The Product Homomorphism Problem and Applications

Balder ten Cate[1] and Victor Dalmau[2]

1 **LogicBlox Inc. and UC Santa Cruz**
2 **Universitat Pompeu Fabra**

Abstract

The *product homomorphism problem* (PHP) takes as input a finite collection of structures $\mathbf{A}_1, \ldots, \mathbf{A}_n$ and a structure $\mathbf{B}$, and asks if there is a homomorphism from the direct product $\mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_n$ to $\mathbf{B}$. We pinpoint the computational complexity of this problem. Our motivation stems from the fact that PHP naturally arises in different areas of database theory. In particular, it is equivalent to the problem of determining whether a relation is definable by a conjunctive query, and the existence of a schema mapping that fits a given collection of positive and negative data examples. We apply our results to obtain complexity bounds for these problems.

1998 ACM Subject Classification H.2 Database Management, G.2 Discrete Mathematics

Keywords and phrases Homomorphisms, Direct Product, Data Examples, Definability, Conjunctive Queries, Schema Mappings

Digital Object Identifier 10.4230/LIPIcs.ICDT.2015.161

1 Introduction

Structure identification [7] refers to the general problem of finding a structural description of some data. When the data is relational, this task is usually formalized as the problem of determining whether a given relation can be represented in a given logical formalism. The CQ-definability problem (also called existential inverse satisfiability problem in [6] and PP-definability problem in [13]) is one of the most studied variants of this problem (see [5, 6, 10, 13]). The input of the CQ-definability problem is a relation S and a finite set $R_1, \ldots, R_m$ of relations over the same domain than S, and the question is to decide whether S is expressible by a conjunctive query over the instance I that consists of $R_1, \ldots, R_m$. This question is not only relevant in the context of databases, but is also pertinent to constraint satisfaction. From a constraint satisfaction perspective, CQ-definability can be viewed as asking whether relation S can be expressed as the projection of the set of solutions of a constraint satisfaction instance that uses relations from $R_1, \ldots, R_m$ in its constraints [5] or, equivalently, whether S belongs to the *expressive power* of $R_1, \ldots, R_n$ [10]. In constraint satisfaction, the notion of expressive power plays an important role in the so-called algebraic approach in the study of the constraint satisfaction problem (see [10]). Another collection of problems that can be viewed as instances of structure identification is the *fitting problem for schema mappings*, where the input is a finite collection of data examples (I, J), where I and J are database instances over a source schema and a target schema, respectively, and the question is whether there exists a schema mapping that fits these data examples.

It turns out that the problems described above are closely related to a certain algorithmic problem from graph theory: given a collection of graphs, or more generally, relational structures $\mathbf{A}_1, \ldots, \mathbf{A}_n, \mathbf{B}$, is there a homomorphism from the direct product $\mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_n$ to $\mathbf{B}$. This problem, which we will refer to as the *product homomorphism problem* (PHP), is

18th International Conference on Database Theory (ICDT'15).
Editors: Marcelo Arenas and Martín Ugarte; pp. 161–176

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

clearly decidable in NExpTime: it suffices to materialize the (exponentially large) direct product $\mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_n$ and guess a homomorphism from it to $\mathbf{B}$. It was recently shown [13] that PHP is NExpTime-complete if the arity of the relations in the schema is unbounded. In this paper, we provide a simplified proof, and we establish that the same lower bound holds under various restrictions, including for a fixed schema. We use this to establish tight complexity bounds for CQ-definability as well as for various variants of the fitting problem for schema mappings.

After reviewing basic definitions in Section 2, we study the product homomorphism problem in Section 3. We then proceed with applications to instance-level query definability (Section 4) and to fitting problems for schema mappings (Section 5). We conclude in Section 6.

2 Preliminaries

Schemas, Structures, Database Instances, and Homomorphisms

A *schema* is a nonempty finite set of relation symbols τ of specified arities. A (finite) *structure* $\mathbf{A}$ of the schema τ (or, τ-structure), consists of a finite set A, called the *domain* of $\mathbf{A}$ and a relation $R^{\mathbf{A}} \subseteq A^r$ for every relation symbol $R \in \tau$ where r is the arity of R. Throughout the paper we will use the same boldface and slanted capital letters to denote a structure and its domain respectively.

Let $\mathbf{A}$ and $\mathbf{B}$ be structures of the same schema τ. A *homomorphism* h from $\mathbf{A}$ to $\mathbf{B}$, denoted $h : \mathbf{A} \to \mathbf{B}$, is a function from A to B, such that for every $R \in \mathbf{R}$ and every tuple $a = (a_1, \ldots, a_r) \in R^{\mathbf{A}}$, we have that tuple $h(a)$, defined as $(h(a_1), \ldots, h(a_r))$, belongs to $R^{\mathbf{B}}$. We shall write $\mathbf{A} \to \mathbf{B}$ to indicate that there is a homomorphism from $\mathbf{A}$ to $\mathbf{B}$. When $\mathbf{A} \to \mathbf{B}$ and $\mathbf{B} \to \mathbf{A}$, then we say that $\mathbf{A}$ and $\mathbf{B}$ are *homomorphically equivalent*.

In Section 4 and Section 5 we will consider applications involving *database instances*. Recall that a database instance is a finite structure with an unspecified domain. In other words, the specification of a database instance includes the relations but does not include the domain. For the purposes of this paper, the difference between finite structures and database instances is inessential. This is because all notions and results in this paper are invariant for homomorphic equivalence, and modulo homomorphic equivalence, the domain of a structure can always be assumed to coincide with the *active domain*, that is, the set of all elements occurring in the relations of the structure. We will therefore freely switch from speaking about structures to speaking about instances in Section 4 and Section 5.

Direct products

An n-ary tuple a on A is any element of A^n (for $n \geq 1$). For every $1 \leq i \leq n$ we shall use $a[i]$ to denote its ith component of a. Let $R \subseteq A^n$ be a n-ary relation on A. Then the *projection*, $\mathrm{pr}_{i_1,\ldots,i_j} R$ over coordinates $i_1, \ldots, i_j \in \{1, \ldots, n\}$ is the j-ary relation defined as $\{(a[i_1], \ldots, a[i_j]) \mid a \in R\}$.

Let $a = (a_1, \ldots, a_n) \in A^n$ and $b = (b_1, \ldots, b_n) \in B^n$ be tuples of the same arity. The *direct product* $a \otimes b$ is the n-ary tuple on $A \otimes B$ defined as $((a_1, b_1), \ldots, (a_n, b_n))$. Similarly, if $R \subseteq A^n$ and $S \subseteq B^n$ are n-ary relations then the direct product, $R \otimes S$ is defined to be $\{a \otimes b \mid a \in R, b \in S\}$. The direct product, $\mathbf{A} \otimes \mathbf{B}$, of $\mathbf{A}$ and $\mathbf{B}$ is the τ-structure with domain $A \otimes B$ such that $R^{\mathbf{A}\otimes\mathbf{B}} = R^{\mathbf{A}} \otimes R^{\mathbf{B}}$ for every $R \in \tau$. We shall use $\Pi_{1 \leq i \leq n} R_i$ as a shorthand of $R_1 \otimes \cdots \otimes R_n$ (note that the $\otimes$ operation is associative up to isomorphism). Furthermore, we shall use R^n to denote $\Pi_{1 \leq i \leq n} R$.

The direct product construction is of fundamental importance in the study of graphs and homomorphisms, as it turns out to capture the meet (or, least upper bound) operation of the lattice of structures ordered by homomorphic embedding. That is, for all structures $\mathbf{B}_1, \ldots, \mathbf{B}_n$, we have (i) $\Pi_{1 \leq i \leq n} \mathbf{B}_i \to \mathbf{B}_i$ for all $1 \leq i \leq n$, and (ii) for all structures $\mathbf{A}$, if $\mathbf{A} \to \mathbf{B}_i$ for all $1 \leq i \leq n$, then $\mathbf{A} \to \Pi_{1 \leq i \leq n} \mathbf{B}_i$.

It is important to distinguish direct products from the construction that is usually referred to as *cartesian product* is database theory. Let $R \subseteq A^r$ and $S \subseteq B^s$ be relations of possibly different arity. The *cartesian product* of R and S is the $r+s$-ary relation on $A \cup B$ defined as

$$R \times S = \{(a_1, \ldots, a_r, b_1, \ldots, b_s) \mid (a_1, \ldots, a_r) \in R, (b_1, \ldots, b_s) \in S\}.$$

Conjunctive queries

An n-ary *conjunctive query* q (for $n \geq 0$) is specified by a first-order formula of the form $\phi(x_1, \ldots, x_n) = \exists y_1, \ldots, y_m(\alpha_1 \wedge \cdots \wedge \alpha_k)$, with $m \geq 0$ and $k \geq 1$, where $\alpha_1, \ldots, \alpha_k$ are relational atomic formulas (i.e., atomic formulas not involving equality), and such that each variable x_i occurs in at least one atom α_j. We denote by $q(\mathbf{A})$ the n-ary relation over A defined by $q(\mathbf{A}) = \{a \in A^n \mid \mathbf{A} \models \phi(a[1], \ldots, a[n])\}$. For simplicity, we assume that the atoms in a conjunctive query do not contain any constants (although our results, suitably adapted, can be shown to apply to queries with constants as well). A fundamental property of conjunctive queries is that they are preserved by homomorphisms: if $h : \mathbf{A} \to \mathbf{B}$ is a homomorphism and $a \in q(\mathbf{A})$, then $(h(a[1]), \ldots, h(a[n])) \in q(\mathbf{B})$.

3 The Product Homomorphism Problem

The *product homomorphism problem* (PHP) takes as input a finite collection of relational structures $\mathbf{A}_1, \ldots, \mathbf{A}_n$ and another relational structure $\mathbf{B}$, all over the same schema, and asks whether there is a homomorphism from the direct product $\mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_n$ to $\mathbf{B}$. This problem is clearly solvable in non-deterministic exponential time. It follows from results in [13] that the problem is NExpTime-complete. The proof, based on a reduction from an exponential tiling problem, uses structures of bounded domain size but relations of unbounded arity. We provide a self-contained proof of NExpTime-hardness of PHP, and we show that it holds already for directed graphs, as well as for structures of bounded arity with a bounded domain size (but without a bound on the number of relations). More precisely, we obtain:

▶ **Theorem 1.** *The PHP is* NExpTime*-complete [13]. The lower bound holds already for*

1. *structures with binary relations and a bounded domain size;*
2. *structures with a single relation and a bounded domain size;*
3. *structures with a single binary relation*

This completes the picture, since PHP is solvable in polynomial time when all three of the above parameters (i.e., number of relations, arity, and domain size) are bounded, as follows from the fact that, in this case, there are, up to isomorphism, only a bounded number of possible structures that can be part of the input.

Theorem 1(1) is proved by an adaptation of the technique used in [13]. Theorem 1(2) is proved by a reduction from 1(1). Theorem 1(3) is proved by a reduction from 1(2).

3.1 Proof of Theorem 1(1)

Proof. The proof will be a reduction from the exponential tiling problem which we now define. A *domino system* is a finite structure $\mathbf{D} = (D, H^{\mathbf{D}}, V^{\mathbf{D}})$ where the elements in its

domain, D, are called *tile types* and $H^{\mathbf{D}}, V^{\mathbf{D}}$ are binary relations on D, called the horizontal and vertical adjacency relation, respectively. For $N > 1$, we use $[N]$ to denote the set $\{0, \ldots, N-1\}$. A *tiling* of $[N] \otimes [N]$ by $\mathbf{D}$ is any mapping $\rho : [N] \otimes [N] \to D$ satisfying the following conditions:

- $(\rho(i,j), \rho(i,j+1)) \in H^{\mathbf{D}}$ for every $i \in [N]$ and $j \in [N-1]$
- $(\rho(i,j), \rho(i+1,j)) \in V^{\mathbf{D}}$ for every $i \in [N-1]$ and $j \in [N]$

The input of an exponential tiling problem is constituted by a domino system $\mathbf{D}$, an integer $N > 1$ (written in binary), and a sequence $T_0, \ldots, T_{n-1} \in D$, with $n \leq N$. The question is whether there is a tiling ρ of $[N] \otimes [N]$ such that $\rho(0, j) = T_j$ for every $j \in [n]$.

It is shown in [4] that the exponential tiling problem is NExpTime-hard. As discussed in [13], there exists a single domino system $\mathbf{D}$ such that the problem remains coNExpTime-hard even if the domino system in the input is required to be $\mathbf{D}$ and $N = 2^m$ for some $m > 1$.

Let us introduce a bit of notation. For every $k \in [N]$, let us denote by $\mathbf{b}_k \in \{0,1\}^m$ the m-bit binary representation of k, which we assume starts with the bit of highest weight. Also, we shall use $\Pi_\ell \mathbf{A}_\ell$ as a shorthand for $\Pi_{0 \leq \ell \leq 2m} \mathbf{A}_\ell$

Given an instance $(\mathbf{D}, N, T_0, \ldots, T_{n-1}), N = 2^m$ of the exponential tiling problem we construct in polynomial time an instance $\mathbf{A}_1, \ldots, \mathbf{A}_{2m}$, $\mathbf{B}$, of the PHP. Each one of the structures, $\mathbf{A}_1, \ldots, \mathbf{A}_{2m}$, has domain $\{0,1\}$. In this way, there is a correspondence between $[N] \otimes [N]$ and the domain of $\Pi_\ell \mathbf{A}_\ell$. More precisely, we associate to every element $(i,j) \in [N] \otimes [N]$ the tuple $(x_1, \ldots, x_{2m}) \in \{0,1\}^{2m}$ where $(x_1, \ldots, x_m) = \mathbf{b}_i$ and $(x_{m+1}, \ldots, x_{2m}) = \mathbf{b}_j$.

The domain of $\mathbf{B}$ will be the set, D, of tile types, so that a map h from $\Pi_\ell A_\ell$ to B can be viewed as a way of assigning a tile type to each position on the $[N] \otimes [N]$ grid. Furthermore, by endowing the structures involved with suitable relations, we will ensure that every such mapping h is an homomorphism from $\Pi_\ell \mathbf{A}_\ell$ to $\mathbf{B}$ if and only if it corresponds to a valid tiling.

For ease of exposition, we will also allow unary relations in the instance of the PHP. It is straightforward to replace the unary relations by binary ones.

Let $\mathcal{H}, \mathcal{V}$ be binary relations on the domain $\{0,1\}^{2m}$ that denote the horizontal and vertical successor relations on $[N] \otimes [N]$. Formally

$$\mathcal{H} = \{(\mathbf{b}_i \mathbf{b}_j, \mathbf{b}_i \mathbf{b}_{j+1}) \mid i \in [N], j \in [N-1]\}$$

$$\mathcal{V} = \{(\mathbf{b}_i \mathbf{b}_j, \mathbf{b}_{i+1} \mathbf{b}_j) \mid i \in [N-1], j \in [N]\}$$

where $\mathbf{b}_i \mathbf{b}_j \in \{0,1\}^{2m}$ denotes the $2m$-ary bit string obtained concatenating $\mathbf{b}_i$ and $\mathbf{b}_j$. Also, let $\mathcal{P}_0 = \{\mathbf{b}_0 \mathbf{b}_0\}, \ldots, \mathcal{P}_{n-1} = \{\mathbf{b}_0 \mathbf{b}_{n-1}\}$ be unary singleton relations on the domain $\{0,1\}^{2m}$ that denote the first n cells in the zero row of $[N] \otimes [N]$.

In order to make our reduction work, we need to somehow make sure that the relations $\mathcal{H}, \mathcal{V}, \mathcal{P}_0, \ldots, \mathcal{P}_{n-1}$ are "available" in the product structure $\Pi_\ell \mathbf{A}_\ell$, by choosing the relations in structures $\mathbf{A}_1, \ldots, \mathbf{A}_{2m}$ appropriately.

Let us say that an r-ary relation R over domain $\{0,1\}^{2m}$ is *factorizable* if it can be represented as a direct product $R_1 \otimes \cdots \otimes R_{2m}$ where each R_ℓ is an r-ary relation over $\{0,1\}$.

Intuitively, this means that if we include in each structure $\mathbf{A}_\ell$ the relation R_ℓ, then the product structure $\Pi_\ell \mathbf{A}_\ell$ will contain the relation R. Each of the unary relations $\mathcal{P}_0, \ldots, \mathcal{P}_n$, being a singleton, is trivially factorizable. Indeed, for every $j \in [n]$, $\mathcal{P}_j = \{0\}^m \otimes \{\mathbf{b}_j[1]\} \otimes \cdots \otimes \{\mathbf{b}_j[m]\}$.

The binary relation $\mathcal{H}$ is *not* factorizable. However, it turns out to be a union of a small number of factorizable relations, which will suffice for our purposes. For each $k \in [m]$, let $\mathcal{H}_k = \mathcal{H} \cap (\{0,1\}^{2m-k-1} \otimes \{0\} \otimes \{1\}^k \otimes \{0,1\}^{2m})$. In words, $\mathcal{H}_k$ is the subrelation of $\mathcal{H}$ that contains all those $(\mathbf{b}_i \mathbf{b}_j, \mathbf{b}_i \mathbf{b}_{j+1}) \in \mathcal{H}$ such that $\mathbf{b}_j$ finishes with a zero followed by k

ones. Note that for every $(\mathbf{b}_i\mathbf{b}_j, \mathbf{b}_i\mathbf{b}_{j+1}) \in \mathcal{H}$, $\mathbf{b}_j$ must contain one zero. It follows that $\mathcal{H} = \bigcup_{k\in[m]} \mathcal{H}_k$. Furthermore, it is easy to see that $\mathcal{H}_k$ factorizes as

$$\mathcal{H}_k = \mathsf{id}^{2m-k-1} \otimes \{(0,1)\} \otimes \{(1,0)\}^k$$

where $\mathsf{id} = \{(0,0),(1,1)\}$ is the equality relation on $\{0,1\}$.

Similarly, we can express $\mathcal{V}$ as $\bigcup_{k\in[m]} \mathcal{V}_k$ where

$$\mathcal{V}_k = \mathsf{id}^{m-k-1} \otimes \{(0,1)\} \otimes \{(1,0)\}^k \otimes \mathsf{id}^m .$$

We are now ready to define the structures $\mathbf{A}_1, \ldots, \mathbf{A}_{2m}$ and $\mathbf{B}$. The scheme consists of the relations $H_0, \ldots, H_{m-1}, V_0, \ldots, V_{m-1}, P_0, \ldots P_{n-1}$. As mentioned above, the domain of $\mathbf{A}_\ell$ is $\{0,1\}$ for every $1 \leq \ell \leq 2m$. For $k \in [n], 1 \leq \ell \leq 2m$, we define

$$P_k^{\mathbf{A}_\ell} = \begin{cases} \{\mathbf{b}_k[\ell - m]\} & \text{if } m < \ell \\ \{0\} & \text{otherwise} \end{cases}$$

For $k \in [m], 1 \leq \ell \leq 2m$, we define

$$H_k^{\mathbf{A}_\ell} = \begin{cases} \{(1,0)\} & 2m - k < \ell \\ \{(0,1)\} & \ell = 2m - k \\ \mathsf{id} & \text{otherwise} \end{cases}$$

$$V_k^{\mathbf{A}_\ell} = \begin{cases} \{(1,0)\} & m - k < \ell \leq m \\ \{(0,1)\} & \ell = m - k \\ \mathsf{id} & \text{otherwise} \end{cases}$$

The domain of structure $\mathbf{B}$ is the set, D, of all tile types. For $k \in [n]$ we define $P_k^{\mathbf{B}} = \{T_k\}$, and for $k \in [m]$, we define $H_k^{\mathbf{B}} = H^{\mathbf{D}}$, and $V_k^{\mathbf{B}} = V^{\mathbf{D}}$.

It follows from the definitions that $P_k^{\Pi_\ell \mathbf{A}_\ell} = \mathcal{P}_k$ for all $k \in [n]$ and that $H_k^{\Pi_\ell \mathbf{A}_\ell} = \mathcal{H}_k$ and $V_k^{\Pi_\ell \mathbf{A}_\ell} = \mathcal{V}_k$ for every $k \in [m]$. It follows that there is a homomorphism from $\Pi_\ell \mathbf{A}_\ell$ to $\mathbf{B}$ if and only if there is a valid tiling. The reduction we have just defined can be easily carried out in polynomial time. ◀

3.2 Proof of Theorem 1(2)

Proof. The proof proceeds by a reduction from the PHP with bounded domain size (Theorem 1(1)). We shall show how to construct, given an instance $\mathbf{A}_1, \ldots, \mathbf{A}_n, \mathbf{B}$ of the PHP another one with only one relation in each structure. Furthermore, the reduction will only increase the domain size of each structure by one.

Let $R_1, \ldots, R_k$ be the relation symbols in the scheme. The most immediate way to do the reduction consists in to replace, in every structure $\mathbf{C}$ among $\mathbf{A}_1, \ldots, \mathbf{A}_n, \mathbf{B}$, relations $R_1^{\mathbf{C}}, \ldots, R_k^{\mathbf{C}}$ by a single relation $R_1^{\mathbf{C}} \times \cdots \times R_k^{\mathbf{C}}$. It is not difficult to see that a mapping $h : \Pi_i A_i \to B$ is an homomorphism in the original instance if and only is an homomorphism in the instance obtained after the transformation. However, this transformation cannot be carried out in polynomial time as computing the cartesian product of k relations requires time exponential on k. To overcome this difficulty we shall apply first an step which will reduce the number of relations to 2.

We can assume that all relation symbols $R_1, \ldots, R_k$ have arity 2. This assumption is not essential but simplifies slightly the presentation. Let $\mathbf{C}$ be any structure among $\mathbf{A}_1, \ldots, \mathbf{A}_n, \mathbf{B}$ and let 0 be a fresh element not occurring in $B \cup \bigcup_{1\leq i\leq n} A_i$. We denote by $\mathbf{C}^*$ the structure with domain $C \cup \{0\}$ with the following relations:

(i) a unary relation $P^{\mathbf{C}^*} = C$
(ii) a $2k$-ary relation $R^{\mathbf{C}^*}$ defined as

$$R^{\mathbf{C}^*} = \{0^{2k}\} \cup \bigcup_{1 \leq j \leq k} \{0^{2(j-1)}\} \times R_j^{\mathbf{C}} \times \{0^{2(k-j)}\}\,.$$

That is, $R^{\mathbf{C}^*}$ contains all-zeroes tuple $(0, \cdots, 0)$, and, for every $1 \leq j \leq k$ and every tuple $(a, b) \in R_j$ $(1 \leq j \leq k)$, the $2k$-ary tuple $(a_1, \ldots, a_{2k})$ where all coordinates are 0 with the exception of a_{2j-1} and a_{2j} which are a and b respectively.

This transformation can be carried out in polynomial time and it increases the domain of each structure with at most one element. We claim that $\Pi_i \mathbf{A}_i^* \to \mathbf{B}^*$ if and only if $\Pi_i \mathbf{A}_i \to \mathbf{B}$.

In one direction, suppose h^* is an homomorphism from $\Pi_i \mathbf{A}_i^*$ to $\mathbf{B}^*$. It follows that h^* must map every element of $\Pi_i A_i = P^{\Pi_i \mathbf{A}_i^*}$ to an element of $B = P^{\mathbf{B}^*}$. It is then easy to see that the restriction h of h^* with domain $\Pi_i A_i$ is in fact a homomorphism from $\Pi_i \mathbf{A}_i$ to $\mathbf{B}$. Indeed, let $1 \leq j \leq k$ and let $(a, b) \in R_j^{\Pi_i \mathbf{A}_i}$. It follows by the definition of R, that $\mathrm{pr}_{2j-1,2j}\, R^{\mathbf{C}^*} = R_j^{\mathbf{C}} \cup \{(0,0)\}$ for every $\mathbf{C}$ among $\mathbf{A}_1, \ldots, \mathbf{A}_n, \mathbf{B}$. Consequently we have that $(a, b) \in \mathrm{pr}_{2j-1,2j}\, R^{\Pi_i \mathbf{A}*_i}$. It follows that h maps (a, b) to a tuple in $R_j^{\mathbf{B}} \cup \{(0,0)\}$. Since this tuple must be in B^2 we are done.

Conversely, suppose h is an homomorphism from $\Pi_i \mathbf{A}_i$ to $\mathbf{B}$. Let h^* be the map from $\Pi_i A_i^*$ to B^* that extends h such that every element in $\Pi_i A_i^*$ containing a 0 is sent to the element 0 of B^*. Formally, $h^*(a)$ is defined to be $h(a)$ whenever $a \in \Pi_i A_i$ and 0 otherwise. We shall see that h^* is an homomorphism from $\Pi_i \mathbf{A}_i^*$ to $\mathbf{B}^*$. Let a be any element in $P^{\Pi_i \mathbf{A}_i^*}$. It follows from the definition of relation P that $a \in \Pi_i A_i$. Then, $h^*(a)$ is $h(a)$ which necessarily belongs to $B = P^{\mathbf{B}^*}$. Now, let $a = (a_1, \ldots, a_{2k})$ be any tuple in $R^{\Pi_i \mathbf{A}_i^*}$ and let J be the set containing all coordinates $j \in \{1, \ldots, 2k\}$ such that $a_j \in \Pi_i A_i$. It follows from the definition of R that J is empty or is of the form $\{a_{2j-1}, a_{2j}\}$ for some $j \in \{1, \ldots, k\}$. If $J = \emptyset$ then $h^*(a) = (0, \ldots, 0) \in R^{\mathbf{B}^*}$ (h is applied component-wise). Now assume that $J = \{a_{2j-1}, a_{2j}\}$. Note that $h^*(a_i)$ is $h(a_i)$ if $i \in \{2j-1, 2j\}$ and 0 otherwise. We have that $(a_{2j-1}, a_{2j}) \in R_j^{\Pi_i \mathbf{A}}$ and hence $(h(a_{2j-1}), h(a_{2j})) \in R_\ell^{\mathbf{B}}$. Consequently, $h^*(a) \in R^{\mathbf{B}}$. ◀

3.3 Proof of Theorem 1(3)

Proof. We shall give a reduction from the PHP with a single relation (Theorem 1(2)). Let $\mathbf{A}_1, \ldots, \mathbf{A}_n, \mathbf{B}$ be an instance of the PHP over a scheme containing a single r-ary relation R. We may assume without loss of generality that, for each structure $\mathbf{C} = (C, R^{\mathbf{C}})$ among $\mathbf{A}_1, \ldots, \mathbf{A}_n, \mathbf{B}$, the projection of $R^{\mathbf{C}}$ to the first coordinate is the entire domain C. This is because we can always replace the r-ary relation R by the $(r+1)$-ary relation $C \times R$. This transformation can be carried out in polynomial time and it does not affect the existence or non-existence of a homomorphism from $\Pi_i \mathbf{A}_i$ to $\mathbf{B}$.

For every $i \in \{1, \ldots, n\}$, we denote by $G(\mathbf{A}_i)$ the digraph defined as follows. The nodes of $G(\mathbf{A}_i)$ include all elements in A_i. Furthermore, for every tuple $t \in R^{\mathbf{A}_i}$, $G(\mathbf{A}_i)$ contains r additional nodes, which we denote $t^1, \ldots, t^r$. Furthermore, we include the following edges:

- (t^j, t^{j+1}) for every $1 \leq j < r$.
- $(t[j], t^j)$ for every $1 \leq j \leq r$.

We define $G(\mathbf{B})$ as the digraph obtained from $\mathbf{B}$ in the same way, except that we further add $r-1$ additional elements $s^1, \ldots, s^{r-1}$ called *sink nodes*. We also add edge (s^j, s^{j+1}) for every $1 \leq j < r-1$. Furthermore we add an edge from every element in B to every sink node.

Claim: There is a homomorphism $h' : \Pi_i G(\mathbf{A}_i) \to G(\mathbf{B})$ if and only if there is a homomorphism $h : \Pi_i \mathbf{A}_i \to \mathbf{B}$.

In the remainder, we prove this claim, which immediately implies the theorem. We start with the more difficult direction. Let h be a homomorphism from $\Pi_i \mathbf{A}_i$ to $\mathbf{B}$. We shall define from h a homomorphism h' from $\Pi_i G(\mathbf{A}_i)$ to $G(\mathbf{B})$. Let $v = (v_1, \ldots, v_n)$ be a node of $\Pi_i G(\mathbf{A}_i)$.

- If $v_i \in A_i$ for every $1 \leq i \leq n$ then we say that v is of "type 1". In this case we define $h'(v) = h(v)$.
- If, for every $1 \leq i \leq n$, $v_i = t_i^{j_i}$ where $t_i \in R^{\mathbf{A}_i}$ and $j_i \in \{1, \ldots, r\}$ then:
  - If, in addition, there exists some j such that $j_i = j$ for every $1 \leq i \leq n$ then we say that v is of "type 2". Note that $\Pi_i t_i$ is a tuple in $R^{P i_i \mathbf{A}_i}$ and hence $h(\Pi_i t_i)$ (where h is applied component-wise) is a tuple in $R^{\mathbf{B}}$. In this case, define $h'(v)$ to be $h(\Pi_i t_i)^j$.
  - Otherwise we say that v is of "type 3" and we set $h(v')$ to the sink node s^j where $j = \min\{j_i \mid 1 \leq i \leq n\}$. Observe that, in this case, necessarily $j \leq r - 1$.
- If v is not of any of the previous types then we say that is of "type 4". In this case, we shall prove there exists a vertex u of type 1 such that for every vertex w of type 2 the following holds:

$$(v, w) \text{ is an edge of } \Pi_i G(\mathbf{A}_i) \Rightarrow (u, w) \text{ is an edge of } \Pi_i G(\mathbf{A}_i)\,.$$

  In this case we set $h'(v) = h'(u)$. Let us define u to be $(u_1, \ldots, u_n)$ where for every $1 \leq i \leq n$, u_i is defined as follows: If $v_i \in A_i$ then set $u_i = v_i$. Otherwise, $v_i = t_i^{j_i}$ for some $t_i \in R^{\mathbf{A}_i}$. Set u_i to be $t_i[j_i + 1]$ if $j_i < r$ and to be any arbitrary element in A_i otherwise. Let $w = (w_1, \ldots, w_n)$ be any node of type 2. We shall prove that for every $1 \leq i \leq n$, if (v_i, w_i) is an edge of $G(\mathbf{A}_i)$ then so is (u_i, w_i). The claim is obvious whenever $u_i = v_i$. Assume now that $v_i = t_i^{j_i}$ for some $t_i \in R^{\mathbf{A}_i}$. Since w is of type 2 it follows that $w_i = t_i^{j_i+1}$. The claim follows from the fact that $G(\mathbf{A}_i)$ contains edge $(t_i[j_i + 1], t_i^{j_i+1})$.

Let us prove that h' is indeed a homomorphism. Let (u, v) be an edge in $\Pi_i G(\mathbf{A}_i)$ and let $u = (u_1, \ldots, u_n)$ and $v = (v_1, \ldots, v_n)$. We shall prove that $(h(u), h(v))$ belongs to $G(\mathbf{B})$ by means of a case analysis on the types of u and v. Notice that v is necessarliy of type 2 or 3 since nodes of type 1 or 4 do not have incoming edges.

- u is of type 1. If v is of type 3 the claim follows from the fact that $G(\mathbf{B})$ has an edge from every element in B to every sink vertex. Assume now that v is of type 2, that is, v is of the form $(t_1^j, \ldots, t_n^j)$. Since (u, v) is an edge of $\Pi_i G(\mathbf{A}_i)$ and u is of type 1 it follows that $u_i = t_i[j]$ for every $1 \leq i \leq n$. Hence $u = (\Pi_i t_i)[j]$ and, since h defines a homomorphism, $h(u)$ is the jth component of $h(\Pi_i t_i)$ (h is applied component-wise). Since $h'(u) = h(u)$, it follows that $G(\mathbf{B})$ contains the edge from $h'(u)$ to $h'(v) = h(\Pi_i t_i)^j$.
- u is of type 2. Then necessarily there exists $t_1, \ldots, t_n$ and j such that $u = (t_1^j, \ldots, t_n^j)$ and $v = (t_1^{j+1}, \ldots, t_n^{j+1})$ and the claim follows directly from the definitions.
- u is of type 3. Then v is necessarily of type 3 as well. Furthermore, it follows that if $h'(u)$ is s^j then necessarily $h'(v) = s^{j+1}$.
- u is of type 4. It follows directly from the definition of $h'(u)$ and the fact that every vertex of type 3 is mapped by h' to a sink node.

Conversely, let h' be a homomorphism from $\Pi_i G(\mathbf{A}_i)$ to $G(\mathbf{B})$. We start by showing that there exists a conjunctive query q with r free variables $x_1, \ldots, x_r$ such that for every $\mathbf{C}$ among $\mathbf{A}_1, \ldots, \mathbf{A}_n, \mathbf{B}$,

$$q(G(\mathbf{C})) = R^{\mathbf{C}}\,. \tag{1}$$

Let $\mathbf{C}$ be among $\mathbf{A}_1, \ldots, \mathbf{A}_n, \mathbf{B}$. Consider first the unary conjunctive query $p(z_1)$ stating that there is a directed path of lenght r starting at z_1. Formally,

$$p(z_1) = \exists z_2, \ldots, z_{r+1}(\bigwedge_{1 \leq j \leq r} E(z_j, z_{j+1}))$$

where E is the edge relation.

By the construction of $G(\mathbf{C})$ it follows that $p(G(C))$ is precisely the projection of $R^{\mathbf{C}}$ to the fist coordinate, which we have assumed to be C. Now we define q to be the conjunctive query

$$q(x_1, \ldots, x_r) = \exists y_1 \ldots y_r(\bigwedge_{1 \leq j \leq r} p(x_j) \bigwedge_{1 \leq j \leq r} E(x_j, y_j) \wedge \bigwedge_{1 \leq j < r} E(y_j, y_{j+1}))$$

where we assume that $p(x_1), \ldots, p(x_n)$ use different existential variables. It is not difficult to see that q satisfies (1).

With the help of q it is very easy to complete the proof. Indeed, Now, let t be any tuple in $R^{\Pi_i \mathbf{A}_i}$. It follows that $t = \Pi_i t_i$, where $t_i \in R^{\mathbf{A}_i}$ for every $1 \leq i \leq n$. It follows from (1) that $t_i \in q(G(\mathbf{A}_i))$ for every $1 \leq i \leq n$. Then, $t \in q(\Pi_i G(\mathbf{A}_i))$ and therefore, since conjunctive queries are preserved by homomorphisms, $h(t)$ belongs to $q(G(\mathbf{B}))$. It follows from (1) that $h(t) \in R^{\mathbf{B}}$.

Note that for every digraph among $G(\mathbf{A}_1), \ldots, G(\mathbf{A}_m), G(\mathbf{B})$ the maximum length of a directed path is r. This will be used in the proof of Theorem 2. ◀

4 First application: instance-level query definability

Instance-level query definability refers to definability of relations inside a given database instance, with respect to some query language. We focus here on the *CQ-definability problem*, which consists in deciding, given a database instance I and a relation S over the active domain of I, whether there is a conjuctive query q such that $q(I) = S$. Recall that a database instance, for present purposes, can be defined as a finite structure (note that conjunctive queries are domain-independent).

It has been long known that the CQ-definability problem is decidable in coNExpTime (see discussion and references in [13]). For the sake of completeness we include a short description of the algorithm that places the problem in coNExpTime. To describe it, we need to introduce the notion of polymorphism. A *polymorphism* of a relation $S \subseteq D^r$ is any operation $f : D^k \to D$, $k \geq 0$ such that the following holds: for every k (not necessarily different) tuples $t_1, \cdots, t_k \in S$, the tuple, $f(t_1, \cdots, t_k)$, obtained by applying f componentwise to $t_1, \cdots, t_k$, belongs also to S. It is well known (see for example [13]) that $q(I) = S$ for some conjunctive query q if and only if every m-ary operation that is a polymorphism of all relations in I is also a polymorphism of S, where m is the size (number of tuples) of S. The coNExpTime algorithm for the CQ-definability is a straightforward application of the previous result: the algorithm, basically, guesses operation f and verifies that f is a polymorphism of all relations in I but not of S.

Additionally, it was shown in [13] that the CQ-definability problem is coNExpTime-complete, even for database instances with a bounded active domain size. However, the proof used relations of arbitrarily large arity. We show that the same problem is coNExpTime-complete for a fixed schema (but without a bound on the size of the domains of the database instances).

▶ **Theorem 2.** *The CQ-definability problem is co*NExpTime*-hard already for unary queries over a fixed schema consisting of a single binary relation.*

Proof. We shall give a reduction from the PHP with a single binary relation (Theorem 1(3)). Let $\mathbf{A}_1, \ldots, \mathbf{A}_n, \mathbf{B}$ be the input of PHP where the schema contains only a binary relation R. Inspection of the proof of Theorem 1(3) shows that we may assume that, in each of these stuctures, the maximum length of a directed path is precisely r, for some fixed natural number r. Let $\mathbf{C}$ be the database instance consisting of the disjoint union of $\mathbf{A}_1, \ldots, \mathbf{A}_n$ and $\mathbf{B}$, extended with the facts $R(a_i, x)$ for all $i \leq n$ and $x \in \mathbf{A}_i$, and $R(b, x)$ for all $x \in \mathbf{B}$, where $a_1, \ldots, a_n$ and b are fresh elements. Observe that each a_i, and also b, by construction, has an outgoing path of length $r+1$, while no other elements have an outgoing path of length $r+1$. We make use of this below. Let $S = \{a_1, \ldots, a_n\}$. Then we claim that $\mathbf{A}_1 \otimes ... \otimes \mathbf{A}_n \to \mathbf{B}$ if and only if S is not definable inside $\mathbf{C}$ by a conjunctive query. In one direction, if $\mathbf{A}_1 \otimes \ldots \otimes \mathbf{A}_n \to \mathbf{B}$ then clearly S is not definable by a conjunctive query, because, by homomorphism preservation, the same conjunctive query would have to select b. On the other hand, if $\mathbf{A}_1 \otimes ... \otimes \mathbf{A}_n \not\to \mathbf{B}$, then we can construct a query q defining S as follows: first we take $q_1 = \exists y_1, \ldots, y_k\, \psi(y_1, \ldots, y_k)$ to be the canonical Boolean conjunctive query of $\mathbf{A}_1 \otimes \ldots \otimes \mathbf{A}_n$, and, then, we define $q(x)$ to be the unary conjunctive query

$$\exists y_1 \ldots, y_k\, (R(x, y_1) \wedge \cdots \wedge R(x, y_k) \wedge \psi(y_1, \ldots, y_k))$$

expressing that q_1 holds in the submodel of $\mathbf{C}$ consisting of all elements reachable (in one step) from the element denoted by x. By construction, $q(\mathbf{C})$ includes all of S and excludes b. It is also easy to see that $q(\mathbf{C})$ contains no elements other than $a_1, \ldots, a_n$ and b (we are using here the fact that structures $\mathbf{A}_1, \ldots, \mathbf{A}_n$ and $\mathbf{B}$ do not contain a path of length $r+1$). Therefore, q defines S. ◀

It is also natural to consider a *generalized* version of CQ-definability, where the input is a finite sequence of pairs $(I_1, S_1), \ldots, (I_n, S_n)$, where each I_i is a database instance and each S_i is a relation over the active domain of I_i (all of the same arity), and the task is to decide whether there is a conjunctive query q, such that, for all $i \leq n$, $q(I_i) = S_i$.

▶ **Theorem 3.** *The generalized CQ-definability problem is co*NExpTime*-complete.*

Proof. The lower bound follows immediately from Theorem 2. For the upper bound: let $(I_1, S_1), \ldots, (I_n, S_n)$ be a given input for the generalized CQ-definability problem. We may assume without loss of generality that the active domains of $I_1, \ldots, I_n$ are disjoint. We extend the schema with an additional binary relation E, and construct a new instance I that is the disjoint union of $I_1, \ldots, I_n$, where E is interpreted as the equivalence relation consisting of all pairs (a, b), such that a and b belong to the active domain of the same instance I_i. Furthermore, let $S = S_1 \cup \cdots \cup S_n$.

Let q be any conjunctive query, over the original schema, such that $q(I_i) = S_i$ for all $i \leq n$. Let q' be obtained from q by adding conjuncts $E(x, y)$ for all pairs of variables x, y occurring (free or bound) in the query q. Then it is easy to show that $q'(I) = S$. Conversely, let q be any conjunctive query (possibly referring to the binary relation E) such that $q(I) = S$. Let q' be obtained from q by dropping all conjuncts involving the relation E. Then it is easy to show that $q'(I_i) = S_i$ for all $i \leq n$. We conclude that $(I_1, S_1), \ldots, (I_n, S_n)$ is a yes-instance for the generalized CQ-definability problem if and only if (I, S) is a yes-instance for the CQ-definability problem. ◀

Analogous to the CQ-definbility problem, one can consider the FO-definability problem, where the task is to decide, given a database instance I and a relation S over the domain of I, whether there is a first-order query q such that $q(I) = S$. This problem was considered, under the name BP-PAIR, in [9], where it was observed that the Banchilon-Paredaens completeness

theorem [3, 12] implies that this problem is in coNP and co-GI-hard, where GI is the class of problems reducible to the graph-isomorphism problem. The same holds for the generalized FO-definability problem, also known as BP-PAIRS. Determining the exact complexity of BP-PAIR and BP-PAIRS is an open problem [9]. Recently, in [2], instance-level definability was studied for regular path queries and for conjunctive regular path queries in the context of graph databases.

5 Second application: the fitting problem for schema mappings

The fitting problem for schema mappings was introduced and studied in [1]. We briefly review the relevant definitions. Fix two disjoint finite relational schemas, $\mathbf{S}$ and $\mathbf{T}$, which we will call the *source schema* and the *target schema*. A *GLAV (Global-and-Local-As-View) constraint* is a first-order sentence of the form

$$\forall \mathbf{x}(\phi(\mathbf{x}) \rightarrow \exists \mathbf{y}\psi(\mathbf{x}, \mathbf{y}))$$

where $\phi(\mathbf{x})$ is a conjunction of one or more atomic relational formulas over the source schema containing all the variables in $\mathbf{x}$, and where $\psi(\mathbf{x}, \mathbf{y})$ is a conjunction of one or more relational atomic formulas over the target schema containing all variables in $\mathbf{y}$. As in the case of conjunctive queries, for simplicity, we assume that the atoms in a GLAV constraint do not contain constants. A *GLAV schema mapping* is a finite set of GLAV constraints.

GLAV schema mappings are used extensively in *data exchange* and in *data integration* [11]. They provide the formal foundation for these data-interoperability tasks by specifying the relationships between the two schemas. Two important special cases of GLAV schema mappings are *GAV (Global-As-View) schema mappings* and *LAV (Local-As-View) schema mappings*. These consists of GAV constraints, and LAV constraints, respectively. A GAV constraint is a GLAV constraint whose consequent $\psi(\mathbf{x})$ consists of a single atomic formula without existential quantifiers, while a LAV constraint is a GLAV constraint whose antecedent $\phi(\mathbf{x}, \mathbf{y})$ consists of a single atomic formula. Finally, a 1GAV schema mapping is a GAV schema mapping such that each relation from the target schema occurs in only one constraint. We can think of a GAV schema mapping (or, a 1GAV schema mapping) as associating, to each target relation, a UCQ view over the source (respectively, a CQ view over the source), while we can think of a LAV schema mapping as associating, to each source relation, a CQ view over the target. GAV, LAV, and GLAV schema mappings are the most widely studied, and the most widely used, types of schema mappings.

Consider a schema mapping M and a pair (I, J), where I and J are database instances over the source $\mathbf{S}$ and the target schema $\mathbf{T}$, respectively. When (I, J), viewed as a single database instance over the union of the two schemas, satisfies the constraints of M, then we say that J is a *solution* of I (with respect to M). We say that J is a *universal solution* for I (with respect to a schema mapping M), if J is a solution for I, and for every solution J' of I, there is a homomorphism from $h : J \rightarrow J'$, where $h(a) = a$ for all a in the active domain of I. It was argued in [8] that universal solutions are the preferred solutions in data exchange. Furthermore, it was shown in [8] that, in the case of GLAV schema mappings, every source instance has a universal solution, and a universal solution can be constructed in polynomial time (data complexity).

Several methodologies have been proposed and used for obtaining schema mappings in practice. In particular, in [1], a *data example*-driven approach to schema mapping design is advocated. An (unlabelled) *data example*, here, is a pair (I, J), where I is a finite structure over the source schema and J is a finite structure over the target schema. Data examples

data examples	LAV	GAV	1GAV	GLAV
universal	NP-cmp [1]	coNP-cmp [1]	coNEXPTIME-cmp*	Π_2^p-cmp [1]
pos. & neg.	coNEXPTIME-cmp*	coNP-cmp*	coNEXPTIME-cmp*	in coN2EXPTIME and coNEXPTIME-hard*

Figure 1 Complexity of variants of the fitting problem for schema mappings (* marks new results).

can be used in different ways to describe a schema mapping. We say that a data example (I, J) is a *positive example* for a schema mapping M if J is a solution for I with respect to M, and it is a *negative example* otherwise. Finally, (I, J) is a *universal example* for M if J is a universal solution for I with respect to M.

The *fitting problem for GLAV (GAV, LAV) schema mappings with positive/negative/universal examples* is the problem where the input is a finite collection of data examples, each labeled as being a positive example or a negative example or a universal example, and the problem is to decide whether there exists a GLAV (GAV, 1GAV, LAV) schema mapping that is consistent with this marking (in other words, that "fits" these data examples). We will follow the literature by considering the *data complexity* of this problem, where the input is the collection of marked data examples, while the source and target schema are assumed to be fixed. The fitting problem for universal examples was studied in [1]. It was shown there that this problem is coNP-complete for GAV schema mappings; NP-complete for LAV schema mappings; and Π_2^p-complete for GLAV schema mappings. It was also shown in [1] that if, for a given set of universal examples, a fitting GLAV (or GAV, LAV) schema mapping exists, then there is one whose size is linear in the combined size of the data examples.

Although it is argued in [1] that universal examples are the most natural and well-behaved type of data examples, it is also important to consider fitting problems where the input is a collection of positive and negative examples. Indeed, for richer schema mapping languages (beyond GLAV), in general, a given source instance may not *have* a universal solution, and hence, we cannot always work with universal examples. Below, we determine the complexity of the fitting problem with positive and negative examples, for the various schema mapping languages introduced above. The main results are summarized in Figure 1.

First, we establish some convenient lemmas. First, the *direct product* construction naturally generalizes to instances with designated elements. Let $n, m \geq 0$ and let $(I_1, \mathbf{a}_1), \ldots, (I_n, \mathbf{a}_n)$, where each I_n is an instance, over the same schema, and where each $\mathbf{a}_i$ is a sequence of m elements of the active domain of I_i ($m \geq 0$). We denote by $\Pi_{1 \leq i \leq n}(I_i, \mathbf{a}_i)$ the pair $(\Pi_{1 \leq i \leq n}(I_i), \mathbf{b})$, where $\mathbf{b}$ is the m-tuple that is the direct product of $\mathbf{a}_1, \ldots, \mathbf{a}_n$.

We will make use of the following fundamental notion from database theory: the *canonical query* of $(I, \mathbf{a})$ is the query $q(\mathbf{x}) = \exists \mathbf{y} \psi(\mathbf{x}, \mathbf{y})$ obtained by associating a first-order variable to each element of the active domain of I, taking ψ to be conjunction of all facts of I using these variables, and existentially quantifying all variables corresponding to elements that do not belong to $\mathbf{a}$. Note that the free variables $\mathbf{x}$ of the query are the variables that correspond to the elements in $\mathbf{a}$.

▶ **Theorem 4.** **(i)** *Let E be a finite collection of positive and negative data examples over a fixed source and target schema. If there exists a LAV schema mapping that fits E, then there exists one of size at most $2^{O(n)}$, where n is the total size of the data examples in E.*

(ii) *The fitting problem for LAV schema mappings with positive and negative examples (over a fixed source and target schema) is co*NEXPTIME*-complete.*

Proof. (i) Let E be a finite set of positive and negative labeled data examples with source and target schemas $\mathbf{S}$ and $\mathbf{T}$. Let F be the finite set consisting of all atomic formulas over $\mathbf{S}$, modulo variable renaming. For every $R(\mathbf{x}) \in F$, let $(J^*_{R(\mathbf{x})}, \mathbf{a}^*_{R(\mathbf{x})})$ be the direct product $\Pi\{(J, \mathbf{a}) \mid (I, J) \in E$ is a positive data example and $I \models R(\mathbf{a})\}$. Let $q_{R(\mathbf{x})}$ be the canonical query of $(J^*_{R(\mathbf{x})}, \mathbf{a}^*_{R(\mathbf{x})})$. Take M^* to be the schema mapping consisting of all LAV constraints of the form $\forall\mathbf{x}(R(\mathbf{x}) \rightarrow q_{R(\mathbf{x})}(\mathbf{x}))$, for $R(\mathbf{x}) \in F$.

Note that the size of M^* is single exponential in $||E||$: since the schemas $\mathbf{S}$ and $\mathbf{T}$ are fixed, the number of formulas over $\mathbf{S}$, modulo variable renaming, is bounded. Moreover, the cardinality of S is bounded linearly by the number of facts in the data examples belonging to E. It follows that M^* consists of polynomially many LAV constraints, each of at most singly exponential size.

Furthermore, it follows from the construction of M^* that for every positively labeled data example $(I, J) \in E$, J is a solution for I with respect to M^*. Indeed, consider any LAV constraint $\forall\mathbf{x}(R(\mathbf{x}) \rightarrow q_{R(\mathbf{x})}(\mathbf{x}))$ of M^*, and suppose $R(\mathbf{a}) \in I$. By construction, $(J^*_{R(\mathbf{x})}, \mathbf{a}^*_{R(\mathbf{x})}) \rightarrow (J, \mathbf{a})$, which means that $q_{R(\mathbf{x})}(\mathbf{a})$ is satisfied in J.

It remains to show that every negatively labeled data example $(I, J) \in E$ falsifies at least one LAV constraint from M^*. For the sake of a contradiction, assume that this is not the case. Let M be the LAV schema mapping that fits E, which we have assumed exists. We know that (I, J) falsifies at least one LAV constraint from M. Let this LAV constraint be of the form $\forall\mathbf{x}(R(\mathbf{x}) \rightarrow \exists\mathbf{y}\psi(\mathbf{x}, \mathbf{y}))$, and let $R(\mathbf{a}) \in I$ be witness to the falsehood of this constraint in (I, J). Since (I, J) satisfies all constraints of M^*, we know that $q_{R(\mathbf{x})}$, as we defined earlier, is satisfied in $(J, \mathbf{a})$. In other words, $(J^*_{R(\mathbf{x})}, \mathbf{a}^*_{R(\mathbf{x})}) \rightarrow (J, \mathbf{a})$. It follows that $\exists\mathbf{y}\psi(\mathbf{x}, \mathbf{y})$ is *not* satisfied in $(J^*_{R(\mathbf{x})}, \mathbf{a}^*_{R(\mathbf{x})})$, in other words, the canonical instance of $\exists\mathbf{y}\psi(\mathbf{x}, \mathbf{y})$ does *not* homomorphically map to $(J^*_{R(\mathbf{x})}, \mathbf{a}^*_{R(\mathbf{x})})$. At the same time, we know that $\exists\mathbf{y}\psi(\mathbf{x}, \mathbf{y})$ maps homomorphically into each factor instance of which $(J^*_{R(\mathbf{x})}, \mathbf{a}^*_{R(\mathbf{x})})$ is the direct product. This contradicts the basic property of direct products we described in Section 2, namely that every structure that maps homomorphically into a collection of structures, maps homomorphically into their direct product. Therefore, we have reached a contradiction.

(ii) The coNExpTime upper bound follows from the proof of item (i): we construct M^* from the given data examples, as described above. As we noted, it follows from the construction that M^* fits all positively labeled data examples in E. Therefore, we only need to verify that M^* fits the negatively labeled data examples in E. We use here the fact that the problem of checking that a LAV constraint is satisfied in a data example belongs to NP (which follows from the fact that LAV constraints are existential FO sentences).

Lower bound: by reduction from Theorem 1(3): let $\mathbf{A}_1, \ldots, \mathbf{A}_n$ and $\mathbf{B}$ be given, with a single binary relation. Let $\mathbf{T}$ be the schema of these structures, and let $\mathbf{S}$ be the schema consisting of a single unary relation P. By the way, in this case, every LAV schema mapping is equivalent to one that consists of a single LAV constraint (this is because there is only one possible left-hand side for the LAV constraints, due to the particular choice of $\mathbf{S}$, and multiple LAV constraints with the same left-hand side can be combined using conjunction of the right-hand sides). Note that this already shows that in essence, here, we are concerned with finding a fitting conjunctive query again (namely the right-hand side of the unique LAV constraint). For each $i \leq n$, consider the data example $(\{P(0)\}, \mathbf{A}_i)$ where 0 is some fresh value. The PHP then reduces to the complement of the question whether there is a LAV schema mapping that fits the positively labeled examples $(\{P(0)\}, \mathbf{A}_i)$ for $i \leq n$ and the negatively labeled example $(\{P(0)\}, \mathbf{B})$. ◀

In contrast, the situation for GAV is quite different:

▶ **Theorem 5.** 1. *Let E be a finite collection of positive and negative data examples over a fixed source and target schema. If there exists a GAV schema mapping that fits E, then there exists one whose size is polynomial in the total size of the data examples in E.*
2. *The fitting problem for GAV schema mappings with positive and negative examples is co*NP*-complete.*

Proof. For the complexity upper bound, we use the following decision procedure: let a finite set of labeled data examples be given. For each negatively labeled example (I, J), we verify that there is a target relation R and a tuple t of appropriate arity over the domain of I such that (i) $R(t)$ is absent in J; and (ii) for every positively labeled example (I', J') (from the given set of examples) and homomorphism $h : I \to I'$, $R(h(t))$ belongs to J'. If this holds, we answer *Yes*, otherwise *No*. Note that item (ii) involves a coNP test. Since this coNP test is performed at most polynomially many times, this places the entire problem in coNP.

If the procedure answers *Yes*, then a fitting GAV schema mapping indeed exists, namely the schema mapping containing, for each negatively labeled example (I, J), the GAV constraint whose left-hand side is the canonical query of I (where each constant is replaced by a corresponding fresh universally quantified variable) and whose right-hand side is the chosen missing fact $R(t)$ (where each constant is again replaced by the corresponding universally quantified variable). It follows from item (ii) that this GAV constraint is indeed satisfied in all the positively labeled examples. The GAV schema mapping consisting of all GAV constraints constructed in this way (one for each negatively labeled data example) therefore fits E. Note that this schema mapping is of poynomial size.

Conversely, if there is a fitting GAV schema mapping, then the above procedure will answer *Yes*: in order for a GAV schema mapping to fit the examples, it must contain, for each negatively labeled example (I, J), a GAV constraint that fails in (I, J). The conclusion of this constraint (together with the homomorphism witnessing its failure in (I, J)) provides the fact $R(t)$ that is missing in J. Since the same GAV constraint holds in all positive data examples, item (ii) above holds as well.

The coNP-hardness is shown by a reduction from (the complement of) the NP-complete graph homomorphism problem. Indeed, is easy to see that, for any two graphs G, G', we have that $G \to G'$ if and only if there is no fitting GAV schema mapping for the set consisting of the positively labeled data example $(G', \emptyset)$ and the negatively labeled data example $(G, \emptyset)$. ◀

1GAV schema mappings have not been previously considered in the context of schema mapping discovery. Therefore, we study the fitting problem both in the case with positive and negative examples and with universal examples.

▶ **Theorem 6.** *The following problems are co*NExpTime*-complete:*
1. *the fitting problem for 1GAV schema mappings with positive and negative examples*
2. *the fitting problem for 1GAV schema mappings with universal examples*

In both settings it holds that, if there is a fitting schema mapping for a given set of data examples, then there is one of whose size is exponential in the total size of the data examples.

Proof. Before we start, we note that, if M is a GAV schema mapping (in particular, a 1GAV schema mapping), then every source instance I has a universal solution J, such that the active domain of J is included in the active domain of I (indeed, the *chase* procedure described in [1] produces such universal solutions for GAV schema mappings). Moreover, if J is a universal solution for I and the active domain of J is included in the active domain of I, the definition of universality implies that J is a subinstance of every solution of I. In addition, for every pair of instances I, J, we have that J is a universal solution for I with

respect to a GAV schema mapping M if and only if J' is a universal solution for I with respect to M, where J' is the subinstance of J induced by the active domain of I. We will make use of these observations below.

First we show how coNExpTime-hardness of the fitting problem for 1GAV schema mappings with universal examples, by reduction from the CQ-definability problem (cf. Theorem 2) Let (I, S) be a given input for the CQ-definability problem (for a fixed schema $\mathbf{S}$), and let n be the arity of the relation S. Let $\mathbf{T}$ be the schema that the single n-ary relation symbol T, and let J be the $\mathbf{T}$-instance given by the relation S. First, observe that, since the target schema $\mathbf{T}$ consists of a single relation, every 1GAV schema mapping, in this case, consists of at most one GAV constraint, which is of the form $\forall \mathbf{x}\mathbf{y}(\phi(\mathbf{x}, \mathbf{y} \rightarrow T\mathbf{x})$ (where there may be a repetition of variables in the T-atom). We can associate to this GAV constraint a conjunctive query $q(\mathbf{x}) = \exists \mathbf{y}\phi(\mathbf{x}, \mathbf{y})$ (and, conversely, each conjunctive query is associated to a GAV schema mapping in this way). It is easy to see that a 1GAV schema mapping M fits the universal example (I, J), if and only if the corresponding conjunctive query $q(\mathbf{x})$ is such that $q(I) = S$.

Next, we reduce the fitting problem for 1GAV schema mappings with universal examples to the fitting problem for 1GAV schema mappings with positive and negative examples. Let E be a collection of universal examples over source and target schemas $\mathbf{S}, \mathbf{T}$. By our earlier observations, we may assume without loss of generality that the data examples $(I, J) \in E$ are such that the active domain of J is included in the active domain of I. We define a set E' of positive and negative examples. For each $(I, J) \in E$, we include in E' as positive examples the pair (I, J) itself; and we include in E' as negative examples all pairs (I, J') where J' is a $\mathbf{T}$-instance over the active domain of I such that $J \not\subseteq J'$. It is easy to show that a 1GAV schema mapping M fits the positive and negative data examples in E' if and only if M fits the universal examples in E. Moreover, the combined side of the data examples in E' is polynomial in the combined size of the data examples in E, given that the schemas $\mathbf{S}, \mathbf{T}$ are fixed.

Finally, we will show that the fitting problem for 1GAV schema mappings with positive and negative examples is in coNExpTime, and we will establish the existence of single exponential size fitting 1GAV schema mappings. Let E be a finite set of data examples over schemas $\mathbf{S}$ and $\mathbf{T}$. We will restrict attention to the case where $\mathbf{T} = \{T\}$ (the generalization to the case with several target relations is straightforward). Let n be the arity of T. For each negatively labeled data example $(I, J) \in E$, by a *missing fact* of (I, J) we will mean a fact $T(\mathbf{a})$, with values $\mathbf{a}$ from the active domain of I, that does *not* belong to J. If a negatively labeled example $(I, J) \in E$ has no missing fact, then it is easy to see that no 1GAV schema mapping (and in fact, no GLAV schema mapping) fits E. Therefore, we may assume that each negatively labeled data example has at least one missing fact. Let F be the set of all functions that map each negatively labeled data example in E to one of its missing facts. Note that F consists of singly exponentially many maps. We can associate to each map $f \in F$ a 1GAV schema mapping M_f, namely the schema mapping consisting of the 1GAV constraint $\forall \mathbf{x}(q_f(\mathbf{x}) \rightarrow T\mathbf{x}))$, where q_f is the canonical query of the direct product $\Pi\{(I, \mathbf{a}) \mid (I, J) \in E$ is a negatively labeled data example and $f(I, J) = T\mathbf{a}\}$. It follows from the construction that M_f fits every negatively labeled data example in E.

Claim: If there is a 1GAV schema mapping that fits E, then, for some $f \in F$, M_f fits E.

Note that the size of M_f is single exponential. The claim also implies the coNExpTime complexity upper bound we are after: to check that there is *no* fitting 1GAV schema mapping for E, it suffices to guess, for each $f \in F$, a positively labeled data example (I, J) and a variable assignment witnessing the fact that $(I, J) \not\models \forall \mathbf{x}(q_f(\mathbf{x}) \rightarrow T\mathbf{x}))$. Note that the exponentially many exponential-size non-deterministic guesses can be collected into a single exponential size non-deterministic guess.

To prove the claim, let M' be a 1GAV schema mapping that fits E, and let its constraint be of the form $\forall\vec{x}(\phi(\mathbf{x}) \to T(\mathbf{x}))$. Let f be the map that sends each negatively labeled data example $(I, J) \in E$ to a missing target fact that witnesses the violation of the constraint in (I, J). Recall that q_f is the canonical query of $(I^*, \mathbf{a}^*) = \Pi\{(I, \mathbf{a}) \mid (I, J) \in E, I \models \phi(\mathbf{a}), J \not\models T(\mathbf{a}), f(I, J) = T(\mathbf{a})\}$. It follows that q_f is falsified in all negatively labeled data examples $(I, J) \in E$, and hence, M' fits all negatively labeled data examples in E (we had already noted that M' fits all positively labeled data examples). ◀

The coNExpTime lower bound for the LAV case applies to the GLAV case as well (the examples involved have a source instance that consists of a single fact, and it is easy to see that, for such data examples, every GLAV constraint is equivalent to a LAV constraint of at most the same size). The upper bound technique used in the LAV case can also be adapted for GLAV schema mappings, but it no longer yields matching complexity and size bounds.

▶ **Theorem 7.** 1. *Let E be a finite collection of data examples over a fixed source and target schema. If there exists a GLAV schema mapping that fits E, then there exists one of size at most $2^{2^{O(n)}}$, where n is the total size of the data examples in E.*
2. *The fitting problem for GLAV schema mappings with positive and negative examples is in co*N2ExpTime *and co*NExpTime*-hard.*

Proof. (sketch) Clearly, a fitting GLAV schema mapping needs to contain at most one constraint per negative example $(I, J) \in E$. The left-hand side of that GLAV constraint can, without loss of generality, be taken to be the canonical conjunctive query of I. The right-hand side must be a CQ that fails in J under the natural variable assignment. Here, we can apply the same technique as in the proof of Theorem 4: we can take the right-hand side of the constraint to be the canonical query of the direct product of all J' with $(I', J') \in E$ is a positive data example and $I \to I'$. The same arguments used in the proof of Theorem 4 show that, if there exists any GLAV schema mapping that fits E, then the GLAV schema mapping constructed here fits E.

Since the number of homomorphisms $I \to I'$ is in general exponential, the above construction, in general, involves taking the direct product of exponentially many instances. This gives us a double exponential size GLAV constraint. By the same arguments used in the proof of Theorem 4, we can derive a coN2ExpTime complexity upperbound for the fitting problem. ◀

6 Conclusion

We provided a detailed classification of the complexity of PHP under various restrictions. We used these results to obtain tight complexity bounds for instance-level query definability problems and for fitting problems for schema mappings. The precise complexity of the fitting problem for GLAV schema mappings with respect to positive and negative data examples is left as an open problem.

Acknowledgements. We are grateful to Ross Willard for discussions on the topic and for comments on an earlier draft. Ten Cate is supported by NSF grant IIS-1217869. Dalmau is supported by MICCIN grant TIN2013-48031-C4-1.

References

1 Bogdan Alexe, Balder ten Cate, Phokion G. Kolaitis, and Wang-Chiew Tan. Designing and refining schema mappings via data examples. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, SIGMOD '11, pages 133–144, New York, NY, USA, 2011. ACM.

2 Timos Antonopoulos, Frank Neven, and Frédéric Servais. Definability problems for graph query languages. In *Proceedings of the 16th International Conference on Database Theory*, ICDT '13, pages 141–152, New York, NY, USA, 2013. ACM.

3 F. Banchilon. On the completeness of query languages for relational databases. In *Proceedings of MFCS*, pages 112–123, 1978.

4 Egon Börger, Erich Grädel, and Yuri Gurevich. *The Classical Decision Problem*. Perspectives in Mathematical Logic. Springer, 1997.

5 Nadia Creignou, Phokion G. Kolaitis, and Bruno Zanuttini. Structure identification of boolean relations and plain bases for co-clones. *J. Comput. Syst. Sci.*, 74(7):1103–1115, 2008.

6 Victor Dalmau. *Computational Complexity of Problems over Generalized Formulas*. PhD thesis, Universitat Politècnica de Catalunya, 2000.

7 Rina Dechter and Judea Pearl. Structure identification in relational data. *Artif. Intell.*, 58(1-3):237–270, 1992.

8 Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. Data exchange: semantics and query answering. *Theoretical Computer Science*, 336(1):89 – 124, 2005. Database Theory.

9 G.H.L. Fletcher, M. Gyssens, J. Paredaens, and D. Van Gucht. On the expressive power of the relational algebra on finite sets of relation pairs. *Knowledge and Data Engineering, IEEE Transactions on*, 21(6):939 –942, june 2009.

10 Peter Jeavons, David A. Cohen, and Marc Gyssens. How to determine the expressive power of constraints. *Constraints*, 4(2):113–131, 1999.

11 Phokion G. Kolaitis. Schema mappings, data exchange, and metadata management. In *Proceedings of the Twenty-fourth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '05, pages 61–75, New York, NY, USA, 2005. ACM.

12 J. Paredaens. On the expressive power of the relational algebra. *Information Processing Letters*, 7(2):107 – 111, 1978.

13 Ross Willard. Testing expressibility is hard. In David Cohen, editor, *CP*, volume 6308 of *Lecture Notes in Computer Science*, pages 9–23. Springer, 2010.