

# Using Trémaux Trees to Compute Small Conjunctive Queries that Separate Positive and Negative Examples

Francesco Kriegel<sup>1,2</sup>

<sup>1</sup>*Institute of Theoretical Computer Science, Technische Universität Dresden, Dresden, Germany*

<sup>2</sup>*Center for Scalable Data Analytics and Artificial Intelligence (ScaDS.AI), Dresden and Leipzig, Germany*

## Abstract

We investigate the problem of computing small conjunctive queries that separate positive from negative examples in ontology-enriched systems. This work builds upon prior research on the query-by-example paradigm, which focused on the existence of separating queries. Here, we go beyond mere existence and study how to construct separating queries that are both correct and compact. Specifically, we define a new recursive notion of homomorphism based on Trémaux trees (normal spanning trees), show that it allows us to extract small separating conjunctive queries from the chase (universal model), and provide an algorithm for this query construction process. Our results offer both theoretical insights and practical tools for making ontology-based query-by-example more usable for end-users.

## 1. Introduction

The query-by-example (QBE) paradigm has recently emerged as a promising approach for making ontology-enriched systems (OES) more accessible to non-expert users. By allowing users to provide sets of positive and negative examples instead of formal queries, QBE bridges the gap between intuitive data exploration and formal query languages like description-logic concepts [1–3], conjunctive queries (CQs) or unions of conjunctive queries (UCQs) [4–8], and first-order formulas [9]. This paper specifically follows earlier work [4], in which foundational results were established regarding the existence of queries that correctly separate the positive examples from the negative ones w.r.t. ontologies formulated in rather expressive description logics (DLs) such as Horn-*ALC* and Horn-*ALC*<sub>T</sub>. However, deciding the existence of a separating query is only the first step. In practical applications, the ultimate goal is to *compute* a concrete query that explains the given examples. More importantly, such queries should ideally be as *small* and as *understandable* as possible. A large or overly complex query may defeat the purpose of QBE as a tool for intuitive data access and explanation.

In this paper, we investigate the computational construction of small separating CQs in OES. Our focus lies not on particular DLs to formulate the ontology but rather on existential rules in general, however we restrict attention to unary and binary predicates. The core technical contribution of this work is a novel use of Trémaux trees (also known as: normal spanning trees)

---

 DL 2025: 38th International Workshop on Description Logics, September 3–6, 2025, Opole, Poland

 francesco.kriegel@tu-dresden.de (Francesco Kriegel)

 <https://tu-dresden.de/inf/lat/francesco-kriegel> (Francesco Kriegel)

 0000-0003-0219-0330 (Francesco Kriegel)

 © 2025 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

to guide the search for separating queries. By leveraging the tree structure, we define a recursive notion of homomorphisms — called Trémaux homomorphisms — and show that existence of a Trémaux homomorphism coincides with existence of a usual homomorphism. These new homomorphisms allow us to develop a new algorithmic approach to extracting small separating queries from the chase of a knowledge base — a canonical representation of its entailments. Our approach not only advances the theoretical understanding of query computation in OES but also opens up new directions for developing more intuitive and compact interfaces for QBE tools.

## 2. Preliminaries

**Signature.** Consider a countable set  $\mathbf{C}$  of *constants* and a countable set  $\mathbf{R}$  of *relations* such that each relation  $R$  in  $\mathbf{R}$  has an *arity*  $\text{ar}(R) \in \mathbb{N}$ . These two sets must be disjoint and together they constitute the *signature*. In Description Logic, constants are usually referred to as *individuals*, unary relations are called (*atomic*) *concepts*, and binary relations are *roles*. Further assume a countably infinite set  $\mathbf{V}$  of *variables* disjoint with the signature. A *term* is either a constant or a variable.<sup>1</sup>

**Syntax.** An *atom* is of the form  $(t_1, \dots, t_n) : R$  where each  $t_i$  is a term and  $R$  is an  $n$ -ary relation. A *substitution* is a partial mapping  $\sigma : \mathbf{V} \rightarrow \mathbf{C} \cup \mathbf{V}$ . If  $\alpha$  is an atom, then  $\alpha\sigma$  denotes the term obtained by simultaneously replacing all occurrences of each variable  $x$  by  $\sigma(x)$  if defined.<sup>2</sup> Given sets  $\mathcal{A}$  and  $\mathcal{B}$  of atoms, a *match* of  $\mathcal{A}$  in  $\mathcal{B}$  is a substitution  $\sigma$  such that  $\mathcal{A}\sigma \subseteq \mathcal{B}$ , where  $\mathcal{A}\sigma := \{\alpha\sigma \mid \alpha \in \mathcal{A}\}$ . We write  $\mathcal{A} \models \mathcal{B}$  if there is such a match. Then  $\models$  is a preorder (i.e. reflexive and transitive) on the set of all sets of atoms. From each match  $\sigma$  of  $\mathcal{A}$  in  $\mathcal{B}$ , we obtain a so-called *homomorphism* from  $\mathcal{A}$  to  $\mathcal{B}$  as the mapping  $h : \mathbf{C} \cup \mathbf{V} \rightarrow \mathbf{C} \cup \mathbf{V}$  where  $h(t) := \sigma(t)$  if the latter is defined and  $h(t) := t$  otherwise; there are no further homomorphisms. Given a set  $\mathcal{A}$  of atoms,  $\text{Var}(\mathcal{A})$  is the set of all variables in  $\mathcal{A}$  and  $\text{Terms}(\mathcal{A})$  consists of all terms in  $\mathcal{A}$ .

**Semantics.** An *interpretation*  $\mathcal{I}$  is a set of atoms, and a *database*  $\mathcal{D}$  is a finite set of atoms.<sup>3</sup> A *tuple-generating dependency* (TGD) or *existential rule* is of the form  $\mathcal{B} \Rightarrow \mathcal{H}$  where the *body*  $\mathcal{B}$  and the *head*  $\mathcal{H}$  are finite sets of atoms.  $\text{Var}(\mathcal{B}) \cap \text{Var}(\mathcal{H})$  is called the *frontier* of  $\mathcal{B} \Rightarrow \mathcal{H}$ . An interpretation  $\mathcal{I}$  is a *model* of a database  $\mathcal{D}$  if  $\mathcal{D} \models \mathcal{I}$ . A TGD  $\mathcal{B} \Rightarrow \mathcal{H}$  is *satisfied* in an interpretation  $\mathcal{I}$  if every match of  $\mathcal{B}$  in  $\mathcal{I}$  can be extended to a match of  $\mathcal{H}$  in  $\mathcal{I}$  (i.e. for each match  $\sigma$  of  $\mathcal{B}$  in  $\mathcal{I}$ , there is a match  $\tau$  of  $\mathcal{H}$  in  $\mathcal{I}$  such that, for each variable  $x \in \text{Var}(\mathcal{B})$ , if  $\sigma(x)$  is defined, then  $\sigma(x) = \tau(x)$ , else  $\tau(x)$  is also undefined). A *knowledge base* (KB) is a pair  $(\mathcal{D}, \mathcal{O})$  consisting of a database  $\mathcal{D}$  and a finite set  $\mathcal{O}$  of TGDs (called *ontology*). An interpretation  $\mathcal{I}$  is a *model* of  $(\mathcal{D}, \mathcal{O})$  if it is a model of  $\mathcal{D}$  and satisfies all TGDs in  $\mathcal{O}$ .

<sup>1</sup>This coincides with the elements of the *term algebra* of type  $\mathbf{C}$  over  $\mathbf{V}$  since constants have the same semantics as nullary function symbols.

<sup>2</sup>Each substitution has a unique extension to a homomorphism  $h$  from the term algebra to itself, which here sends each constant to itself, i.e. we obtain the mapping  $h : \mathbf{C} \cup \mathbf{V} \rightarrow \mathbf{C} \cup \mathbf{V}$  where  $h(x) = \sigma(x)$  for each variable  $x$  and  $h(c) = c$  for each constant  $c$ . With that,  $((t_1, \dots, t_n) : R)\sigma = (h(t_1), \dots, h(t_n)) : R$ .

<sup>3</sup>Other authors disallow variables in databases, but we find them reasonable in order to account for objects that do not have or need a unique identifier to be shared with other knowledge bases.

**Chase.** In the setting considered here, each KB  $(\mathcal{D}, \mathcal{O})$  has a model (i.e. is *consistent*). Such a model can be constructed by means of the chase: it initializes an interpretation  $\mathcal{I}$  with the database  $\mathcal{D}$  and then, simply put, it successively extends  $\mathcal{I}$  whenever there is a TGD in  $\mathcal{O}$  such that there is match of its body in  $\mathcal{I}$  that cannot be extended to a match of the head in  $\mathcal{I}$ . The limit of this construction is also called chase, viz. the *chase* of  $(\mathcal{D}, \mathcal{O})$ , symbol:  $\text{chase}(\mathcal{D}, \mathcal{O})$ . The chase does not terminate for every KB, meaning that  $\text{chase}(\mathcal{D}, \mathcal{O})$  might be countably infinite, and different variants of the chase exist [10]. Chase termination is undecidable [11] but can be guaranteed by restrictions [12].

**Conjunctive Queries.** An  $n$ -ary *conjunctive query* (CQ) is of the form  $(x_1, \dots, x_n) : \mathcal{Q}$  where each  $x_i$  is a variable, called *answer variable*, and  $\mathcal{Q}$  is a finite set of atoms. A mapping  $\sigma : \{x_1, \dots, x_n\} \rightarrow \mathbf{C}$  is an *answer* to this CQ w.r.t. an interpretation  $\mathcal{I}$  if  $\mathcal{Q}\sigma \models \mathcal{I}$  (i.e. if  $\sigma$  can be extended to a match from  $\mathcal{Q}$  to  $\mathcal{I}$ ), and is a *certain answer* to this CQ w.r.t. a KB  $(\mathcal{D}, \mathcal{O})$  if it is an answer to the CQ w.r.t. every model of  $(\mathcal{D}, \mathcal{O})$ . *Query Answering* is the problem consisting of all tuples of (string encodings of) a KB, a CQ, and a mapping such that the mapping is a certain answer to the CQ w.r.t. the KB. In general, query answering is undecidable [13, 14]. Query answering can be done by means of the chase: the certain answers w.r.t.  $(\mathcal{D}, \mathcal{O})$  coincide with the answers w.r.t.  $\text{chase}(\mathcal{D}, \mathcal{O})$ . This is because the chase is *universal* in the sense that it matches every model of the KB.

**Query Containment.** Given CQs  $(x_1, \dots, x_n) : \mathcal{Q}_1$  and  $(x_1, \dots, x_n) : \mathcal{Q}_2$  with the same answer variables, we say that the first is *contained* in the second if, for every interpretation  $\mathcal{I}$ , each answer to the first CQ in  $\mathcal{I}$  is also an answer to the second CQ in  $\mathcal{I}$ . This is the case iff. there is match  $\sigma$  of  $\mathcal{Q}_2$  in  $\mathcal{Q}_1$  that preserves the answer variables, i.e. where  $\sigma(x_i) = x_i$  for each  $i$  [15]. Query containment is NP-complete [15].

**Unions of Conjunctive Queries.** A *union of conjunctive queries* (UCQ) is of the form  $(x_1, \dots, x_n) : \mathcal{Q}_1 \sqcup \mathcal{Q}_2 \sqcup \dots \sqcup \mathcal{Q}_m$  where each  $(x_1, \dots, x_n) : \mathcal{Q}_i$  is a CQ and all these CQs have the same arity and the same answer variables. Answers to this UCQ w.r.t.  $\mathcal{I}$  are all answers to any CQ  $(x_1, \dots, x_n) : \mathcal{Q}_i$  w.r.t.  $\mathcal{I}$ , and similarly for the certain answers w.r.t.  $(\mathcal{D}, \mathcal{O})$ . Given UCQs  $(x_1, \dots, x_n) : \mathcal{P}_1 \sqcup \dots \sqcup \mathcal{P}_\ell$  and  $(x_1, \dots, x_n) : \mathcal{Q}_1 \sqcup \dots \sqcup \mathcal{Q}_m$  with the same answer variables, the first is contained in the second iff., for each index  $i \in \{1, \dots, \ell\}$ , there is some index  $j \in \{1, \dots, m\}$  such that the CQ  $(x_1, \dots, x_n) : \mathcal{P}_i$  is contained in the CQ  $(x_1, \dots, x_n) : \mathcal{Q}_j$  [16].

**Translation to First-order Logic.** Databases, CQs, and TGDs can be syntactically translated into first-order logic by replacing each finite set  $\mathcal{A}$  of atoms by its conjunction  $\bigwedge \mathcal{A}$  and adding quantifiers for the variables. In particular, a database  $\mathcal{D}$  translates to  $\exists x_1. \exists x_2. \dots \exists x_n. \bigwedge \mathcal{D}$  for an arbitrary enumeration  $\text{Var}(\mathcal{D}) = \{x_1, x_2, \dots, x_n\}$ , a CQ  $(x_1, \dots, x_n) : \mathcal{Q}$  translates to  $\exists y_1. \exists y_2. \dots \exists y_m. \bigwedge \mathcal{Q}$  for an arbitrary enumeration  $\text{Var}(\mathcal{Q}) \setminus \{x_1, x_2, \dots, x_n\} = \{y_1, y_2, \dots, y_m\}$ , and a TGD  $\mathcal{B} \Rightarrow \mathcal{H}$  translates to  $\forall x_1. \forall x_2. \dots \forall x_n. (\bigwedge \mathcal{B} \rightarrow \exists y_1. \exists y_2. \dots \exists y_m. \bigwedge \mathcal{H})$  for arbitrary enumerations  $\text{Var}(\mathcal{B}) = \{x_1, x_2, \dots, x_n\}$  and  $\text{Var}(\mathcal{H}) \setminus \text{Var}(\mathcal{B}) = \{y_1, y_2, \dots, y_m\}$ . Assuming The Axiom of Choice, the Löwenheim-Skolem Theorem implies that this translation preserves the semantics – it suffices

to consider countable structures in order to interpret first-order theories over at most countable signatures. Since we can rewrite between countable structures (first-order interpretations) and the above defined interpretations in the obvious way, every first-order model yields a model in the above sense and vice versa.

**Products.** Given finitely many sets  $\mathcal{A}_1, \dots, \mathcal{A}_n$  of atoms, their *product*  $\mathcal{A}_1 \times \dots \times \mathcal{A}_n$  is a set consisting of the atoms  $(f(t_1^1, \dots, t_1^n), \dots, f(t_k^1, \dots, t_k^n)) : R$  for all atoms  $(t_1^1, \dots, t_k^1) : R \in \mathcal{A}_1, \dots, (t_1^n, \dots, t_k^n) : R \in \mathcal{A}_n$ , where  $f : (\mathbf{C} \cup \mathbf{V})^n \rightarrow \mathbf{C} \cup \mathbf{V}$  is an arbitrary bijection such that  $f(c, \dots, c) = c$  for each constant  $c$  and otherwise  $f(u_1, \dots, u_n)$  is a variable (i.e. if the  $u_i$  are not all the same constant). Since  $\mathbf{V}$  is countably infinite, such bijections always exist. In technical considerations we use this function  $f$  only implicitly and rather assume that all atoms in the product are of the form  $((t_1^1, \dots, t_1^n), \dots, (t_k^1, \dots, t_k^n)) : R$ , where constants  $c$  and according tuples  $(c, \dots, c)$  are treated as synonyms.

**Undirected Graphs.** An *undirected graph* (with loops) is a pair  $(V, E)$  consisting of a set  $V$  of *vertices* and a set  $E$  of *edges* such that  $E$  consists of subsets of  $V$  with one or two elements. Edges with one element are called *loops*. A *walk* from a vertex  $v$  to a vertex  $w$  is a sequence  $v_0, v_1, \dots, v_n$  of vertices such that  $v_0 = v, v_n = w$ , and  $\{v_{i-1}, v_i\} \in E$  for each  $i \in \{1, \dots, n\}$ ; its *length* is  $n$ . It is a *path* if all vertices are pairwise distinct, and it is *empty* if  $n = 0$ . We say that a vertex  $v$  is *reachable* from another vertex  $w$  if there is a walk from  $v$  to  $w$ . A graph  $(V, E)$  is *connected* if each vertex is reachable from each other vertex. The *distance* between vertices  $v$  and  $w$  is the smallest length of a path from  $v$  to  $w$ , or  $\infty$  if no such path exists. A *cycle* is a non-empty walk that starts and ends with the same vertex and otherwise consists of pairwise distinct vertices, and we call a graph  $(V, E)$  *acyclic* if it does not contain any cycles. A connected, acyclic undirected graph is usually called an *undirected tree*. In each undirected tree, there is a unique shortest path from each vertex to each other vertex. Furthermore, each undirected tree  $(V, E)$  with a distinguished vertex  $v_0$ , called the *root*, admits a partial order  $\leq$  on  $V$ , namely where  $v \leq w$  if the (unique) shortest path from  $v_0$  to  $v$  can be extended to the (unique) shortest path from  $v_0$  to  $w$ .

**Further Notions.** Consider a set  $\mathcal{A}$  of atoms. Given a set  $U$  of terms, the subset of  $\mathcal{A}$  *generated* by  $U$  is the smallest subset  $\mathcal{B}$  of  $\mathcal{A}$  containing all atoms from  $\mathcal{A}$  that involve some term contained in  $U$  or occurring in some atom in  $\mathcal{B}$ . The *induced graph* of  $\mathcal{A}$  is the undirected graph  $G_{\mathcal{A}} := (V, E)$  where  $V$  consists of all terms and  $E$  consists of all edges  $\{t, u\}$  such that  $t$  and  $u$  occur together in some atom involving a predicate with arity  $\geq 2$  (where  $t$  and  $u$  might be equal). The *distance* in  $\mathcal{A}$  between two terms is the distance between them in  $G_{\mathcal{A}}$ . We call  $\mathcal{A}$  *connected* if the induced graph  $G_{\mathcal{A}}$  is connected. Similarly, a CQ  $(x_1, \dots, x_n) : \mathcal{Q}$  is *connected* if  $\mathcal{Q}$  is connected.

### 3. Learning Conjunctive Queries from Examples

The query-by-example (QBE) paradigm considers a KB  $(\mathcal{D}, \mathcal{O})$  and sets  $P$  and  $N$  of positive and, respectively, negative examples. These examples are mappings from a fixed set of answer

variables  $x_1, \dots, x_n$  to the set of constants. A solution is a (U)CQ that separates the positive from the negative examples in the sense that all mappings in  $P$  are certain answers w.r.t. the KB but none of the negative ones. QBE is useful in situations where users do not have the ability to formulate queries themselves – they can then rather use such a solution query.

As solutions we will only consider *constant-free* (U)CQs, i.e. where no constants occur in the atoms. To this end, we ignore the semantics of constants and rather treat them as if they were variables. Formally, we use *constant-ignoring* homomorphisms from  $\mathcal{A}$  to  $\mathcal{B}$ , which are defined like homomorphisms but without the requirement to leave constants unchanged (i.e.  $h(c) = c$  is not required for each constant  $c$ ).

**Definition 1.** Consider a KB  $(\mathcal{D}, \mathcal{O})$ , variables  $x_1, \dots, x_n$ , and finite sets  $P$  and  $N$  of mappings  $\sigma: \{x_1, \dots, x_n\} \rightarrow \mathbf{C}$ . A (U)CQ with answer variables  $x_1, \dots, x_n$  *separates*  $P$  and  $N$  if all mappings in  $P$  are certain answers to it w.r.t.  $(\mathcal{D}, \mathcal{O})$  but no mapping in  $N$  is a certain answer.

By slightly adapting the proof of Theorem 1 in [4] and further utilizing Lemma 4 in [4] we immediately obtain a proof for the following statement.<sup>4</sup>

**Theorem 2.** Assume a KB  $(\mathcal{D}, \mathcal{O})$ , variables  $x_1, \dots, x_n$ , and finite sets  $P$  and  $N$  of mappings  $\sigma: \{x_1, \dots, x_n\} \rightarrow \mathbf{C}$ , where  $P = \{\sigma_1, \dots, \sigma_p\}$ . Consider the mapping  $\sigma_P: \{x_1, \dots, x_n\} \rightarrow \mathbf{C} \cup \mathbf{V}$  where  $\sigma_P(x_i) := (\sigma_1(x_i), \dots, \sigma_p(x_i))$ . There is a constant-free CQ that separates  $P$  and  $N$  iff. the following two conditions hold:

1. For each variable  $x_i \in \{x_1, \dots, x_n\}$ , the term  $\sigma_P(x_i)$  occurs in some atom of  $\times_{i=1}^p \text{chase}(\mathcal{D}, \mathcal{O})$  (the  $p$ -fold product of the chase).
2. For each  $\tau \in N$ , there is no constant-ignoring homomorphism from  $\times_{i=1}^p \text{chase}(\mathcal{D}, \mathcal{O})$  to  $\text{chase}(\mathcal{D}, \mathcal{O})$  that sends  $\sigma_P(x_i)$  to  $\tau(x_i)$  for each variable  $x_i \in \{x_1, \dots, x_n\}$ .

Condition 2 is equivalent to each of the following conditions, where  $\mathcal{P}$  is the subset of  $\times_{i=1}^p \text{chase}(\mathcal{D}, \mathcal{O})$  generated by the terms  $\sigma_P(x_1), \dots, \sigma_P(x_n)$ :

- 2'. For each  $\tau \in N$ , there is no constant-ignoring homomorphism from  $\mathcal{P}$  to  $\text{chase}(\mathcal{D}, \mathcal{O})$  that sends  $\sigma_P(x_i)$  to  $\tau(x_i)$  for each variable  $x_i \in \{x_1, \dots, x_n\}$ .
- 2". There is a depth  $d \in \mathbb{N}$  such that, for each  $\tau \in N$ , there is no constant-ignoring homomorphism from  $\mathcal{P}|_d$  to  $\text{chase}(\mathcal{D}, \mathcal{O})$  that sends  $\sigma_P(x_i)$  to  $\tau(x_i)$  for each variable  $x_i \in \{x_1, \dots, x_n\}$ , where  $\mathcal{P}|_d$  is the subset of  $\mathcal{P}$  that consists only of the atoms involving terms with a distance of at most  $d$  to some term  $\sigma_P(x_i)$ .

Specifically, it follows that there is a constant-free CQ separating  $P$  and  $N$  iff. there is a connected such CQ. Both above conditions can be decided if the chase terminates – in this case we obtain a CQ that separates  $P$  and  $N$  and is most specific w.r.t. query containment as the query  $(x_1, \dots, x_n): \mathcal{Q}$ , where the atom set  $\mathcal{Q}$  is the product  $\times_{i=1}^p \text{chase}(\mathcal{D}, \mathcal{O})$  specifically constructed with a bijection  $f$  such that all values of  $f$  are variables (since we want a constant-free CQ) and  $f(\sigma_P(x_i)) = x_i$  for each answer variable  $x_i$ .

<sup>4</sup>Actually, Lemma 4 is implicitly used in the proof of Theorem 1, i.e. within [4] Lemma 4 should have been proven before Theorem 1.

Existence of a constant-free separator CQ is undecidable w.r.t.  $\mathcal{ELI}$  KBs [1]. If the KB is expressible in Horn- $\mathcal{ALC}$ , then non-existence of a constant-free separator CQ is complete for non-deterministic exponential time [4]. Furthermore if a CQ exists, then in Condition 2” there is a depth  $d$  that is exponential in the size of the KB — thus there is a connected separating CQ of double exponential size [4]. However, a most specific CQ need not exist w.r.t. Horn- $\mathcal{ALC}$  KBs. As a counterexample consider the database  $\{a:A, b:B, c:C\}$ , the ontology  $\{\{x:A\} \Rightarrow \{(x,y):r, y:A\}, \{x:B\} \Rightarrow \{(x,y):r, y:B\}\}$ , where the first TGD is  $A \sqsubseteq \exists r. A$  in DL notation, positive examples  $\{x_1 \mapsto a, x_1 \mapsto b\}$ , and negative examples  $\{x_1 \mapsto c\}$ . The above conditions are obviously fulfilled, i.e. there exists a separating CQ. However, a most specific CQ would need to contain an infinite  $r$ -chain issuing from the answer variable  $x_1$ , which is impossible.

For the cases where a CQ separator does not exist, there could still be a UCQ separator. Of course, such a UCQ exists iff., for each positive example  $\sigma \in P$ , there is a CQ separating  $\{\sigma\}$  and  $N$ , say  $(x_1, \dots, x_n) : \mathcal{Q}_\sigma$  — a UCQ separating  $P$  and  $N$  is then  $(x_1, \dots, x_n) : \bigsqcup_{\sigma \in P} \mathcal{Q}_\sigma$ . For this reason, existence of a constant-free separator UCQ w.r.t. Horn- $\mathcal{ALC}$  KBs is complete for (deterministic) exponential time [4]. However, due to the excessive usage of disjunction, such an UCQ solution could suffer from over-fitting. Therefore, the number of disjuncts of such a UCQ solution should be minimized.

Determining the minimal number of disjuncts is already NP-hard since we can reduce the minimum-set-cover problem [17] as follows. Consider a set  $P = \{c_1, \dots, c_n\}$  and subsets  $S_1, \dots, S_m \subseteq P$  such that  $S_1 \cup \dots \cup S_m = P$ . A minimum set cover is a size-minimal subset  $I \subseteq \{1, \dots, m\}$  such that  $\bigcup_{i \in I} S_i = P$ . For the reduction, we treat the  $c_i$  as constants and the  $S_j$  as unary relations, and consider a further constant  $d$ , the database  $\mathcal{D} := \{c_i : S_j \mid c_i \in S_j\}$ , positive examples  $P$ , and negative examples  $N := \{d\}$  (where we assume a single answer variable  $x_1$  and do not distinguish between the element  $c_i$  and the mapping  $x_1 \mapsto c_i$ , and likewise for  $d$ ). Then for each subset  $P' \subseteq P$ , there is a CQ separating  $P'$  and  $N$  iff.  $P' \subseteq S_j$  for some  $j$ . Thus a separating UCQ with the fewest disjuncts yields a minimum set cover.

In practice, one might do it the greedy way: determine a first maximal subset  $P_1$  of  $P$  such that a CQ separating  $P_1$  and  $N$  exists, next find a maximal subset  $P_2$  of  $P \setminus P_1$  that can be separated from  $N$  by a CQ, and likewise continue inductively with the remaining positive examples until none is left over.

When we are only interested in a subset  $\mathbf{S}$  of the set  $\mathbf{R}$  of relations to be used in the separator (U)CQs, then Theorem 2 holds accordingly when the  $p$ -fold product  $\times_{i=1}^p \text{chase}(\mathcal{D}, \mathcal{O})$  is replaced by its subset consisting of all atoms with a relation in  $\mathbf{S}$ .

## 4. Trémaux Trees

A Trémaux tree of an undirected graph is a spanning tree such that every edge of the graph connects an ancestor–descendant pair in the tree. Trémaux trees are named after Charles Pierre Trémaux, a 19th-century French author who used a form of depth-first search as a strategy for solving mazes. In computer science they are also called depth-first trees [18], whereas in graph theory they are rather called normal spanning trees [19]. Trémaux trees exist for all finite graphs and can be computed in polynomial time [20] as well as by a randomized NC algorithm [21].



**Definition 3.** Let  $(V, E)$  be a connected undirected graph with loops. Further let  $v_0 \in V$  be a vertex. A *Trémaux tree* of  $(V, E)$  with root  $v_0$  is a subset  $F$  of  $E$  such that

1.  $(V, F)$  is an undirected tree, and
2.  $v \leq w$  or  $w \leq v$  for each edge  $\{v, w\} \in E$ , where  $\leq$  is the induced partial order of  $(V, F)$  for root  $v_0$ .

Although the following result is already known, we want to provide an own proof.<sup>5</sup>

**Proposition 4.** *For each finite connected undirected graph with loops and with a distinguished vertex, a Trémaux tree can be computed in polynomial time.*

*Proof.* Assume that  $(V, E)$  is a connected undirected graph with loops and let  $v_0$  be a distinguished vertex from  $V$ . In the following, we will devise a recursive procedure that produces a Trémaux tree. Firstly, initialize an undirected graph  $(V, F)$  where  $F := E$ . During the run of the procedure, we maintain a set  $U$  of *unprocessed* vertices that we initialize as  $U := V$ . The invariant during the construction is that, for each processed vertex  $v \in V \setminus U$ , there is a unique shortest path from  $v_0$  to  $v$  within  $(V, F)$ . The computation starts by calling the following recursive procedure on the distinguished vertex  $v_0$ .

**Process( $v$ ):** Mark  $v$  as processed by removing  $v$  from  $U$ . The unprocessed neighborhood of  $v$  is the set  $N_U(v) := \{w \mid w \in U \text{ and } \{v, w\} \in E\}$ . Define the undirected graph  $(V', E')$  by

$$V' := \{v\} \cup N_U(v),$$

$$E' := \{\{v, w\} \mid w \in N_U(v)\} \cup \{\{w_1, w_2\} \mid w_1, w_2 \in N_U(v) \text{ and } w_1 \sim w_2\},$$

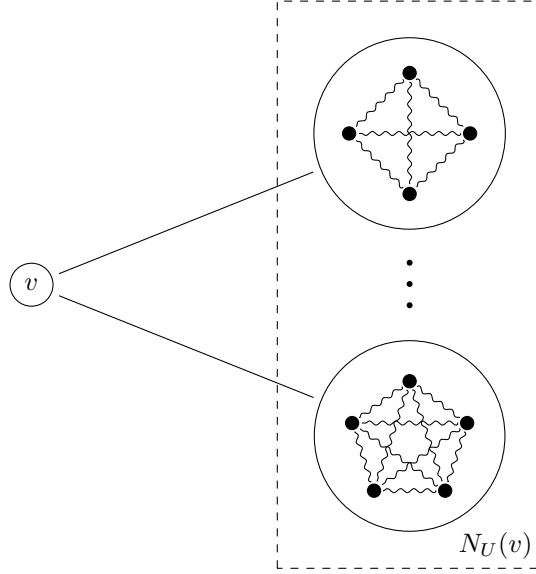
where  $w_1 \sim w_2$  if  $w_1$  is reachable from  $w_2$  in the subgraph  $(V, E)|_U := (U, \{e \mid e \in E \text{ and } e \subseteq U\})$ . Note that  $\sim$  is an equivalence relation on  $N_U(v)$ , i.e. it is reflexive, symmetric, and transitive. Thus, the subgraph of  $(V', E')$  obtained by removing  $v$  is a disjoint union of complete graphs,<sup>6</sup> and all vertices of each such complete graph are connected with  $v$  by an edge. A schematic presentation of the graph  $(V', E')$  is given in Figure 1. For each complete graph, select one vertex  $w$ , delete from  $F$  all edges from  $v$  into that complete graph except  $\{v, w\}$ , and then proceed recursively by calling **Process( $w$ )**. Note that, if  $v_0, v_1, \dots, v$  is the unique shortest path from  $v_0$  to  $v$  within  $(V, F)$ , then  $v_0, v_1, \dots, v, w$  is the unique shortest path from  $v_0$  to  $w$  within  $(V, F)$ , i.e. the invariant is satisfied.

After termination, the invariant is still satisfied and so it follows that the resulting graph  $(V, F)$  is an undirected tree. It remains to show that  $v \leq w$  or  $w \leq v$  for each edge  $\{v, w\} \in E$ , where  $\leq$  is the partial order on  $V$  that is induced by  $(V, F)$  for root  $v_0$ . For this purpose, consider an edge  $\{v, w\} \in E$ .

- If  $\{v, w\}$  has not been deleted, i.e. is contained in  $F$ , then either the unique shortest path from  $v_0$  to  $v$  can be extended to the unique shortest path from  $v_0$  to  $w$  or vice versa. It follows that either  $v \leq w$  or  $w \leq v$ .

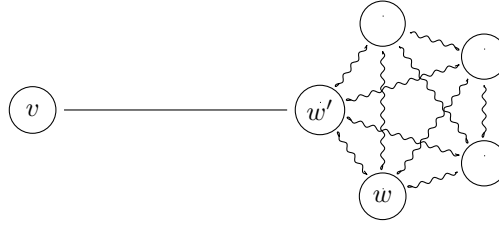
<sup>5</sup>In fact, the author only later recognized that Trémaux trees have already been well investigated.

<sup>6</sup>A complete graph is a graph in which all vertices are connected by an edge.



**Figure 1:** A schematic presentation of the graph  $(V', E')$

- Otherwise, the edge  $\{v, w\}$  has been deleted from  $F$ , i.e. during the call either of  $\text{Process}(v)$  or of  $\text{Process}(w)$ . We only treat the first case, the other is analogous. During the call of



**Figure 2:** After the call of  $\text{Process}(v)$

$\text{Process}(v)$ , a vertex  $w'$  in the complete graph containing  $w$  was selected and all edges from  $v$  into this complete graph except  $\{v, w'\}$  were deleted, see Figure 2. Due to the invariant it follows that, after termination, the unique shortest path from  $v_0$  to  $w$  must go through  $v$  and thus  $v \leq w$  must be satisfied.

We conclude that, after termination, the resulting graph  $(V, F)$  is a Trémaux tree of  $(V, E)$  with root  $v_0$ .

Since each vertex is processed only once, there are only linearly many calls to the procedure  $\text{Process}(\cdot)$ . It is well-known that graph reachability can be decided in polynomial time and thus the intermediate graph  $(V', E')$  during each call to  $\text{Process}(\cdot)$  can be constructed in polynomial time. We conclude that the initial call of  $\text{Process}(v_0)$  terminates in polynomial time.  $\square$



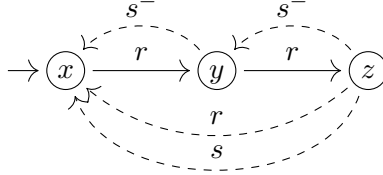
In the remainder of this article we assume that the signature consists only of constants, unary relations, and binary relations.

**Definition 5.** Let  $\mathcal{D}$  be a database and  $t$  a term in  $\mathcal{D}$ . A *Trémaux order* of  $\mathcal{D}$  with root  $t$  is the partial order  $\leq$  on  $\text{Terms}(\mathcal{D})$  induced by some Trémaux tree of the induced graph  $G_{\mathcal{D}}$  with root  $t$ .

According to Proposition 4, Trémaux orders can be computed in polynomial time.

Let  $\mathcal{D}$  be a database and  $t_0$  a term occurring in  $\mathcal{D}$ . Further assume that  $\leq$  is a Trémaux order of  $\mathcal{D}$  with root  $t_0$  and denote by  $\prec$  the *neighborhood relation* of  $\leq$ , i.e.  $t \prec u$  if  $t < u$  and there is no  $v$  such that  $t < v < u$ . Note that, if  $t \prec u$ , then the edge  $\{t, u\}$  must be present, i.e. there is a role  $r$  such that  $\mathcal{D}$  contains at least one of the atoms  $(t, u) : r$  or  $(u, t) : r$ . We will also write  $(t, u) : r^-$  for the latter, where  $r^-$  denotes the inverse of  $r$ , i.e. we do not distinguish between the atoms  $(u, t) : r$  and  $(t, u) : r^-$ . For each  $t \prec u$ , we choose some atom  $(t, u) : R$  in  $\mathcal{D}$ , where  $R$  is a role  $r$  or an inverse  $r^-$ , and then set  $R_{t,u} := R$ . To indicate that  $t \prec u$  and  $R_{t,u} = R$ , we occasionally write  $t \prec_R u$ .

**Example 6.** Consider the database  $\mathcal{D} := \{(x, y) : r, (x, y) : s, (y, z) : r, (y, z) : s, (z, x) : r, (z, x) : s\}$  and let  $x$  be the root term. The induced graph  $G_{\mathcal{D}} := (V, E)$  has vertex set  $V := \{x, y, z\}$  and edge set  $E := \{\{x, y\}, \{y, z\}, \{z, x\}\}$ . A Trémaux tree of  $G_{\mathcal{D}}$  with root  $x$  is  $(V, F)$  with edge set  $F := \{\{x, y\}, \{y, z\}\}$ . The induced partial order  $\leq$ , which is a Trémaux order of  $\mathcal{D}$  with root  $x$ , has the neighborhood relation  $\prec$  where  $x \prec y \prec z$ . We choose  $R_{x,y} := r$  and  $R_{y,z} := r$ . The below figure shows the Trémaux order  $\leq$ , where solid lines represent edges in the Trémaux tree and dashed lines represent the remaining edges.



## 5. Constructing Small Separating Queries

Consider a KB  $(\mathcal{D}, \mathcal{O})$  defined over a signature that consists only of constants, unary relations, and binary relations. Further consider a single answer variable  $x_1$  and finite sets  $P$  and  $N$  of mappings  $\sigma : \{x_1\} \rightarrow \mathbf{C}$ , which are the positive and negative examples, such that there is a constant-free CQ that separates  $P$  and  $N$ . Our goal now is to construct a *small* separating CQ.

By assumption, the two conditions in Theorem 2 must be satisfied. Condition 1 ensures that we can find a term in the  $p$ -fold product of the chase that “describes” all commonalities of the positive examples, viz. in our setting the tuple  $\sigma_P(x_1)$  consisting of all  $\sigma(x_1)$  where  $\sigma$  ranges over  $P$ . Condition 2 ensures that these commonalities are not all fulfilled by any negative example. Together both conditions ensure the existence of a separating CQ.

Specifically by Condition 2<sup>7</sup> there is a depth  $d$  such that it suffices to consider all terms with a distance  $\leq d$  to  $\sigma_P(x_1)$ , and already this finite<sup>7</sup> subset  $\mathcal{P}|_d$  of the  $p$ -fold product of

<sup>7</sup>Since there are only finitely many TGDs, the chase is finitely branching.

$\text{chase}(\mathcal{D}, \mathcal{O})$  does not admit, for any negative example  $\tau$ , a constant-ignoring homomorphism to  $\text{chase}(\mathcal{D}, \mathcal{O})$  that maps  $\sigma_P(x_1)$  to  $\tau(x_1)$ . We then obtain a separating CQ  $x_1 : \mathcal{P} \upharpoonright_d$  when  $\mathcal{P} \upharpoonright_d$  is taken from the product  $\times_{i=1}^P \text{chase}(\mathcal{D}, \mathcal{O})$  specifically constructed with a bijection  $f$  such that  $f(\sigma_P(x_1)) = x_1$  and all values of  $f$  are variables. However, this CQ can be quite large.

Now we exploit the particular structure of a Trémaux order in order to recursively define Trémaux homomorphisms, and then we show that existence of a constant-ignoring homomorphism is equivalent to existence of such a Trémaux homomorphism. Afterwards, we show how a small subset  $\mathcal{Q}$  of  $\mathcal{P} \upharpoonright_d$  can be extracted for which  $x_1 : \mathcal{Q}$  is already a CQ separating  $P$  and  $N$ .

**Definition 7.** Let  $\mathcal{D}$  be a connected database,  $t$  a term in  $\mathcal{D}$ , and  $\mathcal{B}$  a set of atoms. Further let  $\leq$  be a Trémaux order of  $\mathcal{D}$ . A *Trémaux homomorphism* from  $\mathcal{D}$  to  $\mathcal{B}$  up to  $t$  is a partial mapping  $\ell : \text{Terms}(\mathcal{D}) \rightarrow \text{Terms}(\mathcal{B})$  that fulfills the following conditions:

1.  $\ell(u)$  is defined for each  $u \leq t$ .
2.  $\ell(u) : A \in \mathcal{B}$  for each  $u : A \in \mathcal{D}$  where  $A$  is a unary relation and  $u \leq t$ .
3.  $(\ell(u_1), \ell(u_2)) : R \in \mathcal{B}$  for each  $(u_1, u_2) : R \in \mathcal{D}$  where  $R$  is a binary relation (or its inverse),  $u_1 \leq t$ , and  $u_2 \leq t$ .
4.  $\ell$  can be extended to a Trémaux homomorphism from  $\mathcal{D}$  to  $\mathcal{B}$  up to each  $v$  where  $t \prec v$ .

Formally: for each  $v$  where  $t \prec_R v$ , there is some  $w$  such that  $(\ell(t), w) : R \in \mathcal{B}$  and the extended partial mapping  $\ell \cup \{v \mapsto w\}$  is a Trémaux homomorphism from  $\mathcal{D}$  to  $\mathcal{B}$  up to  $v$ .

**Proposition 8.** Let  $\mathcal{D}$  be a connected database,  $\mathcal{B}$  a set of atoms,  $t$  a term in  $\mathcal{D}$ , and  $u$  a term in  $\mathcal{B}$ . Further let  $\leq$  be a Trémaux order of  $\mathcal{D}$  with root  $t$ . The following statements are equivalent.

1. There is a constant-ignoring homomorphism from  $\mathcal{D}$  to  $\mathcal{B}$  that maps  $t$  to  $u$ .
2.  $\{t \mapsto u\}$  is a Trémaux homomorphism from  $\mathcal{D}$  to  $\mathcal{B}$  up to  $t$ .

*Proof.* Regarding the only-if direction, let  $h$  be a homomorphism from  $\mathcal{D}$  to  $\mathcal{B}$  such that  $h(t) = u$ . Therefore  $h$  already satisfies Conditions 2 and 3 in Definition 7 for every term in  $\mathcal{D}$  (not only for those  $\leq t$ ). Condition 4 follows by induction w.r.t.  $\leq$  since, if  $t' \prec_R u'$ , then  $(t', u') : R \in \mathcal{D}$  and thus  $(h(t'), h(u')) : R \in \mathcal{B}$ .

In the converse direction, we obtain a homomorphism in the limit. More specifically, we can lazily build it by Condition 4, i.e. we traverse through  $\mathcal{D}$  along  $\prec$  starting from  $t$  and construct the union of all partial mappings  $\ell \cup \{t' \mapsto u'\}$ . This yields a well-defined mapping since, on the one hand, assignments of terms that are smaller w.r.t.  $\leq$  are never overwritten and, on the other hand, all terms  $u'$  next to a term  $t'$  (i.e. where  $t' \prec u'$ ) can be processed independently of each other due to Condition 2 in Definition 3. Since all terms are reachable from  $t$ , this limit mapping is defined for all terms in  $\mathcal{D}$ .  $\square$

One point worthy of remark is that, since all terms next to a term can be processed independently, we can easily implement a parallel procedure for deciding homomorphism existence when the signature is at most binary.<sup>8</sup>

<sup>8</sup>The author does not know whether this has already been exploited in query answering systems.

Finally, we come back to our goal of constructing a small separating CQ. Proposition 8 yields that, for each  $\tau \in N$ , the partial mapping  $\{\sigma_P(x_1) \mapsto \tau(x_1)\}$  is no Trémaux homomorphism from  $\mathcal{P}|_n$  to  $\text{chase}(\mathcal{D}, \mathcal{O})$  up to  $\sigma_P(x_1)$ , and thus Conditions 2 or 3 in Definition 7 must already be violated by the root  $\sigma_P(x_1)$  or, by following the recursion in Condition 4, one of them must be violated by a term  $\geq \sigma_P(x_1)$ . We use this observation to collect a small but sufficiently large subset of  $\mathcal{P}|_d$  that already witnesses the non-existence of homomorphisms for all negative examples, and afterwards transform this subset into a CQ.

We initialize the subset  $\mathcal{Q}$  of  $\mathcal{P}|_d$  as the empty set. Furthermore, we maintain a mapping  $L$  that assigns to each term in  $\mathcal{P}|_d$  a set of partial mappings, where we initialize  $L(\sigma_P(x_1)) := \{\{\sigma_P(x_1) \mapsto \tau(x_1)\} \mid \tau \in N\}$  and  $L(t) := \emptyset$  for each term  $t \neq \sigma_P(x_1)$ . The invariant is that each set  $L(t)$  will always contain only such mappings  $\ell$  where  $\ell(u)$  is defined for each  $u \leq t$  and that are no Trémaux homomorphisms from  $\mathcal{P}|_d$  to  $\text{chase}(\mathcal{D}, \mathcal{O})$  up to  $t$ . We start with processing  $\sigma_P(x_1)$ , i.e. we call  $\text{Process}(\sigma_P(x_1))$ .

$\text{Process}(t)$ : For each  $\ell \in L(t)$ , do the following.

1. Choose one of the following two instructions and try to execute it. If it cannot be executed, try the other.
  - a) Try to choose a unary atom  $t : A$  in  $\mathcal{P}|_d$  where  $\ell(t) : A$  is not in  $\text{chase}(\mathcal{D}, \mathcal{O})$ , and add the atom  $t : A$  to  $\mathcal{Q}$ .
  - b) Try to choose a binary atom  $(t, u) : R$  in  $\mathcal{P}|_d$  such that  $u \leq t$  and where  $(\ell(t), \ell(u)) : R$  is not in  $\text{chase}(\mathcal{D}, \mathcal{O})$ , and add the atom  $(t, u) : R$  to  $\mathcal{Q}$ .
2. If none of the two above instructions can be executed, then choose a term  $v$  where  $t \prec_R v$  such that, for each term  $w$  where  $(\ell(t), w) : R$  is in  $\text{chase}(\mathcal{D}, \mathcal{O})$ , the extension  $\ell \cup \{v \mapsto w\}$  is no Trémaux homomorphism from  $\mathcal{P}|_d$  to  $\text{chase}(\mathcal{D}, \mathcal{O})$  up to  $v$ . Due to the invariant and Condition 4 in Definition 7, such a term  $v$  must exist. Then, add the atom  $(t, v) : R$  to  $\mathcal{Q}$  and further add  $\ell \cup \{v \mapsto w\}$  to  $L(v)$  for each  $w$  where  $(\ell(t), w) : R$  is in  $\text{chase}(\mathcal{D}, \mathcal{O})$ .

Afterwards, call  $\text{Process}(v)$  for each  $v$  where  $t \prec v$  and  $L(v) \neq \emptyset$ .

Termination of the initial call  $\text{Process}(\sigma_P(x_1))$  is guaranteed since  $\mathcal{P}|_n$  is finite and  $\text{chase}(\mathcal{D}, \mathcal{O})$  is finitely branching. Further note that in Instruction 1 it suffices to consider the atoms at  $t$  since those with terms  $< t$  have already been tried earlier. In the end,  $x_1 : \mathcal{Q}$  is CQ separating  $P$  and  $N$ .

It is easy to see that the above procedure yields a minimal separating CQ (i.e. with a smallest number of atoms) when there is only one negative example. The author claims that with a suitable strategy the procedure can also yield minimal CQs for multiple negative examples, but existing results on verifying extremal separating CQs already imply high computational complexity even without TGDs [6]. Within the framework of PAC-learning, computation of size-minimal separating queries expressible by  $\mathcal{EL}$  concepts has already been considered [2].

**Future Prospects.** In order to expand on this result, it would be interesting to lift the current restriction to only one answer variable of separating CQs and, furthermore, to investigate how higher-arity relations in the signature can be handled.

## Acknowledgments

This work has been supported by Deutsche Forschungsgemeinschaft (DFG) in Project 389792660 (TRR 248: Foundations of Perspicuous Software Systems) and in Project 558917076 (Construction and Repair of Description-logic Knowledge Bases) as well as by the Saxon State Ministry for Science, Culture, and Tourism (SMWK) by funding the Center for Scalable Data Analytics and Artificial Intelligence (ScaDS.AI).

## References

- [1] Maurice Funk, Jean Christoph Jung, Carsten Lutz, Hadrien Pulcini, Frank Wolter. Learning Description Logic Concepts: When can Positive and Negative Examples be Separated? In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*. 2019, pp. 1682–1688. DOI: 10.24963/ijcai.2019/233.
- [2] Balder ten Cate, Maurice Funk, Jean Christoph Jung, Carsten Lutz. SAT-Based PAC Learning of Description Logic Concepts. In: *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China*. 2023, pp. 3347–3355. DOI: 10.24963/IJCAI.2023/373.
- [3] Jean Christoph Jung, Carsten Lutz, Hadrien Pulcini, Frank Wolter. Separating Data Examples by Description Logic Concepts with Restricted Signatures. In: *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning, KR 2021, Online event, November 3-12, 2021*. 2021, pp. 390–399. DOI: 10.24963/KR.2021/37.
- [4] Víctor Gutiérrez-Basulto, Jean Christoph Jung, Leif Sabellek. Reverse Engineering Queries in Ontology-Enriched Systems: The Case of Expressive Horn Description Logic Ontologies. In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*. 2018, pp. 1847–1853. DOI: 10.24963/ijcai.2018/255.
- [5] Balder ten Cate, Victor Dalmau. Conjunctive Queries: Unique Characterizations and Exact Learnability. In: *ACM Trans. Database Syst.* 47.4 (2022), 14:1–14:41. DOI: 10.1145/3559756.
- [6] Balder ten Cate, Victor Dalmau, Maurice Funk, Carsten Lutz. Extremal Fitting Problems for Conjunctive Queries. In: *Proceedings of the 42nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2023, Seattle, WA, USA, June 18-23, 2023*. 2023, pp. 89–98. DOI: 10.1145/3584372.3588655.
- [7] Balder ten Cate, Maurice Funk, Jean Christoph Jung, Carsten Lutz. On the non-efficient PAC learnability of conjunctive queries. In: *Inf. Process. Lett.* 183 (2024), p. 106431. DOI: 10.1016/J.IPL.2023.106431.
- [8] Balder ten Cate, Maurice Funk, Jean Christoph Jung, Carsten Lutz. Fitting Algorithms for Conjunctive Queries. In: *SIGMOD Rec.* 52.4 (2023), pp. 6–18. DOI: 10.1145/3641832.3641834.

- [9] Jean Christoph Jung, Carsten Lutz, Hadrien Pulcini, Frank Wolter. Logical separability of labeled data examples under ontologies. In: *Artif. Intell.* 313 (2022), p. 103785. DOI: 10.1016/J.ARTINT.2022.103785.
- [10] Markus Krötzsch, Maximilian Marx, Sebastian Rudolph. The Power of the Terminating Chase (Invited Talk). In: *22nd International Conference on Database Theory, ICDT 2019, March 26-28, 2019, Lisbon, Portugal*. 2019, 3:1–3:17. DOI: 10.4230/LIPICS.ICDT.2019.3.
- [11] Alin Deutsch, Alan Nash, Jeffrey B. Remmel. The chase revisited. In: *Proceedings of the Twenty-Seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2008, June 9-11, 2008, Vancouver, BC, Canada*. 2008, pp. 149–158. DOI: 10.1145/1376916.1376938.
- [12] Bernardo Cuenca Grau, Ian Horrocks, Markus Krötzsch, Clemens Kupke, Despoina Magka, Boris Motik, Zhe Wang. Acyclicity Notions for Existential Rules and Their Application to Query Answering in Ontologies. In: *J. Artif. Intell. Res.* 47 (2013), pp. 741–808. DOI: 10.1613/jair.3949.
- [13] Catriel Beeri, Moshe Y. Vardi. The Implication Problem for Data Dependencies. In: *Automata, Languages and Programming, 8th Colloquium, Acre (Akko), Israel, July 13-17, 1981, Proceedings*. 1981, pp. 73–85. DOI: 10.1007/3-540-10843-2\_7.
- [14] Ashok K. Chandra, Harry R. Lewis, Johann A. Makowsky. Embedded Implicational Dependencies and their Inference Problem. In: *Proceedings of the 13th Annual ACM Symposium on Theory of Computing, May 11-13, 1981, Milwaukee, Wisconsin, USA*. 1981, pp. 342–354. DOI: 10.1145/800076.802488.
- [15] Ashok K. Chandra, Philip M. Merlin. Optimal Implementation of Conjunctive Queries in Relational Data Bases. In: *Proceedings of the 9th Annual ACM Symposium on Theory of Computing, May 4-6, 1977, Boulder, Colorado, USA*. 1977, pp. 77–90. DOI: 10.1145/800105.803397.
- [16] Yehoshua Sagiv, Mihalis Yannakakis. Equivalences Among Relational Expressions with the Union and Difference Operators. In: *J. ACM* 27.4 (1980), pp. 633–655. DOI: 10.1145/322217.322221.
- [17] Richard M. Karp. Reducibility Among Combinatorial Problems. In: *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*. 1972, pp. 85–103. DOI: 10.1007/978-1-4684-2001-2\_9.
- [18] Shimon Even, Guy Even. *Graph Algorithms*. 2012. DOI: 10.1017/CBO9781139015165.
- [19] Reinhard Diestel. *Graph Theory*. 2025. DOI: 10.1007/978-3-662-70107-2.
- [20] John H. Reif. Depth-First Search is Inherently Sequential. In: *Inf. Process. Lett.* 20.5 (1985), pp. 229–234. DOI: 10.1016/0020-0190(85)90024-9.
- [21] Alok Aggarwal, Richard J. Anderson. A random NC algorithm for depth first search. In: *Comb.* 8.1 (1988), pp. 1–12. DOI: 10.1007/BF02122548.