Extremal Separation Problems for Temporal Instance Queries

Jean Christoph Jung[1] , **Vladislav Ryzhikov**[2] , **Frank Wolter**[3] and **Michael Zakharyaschev**[2]
[1]Department of Computer Science, TU Dortmund University, Germany
[2]School of Computing and Mathematical Sciences, Birkbeck, University of London, UK
[3]Department of Computer Science, University of Liverpool, UK
jean.jung@tu-dortmund.de, {v.ryzhikov,m.zakharyaschev}@bbk.ac.uk, wolter@liverpool.ac.uk

Abstract

The separation problem for a class $\mathcal{Q}$ of database queries is to find a query in $\mathcal{Q}$ that distinguishes between a given set of 'positive' and 'negative' data examples. Separation provides explanations of examples and underpins the query-by-example paradigm to support database users in constructing and refining queries. As the space of all separating queries can be large, it is helpful to succinctly represent this space by means of its most specific (logically strongest) and general (weakest) members. We investigate this extremal separation problem for classes of instance queries formulated in linear temporal logic *LTL* with the operators conjunction, 'next', and 'eventually'. Our results range from tight complexity bounds for verifying and counting extremal separators to algorithms computing them.

1 Introduction

The separation (aka fitting or consistency) problem for a class $\mathcal{Q}$ of queries is to find some $\boldsymbol{q} \in \mathcal{Q}$ that separates a given set $E = (E^+, E^-)$ of positive and negative data examples in the sense that $\mathcal{D} \models \boldsymbol{q}$ for all $\mathcal{D} \in E^+$, and $\mathcal{D} \not\models \boldsymbol{q}$ for all $\mathcal{D} \in E^-$. Separation underpins the query-by-example approach, which aims to support database users in constructing queries and schema mappings with the help of data examples [Alexe *et al.*, 2011; Martins, 2019], inductive logic programming [Cropper *et al.*, 2022], and, more recently, automated feature extraction, where separating queries are proposed as features in classifier engineering [Kimelfeld and Ré, 2018; Barceló *et al.*, 2021]. Separating queries (and, more generally, formulas) also underpin recent logic-based approaches aiming to explain positive and negative data examples given by applications [Sarker *et al.*, 2017; Camacho and McIlraith, 2019; Raha *et al.*, 2022].

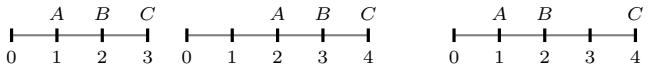
The space of all separating queries, denoted $\mathsf{s}(E, \mathcal{Q})$, forms a convex subset of $\mathcal{Q}$ under the containment (or logical entailment) relation $\boldsymbol{q} \models \boldsymbol{q}'$ between queries $\boldsymbol{q}$, $\boldsymbol{q}'$, and is an instance of the more general version spaces [Mitchell, 1982]. If finite, $\mathsf{s}(E, \mathcal{Q})$ can be represented by its extremal elements: the most specific (logically strongest) and most general (logically weakest) separators in $\mathcal{Q}$. In fact, many known algorithms check the existence of separators by looking for a most specific one [ten Cate and Dalmau, 2015; Barceló and Romero, 2017; Gutiérrez-Basulto *et al.*, 2018; Funk *et al.*, 2019]. This is not surprising as query classes are often closed under $\wedge$, and so the conjunction of all separators gives the unique most specific one. Dually, the unique most general separator is given by the disjunction of all separators if the query class is closed under $\vee$, which is a less common assumption. For the case of first-order queries constructed using $\wedge$ and existential quantifiers (conjunctive queries or CQs), a systematic study of extremal separation has recently been conducted in the award winning [ten Cate *et al.*, 2023].

Here, we study extremal separation for temporal instance queries. Data instances take the form $(\delta_0, \dots, \delta_n)$ describing temporal evolutions, where the δ_i are the sets of atomic propositions that are true at time i. Queries are formulated in the fragment of linear temporal logic *LTL* with the operators $\wedge$, $\bigcirc$ (next), and $\Diamond$ (eventually). These $\bigcirc\Diamond$-*queries* and its subclass of $\Diamond$-*queries* without $\bigcirc$, are obtained by restricting CQs to propositional temporal data and form the core of most temporal query languages proposed in the database and knowledge representation literature [Chomicki and Toman, 2018; Baader *et al.*, 2015; Borgwardt *et al.*, 2015; Artale *et al.*, 2021]. We are particularly interested in subclasses of the classes of $\bigcirc\Diamond$-and $\Diamond$-queries that are not closed under $\wedge$ and take the form of path queries:

$$\boldsymbol{q} = \rho_0 \wedge \boldsymbol{o}_1(\rho_1 \wedge \boldsymbol{o}_2(\rho_2 \wedge \cdots \wedge \boldsymbol{o}_n\rho_n)) \qquad (1)$$

with $\boldsymbol{o}_i \in \{\bigcirc, \Diamond\}$ and conjunctions ρ_i of atoms. The temporal patterns expressed by path queries correspond to common subsequences, subwords, and combinations thereof, which have been investigated in the string pattern matching literature for more than 50 years. Their applications range from computational linguistics to bioinformatics and revision control systems [Bergroth *et al.*, 2000; Chowdhury *et al.*, 2010; Abboud *et al.*, 2015; Blum *et al.*, 2021]. In fact, our results can also be interpreted and applied in that research tradition and our techniques combine both logic and automata-based methods with pattern matching. We note that not admitting $\vee$ in our query languages is crucial for finding this type of patterns and adding it would often trivialise separation.

Example 1. Suppose the first two sequences of events shown below are 'positive', the third one is 'negative', and our task is to explain this phenomenon using path queries. The space of



possible explanations includes $\boldsymbol{q}_1 = \Diamond(A \wedge \bigcirc(B \wedge \bigcirc C))$, $\boldsymbol{q}_2 = \Diamond(A \wedge \bigcirc\bigcirc C)$ and $\boldsymbol{q}_3 = \Diamond(B \wedge \bigcirc C)$, all of which are true at 0 in the positive examples and false in the negative one. In fact, $\boldsymbol{q}_1$ is the unique most specific explanation, while $\boldsymbol{q}_2$ and $\boldsymbol{q}_3$ are the non-equivalent most general ones. On the other hand, there exists no explanation in terms of $\Diamond$ only.

The (bounded-size) separator existence problem for various classes $\mathcal{Q}$ of *LTL*-queries has recently been studied in [Fijalkow and Lagarde, 2021; Raha *et al.*, 2022; Fortin *et al.*, 2023]. Our aim here is to investigate systematically the separator spaces $\mathsf{s}(E, \mathcal{Q})$ by determining the complexity of verifying and counting most specific and general separators and giving algorithms computing them. On our way, we determine the complexity of entailment (aka containment in database theory) between queries and computing weakening and strengthening frontiers, which are the key to understanding $\mathsf{s}(E, \mathcal{Q})$. Intuitively, a weakening/strengthening frontier of $\boldsymbol{q} \in \mathcal{Q}$ is a set of queries properly weaker/stronger than $\boldsymbol{q}$ that form a boundary between $\boldsymbol{q}$ and all of its weakenings/strengthenings in $\mathcal{Q}$.

In detail, we first prove that query containment is in P for the class of $\Diamond$-queries and all of our classes of path queries. Based on this result, we show that strengthening and weakening frontiers can be computed in polytime for path queries. This is also the case for $\Diamond$-queries and weakening frontiers but not for strengthening ones. It follows that checking whether a path query is a most specific/general separator and whether a $\Diamond$-query is a most general one are both in P. In contrast, we establish coNP-completeness of checking whether a $\Diamond$-query is a most specific separator and whether a path query is the unique most specific/general one. Using frontiers, we show for path queries that the existence of unique most specific/general separators is in the complexity class US (for which unique SAT is complete) and that counting the number of most general/specific separators is in $\sharp$P. We show that these upper bounds are tight, sometimes using the rich literature on algorithms for sequences (e.g., longest common subsequences). Our lower bounds mostly require only a bounded number of atomic propositions and an unbounded number of either negative or positive examples.

These complexity results are complemented with algorithms for computing extremal separators in the majority of our query classes. The algorithms use a graph encoding of the input example set, associating (extremal) separators with certain paths in the graph. Complexity-wise, they are optimal, running in polytime if the number of examples is bounded and exponential time otherwise.

Omitted proofs can be found in [Jung *et al.*, 2024].

1.1 Related Work

Being inspired by the investigation of extremal separation for first-order conjunctive queries (CQs) [ten Cate *et al.*, 2023], our results turn out to be very different. For instance, while separability by CQs is NExpTime-complete and separating queries are exponential in the size of the examples (the extremal ones even larger), the extremal separation problems for *LTL*-queries are often complete for SAT-related complexity classes, with separating queries being of polynomial size. For work on separation in the query-by-example paradigm we refer the reader to [Zhang *et al.*, 2013; Weiss and Cohen, 2017; Kalashnikov *et al.*, 2018; Deutch and Gilad, 2019; Staworko and Wieczorek, 2012; Barceló and Romero, 2017; Cohen and Weiss, 2016; Arenas *et al.*, 2016] in the database context and to [Gutiérrez-Basulto *et al.*, 2018; Ortiz, 2019; Cima *et al.*, 2021; Jung *et al.*, 2022] in the context of KR.

Our contribution is also closely related to work on synthesising *LTL*-formulas that explain the positive and negative data examples coming from an application [Lemieux *et al.*, 2015; Neider and Gavran, 2018; Camacho and McIlraith, 2019; Raha *et al.*, 2022; Fortin *et al.*, 2022; Fortin *et al.*, 2023]. While concerned with separability of temporal data instances and, in particular, separability by *LTL*-formulas of small size, the separator spaces $\mathsf{s}(E, \mathcal{Q})$ themselves have not yet been investigated in this context.

2 Data and Queries in *LTL*

Fix some countably-infinite set of unary predicate symbols, called *atoms*. A *signature*, σ, is any finite set of atoms. A (*temporal*) *data instance* is any finite set $\mathcal{D} \neq \emptyset$ of *facts* $A(\ell)$ with an atom A and a *timestamp* $\ell \in \mathbb{N}$, saying that A happened at ℓ. The *size* $|\mathcal{D}|$ of $\mathcal{D}$ is the number of symbols in it, with the timestamps given in *unary*. Let $\max \mathcal{D}$ be the maximal timestamp in $\mathcal{D}$. Where convenient, we also write $\mathcal{D}$ as the word $\delta_0 \dots \delta_{\max \mathcal{D}}$ with $\delta_i = \{A \mid A(i) \in \mathcal{D}\}$. Without loss of generality, we can assume that $\delta_i \neq \emptyset$, using placeholders $\top(i)$ if needed. The *signature* $\mathsf{sig}(\mathcal{D})$ of $\mathcal{D}$ is the set of atoms occurring in it.

We query data instances by means of *LTL*-formulas, called *queries*, that are built from atoms (treated as propositional variables) and the logical constant $\top$ (truth) using $\wedge$ and the temporal operators $\bigcirc$ (next time) and $\Diamond$ (sometime in the future). We consider the following classes of queries:

- $\mathcal{Q}[\bigcirc\Diamond]$: all $\bigcirc\Diamond$-*queries*;
- $\mathcal{Q}[\Diamond]$: all $\Diamond$-*queries* (not containing $\bigcirc$);
- $\mathcal{Q}_p[\bigcirc\Diamond]$: *path* $\bigcirc\Diamond$-*queries* of the form (1), where the conjunctions of atoms ρ_i are often treated as sets and the empty conjunction as $\top$;
- $\mathcal{Q}_p[\Diamond]$: all *path* $\Diamond$-*queries* (not containing $\bigcirc$);
- $\mathcal{Q}_{\boldsymbol{in}}$: *interval-queries* of the form (1) with $\rho_0 = \top$, $\rho_1 \neq \top$, $\boldsymbol{o}_1 = \Diamond$, and $\boldsymbol{o}_i = \bigcirc$, for $i > 1$.

Queries in $\mathcal{Q}_{in}$ single out an interval of a fixed length starting at some time-point ≥ 1; $\boldsymbol{q}_1$–$\boldsymbol{q}_3$ from Example 1 are in $\mathcal{Q}_{in}$. $\mathcal{Q}^\sigma$ is the restriction of a class $\mathcal{Q}$ to a signature σ. The *temporal depth* $tdp(\boldsymbol{q})$ of $\boldsymbol{q}$ is the maximum number of nested temporal operators in $\boldsymbol{q}$. The *signature* $\mathsf{sig}(\boldsymbol{q})$ of $\boldsymbol{q}$ is the set of atoms in $\boldsymbol{q}$; the *size* $|\boldsymbol{q}|$ of $\boldsymbol{q}$ is the number of symbols in it.

The *truth-relation* $\mathcal{D}, n \models \boldsymbol{q}$—saying that $\boldsymbol{q}$ is true in $\mathcal{D}$ at moment $n \in \mathbb{N}$—is defined as usual in temporal logic under the *strict semantics*: $\mathcal{D}, n \models \top$ for all $n \in \mathbb{N}$; $\mathcal{D}, n \models A$ iff $A(n) \in \mathcal{D}$; $\mathcal{D}, n \models \bigcirc \boldsymbol{q}'$ iff $\mathcal{D}, n+1 \models \boldsymbol{q}'$; and $\mathcal{D}, n \models \Diamond \boldsymbol{q}'$ iff $\mathcal{D}, m \models \boldsymbol{q}'$, for some $m > n$. A data instance $\mathcal{D}$ is called a *positive example* for a query $\boldsymbol{q}$ if $\mathcal{D}, 0 \models \boldsymbol{q}$; otherwise, $\mathcal{D}$ is a *negative example* for $\boldsymbol{q}$. Checking whether $\mathcal{D}$ is a positive

(negative) example for our queries $\boldsymbol{q}$ can obviously be done in polytime in $|\mathcal{D}|$ and $|\boldsymbol{q}|$.

We write $\boldsymbol{q} \models \boldsymbol{q}'$ if $\mathcal{D}, 0 \models \boldsymbol{q}$ implies $\mathcal{D}, 0 \models \boldsymbol{q}'$ for all instances $\mathcal{D}$, and $\boldsymbol{q} \equiv \boldsymbol{q}'$ if $\boldsymbol{q} \models \boldsymbol{q}'$ and $\boldsymbol{q}' \models \boldsymbol{q}$, in which case $\boldsymbol{q}$ and $\boldsymbol{q}'$ are *equivalent*. For example, for any query $\boldsymbol{q}$, we have $\Diamond\bigcirc\boldsymbol{q} \equiv \bigcirc\Diamond\boldsymbol{q} \equiv \Diamond\Diamond\boldsymbol{q}$. It follows that every path query in $\mathcal{Q}_p[\bigcirc\Diamond]$ is equivalent to a query of the form (1), in which $\rho_n \neq \top$ and whenever $\rho_i = \top$, $0 < i < n$, then $\boldsymbol{o}_i = \boldsymbol{o}_{i+1}$; in this case we say that $\boldsymbol{q}$ is in *normal form*. Unless indicated otherwise, we assume all path queries to be in normal form.

Sequences. There is a close link between evaluating path queries and algorithms for finding patterns in strings [Crochemore *et al.*, 2007]. A *sequence* is a data instance $\mathcal{D} = \delta_0 \dots \delta_n$ with $\delta_0 = \emptyset$ and $|\delta_i| = 1$, for $i > 0$; a *sequence query* is a path query of the form (1) with $|\rho_0| = 0$ and $|\rho_i| = 1$, for $i > 0$. Querying sequences using sequence queries corresponds to the following matching problems:

- for any sequence query $\boldsymbol{q} \in \mathcal{Q}_p[\Diamond]$ of the form (1), we have $\mathcal{D}, 0 \models \boldsymbol{q}$ iff $\rho_1 \dots \rho_m$ is a *subsequence* of $\mathcal{D}$;
- for any sequence query $\boldsymbol{q} \in \mathcal{Q}_{in}$ of the form (1), we have $\mathcal{D}, 0 \models \boldsymbol{q}$ iff $\rho_1 \dots \rho_m$ is a *subword* of $\mathcal{D}$.

3 Query Containment

The *query containment problem* for a class $\mathcal{Q}$ of queries is to decide whether $\boldsymbol{q} \models \boldsymbol{q}'$, for any given $\boldsymbol{q}, \boldsymbol{q}' \in \mathcal{Q}$. In contrast to conjunctive queries, where query containment is NP-complete [Chandra and Merlin, 1977], query containment is tractable for the majority of query classes defined above:

Theorem 2. *The query containment problems for $\mathcal{Q}_p[\bigcirc\Diamond]$, $\mathcal{Q}[\Diamond]$ (and their subclasses) are all in* P.

To prove Theorem 2, suppose first that we are given two queries $\boldsymbol{q}, \boldsymbol{q}' \in \mathcal{Q}_p[\bigcirc\Diamond]$, where $\boldsymbol{q}$ takes of the form (1) and $\boldsymbol{q}' = \rho'_0 \wedge \boldsymbol{o}'_1(\rho'_1 \wedge \dots \wedge \boldsymbol{o}'_m\rho'_m)$. Denote by $[m]$ the closed interval $[0, m] \subseteq \mathbb{N}$. A function $h\colon [m] \to [n]$ is *monotone* if $h(i) < h(j)$ whenever $i < j$. Then tractability of containment for path queries follows from the criterion below, which is proved in the appendix by induction on $tdp(\boldsymbol{q}')$:

Lemma 3. *Let $\boldsymbol{q}, \boldsymbol{q}' \in \mathcal{Q}_p[\bigcirc\Diamond]$. Then $\boldsymbol{q} \models \boldsymbol{q}'$ iff there is a monotone function $h\colon [m] \to [n]$ such that $h(0) = 0$ and, for all $i \in [m]$, we have $\rho'_i \subseteq \rho_{h(i)}$ and if $\boldsymbol{o}'_{i+1} = \bigcirc$, then $\boldsymbol{o}_{h(i+1)} = \bigcirc$ and $h(i+1) = h(i) + 1$.*

We refer to any function h defined in Lemma 3 as a *containment witness* for the pair $\boldsymbol{q}, \boldsymbol{q}' \in \mathcal{Q}_p[\bigcirc\Diamond]$.

Suppose now $\boldsymbol{q} \in \mathcal{Q}[\Diamond]$. As shown in [Fortin *et al.*, 2022], we can convert $\boldsymbol{q}$ in polytime to an equivalent query in the *normal form* $\rho \wedge \boldsymbol{q}_1 \wedge \dots \wedge \boldsymbol{q}_n$, where ρ is a conjunction of atoms and each $\boldsymbol{q}_i$ is in $\mathcal{Q}_p[\Diamond]$ and starts with $\Diamond$. Tractability of containment for $\mathcal{Q}[\Diamond]$-queries follows from:

Lemma 4. *If $\boldsymbol{q} = \rho \wedge \boldsymbol{q}_1 \wedge \dots \wedge \boldsymbol{q}_n \in \mathcal{Q}[\Diamond]$ is in normal form, $\boldsymbol{q}' \in \mathcal{Q}_p[\Diamond]$ and $\boldsymbol{q} \models \boldsymbol{q}'$, then there is $\boldsymbol{q}_i$, $1 \leq i \leq n$, with $\rho \wedge \boldsymbol{q}_i \models \boldsymbol{q}'$.*

Proof. In the detailed proof given in the appendix, we show that, assuming $\rho \wedge \boldsymbol{q}_i \not\models \boldsymbol{q}'$ for all i, we can convert the $\boldsymbol{q}_i$ into a data instance $\mathcal{D}$ with $\mathcal{D}, 0 \models \boldsymbol{q}$ and $\mathcal{D}, 0 \not\models \boldsymbol{q}'$. ⊣

For queries $\boldsymbol{q} \in \mathcal{Q}[\bigcirc\Diamond]$, Lemma 4 does not hold:

Example 5. Let $\boldsymbol{q} = \boldsymbol{q}_1 \wedge \boldsymbol{q}_2 \wedge \boldsymbol{q}_3 \wedge \boldsymbol{q}_4$, where

$$\begin{aligned}
\boldsymbol{q}_1 &= \Diamond(a \wedge \bigcirc((a \wedge b) \wedge \bigcirc a)),\\
\boldsymbol{q}_2 &= \Diamond(b \wedge \bigcirc((a \wedge b) \wedge \bigcirc b)),\\
\boldsymbol{q}_3 &= \Diamond(a \wedge \bigcirc((a \wedge b) \wedge \bigcirc b)),\\
\boldsymbol{q}_4 &= \Diamond(a \wedge \Diamond(b \wedge \Diamond(a \wedge \Diamond(b \wedge \Diamond(a \wedge \Diamond b))))),
\end{aligned}$$

and $\boldsymbol{q}' = \Diamond(b \wedge \Diamond((a \wedge b) \wedge \Diamond a))$. Then $\boldsymbol{q} \models \boldsymbol{q}'$ but $\boldsymbol{q}_i \not\models \boldsymbol{q}'$, for any i, $1 \leq i \leq 4$.

At the moment, the question whether containment of $\mathcal{Q}[\bigcirc\Diamond]$-queries is tractable remains open.

4 Example Sets and Separating Queries

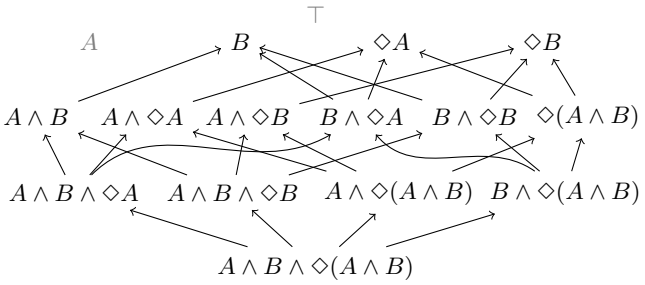
An *example set* E is a pair (E^+, E^-) of finite sets $E^+ \neq \emptyset$ and E^- of data instances. We say that a query $\boldsymbol{q}$ *separates* E if all $\mathcal{D} \in E^+$ are positive examples and all $\mathcal{D} \in E^-$ are negative examples for $\boldsymbol{q}$. Denote by $\mathsf{s}(E, \mathcal{Q})$ the set of queries in a class $\mathcal{Q}$ separating E and call E *$\mathcal{Q}$-separable* if $\mathsf{s}(E, \mathcal{Q}) \neq \emptyset$. Our general aim is to understand the structure of $\mathsf{s}(E, \mathcal{Q})$ for various important query classes $\mathcal{Q}$.

We consider queries in $\mathcal{Q}$ *modulo equivalence*, not distinguishing between $\boldsymbol{q} \equiv \boldsymbol{q}'$. In this case, the relation $\models$ is a partial order on $\mathcal{Q}$. For any $\boldsymbol{q} \in \mathsf{s}(E, \mathcal{Q})$, we clearly have

- $tdp(\boldsymbol{q}) \leq \min\{\max \mathcal{D} \mid \mathcal{D} \in E^+\}$,
- $\mathsf{sig}(\boldsymbol{q}) \subseteq \bigcap\{\mathsf{sig}(\mathcal{D}) \mid \mathcal{D} \in E^+\}$,

so $\mathsf{s}(E, \mathcal{Q})$ is finite. We refer to the $\models$-minimal elements of $\mathsf{s}(E, \mathcal{Q}) \neq \emptyset$ as *most specific $\mathcal{Q}$-separators of E* and to the $\models$-maximal elements as *most general $\mathcal{Q}$-separators of E* (modulo $\equiv$); they comprise the sets $\mathsf{mss}(E, \mathcal{Q})$ and $\mathsf{mgs}(E, \mathcal{Q})$, respectively. If these sets are singletons, we call their only element the *unique most specific* and, respectively, *unique most general $\mathcal{Q}$-separator of E*. Note that the former always exists if $\mathsf{s}(E, \mathcal{Q}) \neq \emptyset$ and $\mathcal{Q}$ is closed under $\wedge$.

Example 6. (i) Suppose $\mathcal{D}^+ = \{A(0), B(0), A(1), B(1)\}$, $\mathcal{D}^- = \{A(0)\}$ and $E = (\{\mathcal{D}^+\}, \{\mathcal{D}^-\})$. The separator space $\mathsf{s}(E, \mathcal{Q}_p[\Diamond])$ is shown below as a Hasse diagram with arrows indicating the partial order $\models$ (and $\top, A \notin \mathsf{s}(E, \mathcal{Q}_p[\Diamond])$), so



the most general $\mathcal{Q}_p[\Diamond]$-separators of E comprise the set $\mathsf{mgs}(E, \mathcal{Q}_p[\Diamond]) = \{B, \Diamond A, \Diamond B\}$ and the unique most specific one is $A \wedge B \wedge \Diamond(A \wedge B)$.

(ii) For the example set $E = (\{\mathcal{D}_1^+, \mathcal{D}_2^+\}, \{\mathcal{D}_1^-\})$ with

$$\mathcal{D}_1^+ = \{A(1), B(2)\}, \mathcal{D}_2^+ = \{B(1), A(2)\}, \mathcal{D}_1^- = \{C(0)\},$$

$\mathsf{s}(E, \mathcal{Q}_p[\Diamond]) = \{\Diamond A, \Diamond B\} = \mathsf{mss}/\mathsf{mgs}(E, \mathcal{Q}_p[\Diamond])$ but there is no unique most specific/general separator. In contrast, $\mathsf{s}(E, \mathcal{Q}[\Diamond]) = \{\Diamond A, \Diamond B, \Diamond A \wedge \Diamond B\}$ has the unique most specific separator $\Diamond A \wedge \Diamond B$ but no unique most general one.

(iii) One can show that $\mathsf{mss}/\mathsf{mgs}(E, \mathcal{Q})$ always contains a *longest/shortest separator* of E in $\mathcal{Q}$ (of largest/smallest temporal depth). To illustrate, let $E = (\{\mathcal{D}_1^+\}, \{\mathcal{D}_1^-, \mathcal{D}_2^-, \mathcal{D}_3^-\})$,

$$\mathcal{D}_1^+ = \{B(0), C(0), A(1)\},$$
$$\mathcal{D}_1^- = \{B(0)\},\ \mathcal{D}_2^- = \{C(0)\},\ \mathcal{D}_3^- = \{A(1)\}.$$

Then $\mathsf{mgs}(E, \mathcal{Q}_p[\Diamond]) = \{B \land C, B \land \Diamond A, C \land \Diamond A\}$, where $B \land C$ of depth 0 is the shortest separator of E in $\mathcal{Q}_p[\Diamond]$.

Our main concern is the following three algorithmic problems for query classes $\mathcal{Q} \subseteq \mathcal{Q}[\bigcirc\Diamond]$ with input E and $\boldsymbol{q} \in \mathcal{Q}$:

most specific/general separator verification: decide whether $\boldsymbol{q}$ is an element of $\mathsf{mss}(E, \mathcal{Q})$ / $\mathsf{mgs}(E, \mathcal{Q})$;

counting most specific/general separators: count the elements of $\mathsf{mss}(E, \mathcal{Q})$ / $\mathsf{mgs}(E, \mathcal{Q})$;

computing a most specific/general separator: construct some query in $\mathsf{mss}(E, \mathcal{Q})$ / $\mathsf{mgs}(E, \mathcal{Q})$.

We are particularly interested in deciding whether there is a unique most specific/general separator and computing it. To achieve our aims, we obviously should be able to decide whether $\boldsymbol{q} \in \mathsf{s}(E, \mathcal{Q})$ (*separator verification*) and whether $\mathsf{s}(E, \mathcal{Q}) \neq \emptyset$ (*separator existence*). As mentioned in Section 2, separator verification is in P. We are also interested in the case when the number of positive or negative examples in E is bounded. The table below summarises the complexities of separator existence for our query classes, where b^+ / b^- means that $|E^+|$ / $|E^-|$ is bounded, and one can assume a bounded signature. Except for $\mathcal{Q}_{in}$, these results are shown in [Fortin *et al.*, 2023]. The proofs use techniques developed for the *longest common subsequence problem* (given $k > 0$ and a set E^+ of sequences, is there a sequence query in $\mathcal{Q}_p[\Diamond]$ of depth $\geq k$ that is a subsequence of all $\mathcal{D} \in E^+$; see Section 2) and separator existence for sequence queries in $\mathcal{Q}_p[\Diamond]$ [Maier, 1978; Fraser, 1996]. The NP-upper bound for $\mathcal{Q}_{in}$ is trivial; the lower one follows from the proof of Theorem 21, and tractability for b^+ is ensured by the observation that there are polynomially-many relevant intervals in the E^+-examples.

separator existence	$\mathsf{b}^+, \mathsf{b}^-$	b^+	b^- or unbounded
$\mathcal{Q}_p[\Diamond]/\mathcal{Q}_p[\bigcirc\Diamond]$	in P	NP-c	NP-c
$\mathcal{Q}[\Diamond]/\mathcal{Q}[\bigcirc\Diamond]/\mathcal{Q}_{in}$	in P	in P	NP-c

This NP-lower bound shows that great care is needed when transferring techniques from the literature on algorithms for sequences to our framework as separability of example sets of sequences using sequence queries in $\mathcal{Q}_{in}$ is easily seen to be in P even for unbounded example sets and signatures.

A key to the extremal separator problems above is the following notions of strengthening and weakening frontiers.

5 Strengthening and Weakening Frontiers

Let $\mathcal{Q}$ be a class of queries and $\boldsymbol{q} \in \mathcal{Q}$. A set $\mathcal{F} \subseteq \mathcal{Q}$ is called a *strengthening frontier* for $\boldsymbol{q}$ in $\mathcal{Q}$ if

- for any $\boldsymbol{q}' \in \mathcal{F}$, we have $\boldsymbol{q}' \models \boldsymbol{q}$ and $\boldsymbol{q}' \not\equiv \boldsymbol{q}$;
- for any $\boldsymbol{q}'' \in \mathcal{Q}$, if $\boldsymbol{q}'' \models \boldsymbol{q}$ and $\boldsymbol{q}'' \not\equiv \boldsymbol{q}$, then there is $\boldsymbol{q}' \in \mathcal{F}$ such that $\boldsymbol{q}'' \models \boldsymbol{q}'$.

A set $\mathcal{F} \subseteq \mathcal{Q}$ is called a *weakening frontier* for $\boldsymbol{q}$ in $\mathcal{Q}$ if

- for any $\boldsymbol{q}' \in \mathcal{F}$, we have $\boldsymbol{q} \models \boldsymbol{q}'$ and $\boldsymbol{q}' \not\equiv \boldsymbol{q}$;
- for any $\boldsymbol{q}'' \in \mathcal{Q}$, if $\boldsymbol{q} \models \boldsymbol{q}''$ and $\boldsymbol{q}'' \not\equiv \boldsymbol{q}$, then there is $\boldsymbol{q}' \in \mathcal{F}$ with $\boldsymbol{q}' \models \boldsymbol{q}''$.

Trivial strengthening/weakening frontiers for $\boldsymbol{q}$ comprise all queries that are properly stronger/weaker than $\boldsymbol{q}$ in $\mathcal{Q}$; our concern, however, is finding small frontiers. We show now that, for $\mathcal{Q}_p^\sigma[\Diamond]$ and $\mathcal{Q}_p^\sigma[\bigcirc\Diamond]$, one can compute a strengthening/weakening frontier for any given $\boldsymbol{q}$ in polytime.

Theorem 7. *Let $\boldsymbol{q} \in \mathcal{Q}_p^\sigma[\bigcirc\Diamond]$ be in normal form* (1)*. A strengthening frontier for $\boldsymbol{q}$ in $\mathcal{Q}_p^\sigma[\bigcirc\Diamond]$ can be computed in polytime by applying once to $\boldsymbol{q}$ one of the following operations, for $i \in [n]$ and $\boldsymbol{q}_i = \rho_i \land \boldsymbol{o}_{i+1}(\rho_{i+1} \land \cdots \land \boldsymbol{o}_n\rho_n)$:*

1. *extend some ρ_i in $\boldsymbol{q}$ by some $A \in \sigma \setminus \rho_i$;*
2. *replace some $\Diamond\boldsymbol{q}_i$ in $\boldsymbol{q}$ by $\Diamond(\top \land \Diamond\boldsymbol{q}_i)$;*
3. *replace some $\boldsymbol{o}_i = \Diamond$ in $\boldsymbol{q}$ by $\bigcirc$ provided that the resulting query is in normal form;*
4. *add $\boldsymbol{o}_{n+1}\rho_{n+1}$ at the end of $\boldsymbol{q}$, where $\boldsymbol{o}_{n+1} = \Diamond$ and $\rho_{n+1} = A$, for some $A \in \sigma$.*

If $\boldsymbol{q} \in \mathcal{Q}_p^\sigma[\Diamond]$, a strengthening frontier for $\boldsymbol{q}$ in $\mathcal{Q}_p^\sigma[\Diamond]$ can be computed in polytime using operations 1, 2, and 4.

Proof. Let $\mathcal{F}$ be the set of queries obtained by a single application of one of these operations to $\boldsymbol{q}$. By Lemma 3, $\boldsymbol{q}' \models \boldsymbol{q}$ and $\boldsymbol{q} \not\equiv \boldsymbol{q}'$ for all $\boldsymbol{q}' \in \mathcal{F}$. Let $\boldsymbol{q}' \models \boldsymbol{q}$ and $\boldsymbol{q} \not\equiv \boldsymbol{q}'$, for some $\boldsymbol{q}' \in \mathcal{Q}_p^\sigma[\bigcirc\Diamond]$ of the form $\boldsymbol{q}' = \rho_0' \land \boldsymbol{o}_1'(\rho_1' \land \cdots \land \boldsymbol{o}_m'(\rho_m'))$. Take a containment witness $h\colon [n] \to [m]$ for $\boldsymbol{q}', \boldsymbol{q}$. If h is surjective, then $n = m$ and $h(i) = i$ for all $i \in [n]$, and so $\rho_i \subseteq \rho_i'$. As $\boldsymbol{q} \not\equiv \boldsymbol{q}'$, either $\boldsymbol{o}_i' = \bigcirc$ and $\boldsymbol{o}_i = \Diamond$, for some $i \in [n]$, or $\rho_i \subsetneq \rho_i'$, for some $i \in [n]$. In the former case, operation 3 gives $\boldsymbol{q}'' \in \mathcal{F}$ with $\boldsymbol{q}' \models \boldsymbol{q}''$; in the latter one, operation 1 gives such a $\boldsymbol{q}''$.

Suppose h is not surjective. If there is $i < n$ such that $h(i+1) - h(i) \geq 2$, then $\boldsymbol{q}' \models \boldsymbol{q}''$ for a $\boldsymbol{q}'' \in \mathcal{F}$ given by operation 2. Otherwise m is not in the range of h, and we get such a $\boldsymbol{q}'' \in \mathcal{F}$ by operation 4. ⊣

Example 8. For $\sigma = \{A, B\}$ and $\boldsymbol{q} = \Diamond(A \land \bigcirc B)$, operations 1–4 give the following strengthening frontier for $\boldsymbol{q}$:

$$\Diamond(A \land B \land \bigcirc B), \Diamond(A \land \bigcirc(A \land B)), \Diamond(\top \land \Diamond(A \land \bigcirc B)),$$
$$\bigcirc(A \land \bigcirc B), \Diamond(A \land \bigcirc(B \land \Diamond A)), \Diamond(A \land \bigcirc(B \land \Diamond B)).$$

A weakening frontier can be constructed by reversing operations 1–3 from Theorem 7 and using a similar argument:

Theorem 9. *Let $\boldsymbol{q} \in \mathcal{Q}_p^\sigma[\bigcirc\Diamond]$ be in normal form* (1)*. A weakening frontier for $\boldsymbol{q}$ in $\mathcal{Q}_p^\sigma[\bigcirc\Diamond]$ can be computed in polytime by applying once to $\boldsymbol{q}$ one of the following operations, for $i \in [n]$ and $\boldsymbol{q}_i = \rho_i \land \boldsymbol{o}_{i+1}(\rho_{i+1} \land \cdots \land \boldsymbol{o}_n\rho_n)$:*

1. *drop some atom from ρ_i;*
2. *replace some $\Diamond(\top \land \Diamond\boldsymbol{q}_i)$ in $\boldsymbol{q}$ by $\Diamond\boldsymbol{q}_i$;*
3. *replace some $\boldsymbol{o}_i = \bigcirc$ by $\Diamond$.*

If $\boldsymbol{q} \in \mathcal{Q}_p^\sigma[\Diamond]$, a weakening frontier for $\boldsymbol{q}$ in $\mathcal{Q}_p^\sigma[\Diamond]$ can be computed in polytime using operations 1 and 2.

Example 10. For $\sigma = \{A, B\}$ and $\boldsymbol{q} = \Diamond(A \land \bigcirc B)$, operations 1-3 give the following weakening frontier for $\boldsymbol{q}$:

$$\Diamond(\top \land \Diamond B), \quad \Diamond A, \quad \Diamond(A \land \Diamond B).$$

Note that the computed weakening frontier can be made smaller by omitting $\Diamond A$, which is weaker than $\Diamond(A \wedge \Diamond B)$.

We next consider frontiers for queries in $\mathcal{Q}^\sigma[\Diamond]$. We say that $\boldsymbol{q} = \rho \wedge \boldsymbol{q}_1 \wedge \cdots \wedge \boldsymbol{q}_n \in \mathcal{Q}^\sigma[\Diamond]$ in normal form is *redundancy-free* if it does not contain $\boldsymbol{q}_i, \boldsymbol{q}_j$ with $i \neq j$ and $\boldsymbol{q}_i \models \boldsymbol{q}_j$. Clearly, for any $\boldsymbol{q} \in \mathcal{Q}^\sigma[\Diamond]$, we can compute an equivalent redundancy-free $\boldsymbol{q}' \in \mathcal{Q}^\sigma[\Diamond]$ in polytime.

Theorem 11. *Let $\boldsymbol{q} = \rho \wedge \boldsymbol{q}_1 \wedge \cdots \wedge \boldsymbol{q}_n \in \mathcal{Q}^\sigma[\Diamond]$ be redundancy-free. A weakening frontier for $\boldsymbol{q}$ in $\mathcal{Q}^\sigma[\Diamond]$ can be computed in polytime by a single application of one of the following operations to $\boldsymbol{q}$:*

1. *drop some atom from ρ;*
2. *replace some $\boldsymbol{q}_i$ by $\bigwedge \mathcal{F}_i$, where $\mathcal{F}_i$ is the weakening frontier for $\boldsymbol{q}_i$ in $\mathcal{Q}^\sigma_p[\Diamond]$ provided by Theorem 9.*

Proof. Let $\mathcal{F}$ be the set of queries defined above. Clearly, $\boldsymbol{q} \models \boldsymbol{q}'$; as $\boldsymbol{q}$ is redundancy-free and in view of Lemma 4, $\boldsymbol{q} \not\equiv \boldsymbol{q}'$ for all $\boldsymbol{q}' \in \mathcal{F}$. Suppose $\boldsymbol{q} \models \boldsymbol{q}'$ and $\boldsymbol{q} \not\equiv \boldsymbol{q}'$, for some $\boldsymbol{q}' = \rho' \wedge \boldsymbol{q}'_1 \wedge \cdots \wedge \boldsymbol{q}'_m$ in $\mathcal{Q}^\sigma[\Diamond]$. By Lemma 4, $\rho' \subseteq \rho$ and, for each j, $1 \leq j \leq m$, there exists $f(j)$, $1 \leq f(j) \leq n$, with $\boldsymbol{q}_{f(j)} \models \boldsymbol{q}'_j$. If $\rho' \subsetneq \rho$, then operation 1 gives a $\boldsymbol{q}'' \in \mathcal{F}$ with $\boldsymbol{q}'' \models \boldsymbol{q}'$. Otherwise, as $\boldsymbol{q} \not\equiv \boldsymbol{q}'$, there is $\boldsymbol{q}_i$ such that $\boldsymbol{q}'_j \not\models \boldsymbol{q}_i$ for all $\boldsymbol{q}'_j$, $1 \leq j \leq m$. Let $\boldsymbol{q}''$ be the query obtained from $\boldsymbol{q}$ by replacing $\boldsymbol{q}_i$ with $\bigwedge \mathcal{F}_i$ by operation 2. To establish $\boldsymbol{q}'' \models \boldsymbol{q}'$, it suffices to show that $f(j) = i$ implies $\bigwedge \mathcal{F}_i \models \boldsymbol{q}_j$. So suppose $f(j) = i$. Then $\boldsymbol{q}'_j \not\models \boldsymbol{q}_i$, and so, by the definition of $\mathcal{F}_i$, there is $\boldsymbol{q}''_i \in \mathcal{F}_i$ with $\boldsymbol{q}''_i \models \boldsymbol{q}_j$. ⊣

However, strengthening frontiers for queries in $\mathcal{Q}^\sigma[\Diamond]$ are not necessarily of polynomial size as shown by the following:

Example 12. We represent queries in $\mathcal{Q}^\sigma[\Diamond]$ of the form (1) as $\rho_0 \dots \rho_n$. For $\sigma = \{A_1, A_2, B_1, B_2\}$, let $\boldsymbol{q}_1 = \emptyset(\boldsymbol{q}\sigma)^n\boldsymbol{q}$, $\boldsymbol{q}_2 = \emptyset\sigma^{2n+1}$, and $\boldsymbol{q} = \{A_1, A_2\}\{B_1, B_2\}$. Using [Fortin *et al.*, 2022, Example 18], one can show that any strengthening frontier for the query $\boldsymbol{q}_1 \wedge \boldsymbol{q}_2$ in $\mathcal{Q}^\sigma[\Diamond]$ is of size $O(2^n)$.

Note also that weakening frontiers for $\boldsymbol{q} \in \mathcal{Q}^\sigma_{in}$ can be computed in polytime using operation 1 in Theorem 9 (if $i = 1$ and $|\rho_i| = 1$, we drop $\top$ and take $\Diamond(\rho_2 \wedge \bigcirc(\rho_3 \wedge \cdots \wedge \bigcirc\rho_n))$). On the other hand, strengthening frontiers can be infinite:

Example 13. All $\Diamond(A \wedge \bigcirc^n A)$, $n > 0$, are in any strengthening frontier for $\boldsymbol{q} = \Diamond A$ in $\mathcal{Q}^\sigma_{in}$, where $\sigma = \{A\}$.

As shown by Theorem 17, the lack of polytime computable strengthening frontiers in $\mathcal{Q}[\Diamond]$ affects the complexity of verifying most specific separators. In contrast, the lack of polytime computable strengthening frontiers in $\mathcal{Q}_{in}$ turns out to be harmless. For $n \geq tdp(\boldsymbol{q})$, call $\mathcal{F} \subseteq \mathcal{Q}$ an *n-bounded weakening/strengthening frontier* for $\boldsymbol{q}$ in $\mathcal{Q}$ if $\mathcal{F}$ is a weakening/strengthening frontier for $\boldsymbol{q}$ in the class $\{\boldsymbol{q}' \in \mathcal{Q} \mid tdp(\boldsymbol{q}') \leq n\}$ (we assume n to be given in unary).

Theorem 14. *Weakening frontiers and n-bounded strengthening frontiers in $\mathcal{Q}^\sigma_{in}$ can be computed in polytime.*

Finally, we observe that Theorems 7 and 9 give an alternative way of computing unique characterisations of queries in $\mathcal{Q}^\sigma_p[\bigcirc\Diamond]$ by data examples in polynomial time compared to [Fortin *et al.*, 2022], which opens another route to studying unique characterisations and exact learning of temporal queries.

6 Complexity

Now we show complexity bounds for the decision and counting problems from Section 4, starting with verification and observing that polytime computable n-bounded frontiers imply tractable verification of most specific/general separators:

Lemma 15. *If an n-bounded strengthening/weakening frontier for $\boldsymbol{q} \in \mathcal{Q}$ is polytime computable in $|\boldsymbol{q}|$ and n, then most specific/general separator verification for $\mathcal{Q}$ is in* P.

Proof. Let $n = \min\{\max \mathcal{D} \mid \mathcal{D} \in E^+\}$. We compute an n-bounded strengthening frontier $\mathcal{F}$ for $\boldsymbol{q}$ in $\mathcal{Q}$ and use that $\boldsymbol{q} \in \mathsf{mss}(E, \mathcal{Q})$ iff $\boldsymbol{q} \in \mathsf{s}(E, \mathcal{Q})$ and $\boldsymbol{q}' \notin \mathsf{s}(E, \mathcal{Q})$ for all $\boldsymbol{q}' \in \mathcal{F}$. The case of $\mathsf{mgs}(E, \mathcal{Q})$ is similar. ⊣

Corollary 16. *Most specific/general separator verification is in* P *for $\mathcal{Q}_p[\Diamond]$, $\mathcal{Q}_p[\bigcirc\Diamond]$, and $\mathcal{Q}_{in}$. Most general separator verification is in* P *for $\mathcal{Q}[\Diamond]$.*

Proof. It follows from Lemma 15 and Theorems 7, 9, 14 that most specific/general separator verification is in P for $\mathcal{Q}_p[\Diamond]$, $\mathcal{Q}_p[\bigcirc\Diamond]$, and $\mathcal{Q}_{in}$. By Theorem 11, most general separation verification is also in P for $\mathcal{Q}[\Diamond]$. ⊣

Most specific separator verification is harder for $\mathcal{Q}[\Diamond]$:

Theorem 17. *For $\mathcal{Q}[\Diamond]$, the most specific separator verification problem coincides with the unique most specific separator verification problem and is* coNP*-complete.*

Proof. As we know, E has a unique most specific separator in $\mathcal{Q}[\Diamond]$ iff $\mathsf{s}(E, \mathcal{Q}[\Diamond]) \neq \emptyset$. Hence most specific separator verification coincides with unique most specific separator verification. Given $\boldsymbol{q}$ and E, we can check in NP that either $\boldsymbol{q} \notin \mathsf{s}(E, \mathcal{Q}[\Diamond])$ (which is in P) or that there is $\boldsymbol{q}' \in \mathsf{s}(E, \mathcal{Q}[\Diamond])$ with $\boldsymbol{q}' \models \boldsymbol{q}$ and $\boldsymbol{q}' \not\equiv \boldsymbol{q}$ (which is in NP). This gives the coNP upper bound. The more involved proof of the lower one (in the appendix) is based on ideas similar to those in the proof of Theorem 21 below. ⊣

In the cases when most specific/general separators are not necessarily unique, we obtain the following:

Theorem 18. *Unique most specific/general separator verification in $\mathcal{Q}_p[\bigcirc\Diamond]$, $\mathcal{Q}_p[\Diamond]$, $\mathcal{Q}_{in}$ as well as unique most general separator verification in $\mathcal{Q}[\Diamond]$ are* coNP*-complete.*

Proof. The upper bounds follow from Corollary 16. The lower ones are by reduction of the problem to decide whether, for a Boolean formula φ and a satisfying assignment $\mathfrak{a}$, there is a satisfying assignment for φ different from $\mathfrak{a}$. We use ideas similar to those in the proof of Theorem 21. ⊣

We now turn to the existence and counting problems. Recall that UNIQUE SAT is the problem to decide whether there is exactly one satisfying assignment for a propositional formula [Blass and Gurevich, 1982]. It is in Δ^p_2, coNP-hard, and complete for the class US of problems solvable by a non-deterministic polynomial-time Turing machine M that accepts iff M has exactly one accepting path. A counting problem is in the class ♯P if there is such M whose number of accepting paths coincides with the number of solutions to the problem [Arora and Barak, 2009].

Theorem 19. *For $\mathcal{Q}_p[\bigcirc\Diamond]$, $\mathcal{Q}_p[\Diamond]$, $\mathcal{Q}_{in}$, counting most specific/general separators is $\sharp P$-complete; the existence of a unique most specific/general separator is* US*-complete. The same holds for most general separators in $\mathcal{Q}[\Diamond]$.*

Theorem 19 follows from the very general and fine-grained complexity results provided by Theorems 20 and 21 below.

Theorem 20. *Let $\mathcal{Q} \subseteq \mathcal{Q}_p[\bigcirc\Diamond]$ be a class with polytime decidable membership and polytime computable n-bounded strengthening/weakening frontiers for queries in $\mathcal{Q}$. Then*

- *counting most specific/general $\mathcal{Q}$-separators is in $\sharp P$;*
- *the existence of a unique most specific/general $\mathcal{Q}$-separator is in* US.

This also holds for most general separators in classes of conjunctions of $\bigcirc\Diamond$-path queries closed under dropping conjuncts and having polytime computable weakening frontiers.

Proof. Given E, construct a TM M that guesses a $\boldsymbol{q} \in \mathcal{Q}$ with $tdp(\boldsymbol{q}) \leq n = \min\{\max \mathcal{D} \mid \mathcal{D} \in E^+\}$ and accepts if $\boldsymbol{q}$ separates E and no $\boldsymbol{q}'$ in the n-bounded frontier for $\boldsymbol{q}$ in $\mathcal{Q}$ separates E. As we know, the required checks are in P. In the second claim, M guesses a query $\boldsymbol{q} = \rho \wedge \Diamond \boldsymbol{q}_1 \wedge \cdots \wedge \Diamond \boldsymbol{q}_l$ with $\boldsymbol{q}_i \in \mathcal{Q}_p[\bigcirc\Diamond]$ and $l \leq |E^-|$. ⊣

For the lower bounds, we take into account the boundedness of $\mathsf{sig}(E)$ and the cardinalities $|E^+|$ and $|E^-|$ of positive and negative examples in E. The next result provides matching lower bounds for Theorem 19 even if the signature is bounded and only one of $|E^+|$ and $|E^-|$ is unbounded, except for $\mathcal{Q}_{in}$ and $\mathcal{Q}[\Diamond]$, where $|E^+|$ has to be unbounded.

Theorem 21. *Let $\mathcal{Q} \subseteq \mathcal{Q}_p[\bigcirc\Diamond]$ be any class of queries containing all $\Diamond\rho$ with a conjunction of atoms ρ. Then*

- *counting most specific/general $\mathcal{Q}$-separators is* $\sharp$P*-hard*;
- *the existence of unique most specific/general separator is* US*-hard*

even for $E = (E^+, E^-)$ and $\sigma = \mathsf{sig}(E)$ with (a) $|E^-| \leq 1$ or (b) $|E^-| \leq 1$, $|\sigma| = 2$ and $\mathcal{Q} = \mathcal{Q}_{in}$ or $\mathcal{Q} \supseteq \mathcal{Q}_p[\Diamond]$, or (c) $|E^+| \leq 4$, $|\sigma| = 3$, $\mathcal{Q} \supseteq \mathcal{Q}_p[\Diamond]$. This result also holds for most general separators in $\mathcal{Q}[\Diamond]$ and $|E^-| = 1$, $|\sigma| = 2$.

Proof. We sketch the proof of the first claim by a parsimonious reduction from SAT for an unbounded number of negative examples (which can be easily merged into one). Take a CNF $\varphi = \psi_1 \wedge \cdots \wedge \psi_k$ with clauses ψ_i over variables $x_1, \ldots, x_n$. We construct $E = (E^+, E^-)$ such that there is a bijection between the satisfying assignments for φ and the separators for E in $\mathcal{Q}[\bigcirc\Diamond]$ (even queries of the form $\Diamond\rho$). The claim follows from the fact that the separating queries are mutually $\models$-incomparable. Define the positive examples $E^+ = \{\mathcal{D}_0, \mathcal{D}'_0, \mathcal{D}_1, \ldots, \mathcal{D}_n\}$ with $2n$ atoms $A_1, \bar{A}_1, \ldots, A_n, \bar{A}_n$ by taking

$$\mathcal{D}_0 = \{A_i(1), \bar{A}_i(1) \mid 1 \leq i \leq n\},$$
$$\mathcal{D}'_0 = \{A_i(2), \bar{A}_i(2) \mid 1 \leq i \leq n\},$$
$$\mathcal{D}_i = \{A_i(1), \bar{A}_i(2), A_j(1), \bar{A}_j(1), A_j(2), \bar{A}_j(2) \mid i \neq j\},$$

for $1 \leq i \leq n$. Let $E^- = \{\mathcal{D}^1_1, \ldots, \mathcal{D}^1_k, \mathcal{D}^2_1, \ldots, \mathcal{D}^2_n\}$, where $\mathcal{D}^1_i$, $1 \leq i \leq k$, comprises $\bar{A}_j(1)$ if x_j does not occur negatively in ψ_i, and $A_j(1)$ if x_j does not occur positively in ψ_i, and $\mathcal{D}^2_i = \{A_j(1), \bar{A}_j(1) \mid j \neq i\}$. For an assignment $\mathfrak{a}$ for $x_1, \ldots, x_n$, let $\rho_\mathfrak{a}$ contain A_i if $\mathfrak{a}(x_i) = 1$ and $\bar{A}_i$, otherwise. Now our claim follows from the following: (i) if $\mathfrak{a}$ satisfies φ, then $\Diamond\rho_\mathfrak{a}$ separates E; (ii) if $\boldsymbol{q} \in \mathcal{Q}_p[\bigcirc\Diamond]$ separates E, then $\boldsymbol{q} \equiv \Diamond\rho_\mathfrak{a}$, for some $\mathfrak{a}$ satisfying φ.

To bound σ and/or E^+, we give parsimonious reductions from SAT and employ techniques for the longest common subsequence problem and separator existence for sequence queries in $\mathcal{Q}_p[\Diamond]$ [Blin *et al.*, 2012; Fraser, 1996]. ⊣

Again, re-using techniques from the literature on algorithms for sequences needs some care. Similar to separability, for example sets of sequences, counting most general/specific sequence $\mathcal{Q}_{in}$-separators is easily seen to be in P even for unbounded signatures and example sets, but Theorem 21 shows that this is not so for $\mathcal{Q}_{in}$ on non-sequence data instances.

Surprisingly, the complexities of counting most general separators and deciding the existence of a unique one diverge if we bound the number of examples, cf. Theorems 23 and 27.

Theorem 22. *Let $\mathcal{Q} = \mathcal{Q}[\Diamond]$ or $\mathcal{Q} \subseteq \mathcal{Q}_p[\bigcirc\Diamond]$ be any class containing all $\Diamond\rho$ with a conjunction of atoms ρ. Then counting most general $\mathcal{Q}$-separators is* $\sharp$P*-hard even for example sets $E = (E^+, E^-)$ with $|E^+| = 2$ and $|E^-| = 1$.*

Proof. The proof is by reduction from counting satisfying assignments for monotone formulas: E^- is as in the proof of Theorem 21 and the positive examples are $\mathcal{D}_0, \mathcal{D}'_0$. ⊣

Theorem 22 does not hold for counting most specific separators in $\mathcal{Q}_{in}$, which is easily seen to be in P if $|E^+|$ is bounded. It remains open whether some version of this theorem holds for most specific separators in $\mathcal{Q}_p[\bigcirc\Diamond]$ or $\mathcal{Q}_p[\Diamond]$.

7 Algorithms

The frontiers defined in Section 5 give a polytime algorithm for computing a most specific/general separator starting from *any* given separator. Suppose $\mathcal{Q} \in \{\mathcal{Q}_p[\Diamond], \mathcal{Q}_p[\bigcirc\Diamond], \mathcal{Q}_{in}\}$, we are given $\boldsymbol{q} \in \mathsf{s}(E, \mathcal{Q})$ and need a most specific separator. By Theorems 7, 9, the length of the longest $\models$-chain in $\mathsf{s}(E, \mathcal{Q}_p[\bigcirc\Diamond])$ is polynomial in $|E|$.[1] We take, if possible, some $\boldsymbol{q}' \in \mathsf{s}(E, \mathcal{Q})$ in the strengthening frontier for $\boldsymbol{q}$, then $\boldsymbol{q}'' \in \mathsf{s}(E, \mathcal{Q})$ in the strengthening frontier for $\boldsymbol{q}'$, etc. This process terminates after polynomially-many steps, returning a most specific $\mathcal{Q}$-separator (and so the unique one, if any).

Thus, we can focus on algorithms deciding the existence of a (unique most specific/general) separator and constructing it. In the next theorem, we compute not just a random input separator, but a longest/shortest one. As strengthening/weakening frontiers contain queries that are not shorter/longer than the input, the procedure above will compute a longest most specific/shortest most general separator.

Theorem 23. *Let $E = (E^+, E^-)$, $\sigma = \mathsf{sig}(E)$, t_+/t_- be the maximum timestamp in E^+/E^-, $c_+ = |E^+|$, $c_- = |E^-|$, and $\mathcal{Q} \in \{\mathcal{Q}_p[\Diamond], \mathcal{Q}_p[\bigcirc\Diamond]\}$. The following can be done in time $O(t_+^{c_+} t_-^{c_-})$:*

(a) *deciding whether $\mathsf{s}(E, \mathcal{Q}) \neq \emptyset$;*

[1] In contrast, the proof of Theorem 21 shows the size of the maximal antichain in $\mathsf{s}(E, \mathcal{Q}_p[\Diamond])$ is in general exponential in $|E|$.

(*b*) *computing a longest/shortest separator in* $\mathsf{s}(E, \mathcal{Q})$;
(*c*) *deciding the existence of a unique most specific* $\mathcal{Q}$*-separator and a unique most general* $\mathcal{Q}_p[\Diamond]$*-separator, and constructing such a separator.*

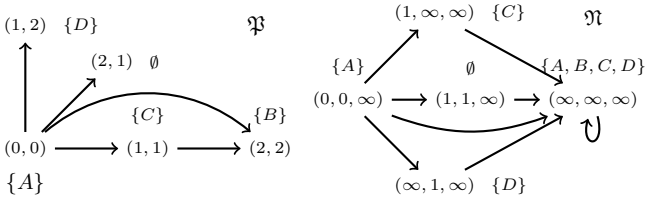
For bounded c_+ *and* c_-*, problem* (a) *is in* NL.

Proof. We only sketch the construction for $\mathcal{Q} = \mathcal{Q}_p[\Diamond]$.

(a) First, we define a directed labelled rooted graph $\mathfrak{P}$ whose paths from the root represent $\mathcal{Q}_p[\Diamond]$-queries with positive examples $E^+ = \{\mathcal{D}_1, \dots, \mathcal{D}_{c_+}\}$. Its nodes are vectors $(n_1, \dots, n_{c_+})$ with $0 \leq n_i \leq \max \mathcal{D}_i$, which are labelled by the sets $\{A \in \sigma \mid A(n_i) \in \mathcal{D}_i \text{ for all } i\}$, and the edges are $(n_1, \dots, n_{c_+}) \to (n'_1, \dots, n'_{c_+})$ with $n_i < n'_i$ for all i. The *root* of $\mathfrak{P}$ is $\bar{0} = (0, \dots, 0)$. To illustrate, let $E = (E^+, E^-)$, where $E^+ = \{\mathcal{D}_1^+, \mathcal{D}_2^+\}$, $E^- = \{\mathcal{D}_1^-, \mathcal{D}_2^-, \mathcal{D}_3^-\}$,

$$\mathcal{D}_1^+ = \{A(0), C(1), D(1), B(2)\},$$
$$\mathcal{D}_2^+ = \{A(0), C(1), B(2), D(2)\},$$
$$\mathcal{D}_1^- = \{A(0), C(1)\},\ \mathcal{D}_2^- = \{A(0), D(1)\},\ \mathcal{D}_3^- = \{B(0)\}.$$

Graph $\mathfrak{P}$ is shown on the left-hand side of the picture below:



Each path starting at $\bar{0}$ gives rise to a $\mathcal{Q}_p^\sigma[\Diamond]$-query with positive examples in E^+: e.g., $(0,0) \to (1,1) \to (2,2)$ gives rise to the query $A \wedge \Diamond(C \wedge \Diamond B)$.

Next, define another graph $\mathfrak{N}$ for $E^- = \{\mathcal{D}_1, \dots, \mathcal{D}_{c_-}\}$. Its nodes are $(n_1, \dots, n_{c_-})$, where $n_i \in [0, \max \mathcal{D}_i] \cup \{\infty\}$, including $\bar{\infty} = (\infty, \dots, \infty)$, which is labelled by σ. The label of $(n_1, \dots, n_{c_-})$ is $\{A \mid A(n_i) \in \mathcal{D}_i, \text{ for all } n_i \neq \infty\}$. The edges are defined in the same way as for $\mathfrak{P}$, with $n_i < \infty$, for any n_i. The root of $\mathfrak{N}$ is $(n_1, \dots, n_l)$, where $n_i = 0$ if $\{A(0) \mid A(0) \in \mathcal{D}, \text{ for all } \mathcal{D} \in E^+\} \subseteq \mathcal{D}_i$ and $n_i = \infty$ otherwise (see the picture above). Let $\boldsymbol{q}$ be a $\mathcal{Q}_p^\sigma[\Diamond]$-query of the form (1) with ρ_0 contained in the root's label. Then all $\mathcal{D}_i \in E^-$ are negative examples for $\boldsymbol{q}$ iff every path starting at the root and having labels $\rho'_0, \dots, \rho'_n$ with $\rho'_i \supseteq \rho_i$, for all $i \leq n$, comes through node $\bar{\infty}$. In our example, every such path for $\boldsymbol{q} = A \wedge \Diamond B$ and $\boldsymbol{q} = A \wedge \Diamond(C \wedge \Diamond B)$ involves $\bar{\infty}$. However, this is not the case for $\boldsymbol{q} = A \wedge \Diamond C$.

Consider now a graph $\mathfrak{P} \otimes \mathfrak{N}$ with nodes $(\boldsymbol{n}, \boldsymbol{m})$, where $\boldsymbol{n}$ is a node in $\mathfrak{P}$ and $\boldsymbol{m}$ a node in $\mathfrak{N}$ with $\boldsymbol{l}(\boldsymbol{n}) \subseteq \boldsymbol{l}(\boldsymbol{m})$, for the labels $\boldsymbol{l}(\boldsymbol{n})$ and $\boldsymbol{l}(\boldsymbol{m})$ of $\boldsymbol{n}$ and $\boldsymbol{m}$. We have an edge $(\boldsymbol{n}, \boldsymbol{m}) \to (\boldsymbol{n}', \boldsymbol{m}')$ in $\mathfrak{P} \otimes \mathfrak{N}$ iff $\boldsymbol{n} \to \boldsymbol{n}'$ in $\mathfrak{P}$, $\boldsymbol{m} \to \boldsymbol{m}'$ in $\mathfrak{N}$, and $\mathfrak{P} \otimes \mathfrak{N}$ has no $(\boldsymbol{n}', \boldsymbol{m}'')$ with $\boldsymbol{m} \to \boldsymbol{m}''$, $\boldsymbol{m}''_i < \boldsymbol{m}'_i$ and $\boldsymbol{m}''_i \neq \infty$, for some i, $\boldsymbol{m}_i$ being the ith coordinate of $\boldsymbol{m}$. One can see that, for any $(\boldsymbol{n}, \boldsymbol{m})$ in $\mathfrak{P} \times \mathfrak{N}$ and $\boldsymbol{n}'$ in $\mathfrak{P}$, there exists at most one edge $(\boldsymbol{n}, \boldsymbol{m}) \to (\boldsymbol{n}', \boldsymbol{m}')$. The root of $\mathfrak{P} \times \mathfrak{N}$ comprises the roots of $\mathfrak{P}$ and $\mathfrak{N}$. In our example, the edges from the root $(\bar{0}, (0,0,\infty))$ of $\mathfrak{P} \otimes \mathfrak{N}$ lead to $((1,2), (\infty,1,\infty))$, $((2,1), (1,1,\infty))$, $((1,1), (1,\infty,\infty))$ and $((2,2), \bar{\infty})$. Given a path

$$\pi = (\bar{0} = \boldsymbol{n}_0, \boldsymbol{m}_0), \dots, (\boldsymbol{n}_n, \boldsymbol{m}_n) \tag{2}$$

let $\boldsymbol{q}_\pi$ be the $\mathcal{Q}_p[\Diamond]$-query of the form (1) with $\rho_i = \boldsymbol{l}(\boldsymbol{n}_i)$ (note that $\boldsymbol{q}_\pi$ is not necessarily in normal form). We call π a *separating path* for E if $\boldsymbol{l}(\boldsymbol{n}_n) \neq \emptyset$ and $\boldsymbol{m}_n = \infty$.

Lemma 24. $\mathsf{s}(E, \mathcal{Q}) \neq \emptyset$ *iff* $\mathfrak{P} \otimes \mathfrak{N}$ *contains a separating path* π*, with* $\boldsymbol{q}_\pi$ *separating* E*.*

In our running example, $\mathfrak{P} \otimes \mathfrak{N}$ has two *separating paths*: $\pi_1 = (\bar{0}, (0,0,\infty))$, $((2,2), \bar{\infty})$ and $\pi_2 = (\bar{0}, (0,0,\infty))$, $((1,1), (1,\infty,\infty))$, $((2,2), \bar{\infty})$, which give rise to the separators $\boldsymbol{q}_{\pi_1} = A \wedge \Diamond B$ and $\boldsymbol{q}_{\pi_2} = A \wedge \Diamond(C \wedge \Diamond B)$.

The existence of a separating path in $\mathfrak{P} \otimes \mathfrak{N}$ can be checked in time $O(t_+^{c_+} t_-^{c_-})$. If c_+ and c_- are bounded, given $(\boldsymbol{n}, \boldsymbol{m})$ and $(\boldsymbol{n}', \boldsymbol{m}')$, we can check in logspace whether $(\boldsymbol{n}, \boldsymbol{m})$ is the root of $\mathfrak{P} \otimes \mathfrak{N}$ and $(\boldsymbol{n}, \boldsymbol{m}) \to (\boldsymbol{n}', \boldsymbol{m}')$, and so the existence of a separating path can be decided in NL.

The proof of point (b) relies on the following observation:

Lemma 25. *If* $\boldsymbol{q}$ *of the form* (1) *separates* E*, then there is a separating path of the form* (2) *with* $\rho_i \subseteq \boldsymbol{l}(\boldsymbol{n}_i)$*, for all* $i \leq n$*.*

It follows that the length of a longest/shortest separator for E coincides with the length of a longest/shortest separating path in $\mathfrak{P} \otimes \mathfrak{N}$, which can be found in polytime.

The proof of point (c) is based on the following criterion:

Lemma 26. *A* $\mathcal{Q}_p[\Diamond]$*-query* $\boldsymbol{q}$ *is a unique most specific* $\mathcal{Q}_p[\Diamond]$*-separator for* E *iff there is a separating path* π *such that* $\boldsymbol{q} = \boldsymbol{q}_\pi$ *and* $\boldsymbol{q}_\pi \models \boldsymbol{q}_\nu$*, for every separating path* ν*.*

In our example, $\boldsymbol{q}_{\pi_2}$ is a unique most specific separator. We show that the criterion of Lemma 26 can be checked in polytime in $\mathfrak{P} \times \mathfrak{N}$. For unique most general separators, the seemingly obvious inversion of $\boldsymbol{q}_\pi \models \boldsymbol{q}_\nu$ does not give a criterion, and a different type of graph is required. ⊣

Finally, we show how Theorem 23 can be used to check the existence of and construct unique most general separators in $\mathcal{Q}[\Diamond]$ (the case of unique most specific ones is trivial).

Theorem 27. *An example set* $E = (E^+, E^-)$ *has a unique most general separator in* $\mathcal{Q}[\Diamond]$ *iff* $\boldsymbol{q} = \bigwedge_{\mathcal{D} \in S} \boldsymbol{q}_\mathcal{D}$ *separates* E*, where* S *is the set of* $\mathcal{D} \in E^-$ *such that* $(E^+, \{\mathcal{D}\})$ *has a unique most general separator,* $\boldsymbol{q}_\mathcal{D}$*, in* $\mathcal{Q}_p[\Diamond]$*. In this case,* $\boldsymbol{q}$ *is a unique most general separator of* E *in* $\mathcal{Q}[\Diamond]$*.*

8 Conclusions

We have conducted a comprehensive complexity analysis of extremal separators in the spaces $\mathsf{s}(E, \mathcal{Q})$ with $\mathcal{Q}$ ranging from various classes of temporal path $\bigcirc\Diamond$-queries to arbitrary $\Diamond$-queries. For arbitrary $\bigcirc\Diamond$-queries, we only know more or less straightforward upper bounds such as Π_2^p for (unique) most general separator verification and Σ_3^p for unique most general separator existence. Establishing tight bounds remains a challenging open problem, which requires a deeper understanding of query containment for these queries.

We also plan to analyse the shortest and longest separators. For instance, Jung *et al*. (2024) show that verifying such separators in $\mathsf{s}(E, \mathcal{Q}_p[\bigcirc\Diamond])$ is coNP-complete—harder than verifying most specific/general ones. An empirical evaluation of our algorithms and more expressive query languages (say, with non-strict $\Diamond$ and 'until') are left for future work.

References

[Abboud *et al.*, 2015] A. Abboud, A. Backurs, and V. V. Williams. Tight hardness results for LCS and other sequence similarity measures. In *Proc. of FOCS*, pages 59–78. IEEE Computer Society, 2015.

[Alexe *et al.*, 2011] B. Alexe, B. ten Cate, Ph. Kolaitis, and W. Ch. Tan. Characterizing schema mappings via data examples. *ACM Trans. Database Syst.*, 36(4):23, 2011.

[Arenas *et al.*, 2016] M. Arenas, G. I. Diaz, and E. Kostylev. Reverse engineering SPARQL queries. In *Proc. of WWW*, pages 239–249, 2016.

[Arora and Barak, 2009] S. Arora and B. Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

[Artale *et al.*, 2021] A. Artale, R. Kontchakov, A. Kovtunova, V. Ryzhikov, F. Wolter, and M. Zakharyaschev. First-order rewritability of ontology-mediated queries in linear temporal logic. *Artif. Intell.*, 299:103536, 2021.

[Baader *et al.*, 2015] F. Baader, S. Borgwardt, and M. Lippmann. Temporal query entailment in the description logic $\mathcal{SHQ}$. *J. Web Semant.*, 33:71–93, 2015.

[Barceló and Romero, 2017] P. Barceló and M. Romero. The complexity of reverse engineering problems for conjunctive queries. In *Proc. of ICDT*, pages 7:1–7:17, 2017.

[Barceló *et al.*, 2021] P. Barceló, A. Baumgartner, V. Dalmau, and B. Kimelfeld. Regularizing conjunctive features for classification. *J. Comput. Syst. Sci.*, 119:97–124, 2021.

[Bergroth *et al.*, 2000] L. Bergroth, H. Hakonen, and T. Raita. A survey of longest common subsequence algorithms. In *Proc. of SPIRE*, pages 39–48, 2000.

[Blass and Gurevich, 1982] A. Blass and Y. Gurevich. On the unique satisfiability problem. *Inf. Control.*, 55(1-3):80–88, 1982.

[Blin *et al.*, 2012] G. Blin, L. Bulteau, M. Jiang, P. Tejada, and S. Vialette. Hardness of longest common subsequence for sequences with bounded run-lengths. In *Proc. of CPM*, volume 7354 of *Lecture Notes in Computer Science*, pages 138–148. Springer, 2012.

[Blum *et al.*, 2021] Ch. Blum, M. Djukanovic, A. Santini, H. Jiang, Chu-Min Li, F. Manyà, and G. Raidl. Solving longest common subsequence problems via a transformation to the maximum clique problem. *Comput. Oper. Res.*, 125:105089, 2021.

[Borgwardt *et al.*, 2015] S. Borgwardt, M. Lippmann, and V. Thost. Temporalizing rewritable query languages over knowledge bases. *J. Web Semant.*, 33:50–70, 2015.

[Camacho and McIlraith, 2019] A. Camacho and Sh. McIlraith. Learning interpretable models expressed in linear temporal logic. In *Proc. of ICAPS*, pages 621–630. AAAI Press, 2019.

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

[Chomicki and Toman, 2018] J. Chomicki and D. Toman. Temporal logic in database query languages. In Ling Liu and M. Tamer Özsu, editors, *Encyclopedia of Database Systems, Second Edition*. Springer, 2018.

[Chowdhury *et al.*, 2010] R. A. Chowdhury, Hai-Son Le, and V. Ramachandran. Cache-oblivious dynamic programming for bioinformatics. *IEEE/ACM Trans. on Computational Biology and Bioinformatics*, 7(3):495–510, 2010.

[Cima *et al.*, 2021] G. Cima, F. Croce, and M. Lenzerini. Query definability and its approximations in ontology-based data management. In *Proc. of CIKM*, pages 271–280. ACM, 2021.

[Cohen and Weiss, 2016] S. Cohen and Y. Y. Weiss. The complexity of learning tree patterns from example graphs. *ACM Trans. Database Syst.*, 41(2):14:1–14:44, 2016.

[Crochemore *et al.*, 2007] M. Crochemore, Ch. Hancart, and T. Lecroq. *Algorithms on strings*. Cambridge University Press, 2007.

[Cropper *et al.*, 2022] A. Cropper, S. Dumancic, R. Evans, and S. H. Muggleton. Inductive logic programming at 30. *Mach. Learn.*, 111(1):147–172, 2022.

[Deutch and Gilad, 2019] D. Deutch and A. Gilad. Reverse-engineering conjunctive queries from provenance examples. In *Proc. of EDBT*, pages 277–288, 2019.

[Fijalkow and Lagarde, 2021] N. Fijalkow and G. Lagarde. The complexity of learning linear temporal formulas from examples. In *Proc. of ICGI*, pages 237–250. PMLR, 2021.

[Fortin *et al.*, 2022] M. Fortin, B. Konev, V. Ryzhikov, Y. Savateev, F. Wolter, and M. Zakharyaschev. Unique characterisability and learnability of temporal instance queries. In *Proc. of KR*, 2022.

[Fortin *et al.*, 2023] M. Fortin, B. Konev, V. Ryzhikov, Y. Savateev, F. Wolter, and M. Zakharyaschev. Reverse engineering of temporal queries mediated by LTL ontologies. In *Proc. of IJCAI*, pages 3230–3238. ijcai.org, 2023.

[Fraser, 1996] C. Fraser. Consistent subsequences and supersequences. *Theor. Comput. Sci.*, 165(2):233–246, 1996.

[Funk *et al.*, 2019] M. Funk, J.-Ch. Jung, C. Lutz, H. Pulcini, and F. Wolter. Learning description logic concepts: When can positive and negative examples be separated? In *Proc. of IJCAI*, pages 1682–1688, 2019.

[Gutiérrez-Basulto *et al.*, 2018] V. Gutiérrez-Basulto, J. Ch. Jung, and L. Sabellek. Reverse engineering queries in ontology-enriched systems: The case of expressive Horn description logic ontologies. In *Proc. of IJCAI-ECAI*, 2018.

[Jung *et al.*, 2022] J. Ch. Jung, C. Lutz, H. Pulcini, and F. Wolter. Logical separability of labeled data examples under ontologies. *Artif. Intell.*, 313:103785, 2022.

[Jung *et al.*, 2024] J. Ch. Jung, V. Ryzhikov, F. Wolter, and M. Zakharyaschev. Extremal separation problems for temporal instance queries. *CoRR* abs/2405.03511, 2024.

[Kalashnikov *et al.*, 2018] D. Kalashnikov, L. Lakshmanan, and D. Srivastava. Fastqre: Fast query reverse engineering. In *Proc. of SIGMOD*, pages 337–350, 2018.

[Kimelfeld and Ré, 2018] B. Kimelfeld and Ch. Ré. A relational framework for classifier engineering. *ACM Trans. Database Syst.*, 43(3):11:1–11:36, 2018.

[Lemieux *et al.*, 2015] C. Lemieux, D. Park, and I. Beschastnikh. General LTL specification mining (T). In *Proc. of ASE*, pages 81–92. IEEE, 2015.

[Maier, 1978] D. Maier. The complexity of some problems on subsequences and supersequences. *J. ACM*, 25(2):322–336, 1978.

[Martins, 2019] D. Martins. Reverse engineering database queries from examples: State-of-the-art, challenges, and research opportunities. *Inf. Syst.*, 83:89–100, 2019.

[Mitchell, 1982] T. Mitchell. Generalization as search. *Artif. Intell.*, 18(2):203–226, 1982.

[Neider and Gavran, 2018] D. Neider and I. Gavran. Learning linear temporal properties. In *Proc. of FMCAD*, pages 1–10. IEEE, 2018.

[Ortiz, 2019] M. Ortiz. Ontology-mediated queries from examples: a glimpse at the DL-Lite case. In *Proc. of GCAI*, pages 1–14, 2019.

[Raha *et al.*, 2022] R. Raha, R. Roy, N. Fijalkow, and D. Neider. Scalable anytime algorithms for learning fragments of linear temporal logic. In *Proc. of TACAS*, volume 13243 of *Lecture Notes in Computer Science*, pages 263–280. Springer, 2022.

[Sarker *et al.*, 2017] Md. K. Sarker, Ning Xie, D. Doran, M. Raymer, and P. Hitzler. Explaining trained neural networks with semantic web technologies: First steps. In *Proc. of NeSy*, 2017.

[Staworko and Wieczorek, 2012] S. Staworko and P. Wieczorek. Learning twig and path queries. In *Proc. of ICDT*, pages 140–154, 2012.

[ten Cate and Dalmau, 2015] B. ten Cate and V. Dalmau. The product homomorphism problem and applications. In *Proc. of ICDT*, pages 161–176, 2015.

[ten Cate *et al.*, 2023] B. ten Cate, V. Dalmau, M. Funk, and C. Lutz. Extremal fitting problems for conjunctive queries. In *Proc. of PODS*, pages 89–98. ACM, 2023.

[Weiss and Cohen, 2017] Y. Y. Weiss and S. Cohen. Reverse engineering SPJ-queries from examples. In *Proc. of PODS*, pages 151–166, 2017.

[Zhang *et al.*, 2013] M. Zhang, H. Elmeleegy, C. M. Procopiuc, and D. Srivastava. Reverse engineering complex join queries. In *Proc. of SIGMOD*, pages 809–820, 2013.