Learning Aggregate Queries Defined by First-Order Logic with Counting

Steffen van Bergerem
Humboldt-Universität zu Berlin, Germany

Nicole Schweikardt
Humboldt-Universität zu Berlin, Germany

Version: 3rd September 2024

Abstract

In the logical framework introduced by Grohe and Turán (TOCS 2004) for Boolean classification problems, the instances to classify are tuples from a logical structure, and Boolean classifiers are described by parametric models based on logical formulas. This is a specific scenario for supervised passive learning, where classifiers should be learned based on labelled examples. Existing results in this scenario focus on Boolean classification. This paper presents learnability results beyond Boolean classification. We focus on multiclass classification problems where the task is to assign input tuples to arbitrary integers. To represent such integer-valued classifiers, we use aggregate queries specified by an extension of first-order logic with counting terms called FOC_1.

Our main result shows the following: given a database of polylogarithmic degree, within quasi-linear time, we can build an index structure that makes it possible to learn FOC_1-definable integer-valued classifiers in time polylogarithmic in the size of the database and polynomial in the number of training examples.

2012 ACM Subject Classification Theory of computation → Database Theory; Theory of computation → Logic; Theory of computation → Finite Model Theory; Computing methodologies → Logical and relational learning; Computing methodologies → Supervised learning

Keywords and phrases Supervised learning, multiclass classification problems, counting logic

1 Introduction

We study the complexity of learning aggregate queries from examples. This is a *classification problem* of the following form. The elements that are to be classified come from a set X, the *instance space*. For a given set V, a V-valued *classifier* on X is a function $c\colon X \to V$. We are given a *training set* S of labelled examples $(x, \lambda) \in X \times V$, i. e., λ is the label assigned to the instance x. The goal is to find a classifier, called a *hypothesis*, that can be used to predict the label of elements from X, including those not given in S. The term *Boolean classification problem* refers to the case where $|V| = 2$ (often, V is $\{1, 0\}$). We use the term *multiclass classification problem* to refer to cases where V may be arbitrarily large. In machine learning, these problems fall into the category of *supervised learning* tasks: we want to learn a function from given input-output pairs. In contrast to this, in unsupervised learning (e. g. clustering), the goal is to learn patterns from unlabelled data [48].

We focus on learning problems related to the framework introduced by Grohe and Turán [35]. There, the instance space X is a set of tuples from a logical structure (that is sometimes called the *background structure*), and the classifiers are Boolean and are described using parametric models based on logical formulas. In this paper, we extend the framework to multiclass classification problems where the classifiers are integer-valued, i. e., $V = \mathbb{Z}$. In the framework that we consider, the background structure is a relational database $\mathcal{A}$, and the instance space X is the set A^k of all k-tuples of elements from the active domain A of $\mathcal{A}$ (also called the *universe* of $\mathcal{A}$). Here, k is a fixed positive integer. One fixes a *parameter length* ℓ (a fixed non-negative integer). A classifier is specified by a pair $p = (t, \bar{w})$, where

$\bar{w} = (w_1, \dots, w_\ell)$ is an ℓ-tuple of elements in A, and t is a *counting term* in the *first-order logic with counting* FOC_1 [34] with free variables $x_1, \dots, x_k, y_1, \dots, y_\ell$. This pair p represents the classifier $c_p \colon X \to \mathbb{Z}$ that assigns to each k-tuple $\bar{a} = (a_1, \dots, a_k) \in X$ the integer i that is obtained by evaluating the counting term t in the database $\mathcal{A}$ while interpreting the variables $x_1, \dots, x_k$ with the elements $a_1, \dots, a_k$ and the variables $y_1, \dots, y_\ell$ with the "parameters" $w_1, \dots, w_\ell$. We will write $h^{\mathcal{A}}_{t,\bar{w}}$ to denote this classifier c_p.

Given a *training set* $S \subseteq A^k \times \mathbb{Z}$, we want to find a pair $p = (t, \bar{w})$ such that the classifier $h^{\mathcal{A}}_{t,\bar{w}}$ is *consistent* with S, i. e., it satisfies $h^{\mathcal{A}}_{t,\bar{w}}(\bar{a}) = i$ for every $(\bar{a}, i) \in S$.

▶ **Example 1.1.** Let $\mathcal{A}$ be a relational database where the active domain A contains authors and publications, the binary relation `Author` contains all pairs (a, p) where a is an author of the publication p, and the binary relation `Citation` contains all pairs (p_1, p_2) where the publication p_1 cites the publication p_2.

Suppose we are given a training set S that consists of a few pairs (a, i) where a is an author and i is the total number of citations of the publications of a. A reasonable classifier for this setting would be a mapping $c \colon A \to \mathbb{Z}$ that assigns to every author a present in the database the total number i of citations of their publications. In our setting, this can be represented as follows. We let $k = 1$ and $\ell = 0$. Since $\ell = 0$, the "parameter" w is fixed to be the empty tuple $()$. Since $k = 1$, we use a counting term with a single free variable x (that will be assigned with authors present in the database). We choose the counting term

$$t(x) \ := \ \#(z_1, z_2).\big(\texttt{Author}(x, z_1) \land \texttt{Citation}(z_2, z_1)\big).$$

Evaluating $t(x)$ for an author x yields the number of tuples (z_1, z_2) such that x is an author of publication z_1, and z_2 is a publication that cites z_1. This is precisely the total number of citations of publications authored by x. Hence, $h^{\mathcal{A}}_{t,()}$ is the desired classifier c.

▶ **Example 1.2.** Suppose we have a database that maintains a list of all cakes colleagues brought to work. We model this as a relational database $\mathcal{A}$ whose active domain A contains persons, IDs of cakes, and types of cake. The binary relation `Brought` contains all pairs (p, c) where p is a person that brought the cake with ID c, and the binary relation `Type` contains all pairs (c, τ) where c is the ID of a cake of type τ (e. g., "chocolate cake", "strawberry cake", "carrot cake", etc). Suppose we want to find a classifier that predicts the popularity of colleagues. For this, via a survey, we gather examples $(p, i) \in A \times \mathbb{Z}$ where p is a person and i is the popularity of the person, and we call the resulting set of labelled examples S. We choose $k = \ell = 1$, so we want to find a classifier that uses a single parameter. According to our own experience at work, it seems conceivable that the following classifier $h^{\mathcal{A}}_{t,w}$ is consistent with S: the parameter w is "chocolate cake" and t is the counting term

$$t(x, y) \ := \ \#(z).\big(\texttt{Brought}(x, z) \land \neg\texttt{Type}(z, y)\big) \ + \ 2 \cdot \#(z).\big(\texttt{Brought}(x, z) \land \texttt{Type}(z, y)\big).$$

Note that t counts the number of cakes brought by person x, where cakes of type y are counted twice, and the variable y will always be assigned the value of the parameter w.

In many application scenarios, the same database is used multiple times with different training sets to learn different classifiers. Thus, we consider a setting in which we are first only given the database, without any training examples. In a precomputation step, we allow gathering information that will be helpful for solving future learning tasks. This precomputation step can be viewed as building an index structure that is designed in order to support solving multiple learning tasks.

In the actual learning phase, we are repeatedly given training sets of labelled examples, and our task is to output a hypothesis that is consistent with the corresponding training set.

For this learning phase, it would be desirable to have algorithms that run efficiently even if the database is too large to fit into the main memory. To achieve this, we are interested in algorithms that require only *local access* to the database, i. e., instead of having random access to the database, a learning algorithm should initially start with the elements given in the training set; subsequently, it may only retrieve the neighbours of elements it already holds in memory. By utilising the memory hierarchy, such local access can be achieved efficiently even in cases where random access is too prohibitive. In the context of learning (concerning Boolean classification problems), this local-access model has been introduced by Grohe and Ritzert [33].

Our contribution

Our main result is an algorithm that builds the index structure in time linear in the size and polynomial in the degree of the database. Afterwards, upon input of concrete training sets, classifiers definable in FOC_1 can be learned in time polynomial in the degree of the database and polynomial in the number of examples given in the training set. Moreover, the classifiers returned by our algorithm can be evaluated in time polynomial in the degree of the database. Furthermore, our algorithms for finding a classifier and for evaluating this classifier do not require random access to the database but only rely on the local-access model.

For databases of polylogarithmic degree (i. e., of degree up to $(\log n)^c$ where c is a constant and n is the size of the database), our main result implies that the index structure can be built in quasi-linear time (i. e., time $n{\cdot}(\log n)^c$); afterwards, FOC_1-definable integer-valued classifiers can be learned in time polylogarithmic (so, in particular, sublinear) in the size of the database and polynomial in the number of training examples.

Previous results in the framework of Grohe and Turán for Boolean classification problems relied on the fact that it suffices to check a constant number of queries while limiting the search space for the parameters to a neighbourhood of a certain radius [33, 7, 10]. For our setting of multiclass classification with aggregate queries, however, this does not hold any more. Hence, a priori, it is not clear that sublinear-time learning algorithms are possible for the multiclass case at all. The main technical challenge towards our learnability result was to find an approach that keeps the number of queries to check small (i. e., polynomial in the degree of the database), while still being able to limit the search space for the parameters to a small neighbourhood around the given training tuples.

Organisation

We provide the necessary definitions concerning FOC_1 in Section 2, and we formally introduce the learning problem that we consider in Section 3. The precise statement of our main result is given in Theorem 3.1. Our proof makes heavy use of the locality properties of the logic FOC_1 shown in [34], including a decomposition of FOC_1-formulas into local formulas. These properties are used in Section 4 to provide our main technical tool for the proof of the main result. Section 5 concludes the paper with a summary and an outlook on future work. In the remainder of this introduction, we give an overview of related work.

Related work

The first-order logic with counting FOC was introduced in [42] and further studied in [34, 7]. This logic extends first-order logic (FO) by the ability to formulate *counting terms* that evaluate to integers, and by *numerical predicates* that allow to compare results of counting terms. It was shown in [42] that the model-checking problem for FOC is fixed-parameter tractable on classes of structures of bounded degree. From [34] it is known that the fixed-parameter tractability of FOC cannot be generalised to even very simple classes of structures of unbounded degree such as unranked trees (under a reasonable assumption in parameterised complexity). However, [34] identified a fragment called FOC_1 for which model-checking of formulas and evaluation of counting terms are fixed-parameter tractable on all nowhere dense classes of structures. The present paper uses counting terms of FOC_1 to represent integer-valued classifiers.

The learning framework we consider has been introduced for Boolean classification problems in [35], which provides information-theoretic learnability results for classes of classifiers that can be specified using FO- and MSO-formulas on restricted classes of structures, such as the class of planar graphs or classes of graphs of bounded degree. Algorithmic aspects of the framework, including the running time of a learning algorithm, were first studied in [33]. The paper showed that Boolean classifiers definable in FO can be learned in sublinear time on structures of polylogarithmic degree. Analogous results have been obtained for MSO on strings [32] and on trees [29], which included a precomputation step to allow for efficient repeated learning. The paper [9] studied the parameterised complexity of the Boolean classification problem and showed that on arbitrary relational structures, learning hypotheses definable in FO is hard for the parameterised complexity class $\mathsf{AW}[*]$ (i. e., subject to a plausible complexity-theoretic assumption, it is not fixed-parameter tractable). The paper also showed that the problem is fixed-parameter tractable if the structures come from a nowhere dense class.

For Boolean classifiers definable in the extension FOCN of FO with counting quantifiers and numerical predicates, [7] obtained a sublinear-time learning algorithm for structures of bounded degree, i. e., classes of structures where the degree is bounded by a constant. Recently, [8] lifted this result to structures of tiny degree, i. e., classes of structures of degree up to $(\log\log n)^c$ for some constant c, where n is the size of the structure. The paper [10] considered a notion of *weighted structures*, which extend ordinary relational structures by assigning weights, i. e. elements from particular rings or abelian groups, to tuples present in the structure. It introduced the expressive logic FOWA, which extends FO by means of aggregating weights and formulating both formulas (that evaluate to "true" or "false") and terms (that "aggregate" weights and evaluate to values in the associated ring or abelian group). For the fragment FOWA_1 (that still extends FO), the paper showed that Boolean classifiers definable by FOWA_1-formulas over weighted background structures of polylogarithmic degree can be learned in sublinear time after quasi-linear-time preprocessing. This lifts the results obtained in [33] for FO to the substantially more expressive logic FOWA_1. As the logic FOC_1 can be embedded in FOWA_1, it follows from [10] that Boolean classifiers definable by FOC_1-formulas over background structures of polylogarithmic degree can be learned in sublinear time after quasi-linear-time preprocessing. The main result of the present paper can be viewed as a generalisation of this to integer-valued classification problems.

The algorithmic results obtained so far within the framework introduced in [35] all focus on *Boolean classification* problems. However, many application scenarios require *multiclass classification* (cf. [21, 14, 36]). In the database systems literature, multiclass classifiers typically are described by *aggregate queries* [51, 52, 53, 54, 45]. In this paper, aggregate queries are represented by the counting terms of FOC_1.

Closely related to the framework we consider is the framework of *inductive logic programming* (ILP) [46, 47, 39, 19, 20]. Both frameworks deal with a *passive supervised learning* setting, where the learning algorithms are given labelled examples. These examples are labelled according to some target concept, and the algorithms should return a hypothesis that approximately matches this target concept. One of the main differences between both frameworks is that we represent the background knowledge by a relational database, whereas in ILP, it is represented in a background theory, i. e., a set of formulas. Related logical learning frameworks have also been studied in formal verification [27, 43, 23, 56, 18].

In the database literature, various approaches to learning queries from examples have been studied, both in passive (such as ours) and active learning settings. In passive learning settings, results often focus on conjunctive queries [37, 38, 6, 40, 5, 55] or consider queries outside the relational database model [50, 11], while we focus on FOC_1, an extension of full first-order logic. In the *active learning* setting introduced by Angluin [4], learning algorithms are allowed to actively query an oracle. Results in this setting [2, 49, 1, 11, 12, 15] again consider various types of queries. Another related subject in the database literature is the problem of learning schema mappings from examples [13, 28, 3, 16, 17].

2 Preliminaries

This section fixes the basic notation used throughout the paper, and it provides the precise syntax and semantics of first-order logic with counting FOC_1 (the latter is taken almost verbatim from [34]). We let $\mathbb{Z}$, $\mathbb{N}$, and $\mathbb{N}_{\geqslant 1}$ denote the sets of integers, non-negative integers, and positive integers, respectively. For $m, n \in \mathbb{Z}$, we let $[m, n] \coloneqq \{\ell \in \mathbb{Z} : m \leqslant \ell \leqslant n\}$ and $[n] \coloneqq [1, n]$. For a k-tuple $\bar{x} = (x_1, \ldots, x_k)$ we write $|\bar{x}|$ to denote its *arity* k. By $()$ we denote the empty tuple, i. e., the tuple of arity 0.

Signatures and structures

A *signature* is a finite set of relation symbols. Every relation symbol R has a fixed *arity* $\operatorname{ar}(R) \in \mathbb{N}$. Let σ be a signature. A *σ-structure* (or *σ-database*) $\mathcal{A}$ consists of a finite set A, called the *universe* of $\mathcal{A}$, and a relation $R(\mathcal{A}) \subseteq A^{\operatorname{ar}(R)}$ for every $R \in \sigma$. Note that signatures may contain relation symbols of arity 0. There are only two 0-ary relations over a set A, namely $\emptyset$ and $\{()\}$, which we interpret as "false" and "true", respectively. We define the *size* $|\mathcal{A}|$ of a σ-structure $\mathcal{A}$ as $|\mathcal{A}| \coloneqq |A|$.

Let σ' be a signature with $\sigma' \supseteq \sigma$. A *σ'-expansion* of a σ-structure $\mathcal{A}$ is a σ'-structure $\mathcal{A}'$ which has the same universe as $\mathcal{A}$ and which satisfies $R(\mathcal{A}') = R(\mathcal{A})$ for all $R \in \sigma$.

A *substructure* of a σ-structure $\mathcal{A}$ is a σ-structure $\mathcal{B}$ with universe $B \subseteq A$ that satisfies $R(\mathcal{B}) \subseteq R(\mathcal{A})$ for every $R \in \sigma$. For a set $X \subseteq A$, the *induced substructure of $\mathcal{A}$ on X* is the σ-structure $\mathcal{A}[X]$ with universe X, where $R(\mathcal{A}[X]) = R(\mathcal{A}) \cap X^{\operatorname{ar}(R)}$ for every $R \in \sigma$.

Gaifman graph, degree, distances, and neighbourhoods

In this paper, when speaking of graphs, we mean undirected simple graphs.

The *Gaifman graph* $G_{\mathcal{A}}$ of a σ-structure $\mathcal{A}$ with universe A is the graph with vertex set $V(G_{\mathcal{A}}) = A$ whose edge set $E(G_{\mathcal{A}})$ contains exactly those edges $\{v, w\}$ where $v, w \in A$, $v \neq w$, and there exists an $R \in \sigma$ and a tuple $(a_1, \ldots, a_{\operatorname{ar}(R)}) \in R(\mathcal{A})$ such that $v, w \in \{a_1, \ldots, a_{\operatorname{ar}(R)}\}$. The *degree* of a σ-structure $\mathcal{A}$ is defined as the degree of the Gaifman graph $G_{\mathcal{A}}$ (i. e., the maximum number of neighbours of a vertex of $G_{\mathcal{A}}$).

The *distance* $\operatorname{dist}^{\mathcal{A}}(a, b)$ between two elements $a, b \in A$ is the minimal number of edges of a path from a to b in $G_{\mathcal{A}}$; if no such path exists, we let $\operatorname{dist}^{\mathcal{A}}(a, b) \coloneqq \infty$. For a tuple $\bar{a} = (a_1, \ldots, a_k) \in A^k$ and an element $b \in A$, we let $\operatorname{dist}^{\mathcal{A}}(\bar{a}, b) \coloneqq \min_{i \in [k]} \operatorname{dist}^{\mathcal{A}}(a_i, b)$.

Consider a k-tuple $\bar{a} = (a_1, \ldots, a_k) \in A^k$ for some $k \in \mathbb{N}_{\geqslant 1}$. For every $r \geqslant 0$, the *ball of radius r* (or *r-ball*) of $\bar{a}$ in $\mathcal{A}$ is the set $N_r^{\mathcal{A}}(\bar{a}) \coloneqq \{b \in A : \operatorname{dist}^{\mathcal{A}}(\bar{a}, b) \leqslant r\}$. The *neighbourhood* *of radius r* (or *r-neighbourhood*) of $\bar{a}$ in $\mathcal{A}$ is the induced substructure $\mathcal{N}_r^{\mathcal{A}}(\bar{a}) \coloneqq \mathcal{A}[N_r^{\mathcal{A}}(\bar{a})]$.

First-order logic with counting FOC_1

Let vars be a fixed, countably infinite set of *variables.* A *σ-interpretation* $\mathcal{I} = (\mathcal{A}, \beta)$ consists of a σ-structure $\mathcal{A}$ and an *assignment* $\beta \colon \mathsf{vars} \to A$, where A denotes the universe of $\mathcal{A}$. For $k \in \mathbb{N}$ and k distinct variables $x_1, \ldots, x_k \in \mathsf{vars}$ and elements $a_1, \ldots, a_k \in A$, we write $\mathcal{I}\frac{a_1,\ldots,a_k}{x_1,\ldots,x_k}$ for the interpretation $(\mathcal{A}, \beta\frac{a_1,\ldots,a_k}{x_1,\ldots,x_k})$, where $\beta\frac{a_1,\ldots,a_k}{x_1,\ldots,x_k}$ is the assignment β' with $\beta'(x_i) = a_i$ for every $i \in [k]$ and $\beta'(z) = \beta(z)$ for all $z \in \mathsf{vars} \setminus \{x_1, \ldots, x_k\}$.

Next, we provide the syntax and semantics of the logic FOC_1 [34]. This logic allows formulating numerical statements based on counting terms and numerical predicates.

For the remainder of this paper, fix a triple $(\mathbb{P}, \mathrm{ar}, [\![.]\!])$, called a *numerical predicate collection*, where $\mathbb{P}$ is a finite set of *predicate names*, ar assigns an *arity* $\mathrm{ar}(\mathsf{P}) \in \mathbb{N}_{\geqslant 1}$ to each $\mathsf{P} \in \mathbb{P}$, and $[\![.]\!]$ assigns a *semantics* $[\![\mathsf{P}]\!] \subseteq \mathbb{Z}^{\mathrm{ar}(\mathsf{P})}$ to each $\mathsf{P} \in \mathbb{P}$. Examples of numerical predicates are the *equality predicate* $\mathsf{P}_=$ with $[\![\mathsf{P}_=]\!] = \{(i,i) : i \in \mathbb{Z}\}$, the *comparison predicate* $\mathsf{P}_\leqslant$ with $[\![\mathsf{P}_\leqslant]\!] = \{(i,j) : i,j \in \mathbb{Z},\ i \leqslant j\}$, or the *prime number predicate* $\mathsf{P}_{\mathsf{prime}}$ with $[\![\mathsf{P}_{\mathsf{prime}}]\!] = \{i \in \mathbb{N} : i \text{ is a prime number}\}$. When analysing the running time of algorithms, we will assume that machines have access to oracles for evaluating the numerical predicates. That is, when given a $\mathsf{P} \in \mathbb{P}$ and a tuple $(i_1, \ldots, i_{\mathrm{ar}(\mathsf{P})})$ of integers, the oracle takes time $O(1)$ to answer if $(i_1, \ldots, i_{\mathrm{ar}(\mathsf{P})}) \in [\![\mathsf{P}]\!]$.

Let σ be a signature. The set of *formulas* and *counting terms* (for short: terms) of $\mathsf{FOC}_1[\sigma]$ is built according to the following rules.

(1) $x_1{=}x_2$ and $R(x_1, \ldots, x_{\mathrm{ar}(R)})$ are *formulas*,[1] where $R \in \sigma$, and $x_1, x_2, \ldots, x_{\mathrm{ar}(R)}$ are variables. We let $\mathrm{free}(x_1{=}x_2) \coloneqq \{x_1, x_2\}$ and $\mathrm{free}\big(R(x_1, \ldots, x_{\mathrm{ar}(R)})\big) \coloneqq \{x_1, \ldots, x_{\mathrm{ar}(R)}\}$.
(2) If φ and ψ are formulas, then $\neg\varphi$ and $(\varphi \vee \psi)$ are also *formulas.* We let $\mathrm{free}(\neg\varphi) \coloneqq \mathrm{free}(\varphi)$ and $\mathrm{free}((\varphi \vee \psi)) \coloneqq \mathrm{free}(\varphi) \cup \mathrm{free}(\psi)$.
(3) If φ is a formula and $x \in \mathsf{vars}$, then $\exists x\, \varphi$ is a *formula.* We let $\mathrm{free}(\exists x\, \varphi) \coloneqq \mathrm{free}(\varphi) \setminus \{x\}$.
(4) If φ is a formula and $\bar{x} = (x_1, \ldots, x_k)$ is a tuple of k pairwise distinct variables, for $k \geqslant 0$, then $\#\bar{x}.\varphi$ is a *counting term.* We let $\mathrm{free}\big(\#(x_1, \ldots, x_k).\varphi\big) \coloneqq \mathrm{free}(\varphi) \setminus \{x_1, \ldots, x_k\}$.
(5) Every integer $i \in \mathbb{Z}$ is a *counting term.* We let $\mathrm{free}(i) = \emptyset$.
(6) If t_1 and t_2 are counting terms, then $(t_1 + t_2)$ and $(t_1 \cdot t_2)$ are also *counting terms.* We let $\mathrm{free}\big((t_1 + t_2)\big) \coloneqq \mathrm{free}\big((t_1 \cdot t_2)\big) \coloneqq \mathrm{free}(t_1) \cup \mathrm{free}(t_2)$.
(7) If $\mathsf{P} \in \mathbb{P}$, $m = \mathrm{ar}(\mathsf{P})$, and $t_1, \ldots, t_m$ are counting terms with $\big|\bigcup_{i=1}^m \mathrm{free}(t_i)\big| \leqslant 1$, then $\mathsf{P}(t_1, \ldots, t_m)$ is a *formula.* We let $\mathrm{free}\big(\mathsf{P}(t_1, \ldots, t_m)\big) \coloneqq \bigcup_{i=1}^m \mathrm{free}(t_i)$.

First-order logic $\mathsf{FO}[\sigma]$ is the fragment of $\mathsf{FOC}_1[\sigma]$ built by using only the rules (1)–(3). We write $(\varphi \wedge \psi)$ and $\forall x\, \varphi$ as shorthands for $\neg(\neg\varphi \vee \neg\psi)$ and $\neg\exists x\, \neg\varphi$. For counting terms t_1 and t_2, we write $(t_1 - t_2)$ as a shorthand for $\big(t_1 + ((-1) \cdot t_2)\big)$.

By FOC_1, we denote the union of all $\mathsf{FOC}_1[\sigma]$ for arbitrary signatures σ. This applies analogously to FO. For FOC_1 the semantics for the rules (1)–(3) are defined in the same way as for FO; the semantics of the remaining rules are as follows. Let $\mathcal{I} = (\mathcal{A}, \beta)$ be a σ-interpretation, and let A denote the universe of $\mathcal{A}$.

(4) $[\![\#\bar{x}.\varphi]\!]^{\mathcal{I}} = \Big|\big\{(a_1, \ldots, a_k) \in A^k \ :\ [\![\varphi]\!]^{\mathcal{I}\frac{a_1,\ldots,a_k}{x_1,\ldots,x_k}} = 1\big\}\Big|$, where $\bar{x} = (x_1, \ldots, x_k)$.
(5) $[\![i]\!]^{\mathcal{I}} = i$, for $i \in \mathbb{Z}$.
(6) $[\![(t_1 + t_2)]\!]^{\mathcal{I}} = [\![t_1]\!]^{\mathcal{I}} + [\![t_2]\!]^{\mathcal{I}}$ and $[\![(t_1 \cdot t_2)]\!]^{\mathcal{I}} = [\![t_1]\!]^{\mathcal{I}} \cdot [\![t_2]\!]^{\mathcal{I}}$.

[1] in particular, if $\mathrm{ar}(R) = 0$ then $R()$ is a formula

(7) $[\![\mathsf{P}(t_1,\dots,t_m)]\!]^{\mathcal{I}} = 1$ if $([\![t_1]\!]^{\mathcal{I}},\dots,[\![t_m]\!]^{I}) \in [\![\mathsf{P}]\!]$, and $[\![\mathsf{P}(t_1,\dots,t_m)]\!]^{\mathcal{I}} = 0$ otherwise.

Examples of counting terms in FOC_1 and their intuitive meaning can be found in Examples 1.1 and 1.2.

An *expression* is a formula or a counting term. For an expression ξ, we write $\xi(z_1,\dots,z_k)$ to indicate that $\mathrm{free}(\xi) \subseteq \{z_1,\dots,z_k\}$. A *sentence* is a formula without free variables. A *ground term* is a counting term without free variables.

For a formula φ and a σ-interpretation $\mathcal{I}$, we write $\mathcal{I} \models \varphi$ to indicate that $[\![\varphi]\!]^{\mathcal{I}} = 1$. Likewise, $\mathcal{I} \not\models \varphi$ indicates that $[\![\varphi]\!]^{\mathcal{I}} = 0$. For a formula $\varphi(x_1,\dots,x_k)$, a σ-structure $\mathcal{A}$, and a tuple $\bar{a} = (a_1,\dots,a_k) \in A^k$, we write $\mathcal{A} \models \varphi[\bar{a}]$ to indicate that $(\mathcal{A},\beta) \models \varphi$ for all assignments β with $\beta(x_i) = a_i$ for all $i \in [k]$. Similarly, for a counting term $t(x_1,\dots,x_k)$, we write $t^{\mathcal{A}}[\bar{a}]$ for the integer $[\![t]\!]^{\mathcal{I}}$. In case that φ is a sentence and t is a ground term, we shortly write $\mathcal{A} \models \varphi$ instead of $\mathcal{A} \models \varphi[()]$, and we write $t^{\mathcal{A}}$ instead of $t^{\mathcal{A}}[()]$.

The *size* $|\xi|$ of an FOC_1-expression ξ is defined as the length of the string representation of ξ, where integers and variables are considered as having length 1. For $m, q \in \mathbb{N}$, we write $\mathsf{FOC}_1[\sigma, m, q]$ to denote the set of all $\mathsf{FOC}_1[\sigma]$-expressions of size at most q and with at most m free variables.

Hypotheses

Let $\mathcal{A}$ be a σ-structure with universe A, let $t(\bar{x},\bar{y})$ be an $\mathsf{FOC}_1[\sigma]$-term, let $k := |\bar{x}| \geqslant 1$, let $\ell := |\bar{y}| \geqslant 0$, and let $\bar{w} \in A^\ell$. The *hypothesis represented in $\mathcal{A}$ by the counting term $t(\bar{x},\bar{y})$ and the parameter $\bar{w}$* is defined as the mapping $f\colon A^k \to \mathbb{Z}$ such that $f(\bar{a}) = t^{\mathcal{A}}[\bar{a},\bar{w}]$ for every $\bar{a} \in A^k$. That is, evaluating the counting term t in $\mathcal{A}$ while assigning the variables $\bar{x}$ to the elements $\bar{a}$ and assigning the variables $\bar{y}$ to the elements $\bar{w}$ yields the integer $f(\bar{a})$. Henceforth, we will write $h^{\mathcal{A}}_{t,\bar{w}}$ to denote this function f. For a set $S \subseteq A^k \times \mathbb{Z}$, we say that $h^{\mathcal{A}}_{t,\bar{w}}$ *is consistent with* S if and only if $h^{\mathcal{A}}_{t,\bar{w}}(\bar{a}) = i$ for all $(\bar{a}, i) \in S$.

3 Learning FOC_1-Definable Aggregate Queries

Let σ, σ' be signatures with $\sigma \subseteq \sigma'$, fix two numbers $k \in \mathbb{N}_{\geqslant 1}$, $\ell \in \mathbb{N}$, and let $T \subseteq \mathsf{FOC}_1[\sigma], T' \subseteq \mathsf{FOC}_1[\sigma']$ be two sets of terms $t(\bar{x},\bar{y})$ with $|\bar{x}| = k$ and $|\bar{y}| = \ell$. We study the following problem.

Learn-with-Precomputation(k, ℓ, T, T')
Precomputation: Given a σ-structure $\mathcal{A}$ with universe A, compute a σ'-expansion $\mathcal{A}'$ of $\mathcal{A}$ and a lookup table whose size is independent of $\mathcal{A}$.
Input: Training set $S \subseteq A^k \times \mathbb{Z}$.
Task: Return a term $t' \in T'$ and a tuple $\bar{w}' \in A^\ell$ such that the hypothesis $h^{\mathcal{A}'}_{t',\bar{w}'}$ is consistent with S. The algorithm may reject if there is no term $t \in T$ and tuple $\bar{w} \in A^\ell$ such that the hypothesis $h^{\mathcal{A}}_{t,\bar{w}}$ is consistent with S.

As described in the introduction, the precomputation phase can be viewed as building an index structure for the given database $\mathcal{A}$. Afterwards, this index structure can be used each time that we receive as input a new training set S. The "learning phase" is what happens after receiving such a set S; the desired output is a hypothesis that is consistent with S. The main contribution of this paper is an efficient solution for the problem Learn-with-Precomputation(k, ℓ, T, T'). Before presenting the exact statement of our result (Theorem 3.1), recall from Section 1 the discussion on the benefits of *local-access* algorithms. In the learning phase (i. e., when receiving a training set), the algorithm we provide does not require random access to the database. Instead, it only needs *local access*, i.e., it only accesses the neighbours of elements that it already holds in memory, initially starting with the elements given in the training set. Here, "neighbours" refers to neighbours in the Gaifman graph $G_{\mathcal{A}}$ of the database $\mathcal{A}$. Formally, *local access* means that the algorithm can access an oracle that answers requests of the form "Is $\bar{v} \in R(\mathcal{A})$?" in constant time and requests of the form "Return a list of all neighbours of v in $\mathcal{A}$" in time linear in the number of neighbours of v. As our machine model, we use a random-access machine (RAM) model, and we consider running times under the uniform-cost measure. This allows us to store an element of the database in a single memory cell and access it in a single computation step.

The main result of this paper is the following theorem.

▶ **Theorem 3.1.** *Let σ be a signature, let $k \in \mathbb{N}_{\geqslant 1}$, let $\ell, q \in \mathbb{N}$, let I be a finite set of integers, and let T be the set of all $\mathsf{FOC}_1[\sigma, k{+}\ell, q]$-terms that only use integers from I.*

There is an extension σ' of σ with relation symbols of arity $\leqslant 1$, and there is a number $q' \in \mathbb{N}$ such that, for the set T' of all $\mathsf{FOC}_1[\sigma', k{+}\ell, q']$-terms, there is an algorithm that solves the problem LEARN-WITH-PRECOMPUTATION(k, ℓ, T, T') *as follows.*

For a σ-structure $\mathcal{A}$ of size n and degree d, the precomputation to compute the σ'-expansion $\mathcal{A}'$ of $\mathcal{A}$ and the associated lookup table takes time $d^{\mathcal{O}(1)} \cdot n$. Afterwards, upon input of a training set S of size s, the algorithm uses only local access to $\mathcal{A}'$, access to the lookup table, and time $(s + d)^{\mathcal{O}(1)}$ to return either a hypothesis consistent with S or the answer "reject". Furthermore, the returned hypothesis can be evaluated in time $d^{\mathcal{O}(1)}$, using only local access to $\mathcal{A}'$ and access to the lookup table.

In particular, this implies that when $\mathcal{A}$ comes from a class of σ-structures of polylogarithmic degree, then the precomputation takes time quasi-linear in n (i. e., $n \cdot (\log n)^{O(1)}$), a hypothesis is found within time polynomial in s and polylogarithmic in n, and it can be evaluated in time polylogarithmic in n.

We remark that the algorithm given in the proof of Theorem 3.1 is not meant to be implemented and used in practice. Instead, Theorem 3.1 serves as a fundamental result that shows the theoretic (and asymptotic) feasibility of learning aggregate queries definable in FOC_1. This is in line with previous work on the descriptive complexity of learning [35, 33, 32, 29, 7, 10, 9, 8] and closely related work on so-called algorithmic meta theorems (see, e. g., [30, 41]).

Before turning to the proof of Theorem 3.1, let us first briefly discuss the situation. In the assumption of the theorem, we require the set I of integers occurring in terms of T to be finite. However, note that we can still represent other integers in T by using rule (6) for FOC_1, that is, addition and multiplication. For example, we could let I be the set of powers of 10 up to some bound, and we could obtain the numbers in between with rather few additions and multiplications. Moreover, requiring I to be finite does not limit the use of counting terms of the form $\#\bar{x}.\varphi$, which may evaluate to arbitrary integers based on the given database. Meanwhile, we could let I contain (large) constants that are specific to the field the data is coming from, which allows using them even if the bound q on the size of the terms in T is small.

Since I is finite and the size of terms in T is bounded by q, up to equivalence, the set T is *finite*. Thus, when given a σ-structure $\mathcal{A}$ with universe A and a training set $S \subseteq A^k \times \mathbb{Z}$, in order to find a hypothesis that is consistent with S, we could proceed as follows. Loop through all terms $t(\bar{x}, \bar{y})$ in T. For each such term t, loop through all tuples $\bar{w} \in A^\ell$, and check whether $t^{\mathcal{A}}[\bar{v},\bar{w}] = \lambda$ for every $(\bar{v},\lambda) \in S$. If so, stop and output t and $\bar{w}$, indicating that $h^{\mathcal{A}}_{t,\bar{w}}$ is a hypothesis consistent with S. If no such hypothesis is found, then stop and output “reject”. This obviously solves the learning problem. However, the time taken to loop through all the $\bar{w} \in A^{\ell}$ is polynomial in $|A|$, and it may not suffice to start with the vertices given in the training set and repeatedly iterate over the neighbours of already found vertices. Note that Theorem 3.1 yields a much more efficient algorithm, which runs in time polynomial in the size of the training set and the degree of the structure. This can be substantially faster than being polynomial in the size of the structure. We achieve this by moving over to a larger signature σ' and a suitable σ'-expansion $\mathcal{A}'$ of $\mathcal{A}$. This is done in such a way that afterwards, we only have to loop through those tuples $\bar{w}$ that are close to the tuples $\bar{v} \in A^k$ that actually occur in the training set S. Meanwhile, the number of terms to check is not constant any more, but it depends on $\mathcal{A}$. However, as we discuss in the proof of Theorem 3.1, this number is polynomial in the degree of $\mathcal{A}$, which yields the desired running time. The exact details are provided by the following Lemma 3.2; this lemma is the main technical tool that allows us to prove Theorem 3.1.

For formulating the lemma, we need the following notation. In a structure $\mathcal{A}$ with universe A and of degree at most d, for every $v \in A$ and any radius $r \in \mathbb{N}$, we have $\left|N^{\mathcal{A}}_r(v)\right| \leqslant \nu_d(r) := 1 + d \cdot \sum_{0\leqslant i<r}(d-1)^i$. Note that $\nu_0(r) = 1$, $\nu_1(r) \leqslant 2$, $\nu_2(r) \leqslant 2r+1$, and $\nu_d(r) \leqslant d^{r+1}$ for all $r \in \mathbb{N}$ and $d \geqslant 3$. In particular, for a fixed radius $r \in \mathbb{N}$, $\nu_d(r)$ is polynomial in d. For all $r \in \mathbb{N}$, it is straightforward to construct an $\mathsf{FO}[\sigma]$-formula $\mathrm{dist}^{\sigma}_{\leqslant r}(x,y)$ such that for every σ-structure $\mathcal{A}$ and all $v,w \in A$, we have $\mathcal{A} \models \mathrm{dist}^{\sigma}_{\leqslant r}[v,w]$ if and only if $\mathrm{dist}^{\mathcal{A}}(v,w) \leqslant r$. To improve readability, we write $\mathrm{dist}^{\sigma}(x,y) \leqslant r$ instead of $\mathrm{dist}^{\sigma}_{\leqslant r}(x,y)$, and $\mathrm{dist}^{\sigma}(x,y) > r$ instead of $\neg\,\mathrm{dist}^{\sigma}_{\leqslant r}(x,y)$.

Let $r \in \mathbb{N}$. An $\mathsf{FOC}_1[\sigma]$-formula $\varphi(\bar{x})$ with free variables $\bar{x} = (x_1,\dots,x_k)$ is *r-local (around $\bar{x}$)* if for every σ-structure $\mathcal{A}$ with universe A and every tuple $\bar{v} = (v_1,\dots,v_k) \in A^k$, we have $\mathcal{A} \models \varphi[\bar{v}] \iff \mathcal{N}^{\mathcal{A}}_r(\bar{v}) \models \varphi[\bar{v}]$. A formula is *local* if it is r-local for some $r \in \mathbb{N}$. This notion of local formulas is identical with the one in [25]. It is very similar to the notion of local formulas by Gaifman [26], although we use a semantic notion instead of Gaifman’s syntactic notion.

For every $k \in \mathbb{N}_{\geqslant 1}$, every graph G with vertex set $[k]$, and every tuple $\bar{x} = (x_1,\dots,x_k)$ of k pairwise distinct variables, we consider the formula

$$\delta^{\sigma}_{G,r}(\bar{x}) := \bigwedge_{\{i,j\}\in E(G)} \mathrm{dist}^{\sigma}(x_i,x_j) \leqslant r \quad\wedge \bigwedge_{\{i,j\}\notin E(G)} \mathrm{dist}^{\sigma}(x_i,x_j) > r.$$

Note that $\mathcal{A} \models \delta^{\sigma}_{G,2r+1}[\bar{v}]$ means that the connected components of the r-neighbourhood $\mathcal{N}^{\mathcal{A}}_r(\bar{v})$ correspond to the connected components of G. Clearly, the formula $\delta^{\sigma}_{G,2r+1}(\bar{x})$ is r-local around its free variables $\bar{x}$.

For two sets A, N with $N \subseteq A$, for $k \in \mathbb{N}$, and two tuples $\bar{w},\bar{w}' \in A^k$, we write $\bar{w} \Rrightarrow_N \bar{w}'$ if $w_i = w'_i$ for all $i \in [k]$ with $w_i \in N$. Note that this notion is not symmetric: for tuples $\bar{w},\bar{w}'$ with $\bar{w} \Rrightarrow_N \bar{w}'$, the tuple $\bar{w}'$ may still have an entry $w'_i \in N$ for some $i \in [k]$ while $w_i \notin N$, so $w_i \neq w'_i$.

Our main technical ingredient for the proof of Theorem 3.1 is the following lemma. Note that in the final statement of the lemma, the graphs H are *connected* and the formulas ψ are local — both conditions are absolutely necessary for our proof of Theorem 3.1.

► **Lemma 3.2.** *Let σ be a signature, let $k \in \mathbb{N}_{\geqslant 1}$, let $\ell, q \in \mathbb{N}$, let $t(\bar{x},\bar{y})$ be an $\mathsf{FOC}_1[\sigma, k{+}\ell, q]$-term, and let I_t be the set of integers that occur in t.*

There is an extension σ_t of σ with relation symbols of arity $\leqslant 1$, and there are numbers $q_t, r_t \in \mathbb{N}$ such that, for every σ-structure $\mathcal{A}$ of size n and degree d, we can compute a σ_t-

expansion $\mathcal{A}_t$ of $\mathcal{A}$ in time $d^{\mathcal{O}(1)}\cdot n$ such that the following is true, where A denotes the universe of $\mathcal{A}$. For all $s \in \mathbb{N}_{\geqslant 1}$, for all $\bar{v}_1, \dots, \bar{v}_s \in A^k$, and for all $\bar{w} \in A^\ell$, there is an $\mathsf{FOC}_1[\sigma_t, k{+}\ell, q_t]$-term $t'(\bar{x}, \bar{y})$ such that for all $\bar{w}' \in A^\ell$ with $\bar{w} \Rrightarrow_{N_t} \bar{w}'$ for $N_t := N^{\mathcal{A}}_{(2r_t+1)(\ell+q)}(\bar{v}_1, \dots, \bar{v}_s)$, we have $t^{\mathcal{A}}[\bar{v}_i, \bar{w}] = (t')^{\mathcal{A}'}[\bar{v}_i, \bar{w}']$ for all $i \in [s]$.

Furthermore, t' is a combination via addition and multiplication of integers in I_t, of integers i with $-1 \leqslant i \leqslant \big(\ell \cdot \nu_d\big((2r_t{+}1)q\big)\big)^q$, and of counting terms of the form $\#\bar{z}'.(\psi \wedge \delta^{\sigma_t}_{H,2r_t+1})$ where ψ is an r_t-local formula in $\mathsf{FO}[\sigma_t]$ and H is a connected graph.

We present the proof of Lemma 3.2 in Section 4. Intuitively, the lemma says that we can translate a term t into a term t' over an extended signature such that the new term only needs those parameters that are close to the examples. By using this lemma, we can prove Theorem 3.1 as follows.

Proof of Theorem 3.1. Let σ, k, ℓ, q, I, T be chosen according to the assumption of the theorem.

Note that, up to logical equivalence, there are only finitely many terms in T. Thus, w.l.o.g. we assume that T is finite and that all terms in T only use $x_1, \dots, x_k, y_1, \dots, y_\ell, z_1, \dots, z_q$ as variables. For each term $t(\bar{x}, \bar{y})$ in T, we apply Lemma 3.2 to obtain an extension $\sigma_t \supseteq \sigma$ and numbers $q_t, r_t \in \mathbb{N}$. W.l.o.g. we assume that the sets $(\sigma_t \setminus \sigma)_{t\in T}$ are pairwise disjoint.

Let $\sigma' := \bigcup_{t\in T} \sigma_t$, let $q' := \max_{t\in T} q_t$, and let T' be the set of all $\mathsf{FOC}_1[\sigma', k{+}\ell, q']$-terms using only $x_1, \dots, x_k, y_1, \dots, y_\ell, z_1, \dots, z_{q'}$ as variables. Furthermore, let $r' := \max_{t\in T} r_t$.

Upon input of a σ-structure $\mathcal{A}$ of size n and degree d, for each $t \in T$ we use Lemma 3.2 to compute a σ_t-expansion $\mathcal{A}_t$ of $\mathcal{A}$ in time $d^{\mathcal{O}(1)}\cdot n$. We let $\mathcal{A}'$ be the σ'-expansion of $\mathcal{A}$ obtained by combining all the structures $\mathcal{A}_t$ for $t \in T$.

In addition, we also compute a lookup table that stores the value $g^{\mathcal{A}'} \in \mathbb{Z}$ for every ground term g that occurs in a term in T' and is of the form $\#\bar{z}'.(\psi \wedge \delta^{\sigma'}_{H,2r_t+1})$, where ψ is an r_t-local formula in $\mathsf{FO}[\sigma']$ (for some $t \in T$) and H is a connected graph.

▷ Claim 3.3. The lookup table can be computed in time $d^{\mathcal{O}(1)}\cdot n$.

Proof. First note that the number of entries in the lookup table is constant and does not depend on $\mathcal{A}$, because the terms in T' have size at most q' and only use variables from a fixed set. Thus, it suffices to show that every single entry of the lookup table can be computed in time $d^{\mathcal{O}(1)}\cdot n$.

Let g be a ground term that occurs in a term in T' and is of the form $\#\bar{z}'.(\psi \wedge \delta^{\sigma'}_{H,2r_t+1})$, where ψ is an r_t-local formula in $\mathsf{FO}[\sigma']$ (for some $t \in T$) and H is a connected graph. Then $m := |\bar{z}'| \leqslant q'$. Let $\tilde{r} := (2r' + 1)m$.

Since H is a connected graph and $(2r_t + 1)m \leqslant (2r' + 1)m = \tilde{r}$, we have $v_2, \dots, v_m \in N^{\mathcal{A}'}_{\tilde{r}}(v_1)$ for all $\bar{v} := (v_1, \dots, v_m) \in A^m$ with $\mathcal{A}' \models \delta^{\sigma}_{H,2r_t+1}[\bar{v}]$. Hence, to compute $g^{\mathcal{A}'}$, we can simply initialise the corresponding entry in the lookup table with 0, iterate over all $v_1 \in A$ and all $v_2, \dots, v_m \in N^{\mathcal{A}'}_{\tilde{r}}(v_1)$, and increase the entry in the lookup table by 1 if and only if $\mathcal{A}' \models (\psi \wedge \delta^{\sigma'}_{H,2r_t+1})[\bar{v}]$ holds. For every fixed $v_1 \in A$, we iterate over at most $\nu_d(\tilde{r})^{m-1} \in d^{\mathcal{O}(1)}$ tuples $(v_2, \dots, v_m)$. Moreover, since ψ and $\delta^{\sigma'}_{H,2r_t+1}$ are r_t-local, for each tuple $\bar{v} = (v_1, v_2, \dots, v_m)$, it holds that $\mathcal{A}' \models (\psi \wedge \delta^{\sigma'}_{H,2r_t+1})[\bar{v}]$ if and only if $\mathcal{N}^{\mathcal{A}'}_{r_t}(\bar{v}) \models (\psi \wedge \delta^{\sigma'}_{H,2r_t+1})[\bar{v}]$. The latter can be checked by building the neighbourhood structure around $\bar{v}$ in time polynomial in d, and then evaluating the formula (of constant size) on the neighbourhood structure by a brute-force algorithm in time polynomial in d.

All in all, we use $d^{\mathcal{O}(1)}\cdot n$ iterations, and every iteration takes time $d^{\mathcal{O}(1)}$, so the overall running time is $d^{\mathcal{O}(1)}\cdot n$. ◁

Now let us assume that we receive an arbitrary training set $S = \{(\bar{v}_1, \lambda_1), \dots, (\bar{v}_s, \lambda_s)\} \subseteq A^k \times \mathbb{Z}$ of size s. Let T^* be the set of all those terms $t'(\bar{x}, \bar{y})$ in T' that satisfy the following: t' is a combination via addition and multiplication of integers in I, of integers i with $-1 \leqslant i \leqslant \big(\ell \cdot \nu_d\big((2r'+1)q\big)\big)^q$, and of counting terms of the form $\#\bar{z}'.(\psi \wedge \delta^{\sigma'}_{H,2r_t+1})$, where ψ is an r_t-local formula in $\mathsf{FO}[\sigma']$ (for some $t \in T$) and H is a connected graph.

▷ Claim 3.4. T^* is of size $d^{\mathcal{O}(1)}$. Furthermore, for every $t'(\bar{x}, \bar{y}) \in T^*$, upon input of $\bar{a} \in A^k$ and $\bar{b} \in A^\ell$, we can compute $(t')^{\mathcal{A}'}[\bar{a}, \bar{b}]$ in time $d^{\mathcal{O}(1)}$ with only local access to $\mathcal{A}'$ and access to the precomputed lookup table.

Proof. Since the terms in T^* have length at most q', the variables come from a fixed finite set, the signature σ' has constant size, and the integers that may occur in a term in T^* come from a set of size polynomial in d, the total number of terms in T^* is also polynomial in d.

For the evaluation of a term t' in T^*, we first consider counting terms $t'(\bar{x}', \bar{y}')$ of the form $\#\bar{z}'.(\psi \wedge \delta^{\sigma'}_{H,2r_t+1})$, where $k' := |\bar{x}'| \leqslant k$, $\ell' := |\bar{y}'| \leqslant \ell$, $m' := |\bar{z}'| \leqslant q'$, ψ is an r_t-local formula in $\mathsf{FO}[\sigma']$ (for some $t \in T$), and H is a connected graph with $V(H) = [k' + \ell' + m']$.

In case that $k'+\ell' = 0$, the term t' is a ground term. Hence, we can simply use the precomputed lookup table to get the value $(t')^{\mathcal{A}'} \in \mathbb{Z}$ in time $O(1)$.

In case that $k'+\ell' > 0$, let $\bar{a} \in A^{k'}$ and $\bar{b} \in A^{\ell'}$. Since H is connected and $r_t \leqslant r'$, for all $\bar{c} = (c_1, \dots, c_{m'})$ with $\mathcal{A}' \models \delta^{\sigma'}_{H,2r_t+1}[\bar{a}, \bar{b}, \bar{c}]$, we have $c_1, \dots, c_{m'} \in N^{\mathcal{A}'}_{(2r'+1)m'}(\bar{a}, \bar{b})$.

Thus, for $\tilde{r} := (2r'+1)m' \leqslant (2r'+1)q'$, we have

$$\begin{aligned}(t')^{\mathcal{A}'}[\bar{a}, \bar{b}] &= \Big|\big\{\bar{c} \in A^{m'} \ : \ \mathcal{A}' \models (\psi \wedge \delta^{\sigma'}_{H,2r_t+1})[\bar{a}, \bar{b}, \bar{c}]\big\}\Big| \\ &= \Big|\big\{\bar{c} \in \big(N^{\mathcal{A}'}_{\tilde{r}}(\bar{a}, \bar{b})\big)^{m'} \ : \ \mathcal{A}' \models (\psi \wedge \delta^{\sigma'}_{H,2r_t+1})[\bar{a}, \bar{b}, \bar{c}]\big\}\Big| .\end{aligned}$$

Furthermore, ψ and $\delta^{\sigma'}_{H,2r_t+1}$ are r_t-local formulas, so $\mathcal{A}' \models (\psi \wedge \delta^{\sigma'}_{2r_t+1,H})[\bar{a}, \bar{b}, \bar{c}]$ if and only if $\mathcal{N}^{\mathcal{A}'}_{r_t}(\bar{a}, \bar{b}, \bar{c}) \models (\psi \wedge \delta^{\sigma}_{2r_t+1,H})[\bar{a}, \bar{b}, \bar{c}]$. Since the size of the neighbourhood is polynomial in d, the evaluation of $(t')^{\mathcal{A}'}[\bar{a}, \bar{b}]$ can be performed by evaluating an $\mathsf{FO}[\sigma']$-formula of constant size on a structure of size polynomial in d for a number of assignments that is polynomial in d. The evaluation of the formula can be done by building the neighbourhood structure around $\bar{a}, \bar{b}$ in time polynomial in d, and then evaluating the formula on the neighbourhood structure by a brute-force algorithm in time polynomial in d.

Now let t' be an arbitrary term in T^*. Then, for all $\bar{a} \in A^k$, $\bar{b} \in A^\ell$, $(t')^{\mathcal{A}'}[\bar{a}, \bar{b}]$ can be evaluated in time polynomial in d with only local access to $\mathcal{A}'$ and access to the precomputed lookup table by first evaluating every counting term in t' as described above and then combining the results and the integers occurring in t' via a constant number of additions and multiplications. ◁

To find a hypothesis that is consistent with S, we loop through all terms $t'(\bar{x}, \bar{y})$ in T^*. For each such term t', we loop through all tuples $\bar{w}' \in N^\ell$ for $N := N^{\mathcal{A}}_{(2r'+1)(\ell+q)}(\bar{v}_1, \dots, \bar{v}_s)$. We check if $(t')^{\mathcal{A}'}[\bar{v}_i, \bar{w}'] = \lambda_i$ for all $i \in [s]$. If so, we stop and output t' and $\bar{w}'$, indicating that $h^{\mathcal{A}'}_{t',\bar{w}'}$ is a hypothesis consistent with S. Otherwise, we stop and output "reject".

▷ Claim 3.5. This algorithm uses only local access to $\mathcal{A}'$ and access to the precomputed lookup table, and it terminates within time $(s + d)^{\mathcal{O}(1)}$. If it outputs a hypothesis, this hypothesis is consistent with S. If it outputs "reject", there is no term $t \in T$ and tuple $\bar{w} \in A^\ell$ such that $h^{\mathcal{A}}_{t,\bar{w}}$ is consistent with S.

Proof. The set N contains at most $s \cdot k \cdot \nu_d\big((2r'+1)(\ell+q)\big)$ elements, which is polynomial in s and d. By Claim 3.4, T^* is of size $d^{\mathcal{O}(1)}$. Thus, in total, we iterate over $(s + d)^{\mathcal{O}(1)}$

hypotheses. For every hypothesis consisting of a term $t' \in T^*$ and a tuple $\bar{w}' \in A^\ell$, by Claim 3.4, we can check in time $d^{\mathcal{O}(1)}{\cdot}s$ if $(t')^{\mathcal{A}'}[\bar{v}_i, \bar{w}'] = \lambda_i$ for all $i \in [s]$. This check uses only local access to $\mathcal{A}'$ and access to the precomputed lookup table. This proves the first statement of the claim. The second statement of the claim is obvious.

To prove the third statement of the claim, let us assume that there exists a term $t \in T$ and a tuple $\bar{w} \in A^\ell$ such that $h^{\mathcal{A}}_{t,\bar{w}}$ is consistent with S. For this particular choice of t and $\bar{w}$, and for the given tuples $\bar{v}_1, \ldots, \bar{v}_s$, Lemma 3.2 yields an $\mathsf{FOC}_1[\sigma_t, k{+}\ell, q_t]$-term $t'(\bar{x}, \bar{y})$ such that for all $\bar{w}' \in A^\ell$ with $\bar{w} \Rrightarrow_{N_t} \bar{w}'$ for $N_t := N^{\mathcal{A}}_{(2r_t+1)(\ell+q)}(\bar{v}_1, \ldots, \bar{v}_s)$, we have

$$t^{\mathcal{A}}[\bar{v}_i, \bar{w}] \;=\; (t')^{\mathcal{A}'}[\bar{v}_i, \bar{w}'] \quad \text{for all } i \in [s].$$

Note that $N_t \subseteq N$. Hence, our algorithm will consider at least one $\bar{w}' \in N^\ell$ such that $\bar{w} \Rrightarrow_{N_t} \bar{w}'$. Let us fix such a $\bar{w}'$. All that remains to be done is to show that $t' \in T^*$ — this will imply that our algorithm will eventually consider $t', \bar{w}'$ and then stop and output t' and $\bar{w}'$, indicating that $h^{\mathcal{A}'}_{t',\bar{w}'}$ is a hypothesis consistent with S.

By our choice of σ' and q', we know that $\sigma' \supseteq \sigma_t$ and $q' \geqslant q_t$. Thus, $t' \in T'$. From Lemma 3.2, we know that t' is a combination via addition and multiplication of integers in I, of integers i with $-1 \leqslant i \leqslant \big(\ell \cdot \nu_d\big((2r_t{+}1)q\big)\big)^q$, and of counting terms of the form $\#\bar{z}'.(\psi \wedge \delta^{\sigma'}_{H,2r_t+1})$, where ψ is an r_t-local formula in $\mathsf{FO}[\sigma_t]$ and H is a connected graph. By our particular choice of T^* and since $r' \geqslant r_t$, we obtain that $t' \in T^*$. This completes the proof of Claim 3.5. $\lhd$

In summary, the proof of Theorem 3.1 is complete. ◀

4 Proof of Lemma 3.2

The proof of Lemma 3.2 heavily relies on the following *localisation theorem for* FOC_1. This theorem is implicit in [34]; here we present it in a way analogous to [10, Theorem 4.7].

▶ **Theorem 4.1** (Localisation Theorem for FOC_1, [34]). *Let $k \in \mathbb{N}$, and let σ be a signature. For every $\mathsf{FOC}_1[\sigma]$-formula $\varphi(x_1, \ldots, x_k)$, there is an extension σ_φ of σ with relation symbols of arity $\leqslant 1$, and an $\mathsf{FO}[\sigma_\varphi]$-formula $\varphi'(x_1, \ldots, x_k)$ that is a Boolean combination of local formulas and statements of the form $R()$ where $R \in \sigma_\varphi$ has arity 0, for which the following holds. There is an algorithm that, upon input of a σ-structure $\mathcal{A}$ of size n and degree d, computes in time $d^{\mathcal{O}(1)}{\cdot}n$ a σ_φ-expansion $\mathcal{A}_\varphi$ of $\mathcal{A}$ such that for all $\bar{v} \in A^k$ (where A denotes the universe of $\mathcal{A}$), we have $\mathcal{A}_\varphi \models \varphi'[\bar{v}] \iff \mathcal{A} \models \varphi[\bar{v}]$.*

For the proof of Lemma 3.2, let $\sigma, k, \ell, q, t(\bar{x}, \bar{y}), I_t$ be chosen according to the assumption of the lemma. In particular, $t(\bar{x}, \bar{y})$ is an $\mathsf{FOC}_1[\sigma, k{+}\ell, q]$-term, and I_t is the set of integers that occur in t.

We first note that it suffices to prove the statement of the lemma for the particular case where $t(\bar{x}, \bar{y})$ is of the form $\#\bar{z}.\varphi(\bar{x}, \bar{y}, \bar{z})$ for some $\mathsf{FOC}_1[\sigma]$-formula φ. Assume for now that the statement of the lemma holds for all terms of this form, and consider an arbitrary $\mathsf{FOC}_1[\sigma, k{+}\ell, q]$-term $t(\bar{x}, \bar{y})$ that is *not* of this form. Then, t is a combination via addition and multiplication of integers from I_t and of counting terms u of the form $\#\bar{z}.\varphi(\bar{x}, \bar{y}, \bar{z})$ for some $\mathsf{FOC}_1[\sigma]$-formula φ. Let U be the set of all these counting terms u occurring in t. We assume that the statement of the lemma holds for each $u \in U$. Hence, for each $u \in U$, we obtain an extension σ_u of σ with relation symbols of arity $\leqslant 1$ and numbers $q_u, r_u \in \mathbb{N}$. W.l.o.g. we assume that the sets $(\sigma_u \setminus \sigma)_{u \in U}$ are pairwise disjoint. Let $\sigma_t := \bigcup_{u \in U} \sigma_u$, $r_t := \max_{u \in U} r_u$, and $q_t := q \cdot \max_{u \in U} q_u$.

Upon input of a σ-structure $\mathcal{A}$ of size n and degree d, for each $u \in U$, the statement of the lemma for u enables us to compute a σ_u-expansion $\mathcal{A}_u$ of $\mathcal{A}$ in time $d^{\mathcal{O}(1)} \cdot n$. We let $\mathcal{A}_t$ be the σ_t-expansion of $\mathcal{A}$ obtained by combining all the structures $\mathcal{A}_u$ for all counting terms $u \in U$. Let A denote the universe of $\mathcal{A}$.

When given $\bar{v}_1, \ldots, \bar{v}_s \in A^k$ and $\bar{w} \in A^\ell$, the statement of the lemma for u yields an $\mathsf{FOC}_1[\sigma_u, k+\ell, q_u]$-term u' for each $u \in U$. We choose t' to be the term obtained from t by replacing every occurrence of a counting term $u \in U$ by the corresponding counting term u'. Clearly t' has length at most q_t. It is not difficult to verify that t' has the desired properties stated in Lemma 3.2.

All that remains to be done is to prove the statement of the lemma for the particular case where $t(\bar{x}, \bar{y})$ is of the form $\#\bar{z}.\varphi(\bar{x}, \bar{y}, \bar{z})$ for some $\mathsf{FOC}_1[\sigma]$-formula φ. Given such a term t, let $m := |\bar{z}|$. Using Theorem 4.1, we obtain an extension $\sigma' := \sigma_\varphi$ of σ with relation symbols of arity $\leqslant 1$ and an $\mathsf{FO}[\sigma']$-formula $\varphi'(\bar{x}, \bar{y}, \bar{z})$ that is a Boolean combination of local formulas and statements of the form $R()$, where $R \in \sigma'$ has arity 0, for which the following holds. Given a σ-structure $\mathcal{A}$ of size n and degree d and with universe A, we can compute a σ'-expansion $\mathcal{A}' := \mathcal{A}_\varphi$ of $\mathcal{A}$ in time $d^{\mathcal{O}(1)} \cdot n$ such that for all $\bar{a} \in A^k$, $\bar{b} \in A^\ell$, and $\bar{c} \in A^m$, we have $\mathcal{A} \models \varphi[\bar{a}, \bar{b}, \bar{c}] \iff \mathcal{A}' \models \varphi'[\bar{a}, \bar{b}, \bar{c}]$. Thus, for $\tilde{t}(\bar{x}, \bar{y}) := \#\bar{z}.\varphi'(\bar{x}, \bar{y}, \bar{z})$, we have $\tilde{t}^{\mathcal{A}'}[\bar{a}, \bar{b}] = t^{\mathcal{A}}[\bar{a}, \bar{b}]$ for all $\bar{a} \in A^k$, $\bar{b} \in A^\ell$.

From the particular shape of φ', we obtain that there exists a number $r' \in \mathbb{N}$ such that φ' is r'-local, i.e., for all $\bar{a} \in A^k$, $\bar{b} \in A^\ell$, $\bar{c} \in A^m$, we have $\mathcal{A}' \models \varphi'[\bar{a}, \bar{b}, \bar{c}] \iff \mathcal{N}_{r'}^{\mathcal{A}'}(\bar{a}, \bar{b}, \bar{c}) \models \varphi'[\bar{a}, \bar{b}, \bar{c}]$.

Let $\mathcal{G}$ be the set of undirected graphs with vertex set $[k + \ell + m]$. For every $G \in \mathcal{G}$, consider the formula $\varphi'_G(\bar{x}, \bar{y}, \bar{z}) := \varphi'(\bar{x}, \bar{y}, \bar{z}) \ \wedge \ \delta^{\sigma'}_{G, 2r'+1}(\bar{x}, \bar{y}, \bar{z})$ and the term $u_G(\bar{x}, \bar{y}) := \#\bar{z}.\varphi'_G(\bar{x}, \bar{y}, \bar{z})$. It is not difficult to verify that for all $\bar{a} \in A^k$, $\bar{b} \in A^\ell$, we have

$$t^{\mathcal{A}}[\bar{a}, \bar{b}] \quad = \quad \tilde{t}^{\mathcal{A}'}[\bar{a}, \bar{b}] \quad = \quad \sum_{G \in \mathcal{G}} (u_G)^{\mathcal{A}'}[\bar{a}, \bar{b}].$$

To further decompose the terms u_G, we use techniques similar to the ones used in [34] to decompose terms into so-called connected local terms. With these, we obtain the following technical lemma. The statement of this lemma as well as its proof are highly non-trivial. It depends on a careful analysis of the connected components of undirected graphs with $k+\ell+m$ nodes. Note that in the final statement of the lemma, the graphs H are *connected* and the formulas ψ are local — both conditions are also stated in Lemma 3.2 and are absolutely necessary for our proof of Theorem 3.1.

▶ **Lemma 4.2.** *Let σ' be a signature and let $r', k, \ell, m \in \mathbb{N}$ with $k+\ell+m \geqslant 1$. Let $\varphi'(\bar{x}, \bar{y}, \bar{z})$ be an r'-local $\mathsf{FO}[\sigma']$-formula with $|\bar{x}| = k$, $|\bar{y}| = \ell$, and $|\bar{z}| = m$. Let $\mathcal{G}$ be the set of all undirected graphs with vertex set $[k + \ell + m]$. Let $G \in \mathcal{G}$ and let*

$$u_G(\bar{x}, \bar{y}) \quad := \quad \#\bar{z}.\big(\varphi'(\bar{x}, \bar{y}, \bar{z}) \wedge \delta^{\sigma'}_{G, 2r'+1}(\bar{x}, \bar{y}, \bar{z})\big).$$

There is a number $q_G \in \mathbb{N}$ such that, for every σ'-structure $\mathcal{A}'$ of degree d and with universe A, for all $s \in \mathbb{N}_{\geqslant 1}$, for all $\bar{v}_1, \ldots, \bar{v}_s \in A^k$, and for all $\bar{w} \in A^\ell$, there exists an $\mathsf{FOC}_1[\sigma', k+\ell, q_G]$-term $u'_G(\bar{x}, \bar{y})$ such that the following is true for $N := N^{\mathcal{A}'}_{(2r'+1)(\ell+m)}(\bar{v}_1, \ldots, \bar{v}_s)$.

For all $\bar{w}' \in A^\ell$ with $\bar{w} \Rrightarrow_N \bar{w}'$, we have

$$(u_G)^{\mathcal{A}'}[\bar{v}_i, \bar{w}] \quad = \quad (u'_G)^{\mathcal{A}'}[\bar{v}_i, \bar{w}'] \quad \text{for all } i \in [s].$$

Furthermore, u'_G is a combination via addition and multiplication of integers i with $-1 \leqslant i \leqslant \Big(\ell \cdot \nu_d\big((2r'+1)m\big)\Big)^m$ and of counting terms of the form $\#\bar{z}'.(\psi \wedge \delta^{\sigma'}_{H, 2r'+1})$, where $|\bar{z}'| \leqslant |\bar{z}|$, for a connected graph H and an r'-local formula ψ.

Before proving Lemma 4.2, we first finish the proof of Lemma 3.2. We apply Lemma 4.2 to the term u_G for all $G \in \mathcal{G}$. For each $G \in \mathcal{G}$ this yields a number $q_G \in \mathbb{N}$. We choose $r_t := r'$ and $\sigma_t := \sigma'$.

For any σ-structure $\mathcal{A}$, we let $\mathcal{A}'$ be the σ'-expansion obtained as described above (by applying Theorem 4.1 to the formula φ). When given tuples $\bar{v}_1, \ldots, \bar{v}_s \in A^k$ and $\bar{w} \in A^\ell$, we obtain from Lemma 3.2 an $\mathsf{FOC}_1[\sigma', k{+}\ell, q_G]$-term $u'_G(\bar{x}, \bar{y})$, for every $G \in \mathcal{G}$. This term u'_G is a combination via addition and multiplication of integers i with $-1 \leqslant i \leqslant \Big(\ell \cdot \nu_d\big((2r'{+}1)m\big)\Big)^m$ and of counting terms of the form $\#\bar{z}'.(\psi \wedge \delta^{\sigma'}_{H,2r'+1})$, where $|\bar{z}'| \leqslant |\bar{z}|$, for a connected graph H and an r'-local formula ψ. We choose

$$t'(\bar{x}, \bar{y}) \;:=\; \sum_{G \in \mathcal{G}} u'_G(\bar{x}, \bar{y})$$

and let $q_t := \sum_{G \in \mathcal{G}} (q_G + 3)$. Note that t' has length $\leqslant q_t$ and that $m \leqslant q$. It is not difficult to verify that the term t' has the desired properties stated in Lemma 3.2.

All that remains to be done to complete the proof of Lemma 3.2 is to prove Lemma 4.2.

Proof of Lemma 4.2. We prove the statement by induction on the number c of connected components of G.

Consider the induction base $c = 1$, i. e., the graph G is connected. We choose q_G to be the length of the term u_G. For choosing the term u'_G, we distinguish between 4 cases.

Case 1: $k = \ell = 0$. In this case, we set $u'_G := u_G$.

Case 2: $k = 0$ and $\ell > 0$. In this case, we set $u'_G := i$ for the particular number $i := (u_G)^{\mathcal{A}'}[\bar{w}] \in \mathbb{Z}$. Since G is connected, all elements in a tuple $\bar{c} \in A^m$ with $\mathcal{A}' \models \delta^{\sigma'}_{G,2r'+1}[\bar{w}, \bar{c}]$ have distance at most $(2r'{+}1)m$ from $\bar{w}$. Therefore, $0 \leqslant i \leqslant \big(\ell \cdot \nu_d\big((2r'{+}1)m\big)\big)^m$.

Case 3: $k > 0$ and $\bar{w} \notin N^\ell$. Since G is connected, for all $i \in [s]$ and all $\bar{c} \in A^m$, it holds that $\mathcal{A}' \not\models \delta^{\sigma'}_{G,2r'+1}[\bar{v}_i, \bar{w}, \bar{c}]$. Hence, for all $i \in [s]$ we have $(u_G)^{\mathcal{A}'}[\bar{v}_i, \bar{w}] = 0$. Therefore, we can choose $u'_G := 0$.

Case 4: $k > 0$ and $\bar{w} \in N^\ell$. In this case, $\bar{w}$ is the only tuple $\bar{w}'$ that satisfies $\bar{w} \Rrightarrow_N \bar{w}'$. Hence, we can choose $u'_G := u_G$.

This completes the induction base.

We now turn to the induction step, where $c > 1$. We assume that the statement of the lemma holds for all graphs G' with fewer than c connected components. Let C_1 be the set of all vertices of G that are in the same connected component as the vertex 1. Let $C_2 := V(G) \setminus C_1$. For each $j \in \{1, 2\}$, let $G[C_j]$ be the induced subgraph of G with vertex set C_j. Clearly, G is the disjoint union of the graphs $G[C_1]$ and $G[C_2]$, and $G[C_1]$ has only one connected component, and $G[C_2]$ has $c{-}1$ connected components. For $j \in \{1, 2\}$, let $\bar{x}_j$ be the tuple of variables of $\bar{x}$ such that their index is contained in C_j, let $\bar{y}_j$ be the tuple of variables of $\bar{y}$ such that their index $+k$ is contained in C_j, and let $\bar{z}_j$ be the tuple of variables of $\bar{z}$ such that their index $+k{+}\ell$ is contained in C_j.

By using the Feferman-Vaught Theorem (cf., [24, 44, 10]), we obtain a decomposition Δ of $\varphi'(\bar{x}, \bar{y}, \bar{z})$ w.r.t. $(\bar{x}_1\bar{y}_1\bar{z}_1; \bar{x}_2\bar{y}_2\bar{z}_2)$. I. e., Δ is a finite non-empty set of pairs of r'-local $\mathsf{FO}[\sigma']$-formulas of the form $(\alpha(\bar{x}_1, \bar{y}_1, \bar{z}_1), \beta(\bar{x}_2, \bar{y}_2, \bar{z}_2))$ such that for all $\bar{a} \in A^k$, $\bar{b} \in A^\ell$, $\bar{c} \in A^m$, we have $\mathcal{A}' \models \varphi'[\bar{a}, \bar{b}, \bar{c}] \iff$ there exists a pair $(\alpha, \beta) \in \Delta$ such that $\mathcal{A}' \models \alpha[\bar{a}_1, \bar{b}_1, \bar{c}_1]$ and $\mathcal{A}' \models \beta[\bar{a}_2, \bar{b}_2, \bar{c}_2]$. Here, $\bar{a}_j, \bar{b}_j, \bar{c}_j$ are defined analogously to $\bar{x}_j, \bar{y}_j, \bar{z}_j$ for $j \in \{1, 2\}$. Moreover, the pairs in Δ are mutually exclusive, i. e., for all $\bar{a} \in A^k$, $\bar{b} \in A^\ell$, $\bar{c} \in A^m$, there is at most one pair $(\alpha, \beta) \in \Delta$ such that $\mathcal{A}' \models \alpha[\bar{a}_1, \bar{b}_1, \bar{c}_1]$ and $\mathcal{A}' \models \beta[\bar{a}_2, \bar{b}_2, \bar{c}_2]$.

Let $\mathcal{G}_{\neg G}$ be the set of graphs $H \neq G$ with vertex set $V(H) = [k{+}\ell{+}m]$ and $G[C_1] = H[C_1]$ and $G[C_2] = H[C_2]$. Note that such graphs H have strictly fewer connected components than G.

For every pair $(\alpha, \beta) \in \Delta$, we set

$$\begin{aligned} t_{\alpha,\beta}(\bar{x},\bar{y}) &:= \#\bar{z}.\big(\alpha(\bar{x}_1,\bar{y}_1,\bar{z}_1) \wedge \beta(\bar{x}_2,\bar{y}_2,\bar{z}_2) \wedge \delta^{\sigma'}_{G,2r'+1}(\bar{x},\bar{y},\bar{z})\big), \\ t_{\alpha,1}(\bar{x}_1,\bar{y}_1) &:= \#\bar{z}_1.\big(\alpha(\bar{x}_1,\bar{y}_1,\bar{z}_1) \wedge \delta^{\sigma'}_{G[C_1],2r'+1}(\bar{x}_1,\bar{y}_1,\bar{z}_1))\big), \\ t_{\beta,2}(\bar{x}_2,\bar{y}_2) &:= \#\bar{z}_2.\big(\beta(\bar{x}_2,\bar{y}_2,\bar{z}_2) \wedge \delta^{\sigma'}_{G[C_2],2r'+1}(\bar{x}_2,\bar{y}_2,\bar{z}_2))\big), \\ t_{\alpha,\beta,\neg G}(\bar{x},\bar{y}) &:= \sum_{H \in \mathcal{G}_{\neg G}} \#\bar{z}.\big(\alpha(\bar{x}_1,\bar{y}_1,\bar{z}_1) \wedge \beta(\bar{x}_2,\bar{y}_2,\bar{z}_2) \wedge \ \delta^{\sigma'}_{H,2r'+1}(\bar{x},\bar{y},\bar{z})\big). \end{aligned}$$

Since the pairs $(\alpha, \beta) \in \Delta$ are mutually exclusive, for all $\bar{a} \in A^k$, $\bar{b} \in A^\ell$, we have $(u_G)^{\mathcal{A}'}[\bar{a},\bar{b}] = \sum_{(\alpha,\beta)\in\Delta} t^{\mathcal{A}'}_{\alpha,\beta}[\bar{a},\bar{b}]$. Furthermore, it is not difficult to verify that

$$t^{\mathcal{A}'}_{\alpha,\beta}[\bar{a},\bar{b}] \quad = \quad t^{\mathcal{A}'}_{\alpha,1}[\bar{a}_1,\bar{b}_1] \cdot t^{\mathcal{A}'}_{\beta,2}[\bar{a}_2,\bar{b}_2] \quad - \quad t^{\mathcal{A}'}_{\alpha,\beta,\neg G}[\bar{a},\bar{b}],$$

where $\bar{a}_j, \bar{b}_j$ are defined analogously to $\bar{x}_j, \bar{y}_j$ for $j \in \{1, 2\}$.

Using the induction hypothesis, we apply the statement of the lemma to $t_{\alpha,1}$, $t_{\beta,2}$, and to every summand of $t_{\alpha,\beta,\neg G}$, and we call the resulting terms $t'_{\alpha,1}$, $t'_{\beta,2}$, and $t'_{\alpha,\beta,\neg G}$.

We set $t'_{\alpha,\beta}(\bar{x},\bar{y}) := t'_{\alpha,1}(\bar{x}_1,\bar{y}_1) \cdot t'_{\beta,2}(\bar{x}_2,\bar{y}_2) - t'_{\alpha,\beta,\neg G}(\bar{x},\bar{y})$ and choose $u'_G(\bar{x},\bar{y}) := \sum_{(\alpha,\beta)\in\Delta} t'_{\alpha,\beta}(\bar{x},\bar{y})$.

It is not difficult to verify that the term u'_G has the desired properties. Moreover, the size of u'_G can be bounded by a number q_G that only depends on u_G (but not on $\mathcal{A}'$). This completes the proof of Lemma 4.2. Hence, also the proof of Lemma 3.2 now is complete. ◀

5 Conclusion

We have studied the complexity of learning aggregate queries from examples. For this, we have extended the framework for Boolean classification problems by Grohe and Turán [35] to multiclass classification problems with integer-valued classifiers. In our setting, such classifiers are represented by a pair $(t, \bar{w})$ where $t(\bar{x}, \bar{y})$ is a counting term in FOC_1 and $\bar{w}$ is a tuple of elements in the universe A of the given database $\mathcal{A}$.

Our main result shows that we can build a suitable index structure on the given database $\mathcal{A}$ during a precomputation step whose running time is linear in the size and polynomial in the degree of $\mathcal{A}$. Afterwards, by utilising this index structure, whenever receiving as input a new training set S, a classifier definable in FOC_1 can be learned in time polynomial in the degree of $\mathcal{A}$ and polynomial in the number of examples given in the training set. The classifiers returned by our algorithm can be evaluated efficiently, in time polynomial in the degree of $\mathcal{A}$. Moreover, after having built the index structure, all our algorithms require only local access to the database.

It seems conceivable that our results can be generalised to the more expressive logic FOWA_1 that operates on weighted structures [10], since the locality results obtained for this logic in [10] are similar to the ones obtained for FOC_1 in [34] and used here.

For the *Boolean* classification problem based on FO, [9] shows that the learning problem is fixed-parameter tractable on nowhere dense classes. This result heavily relies on the fixed-parameter tractability of the evaluation problem for FO on nowhere dense classes [31]. From [34] it is known that on nowhere dense classes also the evaluation problem for formulas and terms of FOC_1 is fixed-parameter tractable. We therefore think that FOC_1 is a good candidate for a logic with a fixed-parameter tractable *integer-valued* classification problem on classes of sparse structures.

In this paper, the task in the learning problem is to find an integer-valued hypothesis that is consistent with the training set. Other publications on the considered framework for *Boolean* classification problems also study settings in which one wants to find a hypothesis that *generalises well* [33, 32, 29, 7, 10, 9, 8]. More specifically, they study *probably approximately correct (PAC)* learning tasks in which the training examples are considered as being generated from a (fixed, but unknown) probability distribution. The task is to find a hypothesis that has a small expected error on new examples generated from the same distribution. While PAC-learning algorithms for Boolean classification problems are typically based on the empirical risk minimisation (ERM) principle [48], results in algorithmic learning theory for multiclass classification [14, 21, 36] are based on different principles and even show that the ERM principle in general does not suffice for PAC learning with an unbounded number of classes [22]. Therefore, we expect it to be quite challenging to obtain PAC-learning results for our framework of multiclass classification problems.

We plan to work on the raised questions in future work.

References

1 Azza Abouzied, Dana Angluin, Christos H. Papadimitriou, Joseph M. Hellerstein, and Avi Silberschatz. Learning and verifying quantified Boolean queries by example. In *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2013*, pages 49–60. ACM, 2013. doi:10.1145/2463664.2465220.

2 Howard Aizenstein, Tibor Hegedüs, Lisa Hellerstein, and Leonard Pitt. Complexity theoretic hardness results for query learning. *Comput. Complex.*, 7(1):19–53, 1998. doi:10.1007/PL00001593.

3 Bogdan Alexe, Balder ten Cate, Phokion G. Kolaitis, and Wang Chiew Tan. Characterizing schema mappings via data examples. *ACM Trans. Database Syst.*, 36(4):23:1–23:48, 2011. doi:10.1145/2043652.2043656.

4 Dana Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1987. doi:10.1007/BF00116828.

5 Pablo Barceló, Alexander Baumgartner, Victor Dalmau, and Benny Kimelfeld. Regularizing conjunctive features for classification. *J. Comput. Syst. Sci.*, 119:97–124, 2021. doi:10.1016/j.jcss.2021.01.003.

6 Pablo Barceló and Miguel Romero. The complexity of reverse engineering problems for conjunctive queries. In *20th International Conference on Database Theory, ICDT 2017*, volume 68 of *LIPIcs*, pages 7:1–7:17, 2017. doi:10.4230/LIPIcs.ICDT.2017.7.

7 Steffen van Bergerem. Learning concepts definable in first-order logic with counting. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019*, pages 1–13. IEEE, 2019. doi:10.1109/LICS.2019.8785811.

8 Steffen van Bergerem. *Descriptive Complexity of Learning.* PhD thesis, RWTH Aachen University, Germany, 2023. doi:10.18154/RWTH-2023-02554.

9 Steffen van Bergerem, Martin Grohe, and Martin Ritzert. On the parameterized complexity of learning first-order logic. In *PODS 2022: International Conference on Management of Data*, pages 337–346. ACM, 2022. doi:10.1145/3517804.3524151.

10 Steffen van Bergerem and Nicole Schweikardt. Learning concepts described by weight aggregation logic. In *29th EACSL Annual Conference on Computer Science Logic, CSL 2021, Ljubljana, Slovenia (Virtual Conference), January 25-28, 2021*, volume 183 of *LIPIcs*, pages 10:1–10:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.CSL.2021.10.

11 Angela Bonifati, Radu Ciucanu, and Aurélien Lemay. Learning path queries on graph databases. In *Proceedings of the 18th International Conference on Extending Database Technology, EDBT 2015*, pages 109–120. OpenProceedings.org, 2015. doi:10.5441/002/edbt.2015.11.

12 Angela Bonifati, Radu Ciucanu, and Slawek Staworko. Learning join queries from user examples. *ACM Trans. Database Syst.*, 40(4):24:1–24:38, 2016. doi:10.1145/2818637.

13 Angela Bonifati, Ugo Comignani, Emmanuel Coquery, and Romuald Thion. Interactive mapping specification with exemplar tuples. *ACM Trans. Database Syst.*, 44(3):10:1–10:44, 2019. doi:10.1145/3321485.

14 Nataly Brukhim, Daniel Carmon, Irit Dinur, Shay Moran, and Amir Yehudayoff. A characterization of multiclass learnability. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022*, pages 943–955. IEEE, 2022. doi:10.1109/FOCS54457.2022.00093.

15 Balder ten Cate and Victor Dalmau. Conjunctive queries: Unique characterizations and exact learnability. In *24th International Conference on Database Theory, ICDT 2021*, volume 186 of *LIPIcs*, pages 9:1–9:24, 2021. doi:10.4230/LIPIcs.ICDT.2021.9.

16 Balder ten Cate, Victor Dalmau, and Phokion G. Kolaitis. Learning schema mappings. *ACM Trans. Database Syst.*, 38(4):28:1–28:31, 2013. doi:10.1145/2539032.2539035.

17 Balder ten Cate, Phokion G. Kolaitis, Kun Qian, and Wang-Chiew Tan. Active learning of GAV schema mappings. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2018*, pages 355–368. ACM, 2018. doi:10.1145/3196959.3196974.

18 Adrien Champion, Tomoya Chiba, Naoki Kobayashi, and Ryosuke Sato. ICE-based refinement type discovery for higher-order functional programs. *J. Autom. Reason.*, 64(7):1393–1418, 2020. doi:10.1007/s10817-020-09571-y.

19 William W. Cohen and C. David Page Jr. Polynomial learnability and inductive logic programming: Methods and results. *New Gener. Comput.*, 13(3&4):369–409, 1995. doi:10.1007/BF03037231.

20 Andrew Cropper, Sebastijan Dumancic, Richard Evans, and Stephen H. Muggleton. Inductive logic programming at 30. *Mach. Learn.*, 111(1):147–172, 2022. doi:10.1007/s10994-021-06089-1.

21 Amit Daniely, Sivan Sabato, and Shai Shalev-Shwartz. Multiclass learning approaches: A theoretical comparison with implications. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012*, pages 494–502, 2012. URL: https://proceedings.neurips.cc/paper/2012/hash/19f3cd308f1455b3fa09a282e0d496f4-Abstract.html.

22 Amit Daniely and Shai Shalev-Shwartz. Optimal learners for multiclass problems. In Maria-Florina Balcan, Vitaly Feldman, and Csaba Szepesvári, editors, *Proceedings of The 27th Conference on Learning Theory, COLT 2014, Barcelona, Spain, June 13-15, 2014*, volume 35 of *JMLR Workshop and Conference Proceedings*, pages 287–316. JMLR.org, 2014. URL: http://proceedings.mlr.press/v35/daniely14b.html.

23 P. Ezudheen, Daniel Neider, Deepak D'Souza, Pranav Garg, and P. Madhusudan. Horn-ICE learning for synthesizing invariants and contracts. *Proc. ACM Program. Lang.*, Volume 2(Issue OOPSLA):131:1–131:25, 2018. doi:10.1145/3276501.

24 Solomon Feferman and Robert L. Vaught. The first-order properties of products of algebraic systems. *Fundamenta Mathematicae*, 47:57–103, 1959.

25 Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006. doi:10.1007/3-540-29953-X.

26 Haim Gaifman. On local and non-local properties. In Jacques Stern, editor, *Proceedings of the Herbrand Symposium*, volume 107 of *Studies in Logic and the Foundations of Mathematics*, pages 105–135. North-Holland, 1982. doi:10.1016/S0049-237X(08)71879-2.

27 Pranav Garg, Christof Löding, P. Madhusudan, and Daniel Neider. ICE: A robust framework for learning invariants. In *Computer Aided Verification - 26th International Conference, CAV*

2014, volume 8559 of *Lecture Notes in Computer Science*, pages 69–87. Springer, 2014. URL: https://doi.org/10.1007/978-3-319-08867-9_5, doi:10.1007/978-3-319-08867-9_5.

28 Georg Gottlob and Pierre Senellart. Schema mapping discovery from data instances. *J. ACM*, 57(2):6:1–6:37, 2010. doi:10.1145/1667053.1667055.

29 Emilie Grienenberger and Martin Ritzert. Learning definable hypotheses on trees. In *22nd International Conference on Database Theory, ICDT 2019*, pages 24:1–24:18, 2019. doi:10.4230/LIPIcs.ICDT.2019.24.

30 Martin Grohe. Logic, graphs, and algorithms. In Jörg Flum, Erich Grädel, and Thomas Wilke, editors, *Logic and Automata: History and Perspectives [in Honor of Wolfgang Thomas]*, volume 2 of *Texts in Logic and Games*, pages 357–422. Amsterdam University Press, 2008.

31 Martin Grohe, Stephan Kreutzer, and Sebastian Siebertz. Deciding first-order properties of nowhere dense graphs. *J. ACM*, 64(3):17:1–17:32, 2017. doi:10.1145/3051095.

32 Martin Grohe, Christof Löding, and Martin Ritzert. Learning MSO-definable hypotheses on strings. In *International Conference on Algorithmic Learning Theory, ALT 2017*, pages 434–451, 2017. URL: http://proceedings.mlr.press/v76/grohe17a.html.

33 Martin Grohe and Martin Ritzert. Learning first-order definable concepts over structures of small degree. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017*, pages 1–12, 2017. doi:10.1109/LICS.2017.8005080.

34 Martin Grohe and Nicole Schweikardt. First-order query evaluation with cardinality conditions. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2018*, pages 253–266, 2018. doi:10.1145/3196959.3196970.

35 Martin Grohe and György Turán. Learnability and definability in trees and similar structures. *Theory Comput. Syst.*, 37(1):193–220, 2004. doi:10.1007/s00224-003-1112-8.

36 Steve Hanneke, Shay Moran, and Qian Zhang. Universal rates for multiclass learning. In *The 36th Annual Conference on Learning Theory, COLT 2023*, volume 195 of *Proceedings of Machine Learning Research*, pages 5615–5681. PMLR, 2023. URL: https://proceedings.mlr.press/v195/hanneke23a.html.

37 David Haussler. Learning conjunctive concepts in structural domains. *Mach. Learn.*, 4:7–40, 1989. doi:10.1007/BF00114802.

38 Kouichi Hirata. On the hardness of learning acyclic conjunctive queries. In *Algorithmic Learning Theory, 11th International Conference, ALT 2000*, volume 1968 of *Lecture Notes in Computer Science*, pages 238–251. Springer, 2000. URL: https://doi.org/10.1007/3-540-40992-0_18, doi:10.1007/3-540-40992-0_18.

39 Jörg-Uwe Kietz and Saso Dzeroski. Inductive logic programming and learnability. *SIGART Bull.*, 5(1):22–32, 1994. doi:10.1145/181668.181674.

40 Benny Kimelfeld and Christopher Ré. A relational framework for classifier engineering. *ACM Trans. Database Syst.*, 43(3):11:1–11:36, 2018. doi:10.1145/3268931.

41 Stephan Kreutzer. Algorithmic meta-theorems. In Javier Esparza, Christian Michaux, and Charles Steinhorn, editors, *Finite and Algorithmic Model Theory*, volume 379 of *London Mathematical Society Lecture Note Series*, pages 177–270. Cambridge University Press, 2011.

42 Dietrich Kuske and Nicole Schweikardt. First-order logic with counting. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017*, pages 1–12. IEEE Computer Society, 2017. doi:10.1109/LICS.2017.8005133.

43 Christof Löding, P. Madhusudan, and Daniel Neider. Abstract learning frameworks for synthesis. In *Tools and Algorithms for the Construction and Analysis of Systems – 22nd International Conference, TACAS 2016*, volume 9636 of *Lecture Notes in Computer Science*, pages 167–185. Springer, 2016. URL: https://doi.org/10.1007/978-3-662-49674-9_10, doi:10.1007/978-3-662-49674-9_10.

44 Johann A. Makowsky. Algorithmic uses of the Feferman-Vaught theorem. *Ann. Pure Appl. Log.*, 126(1-3):159–213, 2004. doi:10.1016/j.apal.2003.11.002.

45 Denis Mayr Lima Martins. Reverse engineering database queries from examples: State-of-the-art, challenges, and research opportunities. *Inf. Syst.*, 83:89–100, 2019. `doi:10.1016/j.is.2019.03.002`.

46 Stephen Muggleton. Inductive logic programming. *New Gener. Comput.*, 8(4):295–318, 1991. `doi:10.1007/BF03037089`.

47 Stephen Muggleton and Luc De Raedt. Inductive logic programming: Theory and methods. *J. Log. Program.*, 19/20:629–679, 1994. `doi:10.1016/0743-1066(94)90035-3`.

48 Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, New York, NY, USA, 2014. `doi:10.1017/CBO9781107298019`.

49 Robert H. Sloan, Balázs Szörényi, and György Turán. Learning Boolean functions with queries. In Yves Crama and Peter L. Hammer, editors, *Boolean Models and Methods in Mathematics, Computer Science, and Engineering*, pages 221–256. Cambridge University Press, 2010. `doi:10.1017/cbo9780511780448.010`.

50 Slawek Staworko and Piotr Wieczorek. Learning twig and path queries. In *15th International Conference on Database Theory, ICDT 2012*, pages 140–154. ACM, 2012. `doi:10.1145/2274576.2274592`.

51 Wei Chit Tan, Meihui Zhang, Hazem Elmeleegy, and Divesh Srivastava. Reverse engineering aggregation queries. *Proc. VLDB Endow.*, 10(11):1394–1405, 2017. URL: `http://www.vldb.org/pvldb/vol10/p1394-tan.pdf`, `doi:10.14778/3137628.3137648`.

52 Wei Chit Tan, Meihui Zhang, Hazem Elmeleegy, and Divesh Srivastava. REGAL+: reverse engineering SPJA queries. *Proc. VLDB Endow.*, 11(12):1982–1985, 2018. URL: `http://www.vldb.org/pvldb/vol11/p1982-tan.pdf`, `doi:10.14778/3229863.3236240`.

53 Quoc Trung Tran, Chee Yong Chan, and Srinivasan Parthasarathy. Query reverse engineering. *VLDB J.*, 23(5):721–746, 2014. `doi:10.1007/s00778-013-0349-3`.

54 Chenglong Wang, Alvin Cheung, and Rastislav Bodik. Synthesizing highly expressive SQL queries from input-output examples. In *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2017*, pages 452–466. ACM, 2017. `doi:10.1145/3062341.3062365`.

55 Yaacov Y. Weiss and Sara Cohen. Reverse engineering SPJ-queries from examples. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2017*, pages 151–166. ACM, 2017. `doi:10.1145/3034786.3056112`.

56 He Zhu, Stephen Magill, and Suresh Jagannathan. A data-driven CHC solver. In *Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2018*, pages 707–721. ACM, 2018. `doi:10.1145/3192366.3192416`.