The Homomorphism Lattice, Unique Characterizations, and Concept Learning *

Balder ten Cate

Google, balder@google.com

Abstract. Various problems arising in the context of *example-driven approaches to query discovery* have turned out to be intimately related to basic structural properties of *the homomorphism lattice of finite structures*, such as density, or the existence of duals. In this keynote, I will review some such connections, and highlight some relevant recent results.

1 The Homomorphism Lattice

Let us consider the partial order $(\mathrm{Str}, \leq_{hom})$, where Str is the set of all finite structures over some schema $\mathcal{S}$, and $\leq_{hom}$ denotes the existence of a homomorphism (that is, $A \leq_{hom} B$ holds if there is a homomorphism from A to B, i.e., a function from the domain of A to the domain of B that preserves structure). For the purpose of this exposition, we will assume that $\mathcal{S}$ is a fixed, finite schema, consisting of relation symbols and constant symbols. We will write $A <_{hom} B$ if $A \leq_{hom} B$ and $B \not\leq_{hom} A$, and we will say that A and B are *homomorphically equivalent* if $A \leq_{hom} B$ and $B \leq_{hom} A$.

This partial order $(\mathrm{Str}, \leq_{hom})$ turns out to have a rich structure and has been the subject of extensive study. For a wonderful textbook on this topic (focusing on the case of directed graphs, where the schema $\mathcal{S}$ consists of a single binary relation and no constant symbols), see [7]. We review here a few relevant results. To start with, we have the following well known fact:

Proposition 1. *The partial order* $(\mathrm{Str}, \leq_{hom})$ *is a* lattice *in the following sense:*

1. *For every finite set of structures* $\mathcal{A}$*, we can construct (in exponential time) a structure* $\bigotimes \mathcal{A}$*, such that,* $\bigotimes \mathcal{A} \leq_{hom} A$ *for every* $A \in \mathcal{A}$*, and such that for every structure* B*, if* $B \leq_{hom} A$ *for every* $A \in \mathcal{A}$ *then* $B \leq_{hom} \bigotimes \mathcal{A}$*.*
2. *For every finite set of structures* $\mathcal{A}$*, we can construct (in polynomial time) a structure* $\bigoplus \mathcal{A}$*, such that* $A \leq_{hom} \bigoplus \mathcal{A}$ *for every* $A \in \mathcal{A}$*, and such that, for every structure* B*, if* $A \leq_{hom} B$ *for every* $A \in \mathcal{A}$ *then* $\bigoplus \mathcal{A} \leq_{hom} B$*.*

The "meet" operation $\bigotimes \mathcal{A}$, here is simply the direct product, and the "join" operation $\bigoplus \mathcal{A}$, in the case without constant symbols, is simply the disjoint union (the definition of $\bigoplus \mathcal{A}$ for structures with constant symbols is a little

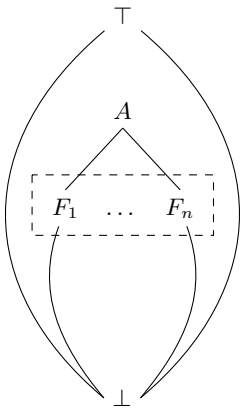


Fig. 1. A frontier in the homomorphism lattice.

more involved, we omit the details here, cf. [11]). We will refer to this lattice as *the Homomorphism Lattice*.

One computational problem in this context that will be relevant below is the *product homomorphism problem*: given a finite set of structures $\mathcal{A}$ and a structure B, the task is to decide whether $\bigotimes \mathcal{A} \leq_{hom} B$ (or, equivalently, whether $C \leq_{hom} A$ for all $A \in \mathcal{A}$ implies $C \leq_{hom} B$). Since the structure $\bigotimes \mathcal{A}$ itself can be constructed in singly exponential time, the product homomorphism problem can be solved in NExpTime. A matching lower bound was established in [14, 10].

Theorem 1 ([14, 10]). *The product homomorphism problem is NExpTime-complete (provided that $\mathcal{S}$ contains a relation of arity at least 2).*

One interesting direction in the study of the homomorphism lattice pertains to *density*. It is known that the homomorphism lattice is not a dense lattice. That is, there exist structures $B <_{hom} A$ for which there is no structure C with $B <_{hom} C <_{hom} A$. Such pairs (B, A) are also known as *gap pairs* [7]. On the other hand, there also exist structures A such that for every $B <_{hom} A$, there is structure C with $B <_{hom} C <_{hom} A$ (an example of the latter kind, in the case without constant symbols, and with a binary relation R, is the single-element structure consisting of the fact $R(a, a)$). Intuitively, we can therefore say that parts of the homomorphism lattice are locally dense, whereas other parts are not. Closely related to this is the concept of a *frontier*.

Definition 1. *A* frontier *for a structure A is a finite collection of structures $\mathcal{F}$ such that*

1. *$F <_{hom} A$ for all $F \in \mathcal{F}$*
2. *For all structures C, if $C <_{hom} A$, then $C \leq_{hom} F$ for some $F \in \mathcal{F}$.*

In other words, a frontier for A is a finite set of structures that separates A from those structures that are homomorphically strictly smaller than A, as depicted in Figure 1.

As it turns out, there is simple necessary and sufficient condition for the existence of a frontier. To define this condition, we must consider the *incidence graph* of a structure A, by which we mean the bipartite multi-graph whose vertices are the elements of the domain of A and the facts of A, and such that an element and a fact are connected by an edge if the element occurs in the fact. If an element occurs multiple times in the same fact, each occurrence generates an edge (hence, this is a multi-graph).

Definition 2. *A structure is* acyclic *if the incidence graph is acyclic, and a structure is* c-acyclic *if every cycle of the incidence graph passes through at least one element that is named by a constant symbol.*

Note that if the schema $\mathcal{S}$ does not contain constant symbols, c-acyclicity is the same as acyclicity.

Theorem 2 ([11]). *Every c-acyclic structure has a frontier. Moreover, given a c-acyclic structure, a frontier can be constructed in polynomial time. Conversely, if a structure A has a frontier, then A is homomorphically equivalent to a c-acyclic structure.*

It also follows from this result that we can test whether a given set of structures is a frontier. In fact, this problem is NP-complete.

Theorem 3. *The following problem is NP-complete: given a structure A and a finite set of structures $\mathcal{F}$, is $\mathcal{F}$ a frontier for A?*

Proof (sketch). For the upper bound, we use the fact that, if A is homorphically equivalent to a c-acyclic structure A', then such A' exists as a substructure of A (we omit the details, but this follows directly from the fact that c-acyclicity is preserved under passage from a structure to its core). Therefore, the problem can be solved in non-deterministic polynomial time as follows: first we guess a substructure A' and we verify that A' is c-acyclic and homomorphically equivalent to A. Note that the existence of such A' is a necessary precondition for $B_1, \ldots, B_n$ to be a frontier of A. Next, we apply Theorem 2 to construct a frontier $\mathcal{F}'$ for A' (and hence for A). Finally, we verify that each $F \in \mathcal{F}$ homomorphically maps to some $F' \in \mathcal{F}'$ and vice versa. It is not hard to see that this non-deterministic algorithm has an accepting run if and only if $\mathcal{F}$ is a frontier for A.

For the lower bound, we reduce from graph 3-colorability. Let A be the structure, over a 3-element domain, that consists of the facts $R(a, b)$ for all pairs a, b with $a \neq b$. In addition, each of the three elements is named by a constant symbol. Since A is c-acyclic, by Theorem 2, it has a frontier $\mathcal{F}$. Now, given any graph G (viewed as a relational structure with binary relation R and without constant symbols), we have that G is 3-colorable if and only if $\mathcal{F}$ is a frontier for the disjoint union of A with G. To see that this is the case, note that if G is 3-colorable, then the disjoint union of A with G is homomorphically equivalent

to A itself, whereas if G is not 3-colorable, then the disjoint union of A with G is strictly greater than A in the homomorphism order. □

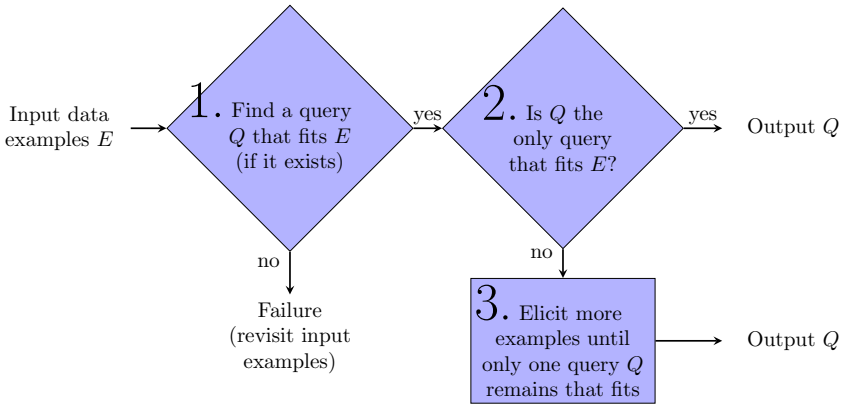
Another, closely related topic in the study of the homomorphism lattice is that of *dualities*. A pair of structures (A, B) are said to be a *duality pair* if $\{C \mid A \not\leq_{hom} C\} = \{C \mid C \leq_{hom} B\}$. Intuitively, this means that the entire class of structures can be partitioned into two disjoint sets: those structures into which A homomorphically maps, and those structures that homomorphically map into B. More generally a pair of finite sets of structures $(\mathcal{A}, \mathcal{D})$ is said to be a generalized homomorphism duality if $\{C \mid A \not\leq_{hom} C$ for all $A \in \mathcal{A}\} = \{C \mid C \leq_{hom} B$ for some $B \in \mathcal{B}\}$. A famous infinitary example of a homomorphism duality is the following: a graph G is 2-colorable (equivalently, has a homomorphism into the 2-element clique) if and only if G does not contain a cycle of odd length (equivalently, does not not have a homomorphism from a structure consisting of a cycle of odd length).

As was shown in [8], generalized homomorphism dualities and frontiers are intimately related, and one can be constructed from the other. In particular, a finite set of structures $\mathcal{A}$ is the left-hand side of a generalized homomorphism duality if and only if (modulo homomorphic equivalence) $\mathcal{A}$ consists of c-acyclic structures [6, 1].

2 Example-Driven Query Discovery

Suppose one wishes to construct a query Q on the basis of a set of *data examples*. For the sake of concreteness, assume that the query Q we wish to construct is a k-ary conjunctive query ($k \geq 0$) over a schema $\mathcal{S}$. For the purpose of our complexity analysis, we will treat the schema and the arity as fixed. Each given data example will be a triple $(I, \mathbf{a}, s)$, where I is a database instance (over the schema $\mathcal{S}$), $\mathbf{a}$ is a k-tuple of values from the active domain of I, and $s \in \{+, -\}$ indicates whether the data example is a positive example or a negative example. We say that a query Q *fits* such a data example $(I, \mathbf{a}, s)$ if $\mathbf{a} \in Q(I)$ (for the case where $s = +$), or $\mathbf{a} \notin Q(I)$ (for the case where $s = -$).

One natural high-level approach for constructing Q might be as in the following flow chart:



Following this approach, three computational tasks naturally arise. The first task is, given an input set of data examples, to decide whether there exists a fitting query, and, if so, produce one. This has also been described a *reverse-engineering* problem [3]: from observed behavior of the query, we must reconstruct a query with that behavior. It turns out that, in the case of conjunctive queries, the existence of a query that fits a given set of data examples, is computationally equivalent to the *product homomorphism problem* that we encountered in the previous section. Indeed (skipping over some minor details related to safety and active domain semantics), we can view a data example for a k-ary query as a structure with k constant symbols, and we can identify a k-ary conjunctive query itself also with a structure with k constant symbols. By the Chandra-Merlin theorem [5], then, a query Q fits a positive (negative) example E precisely if $Q \leq_{hom} E$ (respectively, $Q \not\leq_{hom} E$) where both are viewed as structures.

Now, let E be a set of data examples, and let Q be (the canonical query of) the direct product of all positive examples in E. It is not hard to show that if there exists any query that fits the data examples, then Q must fit the data examples. Moreover, it already follows from the construction that Q fits all positive examples in E, and, in order for Q to fit the negative examples, it must be the case that there is no homomorphism from Q to any of the negative data examples in E. This shows that the existence of a conjunctive query fitting a given set of data examples, reduces to (a conjunction of instances of) the product homomorphism problem. It is not hard to establish a reduction in the other direction as well. Moreover, in the case where a fitting conjunctive query exists, the conjunctive query Q constructed above is guaranteed to be a *greatest fitting query* for the examples, that is, $Q' \leq_{hom} Q$ holds for all conjunctive queries Q' that fit the data examples in E. Therefore, in summary, we have:

Theorem 4 ([10]). *The following problem is CoNexpTime-complete: given a finite set E of data examples, does there exist a conjunctive query that fits E. Moreover, if a fitting conjunctive query exists, then a greatest fitting conjunctive*

query can be constructed in exponential time (where the exponent corresponds to the number of positive examples). [1]

At this point, assuming that a fitting conjunctive query does indeed exists, we have obtained a candidate query Q, which is in fact a greatest fitting conjunctive query. We then arrive at the next question, which is whether Q is the *only* query (up to logical equivalence) that fits E. This second problem turns out, in our specific setting, to be NP-complete. Formally, we say that the data examples in E *uniquely characterize* Q if Q fits E and, furthermore, every conjunctive query that fits E is logically equivalent to Q.

Theorem 5. *The following problem is NP-complete: given a collection E of positive and negative examples, and a greatest fitting conjunctive query Q for E, do the data examples in E uniquely characterize Q?*

Proof (sketch). Theorem 5 is proved by showing that the problem is computationally equivalent to the problem of testing whether a given set of structures is a frontier. In one direction, we claim that the data examples in E uniquely characterize Q if and only if the set $\mathcal{F} = \{(N \otimes Q) \mid N \text{ is a negative example from } E\}$ is a frontier for Q. Note that, here, as before, we take the liberty to skip over minor details by conflating conjunctive queries, data examples, and structures.

First, suppose that Q is uniquely characterized by E. We must show that $\mathcal{F}$ is a frontier for Q. Clearly, $(N \otimes Q) \leq_{hom} Q$. Furthermore, since Q fits the negative example N, we have that $Q \not\leq_{hom} N$ and therefore $Q \not\leq_{hom} (N \otimes Q)$. Finally, let $B <_{hom} Q$, and suppose for the sake of a contradiction that B does not homomorphically map to any structure in $\mathcal{F}$. It follows that B does not map to any negative example N. Therefore, (the canonical query of) B fits all examples in E and is not equivalent to Q, a contradiction. Next, suppose that $\mathcal{F}$ is a frontier for Q. We must show that Q is uniquely characterized by E. Suppose, for the sake of a contradiction, that Q' fits all examples in E and is not equivalent to Q. Since Q is a greatest fitting query, it must be the case that $Q' \leq_{hom} Q$ and $Q \not\leq_{hom} Q'$. Therefore, $Q \leq_{hom} (N \otimes Q)$ for some negative example N, and hence $Q \leq_{hom} N$, a contradiction.

To prove the NP lower bound, we provide an even simpler reduction in the opposite direction. Let A be any structure and $\mathcal{F}$ a finite set of structures such that each structure in $\mathcal{F}$ is homomorphically below A. Then $\mathcal{F}$ is a frontier for A if and only if A is uniquely characterized by the set E that consists of A itself as a positive example, and the structures in $\mathcal{F}$ as negative examples. □

At this point, we know whether the examples in E uniquely characterize Q. If the answer is *Yes*, then we can safely conclude that Q is the query we were after. Otherwise, we arrive at the last of our three problems, namely, eliciting further examples with the aim of identifying our target query with certainty.

[1] The result as stated in [10] is for LAV schema mappings. However, the fitting problem for LAV schema mappings is easily seen to be equivalent to the fitting problem for conjunctive queries.

Here, we enter into the territory of computational learning theory. Without going into details, the first question that arises is whether, by eliciting further examples, we can even hope to arrive at a situation in which we can identify our target conjunctive query with certainty. This question is, of course, equivalent to the question whether the target conjunctive query is uniquely characterizable by finitely many examples. It turns out that the answer is *Yes* precisely if the target query has a finite frontier [11]. By Theorem 2, this holds precisely if the target query is (homomorphically equivalent to) a c-acyclic conjunctive query. If the target query is *not* homomorphically equivalent to a c-acyclic query, then, no matter how many additional data examples we elicit, we will never arrive at a situation in which our set of data examples uniquely characterizes the target query. Remarkably, it turns out that if the target query *is* c-acyclic, then there exists an efficient exact learning algorithm (in the formal sense of Angluin's model of interactive exact learnability [2]), that will identify the target query after asking polynomially many questions, where each question asks whether a given data example is positive or negative for the target query. Following standard terminology from computational learning theory, such questions are called "membership queries", and the formal statement of the result is as follows:

Theorem 6 ([11]). *The class of c-acyclic conjunctive queries is efficiently exactly learnable with membership queries.*

3 Concluding Remarks

This short paper was written with the aim of highlighting some interesting topics and their connections to the structure of the homomorphism lattice. It is by no means a comprehensive survey of work that has been done in this area. In fact, there is considerable literature on example-driven discovery of database queries, description logic ontologies, and schema mappings, as discussed in the related work sections of the papers cited here. See also [9] for a recent survey of different approaches towards learning description logic ontologies.

Since we focused on the case of conjunctive queries in this exposition, one might ask what changes when considering *unions of conjunctive queries*. As it turns out, in this context there is a close correspondence between unions of conjunctive queries and *GAV schema mappings*. Without going into details, this is because a GAV schema mapping can be equivalently viewed as a mapping that assigns to each target relation a union of conjunctive queries over the source schema. Unique characterizations for GAV schema mappings were extensively studied in [1], and some of the results in [1] can be rephrased in terms of unions of conjunctive queries. In particular, in the same way that unique characterizations for conjunctive queries correspond to frontiers in the homomorphism lattice, it follows from results in [1] that unique characterizations for unions of conjunctive queries correspond to generalized homomorphism dualities. Exact learnability (as well as PAC learnability) for GAV schema mappings was studied in [12, 13], and again, the results can be rephrased in terms of unions of conjunctive queries.

We close by briefly mentioning two other interesting problems with close connections with the structure of the homomorphism lattice. The first is *instance-level query definability*, which refers to definability of relations inside a given database instance. More specifically, the *CQ-definability problem*, consists in deciding, given a database instance I and a relation S over the active domain of I, whether there is a conjunctive query Q such that $Q(I) = S$. It turns out that this problem is again computationally equivalent to the product homomorphism problem (cf. [10]). Finally, let us mention one more definability problem of sorts, that arises in the context of ontology-based data access. This is the problem where we are given a ontology-based query (specified with the help of an ontology, and using certain-answer semantics), and we wish to decide whether it is equivalent to an ordinary first-order query (without reference to the ontology). As it turns out, for common ontology languages such as $\mathcal{ALC}$, this decision problem reduces to the question whether a given finite set of structures has a generalized homomorphism duality. See [4] for more details.

References

1. Bogdan Alexe, Balder ten Cate, Phokion G. Kolaitis, and Wang-Chiew Tan. Characterizing schema mappings via data examples. *ACM Trans. Database Syst.*, 36(4):23:1–23:48, December 2011.
2. Dana Angluin. Queries and concept learning. *Mach. Learn.*, 2(4):319–342, April 1988.
3. Pablo Barceló. A theoretical view on reverse engineering problems for database query languages. In Mantas Simkus and Grant E. Weddell, editors, *Proceedings of the 32nd International Workshop on Description Logics, Oslo, Norway, June 18-21, 2019*, volume 2373 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2019.
4. Meghyn Bienvenu, Balder Ten Cate, Carsten Lutz, and Frank Wolter. Ontology-based data access: A study through disjunctive datalog, csp, and mmsnp. *ACM Trans. Database Syst.*, 39(4), December 2015.
5. Ashok K. Chandra Chandra and Philip M. Merlin. Optimal Implementation of Conjunctive Queries in Relational Data Bases. In *ACM Symposium on Theory of Computing (STOC)*, pages 77–90, 1977.
6. Jan Foniok, Jaroslav Nesetril, and Claude Tardif. Generalised dualities and maximal finite antichains in the homomorphism order of relational structures. *Eur. J. Comb.*, 29(4):881–899, 2008.
7. Pavol Hell and Jaroslav Nešetřil. *Graphs and homomorphisms*, volume 28 of *Oxford lecture series in mathematics and its applications*. Oxford University Press, 2004.
8. Jaroslav Nešetřil and Claude Tardif. Duality theorems for finite structures (characterising gaps and good characterisations). *Journal of Combinatorial Theory, Series B*, 80(1):80 – 97, 2000.
9. Ana Ozaki. Learning description logic ontologies: Five approaches. where do they stand? *KI - Künstliche Intelligenz*, 04 2020.
10. Balder ten Cate and Víctor Dalmau. The product homomorphism problem and applications. In Marcelo Arenas and Martín Ugarte, editors, *18th International Conference on Database Theory, ICDT 2015, March 23-27, 2015, Brussels, Belgium*, volume 31 of *LIPIcs*, pages 161–176. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015.

11. Balder ten Cate and Victor Dalmau. Conjunctive queries: Unique characterizations and exact learnability, 2020.
12. Balder ten Cate, Víctor Dalmau, and Phokion G. Kolaitis. Learning schema mappings. *ACM Trans. Database Syst.*, 38(4):28:1–28:31, 2013.
13. Balder ten Cate, Phokion G. Kolaitis, Kun Qian, and Wang-Chiew Tan. Active learning of GAV schema mappings. In Jan Van den Bussche and Marcelo Arenas, editors, *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Houston, TX, USA, June 10-15, 2018*, pages 355–368. ACM, 2018.
14. Ross Willard. Testing expressibility is hard. In David Cohen, editor, *Principles and Practice of Constraint Programming - CP 2010 - 16th International Conference, CP 2010, St. Andrews, Scotland, UK, September 6-10, 2010. Proceedings*, volume 6308 of *Lecture Notes in Computer Science*, pages 9–23. Springer, 2010.