On guarded simulations and acyclic first-order languages

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
Belgium
stijn.vansummeren@ulb.ac.be

Yongming Luo
Eindhoven University of Technology
The Netherlands
y.luo@tue.nl

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

Paul De Bra
Eindhoven University of Technology
The Netherlands
debra@tue.nl

ABSTRACT

An exact structural characterization of the expressive power of the acyclic conjunctive queries is given in terms of guarded simulations. The study of this fragment of first order logic is motivated by the central role it plays in query languages across a wide range of data models. The study of a structural characterization of the language is motivated by the applications of such characterizations, for example, in the design of efficient indexing and query processing strategies. In addition to a presentation of our main result, we discuss the results of a small empirical study which indicate the practicality of guarded simulation based reductions of database instances.

1. INTRODUCTION

The conjunctive queries were recognized early in the study of database query languages as a particularly important fragment of first-order logic (FO) [6]. As the basic language for expressing join patterns between database objects, the conjunctive queries have since continued to play a central role in query language design across all major datamodels: relational, complex object, object-oriented, semi-structured, XML, graph, and RDF data [1, 2]. For example, conjunctive queries appear in the guise of path and star queries, and tree and graph patterns in these various data models.

Already in Chandra and Merlin's first paper on the conjunctive queries, the notion of homomorphisms (i.e., structure preserving mappings) was crucial in reasoning about the language. It is indeed well known that the conjunctive queries are invariant under homomorphisms [26], that is

THEOREM 1. *For tuples $\overline{a}_1$ and $\overline{a}_2$ over constants appearing in respective database instances db_1 and db_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 $\overline{a}_2 \in Q(db_2)$.*

Such *structural* characterizations of the expressive power of query languages play an important role, for example, in the study of indexing data structures to accelerate query processing (e.g., [4, 8, 11, 14, 17, 18, 23, 28]). To be usable, however, these characterizations must be efficiently computable and maintainable under database updates.

Clearly, computing and maintaining all homomorphisms becomes impractical as the size of the database grows. There are, however, useful fragments of FO which have tractable structural characterizations. Indeed, many FO path languages for trees and graphs (e.g., [4, 8, 9, 17, 23, 29]) are characterized by variants of (bi)simulation, tractable structural notions of equivalence which have deep roots both inside and outside of computer science research [27]. In the logic community, the so-called *guarded fragment* of FO was shown to be characterized by a tractable generalized notion of *guarded bisimulation* [3, 24]. Flum et al. have since established expressive equivalence between guarded FO and the acyclic fragment of FO [10], and Leinders et al. have shown that guarded FO corresponds to the semi-join variant of Codd's relational algebra [20].

In the context of the conjunctive queries, is it possible to isolate a useful fragment which similarly admits a tractable structural characterization? Gottlob et al. have established the expressive equivalence of the *acyclic* conjunctive queries and the conjunctive fragment of guarded FO [13]. Clearly, this is a very natural candidate to consider (e.g., such queries appear in the role of tree patterns for XML and path/star patterns for RDF) [10, 12]. To our knowledge, a structural characterization of the acyclic conjunctive queries has not been established; the closest results here are those for positive modal languages established in the 1990's by de Rijke et al. [5, 19] and more recently those of Wu et al. for positive path queries on trees [29].

Contributions and overview. In this paper, we give the first structural characterization of the expressive power of the acyclic conjunctive queries in terms of *guarded simulations*, thereby complementing the structural characterization of acyclic FO in terms of guarded bisimulations. Simulations, which are efficiently computable, have previously found basic applications in data management (e.g., [1, 7,

29]). In addition to a presentation of our main result, we also discuss the results of a preliminary empirical investigation which indicate the practicality of guarded simulations.

We proceed in the paper as follows. In Section 2, we present terminology and notation, and define the notion of guarded simulation. In Section 3, we then present two syntaxes for the acyclic conjunctive queries. Following these preliminaries, we then establish in Section 4 our main results. After this, we discuss in Section 5 the results of the empirical investigation, and then give closing observations in Section 6.

2. STRUCTURES AND GUARDED SIMULATIONS

Schemas and databases. We assume given a fixed universe $\mathcal{U}$ of atomic data values, as well as a fixed, finite set $\mathcal{S}$ of relation symbols. Associated to each relational symbol $r \in \mathcal{S}$ is a natural number, called the *arity* of r.

As usual, a relational database db over $\mathcal{S}$ is an assignment of a finite relation $db(r) \subseteq \mathcal{U}^n$ to each $r \in \mathcal{S}$, where n is the arity of r. We adopt the standard convention of identifying a relational database instance over $\mathcal{S}$ with a finite logical theory consisting of ground facts. Here, a *fact* is a term of the form $r(a_1, \ldots, a_k)$ with $r \in \mathcal{S}$, k the arity of relation name r, and $a_1, \ldots, a_k \in \mathcal{U}$. A database $\mathcal{S}$ can then be identified with the finite set of facts $\{r(\overline{a}) \mid \overline{a} \in db(r), r \in \mathcal{S}\}$.

For convenience we will often denote tuples $(a_1, \ldots, a_k)$ of elements of a set A by $\overline{a}$, and write $\overline{a} \in A$ instead of $\overline{a} \in A^k$. Moreover, if $f\colon A \to B$ is a function, then we denote by $f(\overline{a})$ the tuple obtained by applying f point-wise to each element in $\overline{a}$. Also, if $\overline{a}$ and $\overline{b}$ are tuples of the same arity k, then we denote by $\overline{a} \mapsto \overline{b}$ the relation $\{(a_i, b_i) \mid 1 \leq i \leq k\}$. Finally, if $\overline{a}$ and $\overline{b}$ are two tuples, of arity k and l, respectively (k and l need not be the same), then the *equality type* of $\overline{a}$ and $\overline{b}$, denoted by $eqtp(\overline{a}, \overline{b})$ is defined as follows:

$$eqtp(\overline{a}, \overline{b}) := \{(i, j) \mid a_i = b_j \text{ with } 1 \leq i \leq k \text{ and } 1 \leq j \leq l\}.$$

EXAMPLE 2. *Consider the following two databases, where facts are labeled for ease of reference.*

db_1		db_2	
t_1	$r(a, b, c)$	t_8	$r(o, p, q)$
t_2	$r(b, d, e)$	t_9	$r(p, r, s)$
t_3	$r(c, f, g)$	t_{10}	$r(q, t, u)$
t_4	$r(h, i, j)$	t_{11}	$r(u, v, w)$
t_5	$r(i, k, l)$	t_{12}	$s(z, z, z)$
t_6	$r(j, m, n)$		
t_7	$s(x, y, y)$		

We have that $eqtp(t_1, t_2) = eqtp(t_8, t_9) = \{(2, 1)\}$, *and* $eqtp(t_7, t_7) \subseteq eqtp(t_{12}, t_{12})$.

Guarded simulation. We call a set of atomic data values $X \subseteq \mathcal{U}$ *guarded* in a database db if there is a fact $r(a_1, \ldots, a_n) \in db$ such that $X = \{a_1, \ldots, a_n\}$. In other words, a set is guarded in db if it can be obtained from a tuple in db by forgetting order and removing duplicates. We call a tuple $\overline{a}$ guarded in db if it is built over some guarded set in db. In Example 2, $\{z\}$ is guarded in db_2 (by t_{12}), $\{a, b, c\}$ is guarded in db_1 (by t_1), and (b, a, a, a, a) is guarded in db_1 (by $\{a, b, c\}$).

Let db_1 and db_2 be databases, and let $X, Y \subseteq \mathcal{U}$. A function $f\colon X \to Y$ is a *partial homomorphism* from db_1 to db_2 if for each $r \in \mathcal{S}$ and every $\overline{a} \subseteq X$ (i.e., every tuple built over data values in X) we have that if $r(\overline{a}) \in db_1$ then $r(f(\overline{a})) \in db_2$.

DEFINITION 3. *A* guarded simulation *of database db_1 in database db_2 is a non-empty set I of partial homomorphisms from db_1 to db_2 such that the following condition is satisfied for all $f\colon X \to Y \in I$:*

Forth *for any guarded set Z in db_1 there is a $g \in I$ with domain Z such that g and f agree on the intersection $X \cap Z$.*

Guarded simulation is a variant of guarded bisimulation, due to Andréka et al. [3]. Guarded bisimulations are inspired by both the "classical" notion of bisimulation [27] and Ehrenfeucht-Fraïssé games [21]. An alternative perspective on guarded simulations which will prove useful in the sequel is as follows.

DEFINITION 4. *A* tuple-based simulation *of database db_1 in database db_2 is a non-empty binary relation $T \subseteq db_1 \times db_2$ between the facts of db_1 and db_2 such that:*

1. *T relates only facts with the same relation name (i.e., if $(r(\overline{a}), s(\overline{b})) \in T$ then $r = s$);*
2. *T satisfies the following condition for every $(r(\overline{a}), r(\overline{b})) \in T$*

   Tuple-Forth *for every fact $s(\overline{c}) \in db_1$ there exists $s(\overline{d}) \in db_2$ such that $(s(\overline{c}), s(\overline{d})) \in T$ and $eqtp(\overline{a}, \overline{c}) \subseteq eqtp(\overline{b}, \overline{d})$.*

EXAMPLE 5. *For the instances of Example 2, we have that*

$$T_1 = \{(t_1, t_8), (t_2, t_9), (t_3, t_{10}), (t_4, t_8), (t_5, t_9), (t_6, t_{10}), (t_7, t_{12})\}$$

is a tuple-based simulation of db_1 in db_2, and

$$T_2 = \{(t_1, t_4), (t_2, t_5), (t_3, t_6), (t_4, t_4), (t_5, t_5), (t_6, t_6), (t_7, t_7)\}$$
$$T_3 = \{(t_4, t_1), (t_5, t_2), (t_6, t_3), (t_1, t_1), (t_2, t_2), (t_3, t_3), (t_7, t_7)\}$$

are tuple-based simulations of db_1 in db_1.

Define, for a guarded simulation I,

$$\mathcal{T}[I] := \{(r(\overline{a}), r(f(\overline{a}))) \mid r(\overline{a}) \in db_1, f \in I\}.$$

Also define, for a tuple simulation T,

$$\mathcal{I}[T] := \{\overline{a} \mapsto \overline{b} \mid (r(\overline{a}), r(\overline{b})) \in T\}.$$

For example, the relation $(a, b, c) \mapsto (o, p, q)$ is an element of $\mathcal{I}[T_1]$, for tuple simulation T_1 of Example 5.

We can note the following correspondence between guarded simulation and tuple simulation.

PROPOSITION 6.

1. *If I is a guarded simulation of db_1 in db_2 then $\mathcal{T}[I]$ is a tuple simulation of db_1 in db_2.*
2. *If T is a tuple simulation of db_1 in db_2 then $\mathcal{I}[T]$ is a guarded simulation of db_1 in db_2.*

PROOF. *(1).* Let I be a guarded simulation. We need to show that $\mathcal{T}[I]$ satisfies all conditions of Definition 4:

1. Trivial, by definition of $\mathcal{T}[I]$.

2. The tuple-forth condition follows from the forth condition on I:

   Tuple-Forth Let $(r(\overline{a}), r(f(\overline{a})))$ be a pair in $\mathcal{T}[I]$. Let $s(\overline{c})$ be a fact in db_1. Then $\overline{c}$ forms a guarded set. By the Forth condition of guarded simulation, there must exist some $g \in I$ with domain $\overline{c}$ that agrees with f on all values in $\overline{a} \cap \overline{c}$. Since g is a partial homomorphism, $s(g(\overline{c})) \in db_2$. Moreover, $(s(\overline{c}), s(g(\overline{c}))) \in \mathcal{T}[I]$. Then, since g agrees with f on $\overline{a} \cap \overline{c}$, $eqtp(\overline{a}, \overline{c}) \subseteq eqtp(f(\overline{a}), g(\overline{c}))$, as desired.

(2). Let T be a tuple-based simulation. We need to show that $\mathcal{I}[T]$ satisfies all conditions of Definition 3.

First, we need to show that the elements $\overline{a} \mapsto \overline{b}$ of $\mathcal{I}[T]$ are partial homomorphisms. First note that, by construction, $\overline{a} \mapsto \overline{b}$ is in $\mathcal{I}[T]$ only if $(r(\overline{a}), r(\overline{b})) \in T$ for some $r \in \mathcal{S}$. Let k be the arity of r.

We need to show that $\overline{a} \mapsto \overline{b}$ is (1) a function and (2) preserves facts of db_1 in db_2.

- To see that it is a function, apply the Tuple-forth condition taking $s(\overline{c}) = r(\overline{a})$ in db_1. Then there exists $r(\overline{d})$ in db_2 such that $(r(\overline{a}), r(\overline{d})) \in T$ and $eqtp(\overline{a}, \overline{a}) \subseteq eqtp(\overline{b}, \overline{d})$. Since clearly $(i, i) \in eqtp(\overline{a}, \overline{a})$ for every $1 \leq i \leq k$, this implies that $b_i = d_i$ for $1 \leq i \leq k$. As such, $\overline{b} = \overline{d}$.

  In other words, $eqtp(\overline{a}, \overline{a}) \subseteq eqtp(\overline{b}, \overline{b})$, and hence if $a_i = a_j$ then $b_i = b_j$, for all $1 \leq i \leq j \leq k$. As such, $\overline{a} \mapsto \overline{b}$ is a function.

- Towards (2), consider $s \in \mathcal{S}$ arbitrary, and $\overline{c} \subseteq \overline{a}$. Let f abbreviate $\overline{a} \mapsto \overline{b}$. Suppose that $s(\overline{c}) \in db_1$. We need to show that $s(f(\overline{c})) \in db_2$. By Tuple-Forth, there is some $s(\overline{d})$ in db_2 with $(s(\overline{c}), s(\overline{d})) \in T$ and $eqtp(\overline{a}, \overline{c}) \subseteq eqtp(\overline{b}, \overline{d})$. It remains to show that $\overline{d} = f(\overline{c})$ i.e., that $d_i = f(c_i)$ for every $1 \leq i \leq k$. Fix i arbitrarily. Note that, since $\overline{c} \subseteq \overline{a}$, there exists j_i such that $c_i = a_{j_i}$. Then, because $eqtp(\overline{a}, \overline{c}) \subseteq eqtp(\overline{b}, \overline{d})$, we have $d_i = b_{j_i} = f(a_{j_i}) = f(c_i)$, as desired.

It remains to verify the Forth condition of guarded simulation. Let $f = \overline{a} \mapsto \overline{b}$ be an arbitrary element of $\mathcal{I}[T]$

Forth Let X be a guarded set in db_1. Then there is some fact $s(\overline{c}) \in db_1$ with $X = \{c_1, \ldots, c_l\}$ with l the arity of s. By Tuple-Forth, there exists $s(\overline{d})$ in db_2 with $(s(\overline{c}), s(\overline{d})) \in T$ and $eqtp(\overline{a}, \overline{c}) \subseteq eqtp(\overline{b}, \overline{d})$. Then clearly, since $(s(\overline{c}), s(\overline{d})) \in T$, $\overline{c} \mapsto \overline{d}$ is in $\mathcal{I}[T]$. Moreover, since $eqtp(\overline{a}, \overline{c}) \subseteq eqtp(\overline{b}, \overline{d})$, we know that $\overline{c} \mapsto \overline{d}$ agrees with $\overline{a} \mapsto \overline{b}$ on $\overline{a} \cap \overline{c}$.

□

Note that for simplicity, we have ignored constant data values in simulation. To introduce constants, i.e., the notion that constants can be inspected in relation r on some fixed positions $P_r \subseteq \{1, \ldots, rank(r)\}$, it can further be required in part (1) of Definition 4 that it must be the case that $a_i = b_i$ for each $i \in P_r$. Constants can similarly be incorporated in Definition 3.

3. TWO SYNTAXES FOR THE ACYCLIC CONJUNCTIVE QUERIES

We assume familiarity with the syntax and semantics of first order logic and Codd's relational algebra [21]. The syntax of the guarded conjunctive fragment of first order logic (conjGF) over database schema $\mathcal{S}$ is as follows:

- $r(x_1, ..., x_k)$, where $r \in \mathcal{S}$ is a relation name of arity k and each $x_i, 1 \leq i \leq k$, is a variable, is a (propositional) atomic formula.
- $x = y$, for variables x and y, is an atomic formula.
- if F is an atomic formula, then $F \in$ conjGF.
- if $F, G \in$ conjGF, then $F \wedge G \in$ conjGF.
- if A is a propositional atomic formula, $F \in$ conjGF, the set $free(F)$ of free variables of F is contained in the set of variables of A, and $\overline{x}$ is a list of variables, then $\exists \overline{x}(A \wedge F) \in$ conjGF. Such a formula is said to be "variable guarded" or "strict".
- nothing else is in conjGF.
- if $\exists \overline{x}(A \wedge F) \in$ conjGF and $\overline{x} = (x_1, ..., x_n)$ is a list of variables such that $\{x_1, ..., x_n\} = free(\exists \overline{x}(A \wedge F))$, then $\{\overline{x} \mid \exists \overline{x}(A \wedge F)\}$ is a conjGF query.

The semantics of a conjGF query $q = \{\overline{x} \mid \varphi(\overline{x})\}$ on a database instance db is the set of tuples

$$q(db) = \{v(\overline{x}) \mid db \models \varphi(v(\overline{x})) \text{ and } v \text{ is a valuation over } \overline{x}\}$$

where a "valuation" is a function from a set of variables to the set of atomic data values $\mathcal{U}$.

Note that, following Hirsch [16] and Leinders et al. [20], we restrict our attention in conjGF queries to the so-called strict or variable-guarded formulas, which evaluate to sets of tuples guarded in the same relation.

Towards an alternate syntax for conjGF which we use in the sequel, we take inspiration from Leinders et al. [20], where the equivalence of guarded FO and the semi-join algebra is established.[1] Let conjSA denote the conjunctive fragment of the semi-join algebra, with the following syntax:

- if $r \in \mathcal{S}$ and $k = arity(r)$, then r is a conjSA expression of width k, which we denote by $r : k \in$ conjSA.
- if $e : k \in$ conjSA and $i, j \leq k$, then $\sigma_{i=j}(e) : k \in$ conjSA.
- if $e : k \in$ conjSA, $j \in \mathbb{N}$, and $n_1, ..., n_j$ are each positive integers smaller than or equal to k, then $\pi_{n_1,...,n_j}(e) : j \in$ conjSA.
- if $e_1 : k \in$ conjSA, $e_2 : l \in$ conjSA, and θ is a conjunction of expressions of the form $i = j$, each for some $1 \leq i \leq k$ and $1 \leq j \leq l$, then $e_1 \ltimes_\theta e_2 : k \in$ conjSA.
- nothing else is in conjSA.

The semantics of a conjSA expression e on instance db is the set of tuples $e(db)$ defined as follows:

[1] See also Hirsch's algebraization of guarded FO [16, Section 3.6].

$$e_k^{r(\overline{a})} \quad = \quad \begin{cases} \sigma_{\bigwedge_{i=j \in eqtp(\overline{a},\overline{a})} i=j}(r) & \text{if } k = 0 \\ ((e_0^{r(\overline{a})} \ltimes_{\bigwedge_{i=j \in eqtp(\overline{a},\overline{b}_1)} i=j} e_{k-1}^{s_1(\overline{b}_1)}) \cdots \ltimes_{\bigwedge_{i=j \in eqtp(\overline{a},\overline{b}_n)} i=j} e_{k-1}^{s_n(\overline{b}_n)}) & \text{if } k > 0 \end{cases}$$

Figure 1: Characteristic conjSA expressions of Definition 10.

- if $e := r \in \mathcal{S}$, then $e(db) = db(r)$.
- if $e := \sigma_{i=j}(e')$, then $e(db) = \{t \in e'(db) \mid t[i] = t[j]\}$.
- if $e := \pi_{n_1,\ldots,n_j}(e')$, then $e(db) = \{(t[n_1], ..., t[n_j]) \mid t \in e'(db)\}$.
- if $e := e_1 \ltimes_\theta e_2$, then $e(db) = \{t_1 \in e_1(db) \mid \exists t_2 \in e_2(db)$ such that $t_1[i] = t_2[j]$ for each $i = j \in \theta\}$.

A *query* is a function from database instances to relation instances. By induction on the structure of conjSA expressions and conjGF queries, we can establish the following equivalence between the expressive power of conjSA and conjGF.

PROPOSITION 7. *Let $\mathcal{Q}$ be a query on $\mathcal{S}$. The following are equivalent.*

1. *There exists a* conjSA *expression e such that for all instances db of $\mathcal{S}$, $e(db) = \mathcal{Q}(db)$.*
2. *There exists a* conjGF *query q such that for all instances db of $\mathcal{S}$, $q(db) = \mathcal{Q}(db)$.*

It follows from Proposition 7 and [13, Corollary 3], that

COROLLARY 8. conjSA *and* conjGF *are equivalent in expressive power to the language of strict acyclic conjunctive queries.*[2]

4. STRUCTURAL CHARACTERIZATION OF THE ACYCLIC CONJUNCTIVE QUERIES

We next use Corollary 8 to show that there is a tight relationship between guarded simulation and the acyclic conjunctive queries.

We begin by noting the following invariance lemma, which follows from an induction on the structure of conjSA expressions.

LEMMA 9. conjSA *is invariant under guarded simulations: let db_1 and db_2 be instances of $\mathcal{S}$, and let $\overline{a}_1$ and $\overline{a}_2$ be guarded tuples in db_1 and db_2, respectively. Let I be a guarded simulation of db_1 in db_2 and let $f \in I$ be a partial homomorphism such that $f(\overline{a}_1) = \overline{a}_2$. Then, for every $e \in$ conjSA, it holds that $\overline{a}_1 \in e(db_1)$ implies $\overline{a}_2 \in e(db_2)$.*

The converse of Lemma 9, i.e. that invariance implies the existence of a guarded simulation, is also true, as we show next.

[2]In datalog notation, the *strict* conjunctive queries are those of the form $h \leftarrow p_1, \ldots, p_n$ where all variables of the head h appear together in some subgoal p_i of the body. In the sequel, we just refer to these as conjunctive queries.

DEFINITION 10. *Let db be an instance of $\mathcal{S}$, integer k be non-negative, and $r(\overline{a}) \in db$. Then, given the set*

$$\{s_1(\overline{b}_1), \ldots, s_n(\overline{b}_n)\}$$

of distinct facts in db having non-empty equality type with $r(\overline{a})$, define the family of conjSA *expressions given in Figure 1.*

The parse tree of the expression $e_k^{r(\overline{a})}$ has $\sigma_{\bigwedge_{i=j \in eqtp(\overline{a},\overline{a})} i=j}(r)$ as root, and $\sigma_{\bigwedge_{i=j \in eqtp(\overline{b}_1,\overline{b}_1)} i=j}(s_1), \ldots, \sigma_{\bigwedge_{i=j \in eqtp(\overline{b}_n,\overline{b}_n)} i=j}(s_n)$ as children of the root with edges enforcing for each child $\overline{b}_\ell$ the respective semi-join condition $\bigwedge_{i=j \in eqtp(\overline{a},\overline{b}_\ell)} i = j$. Recurring on each child, the tree continues in this manner to depth k.

EXAMPLE 11. *The $k = 1$ characteristic expression for t_1 of Example 2 is*

$$e_1^{t_1} = ((\sigma_{1=1\wedge 2=2\wedge 3=3}(r) \ltimes_{2=1} \sigma_{1=1\wedge 2=2\wedge 3=3}(r)) \ltimes_{3=1} \sigma_{1=1\wedge 2=2\wedge 3=3}(r))$$

The reader can verify for any $k \geq 0$ that $e_k^{t_1} = e_k^{t_4}$, $t_4 \in e_k^{t_1}(db_1)$, and $t_1 \in e_k^{t_4}(db_1)$.

Clearly:

LEMMA 12. *For every fact $r(\overline{a})$ and every $k \geq 0$ we have $\overline{a} \in e_k^{r(\overline{a})}(db)$.*

These expressions can be optimized (e.g., removing repetition of subqueries by "stratifying" the use of facts), but such is not our concern here. Rather, we have introduced these expressions in order to syntactically characterize those facts which simulate a given fact, as follows.

DEFINITION 13. *Let db_1 and db_2 be database instances. Define the sequence of sets $Z_0, Z_1, \ldots$, with every $Z_k \subseteq db_1 \times db_2$ for $k \geq 0$ as follows:*

$$Z_k = \{(r(\overline{a_1}), r(\overline{a_2})) \mid r(\overline{a_1}) \in db_1, \overline{a_2} \in e_k^{r(\overline{a_1})}(db_2)\}$$

Here $e_k^{r(\overline{a_1})}$ is given as in definition 10.

Note that Z_k is indeed a subset of $db_1 \times db_2$: if $\overline{a_2} \in e_k^{r(\overline{a_1})}(db_2)$ then in particular we must have $\overline{a_2} \in e_0^{r(\overline{a_1})}(db_2)$ and hence, by definition of $e_0^{r(\overline{a_1})}(db_2)$, it must hold that $\overline{a_2} \in db_2(r)$, or equivalently that $r(\overline{a_2}) \in db_2$ as desired.

Also note that for any fact α, any k, and any database db we have $e_{k+1}^\alpha(db) \subseteq e_k^\alpha(db)$. Hence $Z_{k+1} \subseteq Z_k$, for any $k \geq 0$.

LEMMA 14. *Let db_1 and db_2 be database instances, and define the sequence of sets Z_k with $k \geq 0$ as in definition 13. If k is an integer such that (1) $Z_{k+1} = Z_k$ (i.e., Z_k is a fixpoint) and (2) every $\alpha \in db_1$ occurs in the left column of Z_{k+1}, then Z_{k+1} is a tuple simulation of db_1 in db_2.*

We are now ready to establish our main result.

THEOREM 15. *Let db_1 and db_2 be instances of $\mathcal{S}$, integer j be non-negative, and $\overline{a}_1$ and $\overline{a}_2$ be guarded j-tuples in db_1 and db_2, respectively. The following are equivalent.*

1. *For any $e \in$ conjSA, if $\overline{a}_1 \in e(db_1)$ then $\overline{a}_2 \in e(db_2)$.*
2. *There is a guarded simulation I of db_1 in db_2 with partial homomorphism $f \in I$ such that $f(\overline{a}_1) = \overline{a}_2$.*

PROOF. *(1 ⇒ 2).* Since $\overline{a}_1$ is a guarded tuple in db_1, there exists some fact $r(\overline{b}_1) \in db_1$ and positive integers $n_1, \ldots, n_j$ such that $\overline{a}_1 = \pi_{n_1,\ldots,n_j}(\overline{b}_1)$. Then consider the sequence of sets $Z_0, Z_1, Z_2, \ldots$ as defined in Definition 13. Observe that, since $Z_{l+1} \subseteq Z_l$ for any $l \geq 0$, this sequence must converge at some point: i.e., there must exist some k with $Z_{k+1} = Z_k$. Now consider the conjSA expression

$$\mathcal{E} = \pi_{n_1,\ldots,n_j}(e_{k+1}^{r(\overline{b_1})}) \ltimes_{\beta \in db_1}^{\emptyset} e_{k+1}^{\beta}$$

It is readily verified using Lemma 12 that $\overline{a_1} \in \mathcal{E}(db_1)$. By assumption, this implies $\overline{a_2} \in \mathcal{E}(db_2)$. In particular, $\overline{a_2} \in \pi_{n_1,\ldots,n_j}(e_{k+1}^{r(\overline{b_1})})(db_2)$. Hence, there exists some $\overline{b_2} \in e_{k+1}^{r(\overline{b_1})}(db_2)$ with $\overline{a_2} = \pi_{n_1,\ldots,n_j}(b_2)$. As such,

$$(r(\overline{b_1}), r(\overline{b_2})) \in Z_{k+1}.$$

Moreover, by definition of $\mathcal{E}$, there exists, for every fact $\beta = s(\overline{c_1}) \in db_1$, some $\overline{c_2} \in e_{k+1}^{s(\overline{c_1})}(db_2)$. Therefore, for every fact $s(\overline{c_1}) \in db_1$ we have a pair $(s(\overline{c_1}), s(\overline{c_2})) \in Z_{k+1}$. As such, every fact of db_1 occurs in the left column of Z_{k+1}. Hence, Z_{k+1} satisfies the conditions of Lemma 14. As such, Z_{k+1} is a tuple simulation from db_1 to db_2. By Lemma 6, $\mathcal{I}[Z_{k+1}]$ is a guarded simulation from db_1 to db_2. Since $(r(\overline{b_1}), r(\overline{b_2})) \in Z_{k+1}$ and by definition of $\mathcal{I}[\cdot]$ we have $f = \overline{b_1} \mapsto \overline{b_2} \in \mathcal{I}[Z_{k+1}]$. Then $f(\overline{b_1}) = \overline{b_2}$ and hence also $f(\overline{a_1}) = \overline{a_2}$.

(2 ⇒ 1). By Lemma 9. □

We conclude this section by observing the following "van Benthem" style characterization [5] of the acyclic conjunctive queries.

THEOREM 16. *The acyclic conjunctive queries capture precisely the guarded simulation invariant fragment of the conjunctive queries.*

PROOF. (Sketch.) By Corollary 8 and Theorem 15 we have that the acyclic conjunctive queries are invariant under guarded simulations. It remains to show that all guarded simulation invariant conjunctive queries are acyclic.

We establish this by the contrapositive. In datalog notation, let $q = h \leftarrow p_1, \ldots, p_n$ be a cyclic conjunctive query, which we may assume without loss of generality to be minimal, i.e., there is no other query q' equivalent to q having strictly fewer subgoals in its body. It remains to show that q is not invariant under guarded simulation.

As q is cyclic, there must be a cycle in the join graph of q [12]. In other words, there must be pairwise distinct subgoals $p_{c_1}, \ldots, p_{c_k}$ in the body of q such that (1) $k \geq 3$, (2) p_{c_i} and $p_{c_{i+1}}$ share at least one variable, for $1 \leq i < k$, (3) p_{c_k} shares at least one variable with p_{c_1}, and (4) there exists a variable v shared between p_{c_k} and p_{c_1} such that for some $i < k$, v is not shared between p_{c_i} and $p_{c_{i+1}}$.

Let db be the instance of $\mathcal{S}$ consisting of the facts $\{\widehat{p_1}, \ldots, \widehat{p_n}\}$ obtained from the respective subgoals of the body of q by uniformly replacing each distinct variable of q by a distinct constant in $\mathcal{U}$. Let db' be the instance of $\mathcal{S}$ consisting of the facts $\{\widehat{p_1}, \ldots, \widehat{p_n}, \widehat{p'_1}, \ldots, \widehat{p'_n}\}$ where each of $\{\widehat{p_1}, \ldots, \widehat{p_n}\}$ and $\{\widehat{p'_1}, \ldots, \widehat{p'_n}\}$ are distinct instances isomorphic to db except that the constants of $\widehat{p_{c_k}}$ are modified to join with $\widehat{p'_{c_1}}$ (instead of with $\widehat{p_{c_1}}$), the constants of $\widehat{p'_{c_k}}$ are modified to join with $\widehat{p_{c_1}}$ (instead of with $\widehat{p'_{c_1}}$), and these modifications to constants are also uniformly applied to the other facts of $\{\widehat{p_1}, \ldots, \widehat{p_n}\}$ and $\{\widehat{p'_1}, \ldots, \widehat{p'_n}\}$, respectively. In other words, db' has an unfolded copy of the original cycle in db.

Clearly, there exists a guarded simulation of db in db', and $q(db)$ is non-empty by construction. However, $q(db')$ is empty because the cycle $p_{c_1}, \ldots, p_{c_k}$ is not satisfiable in db', by construction. We conclude that q is not invariant under guarded simulations, as desired. □

5. A SMALL EMPIRICAL STUDY

Having established a tight theoretical coupling between the acyclic conjunctive queries and guarded simulation, we next briefly investigate the practicality of the coupling. As discussed in Section 1, an application of such structural characterizations is in the design and implementation of indexing data structures to facilitate efficient query processing.

In the case of acyclic conjunctive queries, we have by Theorem 15 that facts t_1 and t_2 of a database db are indistinguishable in the language if and only if there exist tuple simulations T and T' of db in itself, where $(t_1, t_2) \in T$ and $(t_2, t_1) \in T'$. In this case, we say that the tuples are similar. It is easy to see that similarity of tuples is an equivalence, and hence induces a partition of db into blocks of facts which mutually simulate each other. These blocks, viewed as elements of a reduction of db, are the basis for index data structures tailored specifically to evaluation of acyclic conjunctive queries on db, in the spirit of [4, 8, 14, 17, 23, 28].

Similarity of facts can be computed in polynomial time (e.g., [7, 15]). If, however, simulation-partitions are too large (i.e., close to the size of the original instance), then the obtained reductions are not very useful.

In our work, we are studying the management of data modeled under the RDF standard of the W3C.[3] In particular, we are investigating RDF indexing to support efficient evaluation of SPARQL, the W3C's query language.[4] The heart of SPARQL queries are the so-called basic graph patterns, which are conjunctive queries specialized to RDF [25].

To establish the feasibility of simulation-partition-based indexing for SPARQL evaluation, we performed a preliminary analysis of the reductions achieved when computing the simulation partition of samples from two typical real world RDF data sets: DBPedia[5], an RDF version of the Wikipedia collection, and the RDF version of the Uniprot[6] bioinformatics data set. As sampled data sets, we emphasis that the results obtained are of course preliminary. In order

[3]In essence, an RDF database instance is a finite set of facts of the form $(subject, predicate, object)$. These "triples" are often interpreted as statements of the form "*subject* has relationship *predicate* to *object*." See http://www.w3.org/RDF/

[4]http://www.w3.org/TR/rdf-sparql-query/

[5]http://www.dbpedia.org/

[6]http://www.uniprot.org/

Simulation

	$k=0$	$k=1$	$k=2$	$k=3$	$k=\infty$
DBPedia	0.001	0.005	0.020	0.084	0.508
Uniprot	0.001	0.008	0.035	0.084	0.163

Simulation with Predicate Filtering

	$k=0$	$k=1$	$k=2$	$k=3$	$k=\infty$
DBPedia	0.296	0.725	0.746	0.752	0.752
Uniprot	0.049	0.187	0.210	0.213	0.214

Figure 2: Average reduction factor (i.e., the ratio of the number of blocks in the partition to the number of facts in the original database) of five random samples of 1000 triples from the DBPedia and Uniprot RDF datasets, for neighborhood size k.

to compute simulation on the full datasets, we are currently in the process of developing external memory simulation algorithms, as we have found that current internal memory algorithms have difficulty scaling to large instances.

The results obtained are presented in Figure 2. Considering constants in the predicate position in RDF, following the approach discussed in Section 2, makes sense when values in this position are interpreted as "attribute" names. Indeed, such an interpretation is often the case in practice. Hence, in Figure 2 we give results both with and without such predicate "filtering." We also give results when only considering neighboring nodes within a finite radius k in the tuple-based join graph when computing similarity (i.e., the definition of simulation is recursively applied only to depth k). Such "localization" of the structure considered is often used in index design [8, 17]. We see significant reductions across all variations of similarity, indicating that it is indeed worthwhile to investigate practical applications of these database reductions, in particular for RDF data.

6. CONCLUDING REMARKS

We have given a language-independent characterization of the acyclic conjunctive queries, a fragment of first order logic which is central to the study of database query languages. To our knowledge, this is the first such characterization found. The key notion was that of guarded simulations between database instances. This result complements the established characterization of acyclic first order logic in terms of guarded bisimulations. We also presented results of a preliminary empirical investigation, which indicate the practicality of the characterization.

The study was motivated both internally, as an investigation of a basic fragment of first order logic, and externally by the applications of structural characterizations of the language. There are many interesting directions for future work along both of these lines. We close by listing a few which we are currently pursuing:

- Investigate characterizations of well-behaved generalizations of conjGF, such as the packed guarded fragment [22].
- Current simulation algorithms are designed for internal memory (e.g., [7, 15]). Study efficient similarity computation for databases on external memory
- Study guarded simulation-based indexing data structures and their use in conjunctive query processing.

Acknowledgments

We thank Maarten Marx for his kind assistance with the literature on guarded bisimulations. The research of FP is supported by a FNRS/FRIA scholarship. The research of SV is supported by the OSCB project funded by the Brussels Capital Region. The research of GF, JH, YL, and PD is supported by the Netherlands Organisation for Scientific Research (NWO).

7. REFERENCES

[1] S. Abiteboul, P. Buneman, and D. Suciu. *Data on the web*. Morgan Kaufmann, 2000.

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

[3] H. Andréka et al. Modal languages and bounded fragments of predicate logic. *J Phil Logic*, 27(3), 1998.

[4] A. Arion et al. Path summaries and path partitioning in modern XML databases. *WWW*, 11(1), 2008.

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

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

[7] W. Fan et al. Graph pattern matching: From intractable to polynomial time. *PVLDB*, 3(1), 2010.

[8] G. H. L. Fletcher et al. A methodology for coupling fragments of XPath with structural indexes for XML documents. *Inf Syst*, 34(7), 2009.

[9] G. H. L. Fletcher et al. Relative expressive power of navigational querying on graphs. In *ICDT*, pages 197–207, 2011.

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

[11] R. Goldman and J. Widom. Dataguides: Enabling query formulation and optimization in semistructured databases. In *VLDB*, 1997.

[12] G. Gottlob et al. The complexity of acyclic conjunctive queries. *J ACM*, 48(3), 2001.

[13] G. Gottlob et al. Robbers, marshals, and guards: game theoretic and logical characterizations of hypertree width. *J Comput Syst Sci*, 66(4), 2003.

[14] J. Han, Z. Xie, and Y. Fu. Join index hierarchy: An indexing structure for efficient navigation in object-oriented databases. *IEEE TKDE*, 11(2), 1999.

[15] M. R. Henzinger et al. Computing simulations on finite and infinite graphs. In *FOCS*, 1995.

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

[17] R. Kaushik et al. Covering indexes for branching path queries. In *SIGMOD*, 2002.

[18] A. Kemper and G. Moerkotte. Access support relations: An indexing method for object bases. *Inf Syst*, 17(2):117–145, 1992.

[19] N. Kurtonina and M. de Rijke. Simulating without negation. *J Log Comput*, 7(4):501–522, 1997.

[20] D. Leinders et al. The semijoin algebra and the guarded fragment. *J Log Lang Inf*, 14(3), 2005.

[21] L. Libkin. *Elements of finite model theory*. Springer Verlag, Berlin, 2004.

[22] M. Marx. Queries determined by views: pack your views. In *PODS*, pages 23–30, Beijing, 2007.

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

[24] M. Otto. Highly acyclic groups, hypergraph covers and the guarded fragment. In *LICS*, 2010.

[25] F. Picalausa and S. Vansummeren. What are real SPARQL queries like? In *SWIM*, 2011.

[26] B. Rossman. Homomorphism preservation theorems. *J. ACM*, 55(3), 2008.

[27] D. Sangiorgi. On the origins of bisimulation and coinduction. *ACM TOPLAS*, 31(4), 2009.

[28] P. Valduriez. Join indices. *ACM Trans. Database Syst.*, 12(2):218–246, 1987.

[29] Y. Wu et al. A study of a positive fragment of path queries: Expressiveness, normal form and minimization. *Computer J*, 54(7):1091–1118, 2011.