exactly the same basic structural characteristics such as labels and neighborhood connectivity. Such nodes are called *bisimilar*. Simulation, in contrast, relates pairs of nodes (v, w) such that w has at least, but possibly more of, the basic structural characteristics of v. Nodes for which there is a simulation from one to the other and vice-versa are called *similar*. Nodes that are bisimilar are also similar, but the converse does not necessarily hold (e.g., [29]).

In the context of XML, simulation- and bisimulation-based

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.
Copyright 20XX ACM X-XXXXX-XX-X/XX/XX ...$15.00.

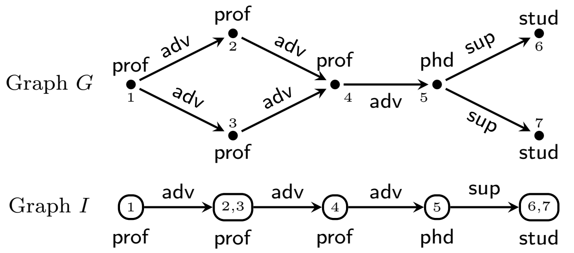


Figure 1: Graphs about academic relations between professors, phd students, and bachelor students, with advisor-of and supervises relationships. Graph I is a simulation-based structural index for G.

structural indexes are obtained by grouping together nodes in the input graph that are similar (respectively, bisimilar). These indexes are known to be *covering* for different fragments of the XPath query language [11,22,27]. That is, given a query in the fragment, its evaluation on the structural index will provide exactly the nodes that would be returned had the query been evaluated on the original data. In Figure 1, for example, the index I is actually a simulation-based index obtained by grouping together the similar nodes in G. (And, as Example 1 has already illustrated, certain queries can be immediately answered on I instead of G.)

Variations of this idea underlying structural indexing have also been used in graph data management to compress [5,10] graph-structured datasets, as well as aid in query processing [18, 25, 30], and data analytics [9].

Given the numerous successful applications of structural indexing in graph databases, one may ask the question: *Is it possible to extend structural indexing from graph databases to arbitrary relational databases?* In this paper and companion work [24, 25], we embark on a formal study of this question, and show that it has an affirmative answer, both from a theoretical and practical perspective.

General methodology. Our study follows the methodology proposed by Fletcher et al. [11] for the design of covering structural indexes for a given target query language $\mathcal{Q}$. This methodology requires the development of the following three components.

(1) A language-independent *structural* characterization of query invariance, characterizing when data objects (in our setting: relational tuples) cannot be distinguished by any query in the target query language $\mathcal{Q}$.

(2) An efficient algorithm to group together data objects that cannot be distinguished by any query in the target language $\mathcal{Q}$.

(3) A data structure (i.e., the index) that exploits this grouping to support query answering by means of the index instead of reverting to the full database.

In this paper, we focus on the conjunctive queries as our target query language, and devote our study to the structural characterization required for component (1). Components (2) and (3) are developed in companion work [24,25]. Actually, we will focus on those conjunctive queries that "select" tuples in the input database rather than compute new tuples. Our focus on this fragment of the conjunctive queries as the target language instead of all conjunctive queries is motivated by the fact that, at least for the purpose of obtaining succinct structural indexes, the class of all conjunctive queries is too large.

To clarify this claim, we note that in graph databases there is a known trade-off between the arity of queries in $\mathcal{Q}$ and the size of the corresponding structural indexes: the index size increases with the arity. In graph databases, for example, $\mathcal{Q}$ is usually a language of node-selecting (i.e., unary) queries. In this setting, the data objects in the methodology of Fletcher et al. are nodes; the structural characterization is given by (bi)similarity; and the structural index data structure is built from the groups of indistinguishable nodes, as illustrated in Example 1. The groups of indistinguishable nodes are necessarily disjoint. Therefore, there can be at most as many groups as there are nodes in the input graph, and, hence, the structural index is always guaranteed to be at most the size of the input graph (although usually much smaller).

Now consider the setting where $\mathcal{Q}$ is a class of k-ary graph queries ($k \geq 2$) instead. Milo and Suciu have shown that essentially the same approach as before can be used to build structural indexes for $\mathcal{Q}$ [22]. However, the data objects become k-tuples of nodes; the structural characterization of indistinguishability is a generalization of (bi)simulation to k-tuples; and the structural index is composed of the groups of indistinguishable k-tuples. Essentially, we are no longer building a summary of the input graph, but a summary of the possible output space of queries in $\mathcal{Q}$—which can be vastly larger than the input graph. In particular, since the number of k-tuples for $k \geq 3$ significantly exceeds the size of the input graph, the number of groups of indistinguishable k-tuples (and hence, the index) exceeds the size of the input graph in practice. Clearly, this defeats the purpose of the structural index as a succinct graph summary.

Since an analogous reasoning applies to the relational setting, we are therefore not interested in a structural characterization of indistinguishability that applies to all conjunctive queries (of arbitrary arity), but in a characterization that is applicable to those conjunctive queries that "select" tuples in the input database. In the literature, these conjunctive queries are known as the *strict* (or *variable-guarded*) conjunctive queries [12]. Formally, a rule-based conjunctive query is strict if all variables in the head occur together in a single atom in the body. (See also Section 2.)

Our focus on the strict conjunctive queries as the target query language implies that we will not be able to answer non-strict queries on the structural index directly. Nevertheless, we show in companion work that query processing of all conjunctive queries can benefit from the presence of these indexes [24,25]. (See also Section 5.)

Overview of approach and main result. What is a good notion of indistinguishability by strict conjunctive queries? It is well known that all conjunctive queries (strict and non-strict) are invariant under homomorphisms (i.e., structure preserving functions from databases to databases), in the following sense.[1]

[1] For the formal development in this paper, it will be convenient to focus on the conjunctive queries that do not mention any constants. All results can be extended to account for the presence of constants, much in the same way as e.g., the classical result on genericity in relational databases can be extended to C-genericity, preserving constants in the finite set C [3].

Theorem 2. *For all databases db_1 and db_2 and all tuples $\overline{a}_1$ and $\overline{a}_2$, if there exists a homomorphism f from db_1 to db_2 such that $f(\overline{a}_1) = \overline{a}_2$, then for every conjunctive query Q, if $\overline{a}_1 \in Q(db_1)$ then also $\overline{a}_2 \in Q(db_2)$.*

Moreover, a seminal result by Rossman states that invariance under homomorphisms is actually a characterization of the conjunctive queries (modulo union).

Theorem 3 ([28])**.** *A query expressible in first order logic (FO) is invariant under homomorphisms on finite structures if, and only if, it is equivalent in the finite to a union of conjunctive queries.*

Homomorphisms are hence the "right" notion for indistinguishability by (strict and non-strict) conjunctive queries on relational databases (which are finite by definition). Unfortunately, however, for component (2) in the methodology of Fletcher et al., we need to be able to *efficiently* group together indistinguishable data objects. This means that we need to group together homomorphic database tuples. Yet, the problem of deciding, given two databases db_1 and db_2 and tuples $\overline{a} \in db_1$ and $\overline{b} \in db_2$, whether there is a homomorphism from db_1 to db_2 that maps $\overline{a}$ to $\overline{b}$, is well-known to be NP-complete [6, 19]. This hence precludes an efficient grouping algorithm.

This raises the question: *is it possible to isolate a useful fragment of the strict conjunctive queries that admits a tractable structural characterization of indistinguishability?* There are two natural research directions one can take to answer this question. First, one can look at known fragments of the conjunctive queries that have previously been shown to be well-behaved in other contexts. Perhaps one of these fragments admits a tractable structural characterization? An immediate candidate fragment here are of course the acyclic conjunctive queries. Second, one can look at generalizations of simulation and bisimulation from graphs to arbitrary relational databases, and investigate what fragment of the strict conjunctive queries are invariant under these generalizations. We will actually show that, in a sense to be made precise below, both directions lead to the same answer.

We are hereby inspired by the following known results. First, in their seminal work, Andréka et al. [2] and Otto [23] have shown that the so-called *guarded fragment* (GF for short) of FO is characterized by *guarded bisimulation*, a generalization of traditional bisimulation to relational databases. This yields the following variation of Theorems 2 and 3.

Theorem 4 ([2, 23])**.** *The GF is invariant under guarded bisimulation. Moreover, a query expressible in FO is invariant under guarded bisimulation on finite structures if, and only if, it is equivalent in the finite to a query expressible in GF.*

Subsequently, Flum et al. [12] have established expressive equivalence between the strict formulae in GF and the strict formulae in the acyclic fragment of FO, and Leinders et al. [20] have shown that strict GF corresponds to the semi-join variant of Codd's relational algebra. In parallel, Gottlob et al. [13] have established the expressive equivalence of the primitive positive fragment of strict GF and the acyclic strict conjunctive queries.

These results hint at the possibility that the acyclic strict conjunctive queries are invariant under a variant of guarded bisimulation. To investigate this, we introduce *guarded simulation*, a variant of guarded bisimulation that is obtained from guarded bisimulation analogously to how classical simulation is obtained from classical bisimulation by dropping the back condition. We show that all acyclic strict conjunctive queries are invariant under guarded simulation.

We actually establish the following stronger result that characterizes the first order definable queries invariant under guarded simulation in terms of a fragment of the conjunctive queries that we call the *freely acyclic conjunctive queries* (FACQs for short).

Theorem 5 (Main Result)**.** *FACQs are invariant under guarded simulation. Moreover, a query expressible in FO is invariant under guarded simulation on finite structures if, and only if, it is equivalent in the finite to a union of FACQs.*

Essentially, a (not necessarily boolean) conjunctive query of the form $head \leftarrow body$ is freely acyclic if the boolean conjunctive query $() \leftarrow head, body$ is acyclic. The classes of acyclic conjunctive queries and freely acyclic conjunctive queries are incomparable (see also Section 3.1.) Nevertheless, the freely acyclic conjunctive queries do include all acyclic strict conjunctive queries, as well as all boolean acyclic conjunctive queries. Our result hence complements the characterization of strict acyclic FO in terms of guarded bisimulation. Since guarded bisimulation and guarded simulation are computable in polynomial time (e.g., [15, 17, 24]), this hence identifies a useful fragment of the (strict) conjunctive queries with a tractable structural characterization.

Fact simulation. To get a better understanding of guarded simulation, we give an alternative definition that we call *fact simulation*. As we have already stated above, guarded simulation is essentially obtained from guarded bisimulation by dropping the back condition. Since guarded bisimulation can be seen as a variant of Ehrenfeucht-Fraissé games, guarded simulation can hence be seen as a game in which the Spoiler always puts pebbles in the same structure (according to a given discipline), and the Duplicator has to construct a partial homomorphism of the placed pebbles in the other structure (within game rule bounds). Inherently, the game configurations are thus more complex than in traditional simulation on graphs. *Fact simulation*, on the other hand, can be viewed as a game where the Spoiler does not put pebbles in one structure but selects entire facts in that structure. The Duplicator is not concerned with constructing partial homomorphisms directly, but responds with facts in the other structure, making sure only to mimic the moves that the Spoiler makes from one fact to the next. In this respect, fact simulation is closer in spirit to classical simulation on graphs which, viewed as a game, has a Spoiler that always occupies a single node in one graph and a Duplicator that has to respond with nodes in the other graph, mimicking the Spoiler's moves. We therefore feel that fact simulation provides a simpler alternative definition to guarded simulation.

Approximations. In graph data management, it is known that, if there are few nodes in graph G that are (bi)similar, then the corresponding structural index of G may be of the same size as G itself, and hence be too large to act as a succinct summary of the structure of G [18]. In such situations, it has been proposed to *approximate* (bi)similarity, and to group nodes in the index with respect to these approximations instead of with respect to full (bi)similarity [11, 18].

Since these concerns about index size can be transferred to the relational setting, it is hence useful to develop approximate versions of guarded simulation.

To this end, we introduce approximations of fact simulation analogously to how approximations of classical simulation are defined. These approximations are proved to be tightly linked to invariance of freely acyclic conjunctive queries whose join tree is of bounded height. In companion work [24,25] we show that these approximations can both be efficiently computed and used to engineer practical guarded simulation-based structural indexes for relational query engines operating on Semantic Web data.

Contributions and organization. In summary, our contributions are as follows. (1) We introduce guarded simulation as a variant of guarded bisimulation, and prove the characterization stated in Theorem 5 (Section 3). (2) We introduce fact simulation as an alternative definition of guarded simulation, and show that approximations of fact simulation are tightly linked to invariance of freely acyclic conjunctive queries of bounded height (Section 4). (3) We show how structural indexes based on (approximations of) fact simulations can be defined (Section 5).

We begin, however, in Section 2 with introducing the required background.

2. PRELIMINARIES

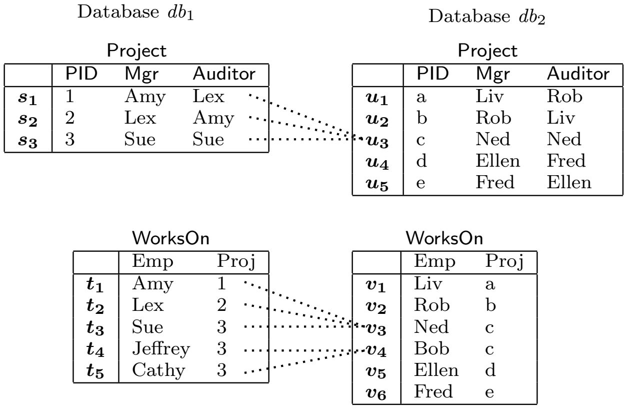
Atoms, facts, and databases. From the outset, we assume given a fixed universe $\mathcal{U}$ of atomic data values, a fixed universe $\mathcal{V}$ of variables, and a fixed set $\mathcal{S}$ of relation symbols, all infinite and pairwise disjoint. We call atomic data values and variables collectively *terms*. Every relation symbol $r \in \mathcal{S}$ is associated with a natural number called the *arity* of r. An *atom* (respectively a *fact*) is an expression of the form $r(a_1, \dots, a_k)$ with $r \in \mathcal{S}$ a relation symbol; k the arity of relation symbol r; and each of the $a_1, \dots, a_k \in \mathcal{V}$ a variable (respectively an atomic data value). A *relational database* over $\mathcal{S}$ is a finite set *db* of facts.

Notation. In what follows, we denote the set of all terms (respectively variables, respectively data values) occurring in a mathematical object X (such as, e.g. an atom, fact, or set of atoms and facts) by $\mathit{terms}(X)$ (resp. $\mathit{var}(X)$, resp. $\mathit{val}(X)$). We write $\mathrm{rel}(\boldsymbol{a})$ for the relation symbol r of atom or fact $\boldsymbol{a} = r(a_1, \dots, a_k)$. We write $|\boldsymbol{a}|$ for the arity k of $\mathrm{rel}(\boldsymbol{a})$ and $\boldsymbol{a}.i$ for the i-th term a_i in $\boldsymbol{a}$, provided $1 \le i \le |\boldsymbol{a}|$. We denote tuples $(a_1, \dots, a_k)$ as $\overline{a}$, and give the natural semantics to $|\overline{a}|$ and $\overline{a}.i$.

The *restriction* of a set A of atoms or facts to a set of terms $X \subseteq \mathcal{U} \cup \mathcal{V}$, denoted $A|_X$, consists of all atoms or facts in A built only from terms in X, $A|_X := \{\boldsymbol{a} \in A \mid \mathit{terms}(\boldsymbol{a}) \subseteq X\}$.

Functions $f\colon X \to Y$ with X and Y sets of terms are extended point-wise to atoms, facts, tuples of terms, and sets thereof. For instance, if $\boldsymbol{a} = r(a_1, \dots, a_k)$ and $\mathit{terms}(\boldsymbol{a}) \subseteq X$ then $f(\boldsymbol{a}) = r(f(a_1), \dots, f(a_k))$. We denote by $f|_Z$ the restriction of the domain of f to the set $X \cap Z$ and, extending this notation to atoms and facts, denote by $f|_{\boldsymbol{a}}$ the restriction of the domain of f to the set $X \cap \mathit{terms}(\boldsymbol{a})$.

We range over atoms by boldface letters drawn from the beginning of the alphabet $(\boldsymbol{a}, \boldsymbol{b}, \dots)$ and facts by boldface letters from the end of the alphabet $(\boldsymbol{r}, \boldsymbol{s}, \dots)$.



Database db_1

Project

	PID	Mgr	Auditor
$\boldsymbol{s_1}$	1	Amy	Lex
$\boldsymbol{s_2}$	2	Lex	Amy
$\boldsymbol{s_3}$	3	Sue	Sue

WorksOn

	Emp	Proj
$\boldsymbol{t_1}$	Amy	1
$\boldsymbol{t_2}$	Lex	2
$\boldsymbol{t_3}$	Sue	3
$\boldsymbol{t_4}$	Jeffrey	3
$\boldsymbol{t_5}$	Cathy	3

Database db_2

Project

	PID	Mgr	Auditor
$\boldsymbol{u_1}$	a	Liv	Rob
$\boldsymbol{u_2}$	b	Rob	Liv
$\boldsymbol{u_3}$	c	Ned	Ned
$\boldsymbol{u_4}$	d	Ellen	Fred
$\boldsymbol{u_5}$	e	Fred	Ellen

WorksOn

	Emp	Proj
$\boldsymbol{v_1}$	Liv	a
$\boldsymbol{v_2}$	Rob	b
$\boldsymbol{v_3}$	Ned	c
$\boldsymbol{v_4}$	Bob	c
$\boldsymbol{v_5}$	Ellen	d
$\boldsymbol{v_6}$	Fred	e

Figure 2: Two company databases. For future reference, facts are labeled with identifiers $(\boldsymbol{s_1}, \boldsymbol{s_2}, \dots)$. The dotted lines indicate a fact simulation (Section 4) between db_1 and db_2.

Definition 6. *If $\boldsymbol{s}$ and $\boldsymbol{t}$ are two facts (resp., atoms), then the* equality type *of $\boldsymbol{s}$ and $\boldsymbol{t}$, denoted $\mathit{eqtp}(\boldsymbol{s}, \boldsymbol{t})$ is the set*

$$\{(i,j) \mid \boldsymbol{s}.i = \boldsymbol{t}.j,\ \text{with } 1 \le i \le |\boldsymbol{s}|, 1 \le j \le \boldsymbol{t}\}.$$

The equality type between two facts hence records the positions on which the facts share a value. To illustrate, referring to the facts in the database db_1 of Figure 2, we have $\mathit{eqtp}(\boldsymbol{s_1}, \boldsymbol{t_1}) = \{(1,2), (2,1)\}$.

Homomorphisms and isomorphisms. Let A and B be sets of facts and atoms. A function $f\colon X \to Y$ is a *homomorphism* from A to B if $\mathit{terms}(A) \subseteq X$ and $f(A) \subseteq B$. It is a *partial homomorphism* if $f(A|_X) \subseteq B$. It is an *isomorphism* if f is bijective, $\mathit{terms}(A) \subseteq X$, and $f(A) = B$.

Conjunctive queries. A (rule-based) conjunctive query (CQ for short) Q consists of a rule of the form

$$Q : \mathit{ans}(\overline{x}) \leftarrow \boldsymbol{a_1},\ \dots, \boldsymbol{a_n},$$

with $\mathit{ans}(\overline{x}), \boldsymbol{a_1}, \dots, \boldsymbol{a_n}$ atoms ($n \ge 0$). The set $\{\boldsymbol{a_1}, \dots, \boldsymbol{a_n}\}$ is called the *body* of Q and is denoted by $\mathit{body}(Q)$. The atom $\mathit{ans}(\overline{x})$ is called the *head* of Q and is denoted by $\mathit{head}(Q)$. It is required that $\mathit{var}(\mathit{head}(Q)) \subseteq \mathit{var}(\mathit{body}(Q))$. We sometimes write $Q(\overline{x})$ to indicate that $\overline{x}$ is the tuple of variables in the head of Q.

A *valuation* μ is a partial function $\mu\colon \mathcal{V} \to \mathcal{U}$. A valuation is an *embedding* of set of atoms A in a database *db* if it is a homomorphism from A to *db*. A valuation μ is an *embedding* of a conjunctive query Q in a database *db* if it is an embedding of $\mathit{body}(Q)$ in *db*. The result of conjunctive query $Q(\overline{x})$ on database *db* is the set $Q(db) := \{\mu(\overline{x}) \mid \mu \text{ is an embedding of } Q \text{ in } db\}$.

Example 7. *Consider the following CQ Q:*

$$\mathit{ans}(\mathit{emp}) \leftarrow \mathsf{Project}(\mathit{pid}, \mathit{mgr}, \mathit{mgr}), \mathsf{WorksOn}(\mathit{pid}, \mathit{emp}).$$

When applied to the databases of Figure 2 it retrieves all the employees who work on a project that is managed and audited by the same person.

A *union of conjunctive queries* (UCQ for short) is a finite set φ of CQs, all with the same head, say $\mathit{ans}(\overline{x})$, which is called the head of φ. The result of UCQ φ on database *db* is the set $\varphi(db) := \bigcup\{Q(db) \mid Q \in \varphi\}$.

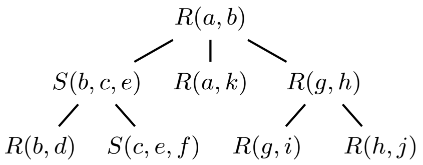


Figure 3: A join tree for the query in Example 9.

An atom or fact $\boldsymbol{a}$ is *boolean* if it does not mention any term. A CQ is boolean if its head is. A CQ Q is *strict* if all variables in the head occur together in a single atom in the body. To illustrate, the query from Example 7 is strict, but the following is not:

$$\begin{aligned} ans(\text{pid}, \text{emp}, \text{mgr}) \leftarrow\ & \mathsf{Project}(\text{pid}, \text{mgr}, \text{mgr}),\\ & \mathsf{WorksOn}(\text{pid}, \text{emp}). \end{aligned}$$

Minimality. A CQ Q is contained in a CQ Q', denoted $Q \subseteq Q'$, if $Q(db) \subseteq Q'(db)$ for all databases db. Q is equivalent to Q', denoted $Q \equiv Q'$ if $Q \subseteq Q'$ and $Q' \subseteq Q$.

A CQ Q is *minimal* if there does not exist an equivalent conjunctive query with fewer atoms in the body. A UCQ φ is minimal if all of its CQs are minimal, and, moreover, $Q \not\subseteq Q'$ for all distinct $Q, Q' \in \varphi$. Obviously, every UCQ has an equivalent one that is minimal.

Acyclicity. The acyclic conjunctive queries were recognized early in the history of database theory as an important subclass of the conjunctive queries that have a PTIME query evaluation problem under combined complexity [1,32]. There are many equivalent definitions of when a conjunctive query is acyclic. Here, we will use two different versions: a definition based on join trees and a definition based on acyclic hypergraphs.

Definition 8 (Join tree)**.** *Let A be a finite set of atoms. A* join tree *for A is a tree T (i.e., a connected acyclic undirected graph) whose nodes are the atoms in A such that, whenever the same variable x occurs in two atoms $\boldsymbol{a}$ and $\boldsymbol{b}$ in A, then x occurs in each atom on the unique path linking $\boldsymbol{a}$ and $\boldsymbol{b}$. A join tree for a conjunctive query Q is a join tree for body(Q).*

Example 9. *Consider the following query:*

$$\begin{aligned} Q : ans(a,b) \leftarrow & R(a,b), S(b,c,e), R(b,d), S(c,e,f),\\ & R(a,k), R(g,h), R(g,i), R(h,j). \end{aligned}$$

A join tree for Q is shown in Figure 3.

Definition 10. *A conjunctive query is* acyclic *if it has a join tree. It is* cyclic *otherwise.*

The query Q from Example 9 is hence acyclic.

Hypergraph acyclicity. A *hypergraph* is a pair $(\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ is a set of nodes and $\mathcal{E}$ is a set of edges (also called hyperedges), which are arbitrary nonempty subsets of $\mathcal{N}$. If Q is a conjunctive query, we define the hypergraph $\mathcal{H}(Q) = (\mathcal{N}, \mathcal{E})$ associated to Q as follows. The set of nodes $\mathcal{N}$ consist of all variables occurring in Q. For each atom $\boldsymbol{a}$ in the body of Q, the set $\mathcal{E}$ contains a hyperedge consisting of all variables occurring in $\boldsymbol{a}$.

It is well-known that a conjunctive query is acyclic if and only if $\mathcal{H}(Q)$ is acyclic. Here, acyclicity of a hypergraph, also referred to as α-acyclicity by Fagin [8], is defined as follows.

A *path* from a node s to a node t in a hypergraph $(\mathcal{N}, \mathcal{E})$ is a sequence of $k \geq 1$ edges $E_1, \ldots, E_k \in \mathcal{E}$ such that: $s \in E_1$, $t \in E_k$, and $E_i \cap E_{i+1} \neq \emptyset$, for every $1 \leq i < k$. Two nodes (or two edges) are *connected* if there is a path from one to the other. A set of nodes (or a set of edges) is connected if all of its pairs of nodes (resp. edges) are connected.

The *reduction* of the hypergraph $(\mathcal{N}, \mathcal{E})$ is obtained by removing from $\mathcal{E}$ each edge that is a proper subset of another edge. A hypergraph is *reduced* if it is equal to its reduction.

Given a hypergraph $(\mathcal{N}, \mathcal{E})$, the set of *partial edges generated by a set of nodes $M \subseteq \mathcal{N}$* is obtained by intersecting the edges in $\mathcal{E}$ with M. That is, the set of partial edges generated by M is the reduction of $\{E \cap M \mid E \in \mathcal{E}\} - \{\emptyset\}$. A set B is said to be a *node-generated set of partial edges* if B is the set of partial edges generated by $M \subseteq \mathcal{N}$, for some M.

Let $\mathcal{F}$ be a connected, reduced set of partial edges, and let E and F be in $\mathcal{F}$. Let $G = E \cap F$. We say that G is an *articulation set* of $\mathcal{F}$ if the set of partial edges $\{H - G \mid H \in \mathcal{F}\} - \{\emptyset\}$ is not connected.

Definition 11 (Hypergraph Acyclicity)**.** *A* block *of a reduced hypergraph is a connected, node-generated set of partial edges with no articulation set. A block is trivial if it contains less than two members. A reduced hypergraph is* acyclic *if all its blocks are trivial. A hypergraph is said to be acyclic if its reduction is.*

Observe that no block can be formed from exactly two partial edges. Indeed, these two edges are either disconnected or their intersection forms an articulation set.

Example 12. *Consider the conjunctive query*

$$Q_2\colon\ ans() \leftarrow R(a,b,c), R(a,b,d), R(a,c,d), R(b,c,d).$$

Its hypergraph $\mathcal{H}(Q_2)$ consists of the following edges:

$$\begin{aligned} E_1 &= \{a,b,c\} & E_2 &= \{a,c,d\}\\ E_3 &= \{a,b,d\} & E_4 &= \{b,c,d\} \end{aligned}$$

Note that $\mathcal{H}(Q_2)$ itself equals the set of partial hyperedges of $\mathcal{H}(Q_2)$ generated by the set $\{a,b,c,d\}$. This set is clearly connected and reduced. Furthermore, it has no articulation set, and it is not trivial. Therefore, $\mathcal{H}(Q_2)$ itself forms a non-trivial block of $\mathcal{H}(Q_2)$. Hence $\mathcal{H}(Q_2)$ is cyclic, and so is Q_2.

3. STRUCTURAL CHARACTERIZATION

Guarded bisimulation is a generalization of classical bisimulation to relational databases introduced by Andréka et al. [2]. (A formal definition of guarded bisimulation is provided in Appendix A for completeness.) Analogously to modal bisimulation, guarded bisimulation is formulated by means of back and forth conditions. In this section, we introduce guarded simulation as a variant of guarded bisimulation without the back condition, and prove Theorem 5. Towards this, we start with the definition of free acyclicity.

3.1 Free Acyclicity

The *extension* of CQ Q, denoted by Q^+, is the CQ obtained by adding *head*(Q) as an atom to the body.

Example 13. *Here are some CQs and their extensions.*

$$\begin{aligned}
Q_1\colon\ & \mathit{ans}(x,z) \leftarrow R(x,y), R(y,z)\\
{Q_1}^+\colon\ & \mathit{ans}(x,z) \leftarrow R(x,y), R(y,z), \mathit{ans}(x,z)\\
Q_2\colon\ & \mathit{ans}(x,y,z) \leftarrow R(x,y), R(y,z), R(x,z)\\
{Q_2}^+\colon\ & \mathit{ans}(x,y,z) \leftarrow R(x,y), R(y,z), R(x,z), \mathit{ans}(x,y,z)
\end{aligned}$$

Observe in particular that Q^+ is always strict.

Definition 14. *CQ Q is* freely acyclic *if Q^+ is acyclic.*

The classes of acyclic CQs (ACQs for short) and freely acyclic CQs (FACQs) are incomparable. Indeed, Q_1 in Example 13 is acyclic but not freely acyclic while Q_2 is freely acyclic, but not acyclic. It is readily verified, however, that for strict queries the two classes coincide (observe that in this case the reduction of $\mathcal{H}(Q)$ coincides with the reduction of $\mathcal{H}(Q^+)$). Since every boolean CQ is strict, the two classes hence also coincide on the boolean queries.

Discussion. The class of FACQs may seem strange at first sight. We find, however, that, from a theoretical viewpoint, they form a natural class of queries since they coincide with the queries expressible in the primitive positive version of the well-known and well-studied *Guarded Fragment* [2, 12, 13, 15, 20] (GF for short) of FO. The formulas of the primitive positive guarded fragment (denoted $\{\wedge, \exists\}$-GF) are built by closing the set of all atoms under conjunction ($\wedge$) and guarded quantification of the form $\exists \overline{x}(\boldsymbol{a} \wedge \varphi)$ where $\overline{x}$ is a finite non-empty sequence of variables, $\boldsymbol{a}$ is an atom, φ is a formula in $\{\wedge, \exists\}$-GF, and all free variables of φ must occur in $\boldsymbol{a}$. To illustrate, $\exists z(R(x,y,z) \wedge S(x,y) \wedge S(x,z))$ is a formula in $\{\wedge, \exists\}$-GF, as is $S(x,y) \wedge S(y,z)$, but $\exists z(S(x,y) \wedge S(y,z))$ is not.

Proposition 15. *A query is expressible by a formula in $\{\wedge, \exists\}$-GF if, and only if, it is equivalent to a FACQ.*

While it is generally acknowledged that ACQs are practically prevalent, it is not clear that the same can be said for FACQs. To get an initial insight into the practical relevance of FACQs, we have analyzed a log of SPARQL queries posed to the DBPedia RDF database between April and July 2010. SPARQL is the standard query language for data in RDF format [26]. In particular, SPARQL queries that use only composition and FILTER in their patterns are CQs. The log contains more than 3 million queries in total, of which over a million remain after duplicate elimination. In total, 2061469 queries in the log are CQs in the above sense (688824 after duplicate removal). Of these, 99.9% are ACQs, and a similarly high number (99.7%) turn out to be FACQs. While of course more real-world query workloads should be investigated to fully establish the practical relevance of FACQs, this at least indicates that such relevance cannot be discarded offhand.

3.2 Guarded Simulation

To formally introduce guarded simulation, we first require the following notions.

Let $X \subseteq \mathcal{U} \cup \mathcal{V}$ be a set of terms and let A be a set of atoms and facts. X is called *guarded* in A if there is some atom or fact $a \in A$ with $X = \mathit{terms}(a)$. For instance, in the database db_1 of Figure 2, the set $\{1, \mathrm{Amy}\}$ is guarded by $\boldsymbol{t_1}$, and $\{3, \mathrm{Sue}\}$ is guarded by $\boldsymbol{s_3}$. In contrast, the set $\{1, 3\}$ is not guarded in db_1, since there is no database fact built from exactly these two values.

We say that two functions $f : X \to Y$ and $g : X' \to Y'$ agree on $X \cap X'$ if $f(x) = g(x)$ for all $x \in X \cap X'$. The notion of guarded simulation is then defined as follows:

Definition 16 (Guarded simulation)**.** *Let db_1 and db_2 be databases. A* guarded simulation *from db_1 to db_2 is a non-empty set $\mathcal{S}$ of finite partial homomorphisms from db_1 to db_2 such that the following forth condition is satisfied.*

- *For every $f\colon X \to Y \in \mathcal{S}$ and for every set X' guarded in db_1, there exists a partial homomorphism $g\colon X' \to Y' \in \mathcal{S}$ such that g and f agree on $X \cap X'$. (**Guarded Simulation Forth**).*

We say that $(\mathit{db}_1, \overline{a})$ is *guardedly simulated by* $(\mathit{db}_2, \overline{b})$, denoted $\mathit{db}_1, \overline{a} \preceq_g \mathit{db}_2, \overline{b}$, if there exists a guarded simulation $\mathcal{S}$ from db_1 to db_2 and a partial homomorphism $h \in \mathcal{S}$ such that $h(\overline{a}) = \overline{b}$. Note that hence $\overline{a}$ and $\overline{b}$ must be of the same arity.

Example 17. *Let db_1 and db_2 be the two databases shown in Figure 2. The following set $\mathcal{S}$ of partial homomorphisms is a guarded simulation from db_1 to db_2:*

$$\begin{aligned}
(1, \mathrm{Amy}, \mathrm{Lex}) &\mapsto (\mathrm{c}, \mathrm{Ned}, \mathrm{Ned})\\
(2, \mathrm{Lex}, \mathrm{Amy}) &\mapsto (\mathrm{c}, \mathrm{Ned}, \mathrm{Ned})\\
(3, \mathrm{Sue}) &\mapsto (\mathrm{c}, \mathrm{Ned})\\
(\mathrm{Amy}, 1) &\mapsto (\mathrm{Ned}, \mathrm{c})\\
(\mathrm{Lex}, 2) &\mapsto (\mathrm{Ned}, \mathrm{c})\\
(\mathrm{Jeffrey}, 3) &\mapsto (\mathrm{Bob}, \mathrm{c})\\
(\mathrm{Cathy}, 3) &\mapsto (\mathrm{Bob}, \mathrm{c}).
\end{aligned}$$

Let us check the guarded simulation forth property for the particular partial homomorphism

$$f\colon (1, \mathrm{Amy}, \mathrm{Lex}) \mapsto (\mathrm{c}, \mathrm{Ned}, \mathrm{Ned}).$$

We must consider all guarded sets X' of db_1:

- *if X' is $\{1, \mathrm{Amy}, \mathrm{Lex}\}$ we choose g as f;*
- *if X' is $\{2, \mathrm{Lex}, \mathrm{Amy}\}$ we choose g as $(2, \mathrm{Lex}, \mathrm{Amy}) \mapsto (\mathrm{c}, \mathrm{Ned}, \mathrm{Ned})$ (clearly, the intersection of X and X' is $\{\mathrm{Lex}, \mathrm{Amy}\}$ and both f and g map Lex $\mapsto$ Ned and Amy $\mapsto$ Ned);*
- *if X' is $\{\mathrm{Amy}, 1\}$ we choose g as $(\mathrm{Amy}, 1) \mapsto (\mathrm{Ned}, \mathrm{c})$ (the intersection of X and X' is $\{\mathrm{Amy}, 1\}$ and both f and g map Amy $\mapsto$ Ned and $1 \mapsto \mathrm{c}$);*
- *if X' is $\{\mathrm{Lex}, 2\}$ we choose g as $(\mathrm{Lex}, 2) \mapsto (\mathrm{Ned}, \mathrm{c})$ (the intersection of X and X' is $\{\mathrm{Lex}\}$ and both f and g map Lex $\mapsto$ Ned);*
- *in all other cases, the intersection of X' and X is empty and we are free to choose $g\colon X' \to Y'$ (since the intersection is empty, any such g will do).*

Using analogous arguments, the guarded simulation forth property can be checked for the other partial homomorphisms. Hence, $\mathcal{S}$ is a guarded simulation. Moreover, since $f(\boldsymbol{s_1}) = \boldsymbol{u_3}$, we have $\mathit{db}_1, (1, \mathrm{Amy}, \mathrm{Lex}) \preceq_g \mathit{db}_2, (\mathrm{c}, \mathrm{Ned}, \mathrm{Ned})$.

Readers familiar with the work by Chen and Dalmau [7] may observe that a guarded simulation is nothing more than a so-called compact winning strategy for the existential k-cover game between two relational structures, for the special case where $k = 1$. Chen and Dalmau link the existence of winning strategies for the k-cover game to invariance by conjunctive queries of so-called *coverwidth* (also known as generalized hypertree width) at most k. Since it is known that the conjunctive queries of coverwidth ≤ 1 are exactly the ACQs (e.g., [7,13]), it is not difficult to obtain the following from their results.

Proposition 18. *The following are equivalent.*

- $db_1, \overline{a} \preceq_g db_2, \overline{b}$
- *For all FACQs Q, if $\overline{a} \in Q(db_1)$ then $\overline{b} \in Q(db_2)$.*

3.3 Characterizing invariance under guarded simulation

Proposition 18 implies that the FACQs are invariant under guarded simulation. It also implies that any FO definable query that is equivalent to a union of FACQs must be invariant under guarded simulation. To obtain Theorem 5, therefore, it remains to prove that any FO definable query that is invariant under guarded simulation is equivalent to a union of FACQs. We devote the rest of this section to this proof, which starts with the following observation.

Proposition 19. *If φ is a FO formula invariant under guarded simulation (on finite databases) then φ is equivalent (in the finite) to a UCQ.*

Proof. Every homomorphism gives rise to a guarded simulation. Indeed, if h is a homomorphism from db_1 to db_2 that maps $\overline{a}$ to $\overline{b}$ then it is readily verified that the set $\mathcal{S} := \{h|_{\overline{a}}\} \cup \{h|_X \mid X \text{ guarded in } db_1\}$ is a guarded simulation from $db_1 \cup \{ans(\overline{a})\}$ to $db_2 \cup \{ans(\overline{b})\}$. Hence $db_1, \overline{a} \preceq_g db_2, \overline{b}$, since $h|_{\overline{a}} \in \mathcal{S}$ maps $\overline{a}$ to $\overline{b}$. Then since φ is invariant under guarded simulations, it is also invariant under homomorphisms. By Rossman's theorem (Theorem 3), φ is hence equivalent to a UCQ. □

Now fix throughout the remainder of this section an FO formula $\varphi(\overline{x})$ invariant under guarded simulation. By Proposition 19 we may assume w.l.o.g. that φ is a UCQ. Furthermore, we may assume w.l.o.g. that this UCQ is minimal.

Now assume for the purpose of contradiction that no union of FACQs expresses φ. Then in particular there exists some CQ $Q(\overline{x})$ in φ that is not freely acyclic, i.e., Q^+ is cyclic. From Q we will construct pairs $(canondb, \overline{a})$ and $(unrolldb, \overline{b})$ such that $canondb, \overline{a} \preceq_g unrolldb, \overline{b}$ and $\overline{a} \in \varphi(canondb)$ but $\overline{b} \notin \varphi(unrolldb)$. Then obviously, φ is not invariant under guarded simulation, yielding the desired contradiction. The definition of *canondb* and *unrolldb* is as follows.

The canonical database. The database *canondb* is simply what is normally called the "canonical database" (or "frozen" database) for Q in the theory of conjunctive queries. Formally, fix for every variable $x \in Q$ a unique data value $x^* \in \mathcal{U}$ such that the function *freeze* mapping $x \mapsto x^*$ for all $x \in var(Q)$ is a bijection. Let $canondb := freeze(body(Q))$ and $\overline{a} := freeze(\overline{x})$. By construction, *freeze* is an embedding of Q in *canondb*. Therefore,

Lemma 20. *$\overline{a} \in Q(canondb) \subseteq \varphi(canondb)$.*

The unrolled database. Since Q^+ is cyclic the hypergraph $\mathcal{H}(Q^+)$ contains a nontrivial block. Fix such a nontrivial block $\mathcal{B}$, as well as a distinguished hyperedge $F \in \mathcal{B}$. Let $\{x_1, \dots, x_n\}$ be the variables mentioned in Q. We fix a set $U = \{{x_1}^\circ, \dots, {x_n}^\circ, {x_1}^\bullet, \dots, {x_n}^\bullet\} \subseteq \mathcal{U}$ of pairwise distinct values. In what follows, we call ${x_i}^\circ$ the *white colored version* of x_i, and ${x_i}^\bullet$ the *black colored version* of x_i.

Let $var(\mathcal{B})$ denote the set of all variables that are mentioned in the hyperedges of block $\mathcal{B}$. We define for every $V \subseteq var(\mathcal{B})$ the function $clr_V \colon var(Q) \to U$ by:

$$\begin{aligned} clr_V(v) &= v^\circ & & v \notin var(\mathcal{B}) \text{ or } v \in V \\ clr_V(v) &= v^\bullet & & v \in var(\mathcal{B}) \text{ and } v \notin V. \end{aligned}$$

Intuitively, clr_V is a function that maps variables to values by "coloring" the variables. Variables not mentioned in $\mathcal{B}$ are colored white, while a variable v mentioned in $\mathcal{B}$ is colored white if v is in V, and black otherwise.

Definition 21 (Covering)**.** *Let E, E', and V be three sets of variables. We say that E covers E' w.r.t. V, denoted $E \sqsupseteq_V E'$, if $E \cap V \supseteq E' \cap V$.*

We abbreviate $E \sqsupseteq_{var(\mathcal{B})} E'$ by $E \sqsupseteq E'$ and write $E \sqsupset E'$ and $E \sqsupset_V E'$ to denote the corresponding strict relations.

Definition 22 (Maximum intersections)**.** *Let $\mathcal{B}_{\sqsupset_F}$ denote the set of all partial hyperedges $E \in \mathcal{B} \setminus \{F\}$ that have a maximal intersection with F among the hyperedges in $\mathcal{B} \setminus \{F\}$. That is, $\mathcal{B}_{\sqsupset_F}$ consists of all $E \in \mathcal{B} \setminus \{F\}$ for which there does not exist $E' \in \mathcal{B} \setminus \{F\}$ with $E' \sqsupset_F E$. Let $\mathcal{M}_\cap$ be the set of maximum intersections of partial hyperedges of $\mathcal{B} \setminus \{F\}$ with F, $\mathcal{M}_\cap := \{E \cap F \mid E \in \mathcal{B}_{\sqsupset_F}\}$.*

Note that, since $\mathcal{B}$ is nontrivial, the cardinality of $\mathcal{M}_\cap$ is at least 2, and all intersections in $\mathcal{M}_\cap$ are nonempty. Also note that for any $A \in \mathcal{M}_\cap$, we have $F \supsetneq A$ and hence $F \sqsupseteq A$.

Example 23. *Consider the query*

$$Q_1 \colon ans() \leftarrow R(a,b,d), R(c,a,d), S(b,c,d,e), T(e,f), T(f,g).$$

It is readily verified that the set

$$\mathcal{B}_1 = \{\{a,b,d\}, \{b,c,d\}, \{c,a,d\}\}$$

forms a block of $\mathcal{H}({Q_1}^+)$. Consider the hyperedge $F = \{b,c,d\}$ of this block. Then $\mathcal{M}_\cap = \{\{b,d\}, \{c,d\}\}$, resulting from the intersections with the hyperedges $\{a,b,d\}$ and $\{c,a,d\}$ respectively.

Next, consider the query

$$Q_2 \colon ans() \leftarrow R(a,b,c), R(a,b,d), R(a,c,d), R(b,c,d)$$

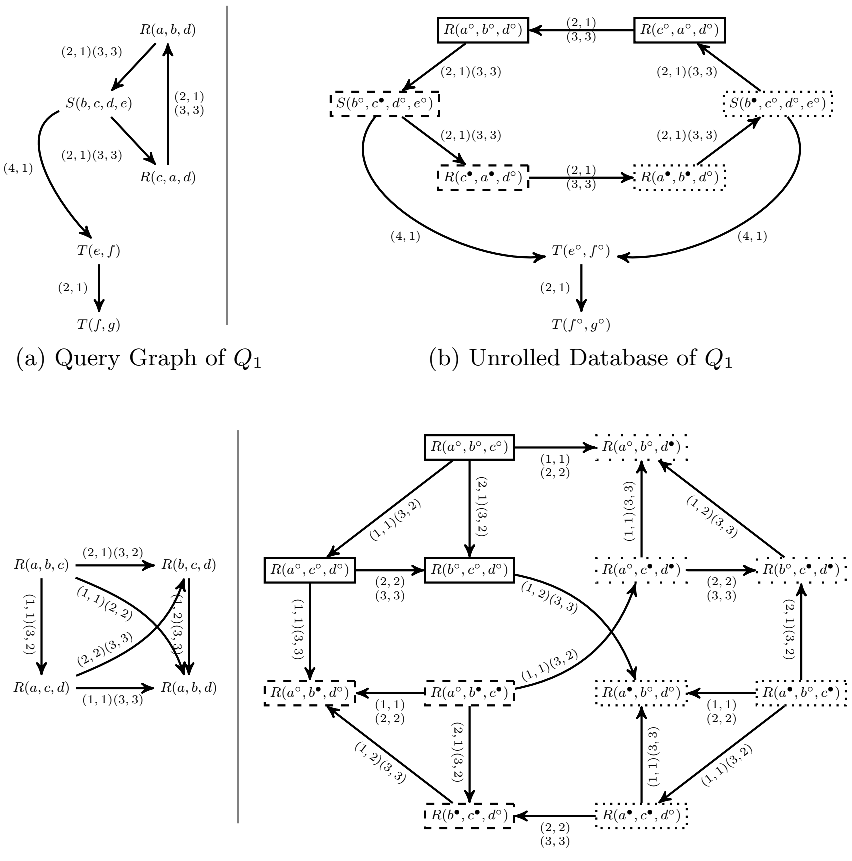
from Example 12. It is readily verified that the set

$$\mathcal{B}_2 = \{\{a,b,c\}, \{a,b,d\}, \{a,c,d\}, \{b,c,d\}\}$$

forms a block of $\mathcal{H}({Q_2}^+)$ (cf., e.g., Example 12). Consider the hyperedge $F = \{a,b,d\}$ of this block. Then $\mathcal{M}_\cap$ is the set $\{\{a,b\}, \{a,d\}, \{b,d\}\}$, resulting from the intersections with the hyperedges $\{a,b,c\}$, $\{a,c,d\}$, and $\{b,c,d\}$ respectively.

We now turn to the construction of *unrolldb*.

Definition 24 (Unrolled database)**.** *Define $\mathcal{F}$ to be the set of functions that contains*


$R(a,b,d)$
$S(b,c,d,e)$
$R(c,a,d)$
$T(e,f)$
$T(f,g)$
$(2,1)(3,3)$
$(2,1)$ $(3,3)$
$(4,1)$
$(2,1)$
$R(a^\circ,b^\circ,d^\circ)$
$R(c^\circ,a^\circ,d^\circ)$
$S(b^\circ,c^\bullet,d^\circ,e^\circ)$
$S(b^\bullet,c^\circ,d^\circ,e^\circ)$
$R(c^\bullet,a^\bullet,d^\circ)$
$R(a^\bullet,b^\bullet,d^\circ)$
$T(e^\circ,f^\circ)$
$T(f^\circ,g^\circ)$
$R(a,b,c)$
$R(b,c,d)$
$R(a,c,d)$
$R(a,b,d)$
$(2,1)(3,2)$
$(1,1)(3,2)$
$(1,1)(2,2)$
$(2,2)(3,3)$
$(1,2)(3,3)$
$(1,1)(3,3)$
$R(a^\circ,b^\circ,c^\circ)$
$R(a^\circ,b^\circ,d^\bullet)$
$R(a^\circ,c^\circ,d^\circ)$
$R(b^\circ,c^\circ,d^\circ)$
$R(a^\circ,c^\bullet,d^\bullet)$
$R(b^\circ,c^\bullet,d^\bullet)$
$R(a^\circ,b^\bullet,d^\circ)$
$R(a^\circ,b^\bullet,c^\bullet)$
$R(a^\bullet,b^\circ,d^\circ)$
$R(a^\bullet,b^\circ,c^\bullet)$
$R(b^\bullet,c^\bullet,d^\circ)$
$R(a^\bullet,c^\bullet,d^\circ)$
$(1,1)$ $(2,2)$
$(2,2)$ $(3,3)$
$(2,1)(3,2)$
$(1,2)(3,3)$

(a) Query Graph of Q_1

(b) Unrolled Database of Q_1

(c) Query Graph of Q_2

(d) Unrolled database of Q_2

Figure 4: Illustration of the unrolled database construction. (a) illustrates query Q_1 of Example 23; (b) illustrates the unrolling of Q_1 with respect to $\mathcal{B}_1$ and partial hyperedge $F = \{b, c, d\}$. (c) illustrates query Q_2 of Example 23; (d) illustrates the unrolling of Q_2 with respect to $\mathcal{B}_2$ and the partial hyperedge $F = \{a, b, d\}$.

- *$clr_{var(B)}|_X$, for each hyperedge X in $\mathcal{H}(Q^+)$ such that $F \not\sqsupseteq X$, and*
- *$clr_A|_X$, for each set $A \in \mathcal{M}_\cap$, and each hyperedge $X \in \mathcal{H}(Q^+)$ such that either $F \sqsupseteq X$ or $A \not\sqsupseteq_F X$.*

The database unrolldb is built from these functions as follows.

$$unrolldb := \{f(\boldsymbol{a}) \mid \boldsymbol{a} \in body(Q), f \in \mathcal{F}, \\ var(\boldsymbol{a}) \subseteq dom(f))\}.$$

For an atom $\boldsymbol{a}$ we refer to the facts $f(\boldsymbol{a})$ with $f \in \mathcal{F}$ and $var(\boldsymbol{a}) \subseteq dom(f)$ as the *copies* of $\boldsymbol{a}$. Observe that, by definition of $\mathcal{F}$, the following holds for every atom $\boldsymbol{a}$ that contains all variables in F (i.e., $F \subseteq var(\boldsymbol{a})$). For every copy of $\boldsymbol{a}$ there exists a set $A \in \mathcal{M}_\cap$ such that all variables in $A \cap F$ are colored white, and all variables in $F \setminus A$ are colored black. (Recall that A is a strict subset of F by definition of $\mathcal{M}_\cap$.) This property is crucial to establish that *unrolldb* has been constructed as desired (in particular, to establish Proposition 28).

Example 25. *Continuing Example 23, Figure 4 shows the unrolled database of queries Q_1 and Q_2, with respect to a particular distinguished edge F. The nodes of the graphs represent the atoms of the query and the facts of the unrolled database. Lowercase letters in a node denote variables. Edges between atoms (or facts) are labeled with their equality type (recall Definition 6). For clarity of presentation, some edges have been suppressed. In particular, self-loops, and edges that are the inverse of displayed edges are not included. Furthermore, edges with labels not present in the original query graph are suppressed in the illustration of the unrolled database. In the unrolled database, different styles of nodes are used to distinguish copies originating from distinct partial hyperedges of $\mathcal{M}_\cap$.*

For the unrolling 4(b), we consider the block $\mathcal{B}_1$ of Example 23 and $F = \{b, c, d\}$. Observe that variables that are not in $\mathcal{B}_1$ (namely: e, f, and g) are only colored white, and that the atoms built on these three variables are copied to a single fact. Also observe that d only gets colored white, as it is shared by all the partial hyperedges in $\mathcal{M}_\cap$. Finally, observe that, consistent with what we have noted before, for every copy of the atom $S(b, c, d, e)$ (which contains all the variables in F) we can identify $A \in \mathcal{M}_\cap$ such that in the copy all variables in A are copied white and all variables in $F \setminus A$ are copied black. Indeed, for the copy $S(b^\circ, c^\bullet, d^\circ, c^\circ)$, take $A = \{b, d\}$. For the copy $S(b^\bullet, c^\circ, d^\circ, c^\circ)$, take $A = \{c, d\}$.

For the unrolling 4(d), we consider the block $\mathcal{B}_2$ of Example 23 and $F = \{a, b, d\}$ .The unrolling contains three copies *of each fact.*

Note in particular that it is not possible to embed Q_1 (resp. Q_2) into the unrolled database of Q_1 (resp. Q_2). Indeed, to construct such an embedding, we would essentially have to find an edge-label-preserving graph homomorphism of the graph in Figure 4(a) (resp. Figure 4(c)) to the graph in Figure 4(b) (resp. Figure 4(d)), which is readily verified to be impossible.

By definition of Q^+, $\mathcal{H}(Q^+)$ contains a hyperedge X with $\mathit{var}(\overline{x}) \subseteq X$. Now observe that, by construction, $\mathcal{F}$ contains for every hyperedge X of $\mathcal{H}(Q^+)$ a function f with domain X. Fix $f \in \mathcal{F}$ with $\mathit{var}(\overline{x}) \subseteq \mathit{dom}(f)$ arbitrarily and let $\overline{b} = f(\overline{x})$.

Let freeze^{-1} denote the inverse of *freeze*. The following lemmas and propositions show that $(\mathit{canondb}, \overline{a})$ and $(\mathit{unrolldb}, \overline{b})$ have been constructed as desired.

Lemma 26. *The set $\mathcal{S} = \{f \circ \mathit{freeze}^{-1} \mid f \in \mathcal{F}\}$ is a guarded simulation of canondb in unrolldb.*

Proof sketch. It suffices to prove that each $f \in \mathcal{F}$ is a partial homomorphism from $\mathit{body}(Q)$ into *unrolldb* and that $\mathcal{F}$ satisfies the guarded forth condition. Indeed, since freeze^{-1} is an isomorphism from *canondb* to $\mathit{body}(Q)$, $\mathcal{S}$ will then be a set of partial homomorphisms from *canondb* into *unrolldb* that satisfy the guarded forth condition. Establishing that each $f \in \mathcal{F}$ is a partial homomorphism is straightforward; establishing the guarded forth condition is done by a technical case analysis. □

Proposition 27. *$\mathit{canondb}, \overline{a} \preceq_g \mathit{unrolldb}, \overline{b}$.*

Proof. Clearly $\overline{b} = f(\mathit{freeze}^{-1}(\overline{a}))$. Hence $\mathit{canondb}, \overline{a} \preceq_g \mathit{unrolldb}, \overline{b}$ since $\mathcal{S} = \{g \circ \mathit{freeze}^{-1} \mid g \in \mathcal{F}\}$ is a guarded simulation of *freeze* in *unrolldb* by Lemma 26, and since $f \circ \mathit{freeze}^{-1} \in \mathcal{S}$ maps $\overline{a} \mapsto \overline{b}$. □

Proposition 28. *$\overline{b} \notin Q(\mathit{unrolldb})$.*

Proof sketch. The proof is by contradiction. The essential reasoning (glossing over many important details) is as follows. Let $\mathit{ans}(\overline{x})$ be the head of Q and let $\mathit{unrolldb}^+$ denote $\mathit{unrolldb} \cup \{\mathit{ans}(\overline{b})\}$.

- First, we show that if $\overline{b} \in Q(\mathit{unrolldb})$ then there must also exist an embedding h of Q^+ in $\mathit{unrolldb}^+$ that maps

  $$x \mapsto x^\circ \qquad \text{or} \qquad x \mapsto x^\bullet,$$

  for every $x \in \mathit{var}(Q)$. In particular, x will not be mapped to a colored version of another variable. As a consequence, we can establish that h maps each atom $\boldsymbol{a} \in \mathit{body}(Q^+)$ to a copy of $\boldsymbol{a}$ in $\mathit{unrolldb}^+$, and not to a copy of some other atom.

- Then, since F is a partial hyperedge of $\mathcal{H}(Q^+)$ there exists some atom $\boldsymbol{a}$ in Q^+ that contains all variables in F. Since, by the first bullet, h maps atoms in $\mathit{body}(Q^+)$ to their copies in $\mathit{unrolldb}^+$, we know in particular that $h(\boldsymbol{a})$ is a copy of $\boldsymbol{a}$. Then, since $\boldsymbol{a}$ contains all variables in F, there exists some $A \in \mathcal{M}_\cap$ such that every variable in $A \subsetneq F$ is colored white in $h(\boldsymbol{a})$ and every variable in $F \setminus A$ is colored black in $h(\boldsymbol{a})$.

- Since $A \in \mathcal{M}_\cap$ there exists $E_1 \in \mathcal{B}$ such that $A = E_1 \cap F$. Moreover, since $\mathcal{B}$ is a block of $\mathcal{H}(Q^+)$, A cannot be an articulation set of $\mathcal{B}$. As such, there must exist a path $E_1, \dots, E_n, F \in \mathcal{B}$ that does not need to traverse any node in A. That is, $(E_i \cap E_{i+1}) \setminus A \neq \emptyset$ for $1 \leq i < n$, and $(E_n \cap F) \setminus A \neq \emptyset$.

- Now, it is possible to establish that $h(E_i)$ consists only of white colored variables, for all $1 \leq i \leq n$. This yields the desired contradiction. Indeed, since $(E_n \cap F) \setminus A$ is nonempty there is some variable x that is both in E_n and F, but not in A. Since $x \in E_i$, h must map $x \mapsto x^\circ$. On the other hand, since $x \in F \setminus A$, we have already established before that h must map $x \mapsto x^\bullet$. □

Proposition 29. *$\overline{b} \notin \varphi(\mathit{unrolldb})$.*

Crux. We already know that $\overline{b} \notin Q(\mathit{unrolldb})$ by Proposition 28. Suppose, for the purpose of contradiction, that there is some other CQ $Q' \in \varphi$ such that $\overline{b} \in Q'(\mathit{unrolldb})$. In particular, there exists an embedding h from Q' into *unrolldb* such that $h(\overline{x}) = \overline{b}$. Now, consider the function decopy: $\mathit{im}(h) \to \mathit{var}(Q)$ such that

$$\begin{aligned} \mathrm{decopy}(x^\bullet) &= x \qquad \text{for every } x^\bullet \in \mathit{im}(h) \\ \mathrm{decopy}(x^\circ) &= x \qquad \text{for every } x^\circ \in \mathit{im}(h). \end{aligned}$$

Observe that by definition of *unrolldb*, Q contains an atom $\mathrm{decopy}(\boldsymbol{s})$ for each fact $\boldsymbol{s} \in \mathit{unrolldb}$ built over the image of h. Hence $(\mathrm{decopy} \circ h)$ is a homomorphism of $\mathit{body}(Q')$ into $\mathit{body}(Q)$. Furthermore, $\mathrm{decopy}(\overline{b}) = \overline{x}$ since $\overline{b} = f(x)$ for some $f \in \mathcal{F}$. By Chandra and Merlin's classical result [6], this implies that Q is contained in Q', contradicting the fact that φ is minimal. □

Since $\mathit{canondb}, \overline{a} \preceq_g \mathit{unrolldb}, \overline{b}$ and $\overline{a} \in \varphi(\mathit{canondb})$ but $\overline{b} \notin \varphi(\mathit{unrolldb})$ we have our desired contradiction: φ is not invariant under guarded simulation. This finishes the proof of Theorem 5.

4. GUARDED VS FACT SIMULATION

We next present an alternate definition for guarded simulation, called *fact simulation*, and show that fact simulation naturally yields approximations that are tightly linked to invariance of freely acyclic conjunctive queries whose join tree is of a specific bounded height.

4.1 Fact simulation

Definition 30. *A* fact simulation *of database db_1 in database db_2 is a nonempty binary relation $\mathcal{F} \subseteq db_1 \times db_2$ between the facts of db_1 and db_2 such that for all facts $\boldsymbol{s} \in db_1$ and $\boldsymbol{t} \in db_2$ with $\boldsymbol{s}\mathcal{F}\boldsymbol{t}$:*

- *$\boldsymbol{s}$ and $\boldsymbol{t}$ carry the same relation symbol, i.e., $\mathrm{rel}(\boldsymbol{s}) = \mathrm{rel}(\boldsymbol{t})$;*

- *for all $\boldsymbol{s}' \in db_1$ there exists $\boldsymbol{t}' \in db_2$ with $\mathit{eqtp}(\boldsymbol{s}, \boldsymbol{s}') \subseteq \mathit{eqtp}(\boldsymbol{t}, \boldsymbol{t}')$ and $\boldsymbol{s}' \mathcal{F} \boldsymbol{t}'$.*

Example 31. *To illustrate, the dotted lines in Figure 2 show a fact simulation $\mathcal{F}$ of database db_1 in db_2. Note that fact simulation is necessarily total on db_1 (i.e., every fact of db_1 occurs in $\mathcal{F}$).*

Now, let db_1 and db_2 be two databases, and let $\boldsymbol{s}$ and $\boldsymbol{t}$ be facts. We say that $(db_1, \boldsymbol{s})$ is *fact simulated by* $(db_2, \boldsymbol{t})$, denoted $db_1, \boldsymbol{s} \preceq_f db_2, \boldsymbol{t}$, if there exists a fact simulation $\mathcal{F}$ of $db_1 \cup \{\boldsymbol{s}\}$ in $db_2 \cup \{\boldsymbol{t}\}$ with $\boldsymbol{s}\,\mathcal{F}\,\boldsymbol{t}$. Moreover if $\overline{a}$ and $\overline{b}$ are tuples of data values, then $db_1, \overline{a} \preceq_f db_2, \overline{b}$ if $db_1, \mathit{ans}(\overline{a}) \preceq_f db_2, \mathit{ans}(\overline{b})$ with *ans* a relation symbol of the same arity as $\overline{a}$ and $\overline{b}$ that does not occur in db_1 or db_2.

We require the following notions to establish that fact simulation is equivalent to guarded simulation. Let $\boldsymbol{s} \rightarrowtail \boldsymbol{t}$ denote, for every $\boldsymbol{s} = s(a_1, \dots, a_k)$ and $\boldsymbol{t} = t(b_1, \dots, b_l)$, the relation $\{(a_i, b_i) \mid 1 \leq i \leq \min(k, l)\}$. When we are sure that $\boldsymbol{s} \rightarrowtail \boldsymbol{t}$ is a function, we use common notation for functions, such as $(\boldsymbol{s} \rightarrowtail \boldsymbol{t})(a)$ to denote the unique value associated to a by the function $\boldsymbol{s} \rightarrowtail \boldsymbol{t}$.

Now define, for a guarded simulation $\mathcal{S}$,

$$\mathcal{F}[\mathcal{S}] := \{(\boldsymbol{s}, f(\boldsymbol{s})) \mid f\colon X \to Y \in \mathcal{S}, \boldsymbol{s} \in db_1, \text{ and } \mathit{val}(\boldsymbol{s}) \subseteq X\}.$$

Also define, for a fact simulation $\mathcal{F}$,

$$\mathcal{S}[\mathcal{F}] := \{\boldsymbol{s} \rightarrowtail \boldsymbol{t} \mid (\boldsymbol{s}, \boldsymbol{t}) \in \mathcal{F}\}.$$

For example, the relation $(1, \mathrm{Amy}, \mathrm{Lex}) \rightarrowtail (\mathrm{c}, \mathrm{Ned}, \mathrm{Ned})$ is an element of $\mathcal{S}[\mathcal{F}]$, for fact simulation $\mathcal{F}$ of Example 31.

The following proposition establishes the correspondence between guarded simulation and fact simulation.

Proposition 32. *1. If $\mathcal{S}$ is a guarded simulation of db_1 in db_2 then $\mathcal{F}[\mathcal{S}]$ is a fact simulation of db_1 in db_2.*

2. If $\mathcal{F}$ is a fact simulation of db_1 in db_2 then $\mathcal{S}[\mathcal{F}]$ is a guarded simulation of db_1 in db_2.

It follows that fact simulation provides an alternative definition for guarded simulation, in the following sense.

Theorem 33. *For databases db_1 and db_2 and tuples $\overline{a}$ and $\overline{b}$ it holds that $db_1, \overline{a} \preceq_g db_2, \overline{b}$ iff $db_1, \overline{a} \preceq_f db_2, \overline{b}$.*

4.2 Approximate fact simulation

In applications of classical simulation in data management it is known that, if there are few nodes in graph G that are similar, then the corresponding structural index of G may be of the same size as G itself, and hence be too large to act as a succinct summary of the structure of G [18]. In such situations, it has been proposed to approximate simulations and to group nodes in the index with respect to these approximations instead of with respect to full simulation [18]. Towards a suitable approximation of guarded simulation, we introduce the following version of fact simulation.

Definition 34 (Approximate fact simulation). *Let db_1 and db_2 be two databases. A* depth-k approximation of fact simulation *of db_1 in db_2, or* fact k-simulation *for short, is a sequence $\mathcal{F}_k \subseteq \mathcal{F}_{k-1} \subseteq \cdots \subseteq \mathcal{F}_0$ of binary relations such that $\mathcal{F}_0$ consists of all pairs $(\boldsymbol{s}, \boldsymbol{t}) \in db_1 \times db_2$ with $\mathrm{rel}(\boldsymbol{s}) = \mathrm{rel}(\boldsymbol{t})$ and $\mathit{eqtp}(\boldsymbol{s}, \boldsymbol{s}) \subseteq \mathit{eqtp}(\boldsymbol{t}, \boldsymbol{t})$; and the following property holds for every $1 \leq j \leq k$ and all $\boldsymbol{s}$ and $\boldsymbol{t}$ with $\boldsymbol{s}\,\mathcal{F}_j\,\boldsymbol{t}$.*

- *For every $\boldsymbol{s'} \in db_1$ there exists $\boldsymbol{t'}$ in db_2 such that $\mathit{eqtp}(\boldsymbol{s}, \boldsymbol{s'}) \subseteq \mathit{eqtp}(\boldsymbol{t}, \boldsymbol{t'})$ and $\boldsymbol{s'}\,\mathcal{F}_{j-1}\,\boldsymbol{t'}$ **(Fact Forth)**.*

We say that $(db_1, \boldsymbol{s})$ is *k-fact simulated* by $(db_2, \boldsymbol{t})$, denoted $db_1, \boldsymbol{s} \preceq_f^k db_2, \boldsymbol{t}$, if there exists a fact k-simulation $\mathcal{F}_k \subseteq \mathcal{F}_{k-1} \subseteq \cdots \subseteq \mathcal{F}_0$ from $db_1 \cup \{\boldsymbol{s}\}$ to $db_2 \cup \{\boldsymbol{t}\}$ with $\boldsymbol{s}\,\mathcal{F}_k\,\boldsymbol{t}$. The notion of *$k$-fact similarity* between $(db_1, \overline{a})$ and $(db_2, \overline{b})$ with $\overline{a}$ and $\overline{b}$ tuples is now defined in the obvious way.

Observe that $db_1, \boldsymbol{s} \preceq_f db_2, \boldsymbol{t}$ iff $db_1, \boldsymbol{s} \preceq_f^k db_2, \boldsymbol{t}$ for every $k \geq 0$.

We now link approximate guarded simulation to indistinguishability by FACQs of bounded height. Here, the *height* of a FACQ is defined as follows. Recall that in graph theory, the *distance* between two connected nodes u and v in an undirected graph G is the length of a shortest path between u and v. The *eccentricity* of u in G, denoted $\mathit{ecc}(u, G)$ is the maximum distance of u to any other node to which it is connected.

Example 35. *Consider, for instance, the join tree T of Figure 3. The eccentricity of $R(a, b)$ in T is 2 while the eccentricity of $R(g, h)$ in T is 3.*

Definition 36. *Let A be a set of atoms. When A is acyclic, the eccentricity of atom $\boldsymbol{a} \in A$, denoted $\mathit{ecc}(\boldsymbol{a}, A)$ is the minimum eccentricity of $\boldsymbol{a}$ among all join trees T for A. The* height *of a FACQ Q is the eccentricity of $\mathit{head}(Q)$ in $\mathit{body}(Q^+)$.*

In other words, the height of Q is the minimum height of any join tree T for $\mathit{body}(Q^+)$, when considered as being rooted at $\mathit{head}(Q)$.

Example 37. *Continuing Example 35, query Q of Example 9 has a height of 3.*

Proposition 38. *Let $k \geq 0$ be a natural number. The following are equivalent.*

(1) $db_1, \overline{a} \preceq_f^k db_2, \overline{b}$

(2) For all FACQs Q of height $\leq k$, if $\overline{a} \in Q(db_1)$ then $\overline{b} \in Q(db_2)$.

Note that Proposition 38 implies Proposition 18 (yet the converse is not true).

Closing remark. We close this section with the following important remark. It is obviously possible to define approximations of guarded simulation in an analogous way as approximations of fact simulation: a depth-k approximation is a sequence $\mathcal{S}_k \subseteq \mathcal{S}_{k-1} \subseteq \cdots \subseteq \mathcal{S}_0$ of partial homomorphisms such that each $\mathcal{S}_i$ satisfies the guarded forth property to $\mathcal{S}_{i-1}$, for $i \geq 1$. While full guarded simulation coincides with fact simulation (cf. Theorem 33), their approximations do not. In particular, $\preceq_g^0$ is distinct from $\preceq_f^0$. Indeed, consider $db_1 = \{r(a, b), r(b, a)\}$ and $db_2 = \{r(1, 2)\}$. Note that there is no partial homomorphism from db_1 to db_2 with domain $\{a, b\}$. Therefore, db_1 is not guarded 0-simulated by db_2. Yet, $db_1 \times db_2$ is a fact 0-simulation of db_1 in db_2.

5. GUARDED STRUCTURAL INDEXING

Recall from the Introduction that in graph data management, a structural index is a compact representation of a data graph. Typically, this compact representation is obtained by grouping the nodes in the input graph that are similar or bisimilar. Structural characterizations of query invariance then enable efficient retrieval of the relevant nodes of the graph for various graph query languages.

In this section we analogously define guarded structural indexes as compact representations of relational data obtained by grouping facts according to guarded similarity.

Our structural characterization of FACQ invariance then provides the means to answer strict ACQs directly on the index.

Towards a formal definition of a guarded simulation-based structural index, we first define guarded similarity, and approximations thereof. Hereto, recall from Section 4 that fact simulation is an alternative definition to guarded simulation. For our definition of the index we will therefore work exclusively with fact simulation.

Definition 39. *$(db_1, \boldsymbol{s})$ is called* fact similar *to $(db_2, \boldsymbol{t})$, denoted $db_1, \boldsymbol{s} \sim_f db_2, \boldsymbol{t}$ if both $db_1, \boldsymbol{s} \preceq_f db_2, \boldsymbol{t}$ and $db_2, \boldsymbol{t} \preceq_f db_1, \boldsymbol{s}$.*

The approximate version, called *k-fact similarity*, denoted $db_1, \boldsymbol{s} \sim_f^k db_2, \boldsymbol{t}$ is defined analogously.

Note that, when applied to the same database db, fact simulation (i.e., the relation $\{(\boldsymbol{s}, \boldsymbol{t}) \mid \boldsymbol{s}, \boldsymbol{t} \in db$ and $db, \boldsymbol{s} \preceq_f db, \boldsymbol{t}\})$ is a preorder and fact similarity (i.e., the relation $\{(\boldsymbol{s}, \boldsymbol{t}) \mid \boldsymbol{s}, \boldsymbol{t} \in db$ and $db, \boldsymbol{s} \sim_f db, \boldsymbol{t}\})$ is an equivalence relation. The same holds for the approximations.

Definition 40. *A* guarded structural index *(or guarded index for short) for a database db is a pair $(db_\downarrow, \mathrm{lab})$ with $db_\downarrow$ a database and* lab: *$db_\downarrow \to 2^{db}$ a function that maps each fact of $db_\downarrow$ to a subset of db.*

The intuition in this definition is that facts in $db_\downarrow$ will serve as the summary of the structure of facts in db, and that lab associates the facts in $db_\downarrow$ to the facts in db that they summarize.

The guarded index based on (full) guarded simulation, denoted $\textsc{sim}_g(db)$, is then defined as follows.

Definition 41. *The* guarded simulation index *for db is a guarded structural index $\textsc{sim}_g(db) = (db_\downarrow, \mathrm{lab})$ such that:*
(1) $db_\downarrow$ is the smallest database such that for every $\boldsymbol{t} \in db$ there exists a fact $\boldsymbol{u} \in db_\downarrow$ with $db, \boldsymbol{t} \sim_f db_\downarrow, \boldsymbol{u}$.
(2) lab *is the function that maps each fact $\boldsymbol{u} \in db_\downarrow$ to the set $\{\boldsymbol{s} \in db \mid db, \boldsymbol{s} \sim_f db_\downarrow, \boldsymbol{u}\}$.*

The k-th approximation of this index is analogously defined by replacing $\sim_f$ in the definition above by $\sim_f^k$, and is denoted $\textsc{sim}_g^k(db)$. Note that the size of $db_\downarrow$ in $\textsc{sim}_g(db)$ and $\textsc{sim}_g^k(db)$ is always bounded by the size of db, but is potentially much smaller.

The following proposition shows that strict ACQs can be answered directly on the guarded-simulation structural index. It follows almost directly from Proposition 38 and the fact that strict ACQs are freely acyclic. Analogous statements hold for the approximations.

Proposition 42 (Cover)**.** *Let $Q(\overline{x})$ be a strict ACQ, let db be a database, and let $\textsc{sim}_g(db) = (db_\downarrow, \mathrm{lab})$. Since Q is strict there exists some $\boldsymbol{a} \in body(Q)$ with $\overline{x} \subseteq var(\boldsymbol{a})$. Let $i_1, \dots, i_n$ be natural numbers such that $\overline{x} = (\boldsymbol{a}.i_1, \dots, \boldsymbol{a}.i_n)$. Then,*

$$Q(db) = \{(\boldsymbol{s}.i_1, \dots, \boldsymbol{s}.i_n) \mid \mu \text{ embedding of } Q \text{ in } db_\downarrow, \\ \boldsymbol{s} \in \mathrm{lab}(\mu(\boldsymbol{a})\}.$$

Since the size of $db_\downarrow$ in $\textsc{sim}_g(db)$ is always bounded by the size of db, evaluating Q by calculating the right-hand side is potentially much faster than evaluating Q on db directly, provided that lab is efficiently implemented.

In companion work [24, 25], we show that a variant of the above index structure and a more elaborate evaluation strategy can improve the efficiency of evaluating not only strict ACQs, but arbitrary conjunctive queries (that cannot be answered directly from the index). The crux to this evaluation strategy is given by the following observation. Analogous statements hold for the depth k-approximations with $k \geq 1$.

Proposition 43. *Let Q be an arbitrary CQ, let db be a database, and let $\textsc{sim}_g(db) = (db_\downarrow, \mathrm{lab})$. Let $\mu \colon \mathcal{V} \to \mathcal{U}$ be a valuation. The following are equivalent.*

(1) Valuation μ is an embedding of Q in db.

(2) There exists an embedding $\mu_\downarrow$ of Q in $db_\downarrow$ such that $\mu(\boldsymbol{a}) \in \mathrm{lab}(\mu_\downarrow(\boldsymbol{a}))$, for each atom $\boldsymbol{a} \in body(Q)$.

Proposition 43 allows us to prune the space that we normally need to search to establish embeddings of a conjunctive query Q in db. To illustrate this claim, let us first introduce the following notation. Let $[\![\boldsymbol{a}]\!]_{db}$ denote the set of valuations $\{\mu\colon var(\boldsymbol{a}) \to \mathcal{U} \mid \mu(\boldsymbol{a}) \in db\}$, for any atom $\boldsymbol{a}$. Define the join $\Omega_1 \bowtie \Omega_2$ of two sets Ω_1 and Ω_2 of valuations to be the set of valuations $\mu_1 \cup \mu_2$ where $\mu_1 \in \Omega_1, \mu_2 \in \Omega_2$, and $\mu_1(x) = \mu_2(x)$ for all common variables $x \in dom(\mu_1) \cap dom(\mu_2)$.

Now observe that, to construct the embeddings of a CQ Q with body $\boldsymbol{a_1}, \dots, \boldsymbol{a_n}$ in database db we essentially compute $[\![\boldsymbol{a_1}]\!]_{db} \bowtie \cdots \bowtie [\![\boldsymbol{a_n}]\!]_{db}$. Proposition 43 tells us, however, that we do not need to consider joining the entire sets $[\![\boldsymbol{a_i}]\!]_{db}$. In particular, in a pre-processing step we can first evaluate Q on $db_\downarrow$ to find all embeddings of Q in $db_\downarrow$. If $db_\downarrow$ is small, this can be done quickly compared to the evaluation of Q on the entire database db. Then, for each obtained embedding $\mu_\downarrow$ we compute $[\![\boldsymbol{a_1}]\!]_{\mathrm{lab}(\mu_\downarrow(\boldsymbol{a_1}))} \bowtie \cdots \bowtie [\![\boldsymbol{a_n}]\!]_{\mathrm{lab}(\mu_\downarrow(\boldsymbol{a_n}))}$ and add it to the final result. Note that the sets $[\![\boldsymbol{a_i}]\!]_{\mathrm{lab}(\mu_\downarrow(\boldsymbol{a_i}))}$ to be joined are potentially much smaller than the original sets $[\![\boldsymbol{a_i}]\!]_{db}$. Also note that we typically need to perform multiple such joins (one for each $\mu_\downarrow$). When done eagerly and naively, this approach may therefore actually slow query processing. Fortunately, however, classical cost-based optimizers can be modified in order to ascertain when this approach is beneficial [24, 25].

Acknowledgments. We thank all reviewers, Paul De Bra, and Yongming Luo for their constructive and insightful comments on previous versions of this paper. The work of François Picalausa was supported by a FRIA research assistant scholarship of the FNRS.

6. REFERENCES

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of databases.* Addison-Wesley, 1995.

[2] H. Andréka, I. Németi, and J. van Benthem. Modal languages and bounded fragments of predicate logic. *J. Phil. Log.*, 27(3):217–274, 1998.

[3] C. Beeri, T. Milo, and P. Ta-Shma. Towards a language for the fully generic queries. In *DBPL 1997*, pages 239–259. Springer, 1997.

[4] P. Blackburn, M. de Rijke, and Y. Venema. *Modal logic.* Cambridge University Press, 2001.

[5] P. Buneman, M. Grohe, and C. Koch. Path queries on compressed XML. In *VLDB 2003*, pages 141–152, 2003.

[6] A. K. Chandra and P. M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *STOC 1977*, pages 77–90. ACM, 1977.

[7] H. Chen and V. Dalmau. Beyond hypertree width: Decomposition methods without decompositions. In *CP*, volume 3709, pages 167–181. Springer, 2005.

[8] R. Fagin. Degrees of acyclicity for hypergraphs and relational database schemes. *J. ACM*, 30(3):514–550, 1983.

[9] W. Fan. Graph pattern matching revised for social network analysis. In *ICDT 2012*, pages 8–21. ACM, 2012.

[10] W. Fan, J. Li, X. Wang, and Y. Wu. Query preserving graph compression. In *SIGMOD 2012*, pages 157–168. ACM, 2012.

[11] G. H. L. Fletcher, D. Van Gucht, Y. Wu, M. Gyssens, S. Brenes, and J. Paredaens. A methodology for coupling fragments of XPath with structural indexes for XML documents. *Inf. Syst.*, 34(7):657–670, 2009.

[12] J. Flum, M. Frick, and M. Grohe. Query evaluation via tree-decompositions. *J. ACM*, 49(6):716–752, 2002.

[13] G. Gottlob, N. Leone, and F. Scarcello. Robbers, marshals, and guards: game theoretic and logical characterizations of hypertree width. *JCSS*, 66(4):775–808, 2003.

[14] G. Gou and R. Chirkova. Efficiently querying large XML data repositories: a survey. *TKDE*, 19(10):1381–1403, 2007.

[15] E. Grädel, C. Hirsch, and M. Otto. Back and forth between guarded and modal logics. *TOCL*, 3(3):418–463, 2002.

[16] M. Gyssens, J. Paredaens, D. V. Gucht, and G. H. L. Fletcher. Structural characterizations of the semantics of XPath as navigation tool on a document. In *PODS 2006*, pages 318–327. ACM, 2006.

[17] C. Hirsch. *Guarded logics: algorithms and bisimulation*. PhD thesis, TU Aachen, 2002.

[18] R. Kaushik, P. Shenoy, P. Bohannon, and E. Gudes. Exploiting local similarity for indexing paths in graph-structured data. In *ICDE 2002*, pages 129–140. IEEE, 2002.

[19] P. G. Kolaitis and M. Y. Vardi. Conjunctive-query containment and constraint satisfaction. In *PODS 1998*, pages 205–213. ACM, 1998.

[20] D. Leinders, M. Marx, J. Tyszkiewicz, and J. V. den Bussche. The semijoin algebra and the guarded fragment. *JOLLI*, 14(3):331–343, 2005.

[21] A. Matono, T. Amagasa, M. Yoshikawa, and S. Uemura. A path-based relational RDF database. In *ADC 2005*, pages 95–103. Australian Computer Society, 2005.

[22] T. Milo and D. Suciu. Index structures for path expressions. In *ICDT 1999*, pages 277–295. Springer, 1999.

[23] M. Otto. Highly acyclic groups, hypergraph covers, and the guarded fragment. *J. ACM*, 59(1):5:1–5:40, 2012.

[24] F. Picalausa. *Guarded structural indexes: theory and application to relational RDF databases*. PhD thesis, Université Libre de Bruxelles, 2013.

[25] F. Picalausa, Y. Luo, G. H. L. Fletcher, J. Hidders, and S. Vansummeren. A structural approach to indexing triples. In *ESWC*, volume 7295, pages 406–421. Springer, 2012.

[26] E. Prud'hommeaux and A. Seaborne. SPARQL query language for RDF. Technical report, W3C Recommendation, 2008.

[27] P. Ramanan. Covering indexes for XML queries: bisimulation − simulation = negation. In *VLDB 2003*, pages 165–176, 2003.

[28] B. Rossman. Homomorphism preservation theorems. *J. ACM*, 55(3):15:1–15:53, 2008.

[29] D. Sangiorgi. *Introduction to bisimulation and coinduction*. Cambridge University Press, 2012.

[30] T. Tran, G. Ladwig, and S. Rudolph. Managing structured and semistructured RDF data using structure indexes. *TKDE*, 25(9):2076–2089, 2013.

[31] O. Udrea, A. Pugliese, and V. S. Subrahmanian. GRIN: a graph based RDF index. In *AAAI 2007*, pages 1465–1470, 2007.

[32] M. Yannakakis. Algorithms for acyclic database schemes. In *VLDB 1981*, pages 82–94. IEEE, 1981.

APPENDIX

A. GUARDED BISIMULATION

The definition of guarded bisimulation due to Andréka et al. [2] is recalled here for completeness. Given two sets A and B of facts and atoms, a function $f\colon X \to Y$ is a *partial isomorphism* from A to B if it is bijective and $f(A|_X) = B|_Y$.

Definition 44 (Guarded bisimulation)**.** *Let db_1 and db_2 be databases. A* guarded bisimulation *from db_1 to db_2 is a nonempty set $\mathcal{I}$ of finite partial isomorphisms from db_1 to db_2 such that the following forth and back conditions are satisfied.*

- *For every $f\colon X \to Y \in \mathcal{I}$ and for every set X' guarded in db_1, there exists a partial isomorphism $g\colon X' \to Y' \in \mathcal{I}$ such that g and f agree on $X \cap X'$. (**Guarded Bisimulation Forth**).*
- *For every $f\colon X \to Y \in \mathcal{I}$ and for every set Y' guarded in db_2, there exists a partial isomorphism $g\colon X' \to Y' \in \mathcal{I}$ such that g^{-1} and f^{-1} agree on $Y \cap Y'$. (**Guarded Bisimulation Back**)*