Bounded Fitting for Expressive Description Logics

Maurice Funk[1] , **Jean Christoph Jung**[2] , **Tom Voellmer**[2,3]
[1]Leipzig University and ScaDS.AI Center Dresden/Leipzig
[2]TU Dortmund University
[3]Center for Trustworthy Data Science and Security, University Alliance Ruhr
maurice.funk@uni-leipzig.de, {jean.jung, tom.voellmer}@tu-dortmund.de,

Abstract

Bounded fitting is an attractive paradigm for learning logical formulas from labeled data examples that offers PAC-style generalization guarantees and can often be implemented leveraging SAT solvers. It has been successfully applied to learning concepts of the description logic $\mathcal{ALC}$. We study bounded fitting for learning concepts in expressive description logics that extend $\mathcal{ALC}$ with inverse roles, qualified number restrictions, and feature comparisons. We investigate under which conditions bounded fitting keeps its favorable theoretical properties in this setting, and implement it using a SAT solver. We compare our tool with state-of-the-art concept learners with encouraging results, demonstrating that it is a practical approach to expressive concept learning.

1 Introduction

Fitting logical formulas to labeled data examples is the basic task of computing, given some data instance I, positive examples P, and negative examples N, a formula φ *fitting* P, N in I, that is, $\varphi(a)$ holds in I for all $a \in P$ but for no $a \in N$. It underlies many higher level tasks such as example-driven query specification, data exploration, explaining the labeling of the examples, and knowledge acquisition [Martins, 2019; ten Cate *et al.*, 2023a; Zloof, 1975]. A recently prominent approach to finding fitting formulas is *bounded fitting*, which rests on the idea to test whether there are fitting formulas of increasing size [ten Cate *et al.*, 2023b; Funk *et al.*, 2025; Neider and Gavran, 2018; Pommellet *et al.*, 2024]. Algorithm 1 illustrates the idea for a general logic $\mathcal{L}$. Bounded fitting is an attractive paradigm both from a theoretical and a practical perspective. Indeed, it is *complete*, meaning that it will find a fitting formula whenever there is one, and always returns a fitting formula of *minimal size*. The latter is desired by users since they typically prefer small fittings, and implies that bounded fitting is an *Occam algorithm* [Blumer *et al.*, 1987], which in turn guarantees that the found concept generalizes well to unseen examples in the sense of Valiant's probably approximately correct (PAC) learning [Valiant, 1984; Blumer *et al.*, 1989]. Finally, bounded fitting can often be implemented with the help of a SAT solver, allowing bounded fitting algorithms to perform well in practice.

Algorithm 1 Bounded Fitting for logic $\mathcal{L}$.

Input: Data instance I, examples P, N

1: **for** $k := 1, 2, \ldots$ **do**
2: **if** there is $\varphi \in \mathcal{L}$ of size k that fits P, N in I **then**
3: **return** φ

In this paper, we are interested in description logic (DL) concepts, an important class of logical formulas used for conceptual modeling, querying knowledge bases, and as a building block of ontologies [W3C, 2012]. The importance of DLs in data management has lead to the development of several tools for *DL concept learning*, that is, the task of finding a fitting DL concept given some knowledge base [Funk *et al.*, 2019; Lehmann and Hitzler, 2010]. Existing tools are based on different techniques such as refinement operators as used in DL-Learner [Bühmann *et al.*, 2018], evolutionary algorithms as used in EvoLearner [Heindorf *et al.*, 2022], or decision trees as in TDL [Demir *et al.*, 2025]. Recently, bounded fitting was demonstrated to be a suitable alternative for learning concepts formulated in the standard DLs $\mathcal{EL}$ and $\mathcal{ALC}$ [ten Cate *et al.*, 2023b; Funk *et al.*, 2025].

However, these DLs do not allow to express essential properties present in real-world data such as inverse roles, counting, and numeric features. Hence, in this paper we consider the DL $\mathcal{ALCQI}(f)$, which is the target language in EvoLearner and DL-Learner and which extends $\mathcal{ALC}$ with qualified number restrictions (abbreviated with $\mathcal{Q}$), inverse roles ($\mathcal{I}$), and feature comparisons (f), a simple mean to access data properties. One typical $\mathcal{ALCQI}(f)$ concept is

$$\text{Elephant} \sqcap \text{Male} \sqcap ((\text{weight} \geq 3t) \sqcup (\geq 2\ \text{childOf}^{-}.\text{Female}))$$

expressing the concept of male elephants that weigh at least three tons or have at least two daughters. The additional expressiveness over $\mathcal{ALC}$ allows to learn more interesting and meaningful concepts.

We make three main contributions. First, we study the fitting problem for $\mathcal{ALCQI}(f)$ and show that one can compute in polynomial time a concept that fits a maximal number of the given examples. The algorithms follow standard ideas based on computing maximal bisimulations, but the fact that one can efficiently compute optimal fitting concepts was surprising to us. While such optimal fitting algorithms are good news from a pure fitting perspective, under standard complexity

assumptions they cannot be Occam algorithms or have PAC-style generalization guarantees.

We thus investigate as our second contribution how to extend the bounded fitting paradigm to $\mathcal{ALCQI}(f)$. To this end, we show first that the *size-restricted fitting problem* that has to be solved in Line 2 of Algorithm 1 is NP-complete for $\mathcal{ALCQI}(f)$, as it is for $\mathcal{ALC}$ and many of its fragments [Funk *et al.*, 2025]. This paves the way to the use of SAT solvers. We then observe that inverse roles can be reduced away in a simple fashion. In contrast, adapting bounded fitting to number restrictions and feature comparisons while having a realistic implementation based on SAT solvers in mind is more challenging. We resort to allow for number restrictions and feature comparisons in a controlled and incremental way. While this is in spirit of *Occam's razor*, which demands to prefer simpler concepts, and can be implemented using a SAT solver, it is not immediately clear that it results in Occam algorithms as formalized by Blumer *et al.* (1989). Our main results are then sufficient conditions for when the constructed algorithms are indeed Occam, (and thus have PAC-style guarantees.

As our third contribution, we implement bounded fitting for $\mathcal{ALCQI}(f)$, using a SAT solver to solve the size-restricted fitting problem. In order to achieve competitive performance, we describe two techniques that speed up SAT-based bounded fitting. First, we propose that bisimilarity can be exploited as a preprocessing step to reduce the size of the input database. This is not specific to bounded fitting and can potentially be used to improve other concept learning systems as well. Second, we observe that bounded fitting allows easy parallelization and implement it to use multiple CPU cores.

We evaluate our implementation on several benchmarks and compare it to the state-of-the-art concept learners EvoLearner and TDL. The experiments show that the concepts learned by our implementation generalize at least as good as the ones found by EvoLearner and TDL on the standard structured machine learning (SML) benchmarks. Moreover, using a newly generated benchmark, we show that our tool is better in finding fitting $\mathcal{ALCQ}$ concepts than other systems. Finally, we confirm that the presented optimization are effective in practice as well. Overall, our evaluation demonstrates that bounded fitting works well in practice also for expressive DLs.

Missing proofs can be found in the full version [Funk *et al.*, 2026].

Related Work. The works most relevant related to this paper are [ten Cate *et al.*, 2023b] and [Funk *et al.*, 2025], which introduce the bounded fitting framework for the DLs $\mathcal{EL}$ and $\mathcal{ALC}$, respectively. Beyond the concept learners mentioned above, there is a series of other tools that rely on more classical machine learning techniques such as trained search heuristics and neural concept synthesis like ROCES [Kouagou *et al.*, 2024], NCES/NCES2 [Kouagou *et al.*, 2023], and Drill [Demir and Ngonga Ngomo, 2023]. As they require a separate pretraining step, it is difficult to make a meaningful comparison with them. Other concept learners are DL-Foil [Rizzo *et al.*, 2020] and SPaCel [Tran *et al.*, 2017].

2 Background

We introduce syntax and semantics of the description logic $\mathcal{ALCQI}(f)$, the extension of $\mathcal{ALCQI}$ [Baader *et al.*, 2017] with simple feature comparisons. Let $\mathsf{N_C}$, $\mathsf{N_R}$, $\mathsf{N_F}$, and $\mathsf{N_I}$ be mutually disjoint and countably infinite sets of *concept names*, *role names*, *feature names*, and *individual names*, respectively. We assume that each feature name f has a *domain* $\mathsf{dom}(f)$, the set of values the feature f can take. We further assume a total order $\leq$ in every domain. An $\mathcal{ALCQI}(f)$ *concept* C is defined according to the syntax rule

$$C, D ::= \top \mid A \mid \neg C \mid C \sqcap D \mid (\bowtie n\ R.C) \mid (f \bowtie v)$$

where A ranges over concept names, $\bowtie\ \in \{\leq, \geq\}$, R is a role name r or an *inverse role* $R = r^-$, $n \geq 0$, $f \in \mathsf{N_F}$ is a feature name, and $v \in \mathsf{dom}(f)$. We call concepts of the shape $(\bowtie n\ R.C)$ *qualified number restrictions* and concepts of shape $(f \bowtie v)$ *feature comparisons*. We use the standard abbreviations $\bot$, $C \sqcup D$, $\exists R.C$, and $\forall R.C$ for $\neg\top$, $\neg(\neg C \sqcap \neg D)$, $(\geq 1\ R.C)$, and $(\leq 0\ R.\neg C)$, respectively. We also consider the usual restrictions of $\mathcal{ALCQI}(f)$ that are obtained by disallowing certain constructors in the syntax. For example, $\mathcal{ALCQ}$ is obtained by disallowing inverse roles r^- and feature comparison, while $\mathcal{ALCI}(f)$ is obtained by disallowing qualified number restrictions.

A *database* I is a finite set of facts of the form $A(a)$, $r(a, b)$, and $f(a, v)$ for $A \in \mathsf{N_C}$, $r \in \mathsf{N_R}$, $f \in \mathsf{N_F}$, $a, b \in \mathsf{N_I}$, and $v \in \mathsf{dom}(f)$. We let $\mathsf{adom}(I)$ denote the set of all individual names that occur in I. Moreover, we assume that for every $a \in \mathsf{adom}(I)$ and $f \in \mathsf{N_F}$, I contains at most one fact of shape $f(a, v)$. We write $r^-(b, a) \in I$ if $r(a, b) \in I$, and define $\mathsf{succ}_I(a, R)$ as the set

$$\mathsf{succ}_I^R(a) = \{b \mid R(a, b) \in I\}.$$

The semantics C^I of an $\mathcal{ALCQI}(f)$ concept C in database I is defined inductively as follows. In the inductive base, we set $\top^I = \mathsf{adom}(I)$ and $A^I = \{a \mid A(a) \in I\}$. In the inductive step, we define $(\neg C)^I = \mathsf{adom}(I) \setminus C^I$, $(C \sqcap D)^I = C^I \cap D^I$, and

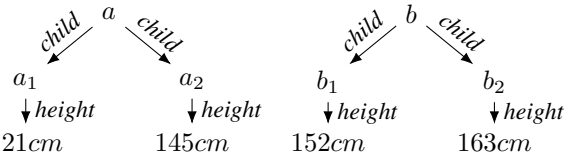
$$(\bowtie n\ R.C)^I = \{a \in \mathsf{adom}(I) \mid |\mathsf{succ}_I^R(a) \cap C^I| \bowtie n\},$$
$$(f \bowtie v)^I = \{a \in \mathsf{adom}(I) \mid f(a, v') \in I \text{ and } v' \bowtie v\}.$$

A *signature* is a finite set Σ of concept, role, and feature names. A Σ-database is a database in which all facts mention only concept, role, and feature names from Σ. For a syntactic object O such as a concept or a database, we let $\|O\|$ denote the *size* of O, that is, the length of a string representation of O in a suitable fixed alphabet in which concept/role/feature names contribute 1. We assume unary coding of numbers in qualified number restrictions.

Fitting Problems. Let I be a database and $P, N \subseteq \mathsf{adom}(I)$ be sets of individuals called *positive* and *negative examples*. We say that an $\mathcal{ALCQI}(f)$ concept C *fits* P, N *in* I if $a \in C^I$ for each $a \in P$ and $b \notin C^I$ for each $b \in N$. The *fitting problem* for $\mathcal{ALCQI}(f)$ is to decide, given I, P, N, whether there is an $\mathcal{ALCQI}(f)$ concept that fits P, N in I. A *fitting algorithm for* $\mathcal{ALCQI}(f)$ takes as input I, P, N as above, and returns an $\mathcal{ALCQI}(f)$ concept that fits P, N in I, if it exists.

The fitting problem and fitting algorithms for fragments of $\mathcal{ALCQI}(f)$ are defined accordingly. Observe that we make the *closed world assumption* in contrast to other works which study the fitting problem for knowledge bases, see, e.g. [Funk *et al.*, 2019; Jung *et al.*, 2022]).

Example 1. *Consider the following database I:*



The $\mathcal{ALCQI}(f)$ concept $(\leq 1\ child.(height \geq 140cm))$ fits $\{a\}, \{b\}$ in I. Further, there is no $\mathcal{ALCQI}$ concept that fits $\{a\}, \{b\}$.

Occam Algorithms. We recall the standard notion of *Occam algorithms* following [Blumer *et al.*, 1989]. Let $\mathcal{C}$ be a set of DL concepts, I be a database and $S \subseteq \mathsf{adom}(I)$ be a set of examples. We say that $\mathcal{C}$ *shatters* S *over* I if for every $S' \subseteq S$ there is a $C \in \mathcal{C}$ such that $S' = S \cap C^I$. The *VC-Dimension* of a set of concepts $\mathcal{C}$ is the largest number $d \geq 0$ such that there is a database I and a set of examples S with cardinality d that is shattered by $\mathcal{C}$.

Let $\mathbf{A}$ be a fitting algorithm. For a signature Σ and $s, m \geq 1$, let $\mathcal{H}^{\mathbf{A}}(\Sigma, s, m)$ be the set of all concepts returned by $\mathbf{A}$ when started on some input I, P, N such that I is a Σ-database, $|P| + |N| = m$, and there is a concept C with $\|C\| \leq s$ that fits P, N in I. We call $\mathbf{A}$ an *Occam algorithm* if there is a polynomial p and $\alpha \in [0, 1)$ such that for all Σ and $s, m \geq 1$ the VC-Dimension of $\mathcal{H}^{\mathbf{A}}(\Sigma, s, m)$ is at most $p(s, |\Sigma|) \cdot m^{\alpha}$.

It is well known that that every Occam algorithm has PAC-style generalization meaning that it is a *sample-efficient PAC learning algorithm*. As the precise definition of that notion is not important for the main part of the paper, we refer the reader to [Blumer *et al.*, 1989] or to the supplementary material.

3 The Fitting Problem for $\mathcal{ALCQI}(f)$

We begin our investigation by showing that the fitting problem for $\mathcal{ALCQI}(f)$ and its fragments is efficiently solvable.

Theorem 1. *The fitting problems for $\mathcal{ALCQI}(f)$, $\mathcal{ALCQ}(f)$, $\mathcal{ALCQI}$, and $\mathcal{ALCQ}$ are decidable in polynomial time.*

This can be shown using standard arguments. Since some of these will be important later, we sketch the main steps here; for details see the supplementary material. We first show the theorem for $\mathcal{ALCQ}$ and then for the remaining languages by a reduction to $\mathcal{ALCQ}$. We rely on the notion of bisimulations.

Definition 1. *Let I be a database. An $\mathcal{ALCQ}$* bisimulation *on I is a relation $S \subseteq \mathsf{adom}(I) \times \mathsf{adom}(I)$ such that the following conditions are satisfied for all $(a, b) \in S$:*

1. *for all $A \in \mathsf{N_C}$, $A(a) \in I$ if and only if $A(b) \in I$;*
2. *for all $r \in \mathsf{N_R}$ and $D \subseteq \mathsf{succ}^r_I(a)$, there is $E \subseteq \mathsf{succ}^r_I(b)$ such that S contains a bijection between D and E;*
3. *for all $r \in \mathsf{N_R}$ and $E \subseteq \mathsf{succ}^r_I(b)$, there is $D \subseteq \mathsf{succ}^r_I(a)$ such that S contains a bijection between D and E.*

We call $a, b \in \mathsf{adom}(I)$ $\mathcal{ALCQ}$ bisimilar*, written $I, a \sim I, b$, if there is an $\mathcal{ALCQ}$ bisimulation S on I with $(a, b) \in S$.*

It is well-known that the existence of fitting $\mathcal{ALCQ}$ concepts is characterized by the non-existence of $\mathcal{ALCQ}$ bisimulations between positive and negative examples. More precisely, there is an $\mathcal{ALCQ}$ concept fitting P, N in I iff $I, a \not\sim I, b$ for all $a \in P, b \in N$ [de Rijke, 2000]. Since bisimilarity can be decided in polynomial time, Theorem 1 for $\mathcal{ALCQ}$ follows.

It remains to provide the promised reductions.

Lemma 1. *There are polynomial-time reductions from the fitting problem for $\mathcal{ALCQI}(f)$ to that for $\mathcal{ALCQ}(f)$, and from the fitting problem for $\mathcal{ALCQ}(f)$ to that of $\mathcal{ALCQ}$.*

Proof. For the first reduction, let I, P, N be an input to the fitting problem for $\mathcal{ALCQI}(f)$. Introduce a fresh role name $\overline{r}$ for every role name r that occurs in I. Then then there is an $\mathcal{ALCQI}(f)$ concept fitting P, N in I iff there is an $\mathcal{ALCQ}(f)$ concept fitting P, N in $J = I \cup \{\overline{r}(b, a) \mid r(a, b) \in I\}$.

For the second reduction, let I, P, N be an input to the fitting problem for $\mathcal{ALCQ}(f)$. Introduce a fresh concept name $A_{f \geq v}$ for every feature f and value v such that there is a fact $f(a, v) \in I$. Let J be the database consisting of all facts of shape $A(a)$ and $r(a, b)$ from I and all facts $A_{f \geq v}(a)$ such that there is a fact $f(a, v') \in I$ with $v \geq v'$. It is routine to verify that there is an $\mathcal{ALCQ}(f)$ concept fitting P, N in I if and only if there is an $\mathcal{ALCQ}$ concept fitting P, N in J. □

We show next that the algorithms used to prove Theorem 1 can be made constructive in a strong sense. Note that we allow DAG representation of concepts when constructing them.

Theorem 2. *There are polynomial time fitting algorithms for $\mathcal{ALCQI}(f)$, $\mathcal{ALCQ}(f)$, $\mathcal{ALCQI}$, $\mathcal{ALCQ}$ which always compute a concept that fits a maximal overall number of positive and negative examples.*

While such optimal fitting algorithms are natural candidates to be used in practice, they cannot be sample-efficient PAC learning algorithms, under standard complexity assumptions. Indeed, if a polynomial time fitting algorithm for $\mathcal{ALCQI}(f)$ was a sample-efficient PAC learning algorithm, it could be used as an efficient PAC learning algorithms for propositional logic which likely does not exist [Kearns and Valiant, 1994]. Hence, the algorithms in Theorem 2 are also not Occam.

4 Bounded Fitting for $\mathcal{ALCQI}(f)$

Motivated by the fact that standard algorithms are not Occam or sample-efficient, we apply the bounded fitting paradigm given in Algorithm 1 to $\mathcal{ALCQI}(f)$, and investigate under which conditions the favorable properties are preserved. As we are interested in realistic implementations leveraging SAT solvers, we first establish that the core problem to be solved in Line 2 of Algorithm 1 is indeed NP-complete for $\mathcal{ALCQI}(f)$.

Formally, the *size-restricted fitting problem* is defined as follows. Given database I, positive and negative examples P, N and $k \geq 0$, decide whether there is an $\mathcal{ALCQI}(f)$ concept C of size $\|C\| \leq k$ that fits P, N in I. In the context of bounded fitting, it is natural to assume k to be coded in unary.

Theorem 3. *The size-restricted fitting problem for $\mathcal{ALCQI}(f)$ is* NP*-complete. Hardness already holds for inputs I, P, N where I is an $\{A, r, s\}$-database for $A \in \mathsf{N_C}$ and $r, s \in \mathsf{N_R}$, and P, N are sets with $|P| = |N| = 1$.*

Algorithm 2 Bounded fitting for $\mathcal{ALCQ}$.

Input: Data instance I, examples P, N

1: **for** $k := 1, 2, \ldots$ **do**
2: **if** there is $C \in \mathcal{ALCQ}^k_{g(k)}$ that fits P, N in I **then**
3: **return** C

The upper bound is established by the usual *guess-and-check* argument: given I, P, N, non-deterministically guess a candidate concept of size k, and deterministically verify that it fits P, N in I in time polynomial in k and the size of I. The proof of the lower bound is more challenging. It extends the proof of the same result for $\mathcal{ALC}$ [Funk *et al.*, 2025], but requires care to handle inverse roles and number restrictions.

While the NP-upper bound from Theorem 3 is encouraging from the theoretical perspective, a direct reduction to SAT in the style of Cook's theorem [Cook, 1971] is hardly practical. Previous reductions were based on the idea of measuring the complexity of the concept sought in stage k of bounded fitting in terms of the number of nodes in the syntax tree or, equivalently, the number of logical operators [ten Cate *et al.*, 2023b; Funk *et al.*, 2025]. The SAT solver then "guesses" the labels of the nodes in the syntax tree and evaluates whether the represented concept fits. This idea does not easily to $\mathcal{ALCQI}(f)$ since the numbers that occur in qualified number restrictions and feature comparisons should in some way contribute to the complexity of the concepts. We discuss below the additions $\mathcal{I}$, $\mathcal{Q}$, f to $\mathcal{ALC}$ separately for the sake of clarity. Our bounded fitting algorithm for $\mathcal{ALCQI}(f)$ combines all ideas.

We start with inverse roles. Observe that the first reduction in Lemma 1 provides a transparent way of dealing with them. Consider the following algorithm $\mathbf{A}_\mathcal{I}$. Let I, P, N be an input for the fitting problem for $\mathcal{ALCI}$. Then, $\mathbf{A}_\mathcal{I}$ computes J, P, N for J defined as in the proof of Lemma 1, and starts bounded fitting (for $\mathcal{ALC}$) on input J, P, N. If it finds an $\mathcal{ALC}$ concept in this way, it outputs the corresponding $\mathcal{ALCI}$ concept. It follows from the reduction and the fact that bounded fitting for $\mathcal{ALC}$ is a complete Occam algorithm that $\mathbf{A}_\mathcal{I}$ is a complete Occam algorithm as well.

We next consider qualified number restrictions. Note that under the discussed measure of size (being the number of logical operators), there are, in principle, infinitely many concepts of a given size. To make the space of concepts considered in each stage of bounded fitting finite, we have to bound the numbers occurring in the concepts. To this end, we allow in stage k numbers up to $g(k)$ in concepts, for some function g. The idea is illustrated in Algorithm 2, where $\mathcal{ALCQ}^k_{g(k)}$ denotes the set of $\mathcal{ALCQ}$ concepts that allow for at most k logical operators and numbers in number restrictions at most $g(k)$. This approach is in the spirit of Occam's razor if g is an increasing function as this reflects that larger numbers in number restrictions are considered more complex. Depending on the domain, the user may choose different functions reflecting their view on the importance of number restrictions. The following theorem provides the user with a range of functions g which make Algorithm 2 a complete Occam algorithm.

Theorem 4. *Let $g \in \Omega(k) \cap O(2^{p(k)})$ for some polynomial p. Then Algorithm 2 is a complete Occam algorithm.*

Completeness follows from $g \in \Omega(k)$. For showing that Algorithm 2 is Occam we carefully analyze its space $\mathcal{H}^\mathbf{A}(\Sigma, s, m)$ and show that its VC-dimension is polynomially bounded for any g as in the theorem.

We next look at feature comparisons. Similar to the number restrictions, there are potentially infinitely many concepts with a given number of logical operators, since features may have an infinite domain. Similar to what was done for $\mathcal{ALCQ}$, we could obtain an Occam algorithm by allowing in every stage k of bounded fitting for, say, k bits in the representation of the feature values. From a practical perspective, however, we believe that different feature values should contribute in the same way to a learned concept, e.g., the complexity of feature comparisons (salary $\geq n$) should not depend on n. This perspective suggests to remove the feature comparisons via the reduction given in Lemma 1, similar to what we did for inverses. Unfortunately, this does not lead to an Occam algorithm. Let $\mathbf{A}_f$ be the algorithm obtained by first applying the reduction of fitting in $\mathcal{ALC}(f)$ to fitting in $\mathcal{ALC}$ from Lemma 1 and then applying bounded fitting for $\mathcal{ALC}$ to the result. Clearly, $\mathbf{A}_f$ is complete, but:

Theorem 5. *Algorithm $\mathbf{A}_f$ is not an Occam algorithm.*

The intuitive reason for this is that for sufficiently large s, m, one can find inputs to the algorithm so that $\mathbf{A}_f$ returns concepts of the form $\exists r.((f \leq j) \sqcap (f \geq j))$, for any $j \in \mathbb{N}$. While an infinite space $\mathcal{H}^{\mathbf{A}_f}(\Sigma, s, m)$ does not rule out being an Occam algorithm, we show that these concepts shatter exponentially sized sets of examples, making the VC dimension not polynomially bounded. Indeed, we show the stronger statement that $\mathbf{A}_f$ is not a sample-efficient PAC learning algorithm.

The proof of Theorem 5 relies on the presence of an unbounded number of feature values and on databases with unbounded outdegree. We next show that both these assumptions are necessary to achieve exponential VC-dimension. A class of databases has *bounded outdegree* if there is a constant ℓ such that for all $a \in \mathsf{adom}(I)$, there are at most ℓ facts of shape $r(a, b) \in I$. Similarly, the class of databases has *bounded domains* if the cardinalities of all feature domains are bounded by some constant.

Theorem 6. *Algorithm $\mathbf{A}_f$ is an Occam Algorithm if input databases have bounded outdegree or bounded domains.*

If domains are bounded, one can replace features with a fixed number of concept names. For bounded outdegree, we observe that for a concept C of a given size s, there is an at most exponential number of feature values "visible" by C in any given set of m examples. This allows us to bound the VC-dimension of the space $\mathcal{H}^{\mathbf{A}_f}(\Sigma, s, m)$ over databases of bounded outdegree.

From a practical point of view, the preconditions in Theorem 6 are not guaranteed to hold in all datasets, but in relevant cases, e.g., for features that describe days of the week, age of employees, or for roles such as child or spouse. A limitation in practice, however, is that using this route introduces a potentially large number of concept names which, as we shall see, is reflected in an increased number of propositional variables. We describe in Section 5.2 how we address this problem in our tool.

5 Implementation

We implemented bounded fitting for $\mathcal{ALCQI}(f)$. Our implementation follows the same general approach as ALCSAT [Funk *et al.*, 2025]: in each stage of bounded fitting, the size-restricted fitting problem is solved by checking satisfiability of a suitable propositional formula, using a SAT solver. Users may specify the fragment of $\mathcal{ALCQI}(f)$ that they are interested in. Moreover, our tool implements an approximation strategy that tries to compute a concept that fits the maximum number of examples. In the remainder of the section, we describe the reduction to SAT for $\mathcal{ALCQ}$, discuss how we handle inverses ($\mathcal{I}$) and features (f), and describe two optimizations.

5.1 SAT Encoding for $\mathcal{ALCQ}$

We briefly recall some details of ALCSAT's implementation [Funk *et al.*, 2025] in order to explain how we extend it to $\mathcal{ALCQ}$. Let I, P, N, k be an instance of the size-restricted fitting problem for $\mathcal{ALC}$. ALCSAT constructs a propositional formula φ that is satisfiable if and only if there exists an $\mathcal{ALC}$ concept C of size k that fits P, N in I. Moreover, such a fitting concept can directly be obtained from a model of φ. Let Σ_{C} and Σ_{R} be the sets of concept and role names that occur in I. The formula φ encodes the syntax tree of an $\mathcal{ALC}$ concept with k nodes and its semantics such that in a model of φ, for all $i, j \in \{1, \dots, k\}$, $a \in \mathsf{adom}(I)$ and v a node label from $\mathcal{V} = \{\top, \bot, \neg, \sqcap, \sqcup\} \cup \Sigma_{\mathsf{C}} \cup \{\exists r, \forall r \mid r \in \Sigma_{\mathsf{R}}\}$, the variable $x_{i,v}$ is true iff node i of the syntax tree is labeled with v, the variable $y_{1,i,j}$ is true iff node i has the single successor j, the variable $y_{2,i,j}$ is true iff node i has the two successors j and $j+1$, and the variable $z_{i,a}$ is true iff $a \in C_i^I$ where C_i is the concept represented by node i in the syntax tree. For example, the semantics of existential restrictions is encoded by clauses that express

$$x_{i,\exists r} \wedge y_{1,i,j} \wedge z_{i,a} \rightarrow \bigvee \{z_{j,b} \mid r(a,b) \in I\}$$

for $1 \leq i, j \leq k$, $r \in \Sigma_r$, $a \in \mathsf{adom}(I)$. The requirement that the concept C_1 fits P, N in I is then encoded using $\bigwedge_{a \in P} z_{1,a}$ and $\bigwedge_{a \in N} \neg z_{1,a}$.

Our implementation extends this encoding by allowing additional vertex labels for qualified number restrictions and by encoding their semantics. Given a bound $g(k)$ (recall Algorithm 2 from the previous section) for numbers in qualified number restrictions, we use node labels $(\leq n\ r)$ and $(\geq n\ r)$ for all $r \in \Sigma_r$ and n with $0 \leq n \leq g(k)$. Then, we encode the semantics of qualified number restrictions by adding clauses that encode

$$x_{i,(\geq n\ r)} \wedge y_{1,i,j} \wedge z_{i,a} \rightarrow (|\{z_{j,b} \mid r(a,b) \in I\}| \geq n)$$
$$x_{i,(\geq n\ r)} \wedge y_{1,i,j} \wedge \neg z_{i,a} \rightarrow (|\{z_{j,b} \mid r(a,b) \in I\}| \leq n-1)$$

for all i, j with $1 \leq i, j \leq k$, $0 \leq n \leq g(k)$, $r \in \Sigma_r$ and $a \in \Delta^I$ to φ. To encode the cardinality constraints in the above formulas as clauses, we employ a sequential counter encoding, which for a bound of k on n literals uses $O(n \cdot k)$ additional variables and clauses [Sinz, 2005]. In total, this results in $O(k^2 \cdot |I| \cdot |\Sigma| \cdot g(k)^2)$ additional clauses.

5.2 Handling Inverses and Features

We treat inverses as described in the previous section, that is, via the (first) reduction in Lemma 1. It is worth noting that EvoLearner implements the same strategy [Heindorf *et al.*, 2022].

To handle features, we implement a variant of the (second) reduction in Lemma 1. Instead of introducing a single concept name $A_{f \geq v}$ for *every value* v that occurs in I, we introduce such concept names only for a *restricted number* n_f of these values. This alone would result in an incomplete algorithm (as we may miss fitting concepts), so we incrementally increase the number n_f along with the stages of bounded fitting. In practice we split the $\mathsf{dom}(f)$ into n_f equally sized intervals.

DL-Learner follows a similar approach [Lehmann, 2010]. EvoLearner extends that by an entropy-based selection of the values given at the examples P, N [Heindorf *et al.*, 2022].

5.3 Optimizations

We present two optimizations. The first relies on $\mathcal{ALCQ}$ bisimulations and is not specific to bounded fitting; it can in principle be used with all other tools. The idea is to exploit that $\mathcal{ALCQ}$ concepts cannot distinguish between $\mathcal{ALCQ}$ bisimilar individuals, and hence to replace the input database I with a smaller database I' in which few elements are bisimilar; we need to keep copies of bisimilar elements, due to the qualified number restrictions.

For a database I and for every $a \in \mathsf{adom}(I)$, let $[a]$ denote the $\sim$-equivalence class of a in I. The *$\sim$-quotient* of I is the database I' over individuals $\langle [a], i \rangle$ with $a \in \mathsf{adom}(I)$ and

$$1 \leq i \leq \max_{b \in \mathsf{adom}(I), r \in \mathsf{N_R}} |\mathsf{succ}_I^r(b) \cap [a]|,$$

and the following facts for all $A \in \mathsf{N_C}$ and $r \in \mathsf{N_R}$:

$$A(\langle [a], i \rangle) \text{ if } A(a) \in I$$
$$r(\langle [a], i \rangle, \langle [b], j \rangle) \text{ if } j \leq |\mathsf{succ}_I^r(a) \cap [b]|.$$

Using the observation that I, a is $\mathcal{ALCQ}$ bisimilar to $I', \langle a, i \rangle$ for any i, one can show the following.

Lemma 2. *Let I, P, N be a fitting problem instance for $\mathcal{ALCQ}$. Every $\mathcal{ALCQ}$ concept C fits P, N in I if and only if C fits $\{\langle [p], 1 \rangle \mid p \in P\}, \{\langle [n], 1 \rangle \mid n \in N\}$ in I'.*

Hence, on input I, P, N our implementation computes I', and then runs bounded fitting on I'. To compute the $\mathcal{ALCQ}$ bisimulation equivalence classes in I, we use a variant of the color refinement algorithm (see e.g. [Grohe *et al.*, 2021]), which can be implemented to run in $O(|I| \log |I|)$.

Our second optimization concerns parallelization. By design, bounded fitting is suitable for parallelization: we can simply start each stage in its own thread. We follow a slightly different route that was proposed in the context of fitting LTL formulas [Riener, 2019]. We split the search space at a given stage k into several pieces and start search in each subspace on a different core based on the topology of the concepts. For this purpose we enumerate (outside the SAT solver) all possible topologies of syntax trees, split them into the number of available threads and start bounded fitting in each thread with the additional constraint to consider only the given topologies.

6 Experiments

We evaluate different aspects of our tool on the SML-Benchmarks [Westphal *et al.*, 2019] and on benchmarks for $\mathcal{ALCQ}$ fitting we generated from the YAGO 4.5 knowledge base [Suchanek *et al.*, 2024]. All experiments were executed on a MacBook Pro with M3 Pro chip and 36GB RAM.[1]

We refer to accuracy and F1-score of concepts C on examples P, N in I. With accuracy we mean the percentage of elements of P and N that are labeled correctly by C, formally $\frac{|C^I \cap P| + |N \setminus C^I|}{|P| + |N|}$. As usual, with F1-score we mean the harmonic mean of precision and recall of C.

6.1 Evaluating Generalization

We compare the ability of our implementation to produce $\mathcal{ALCQ}(f)$ concepts that generalize to unseen examples on the standard SML-benchmarks with the state-of-the art concept learning implementations EvoLearner and TDL[2] as well as the algorithm underlying Theorem 2. We do not compare against the standard DL-Learner algorithms CELOE and OCEL, as EvoLearner is known to deliver concepts of higher F1-score [Heindorf *et al.*, 2022]. We configured all systems to learn $\mathcal{ALCQ}(f)$ concepts with number restrictions up to a cardinality of 3. In the case of TDL this means disabling nominals. This places TDL at a disadvantage as it does not support numerical features (only Boolean features). We allowed all tools to run for 5 minutes. TDL and the algorithm underlying Theorem 2 always terminated before; for the anytime algorithms EvoLearner and our tool, we used the best concept produced so far.

Table 1 shows the results (mean and standard deviation over all runs) of 10-fold cross validation measuring the resulting concept size and F1-score on each data set. We omit results for the Mutagenesis and Nctrer benchmarks, as they possess a small perfect fitting $\mathcal{ALCQ}(f)$ concept that is almost always discovered by the systems. We remark that no implementation performs particularly well on the Carcinogenesis benchmark, where already the concept $\top$ achieves an F1-score of 0.71.

Overall these results demonstrate that bounded fitting performs competitively with state-of-the-art concept learners and is suitable to produce good concepts in practice. As expected, the algorithm underlying Theorem 2 produces very large concepts that overfit and do not generalize.

6.2 Evaluating Fitting for Number Restrictions

We compare the ability of our implementation to find fitting $\mathcal{ALCQ}$ concepts with that of EvoLearner and TDL. To this end, we created a new benchmark which contains difficult concept learning problems, where qualified number restrictions are essential to separate the positive examples from the negative examples. That is, it contains only instances I, P, N such that there is an $\mathcal{ALCQ}$ concept fitting P, N, but no $\mathcal{ALC}$-concept.

[1]Our implementation, the benchmarks, and instructions to reproduce our experiments are available at https://github.com/SAT-based-Concept-Learning/ALCSAT.

[2]We use the implementations of these systems available in OntoLearn https://github.com/dice-group/Ontolearn development version fd38beec65e85e56e50733d404360b1feb5f5a68, as TDL of version 0.9.2 of OntoLearn reported wrong accuracies

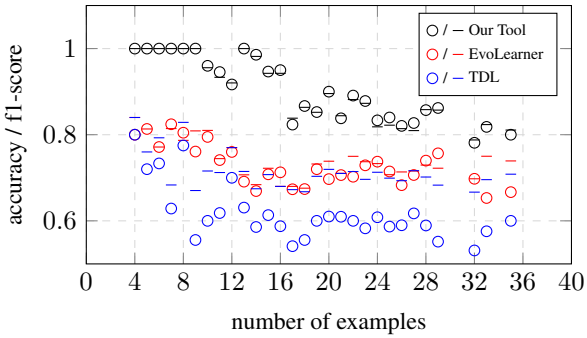


Figure 1: Accuracies (circles) and F1-scores (hyphens) for $\mathcal{ALCQ}$.

For this, we extract examples from the YAGO 4.5 knowledge base [Suchanek *et al.*, 2024], focusing on the relations `children`, `spouse`, and `gender`. We first construct from each $\mathcal{ALC}$-bisimulation equivalence class a single benchmark, by partitioning the equivalence class into $\mathcal{ALCQ}$ bisimulation equivalence classes, taking representatives of half the $\mathcal{ALCQ}$ bisimulation equivalence classes as positive examples and representatives of the other half as negative examples. In a second step, we generate benchmarks with a larger number of examples by combining the positive and negative examples of two benchmarks generated in the first step. We generate 120 benchmarks with the number of examples ranging from 4 to 35. An example of a fitting $\mathcal{ALCQ}$ concept in these benchmarks is the concept $(\leq 3\ \texttt{children}.\top) \sqcap (\leq 1\ \texttt{children}.\exists\ \texttt{spouse}.\top)$. We believe that these benchmarks are of independent interest to evaluate the ability of concept learning implementations to find good $\mathcal{ALCQ}$ on real-world data.

Figure 1 shows the result of comparing mean accuracy and F1-score of the best concept produced within 5 minutes (as in Sec. 6.1) by TDL, EvoLearner (configured to optimize accuracy or F1-score), and our tool on these benchmarks, ordered by increasing number of examples. In all benchmarks, our tool is able to find concepts of higher accuracy and F1-score, in some cases even discovering perfect fittings. This indicates that our approach to handle qualified number restrictions has advantages over other tools, for fitting real-world examples.

6.3 Effect of Features Value Selection

We evaluate the effect of our approach to fit $\mathcal{ALCQ}(f)$ concepts by reduction to the fitting problem for $\mathcal{ALCQ}$. For this, we run our implementation of bounded fitting up to size 8 on SML-benchmarks and vary the number of selected feature values. We use the SML-Benchmarks Mammographic, Mutagenesis, and Suramin, which contain features with domains of size 74, 529, and 157, respectively. Other SML-Benchmarks do not use numeric features to the same extent.

The measured accuracy on the training data and running time in seconds for different number of values are shown in Table 2, averaged over three runs. The first row for 0 values reports results when only Boolean features are used. As expected, increasing the number of values allows bounded fitting to find concepts that achieve higher accuracy, but this generally also increases running time, due to additional concept

	Carcinogenesis	Hepatitis	Lymphography	Mammographic	Premierleague	Pyrimidine
EvoLearner	5.50 ± 1.51 0.71 ± 0.01	4.50 ± 1.35 0.77 ± 0.07	17.40 ± 9.03 0.84 ± 0.08	3.20 ± 1.48 0.78 ± 0.05	5.00 ± 0.94 1.00 ± 0.00	6.40 ± 1.26 0.91 ± 0.14
TDL	879.60 ± 132.97 0.62 ± 0.07	1436.80 ± 46.49 0.83 ± 0.05	112.10 ± 23.33 0.82 ± 0.11	5780.70 ± 302.98 0.69 ± 0.03	298.40 ± 35.22 0.47 ± 0.37	15.90 ± 3.78 0.82 ± 0.24
Theorem 2	4715.60 ± 455.32 0.56 ± 0.06	53269.80 ± 1549.82 0.53 ± 0.06	330.80 ± 18.98 0.74 ± 0.13	1925.40 ± 72.76 0.76 ± 0.05	23.40 ± 1.26 1.00 ± 0.00	60.80 ± 3.94 0.90 ± 0.16
Our Tool	6.00 ± 0.00 0.73 ± 0.10	5.00 ± 0.00 0.81 ± 0.03	10.00 ± 0.00 0.83 ± 0.08	9.10 ± 0.32 0.81 ± 0.05	2.00 ± 0.00 0.97 ± 0.05	5.90 ± 0.32 0.95 ± 0.12

Table 1: Concept sizes (first line) and F1-scores (second line) of concept learning tools on the SML-benchmarks.

	Mammographic		Suramin		Mutagenesis	
Values	ACC	t	ACC	t	ACC	t
0	0.79	5.6	0.82	9.7	0.88	13
2	0.80	24	0.82	11.7	0.95	8.9
5	0.84	24.2	0.82	14.2	1.00	2
10	0.84	10.5	0.88	14.9	1.00	2.3
20	0.84	15.8	0.88	16.9	1.00	0.3
1000	0.84	47.9	0.94	16.9	1.00	0.6

Table 2: Accuracy and running time in seconds of bounded fitting up to size 8 with different numbers of values. Average of 3 runs.

	Mammographic		Hepatitis		Lymphography	
	t	$\|I\|$	t	$\|I\|$	t	$\|I\|$
$\mathcal{ALC}$	10	975	36.9	6812	18.9	148
	5.8	94	1.6	9	19	148
$\mathcal{ALC}(f)$	17	975	936.3	6812	19.8	148
	9.2	364	502	3720	19.5	148
$\mathcal{ALCQ}(f)$	19.4	975	3668.9	6812	19.3	148
	10.5	364	2816.7	5253	19.3	148
$\mathcal{ALCQI}(f)$	60.9	975	9063.4	6812	18.8	148
	40.7	975	9123.6	6812	18.7	148

Table 3: Running time in seconds of bounded fitting up to size 8 and domain sizes without preprocessing (first line) and with preprocessing (second line). Average of 3 runs.

names. However, the running time does not always monotonically increase, as the running time of the SAT solver varies unpredictably with small changes. These results show that our approach to encode values to obtain a propositional formula is effective, and that on the SML-benchmark already a small number of such values suffices.

6.4 Effect of Optimizations

We evaluate the effect of the two presented optimizations. First, we run our implementation of bounded fitting up to size 8 on SML-benchmarks, and measure the required running time with and without the bisimulation-based preprocessing. We also vary the fragment of $\mathcal{ALCQI}(f)$ and measure the number of individuals before and after preprocessing.

Table 3 shows the results on the Mammographic, Hepatitis and Lymphography benchmarks. Again, we report the average of the running time over three runs. The measured number of individuals shows that on some benchmarks like Lymphography, the preprocessing has no effect, but also does not slow down the implementation. On other benchmarks, like Mammographic, the number of individuals is reduced which directly results in faster computation. Naturally, when fitting an $\mathcal{ALC}$ concept more individuals can be identified as when fitting a more expressive concept. This indicates that exploiting bisimilarity is an effective preprocessing step for fitting algorithms, especially when fitting a concept from a less expressive logic.

Second, we determine the efficacy of the implemented parallelization scheme by comparing the running times of our implementation with different number of threads on a selection of the SML-benchmarks. We measure the needed running time on the Mammographic, Carcinogenesis and Lymphography benchmarks to run bounded fitting up to size 8 with $t = 1, 2, 4$ and 8 worker threads.

We report full results in the long version [Funk *et al.*, 2026], and only mention the results on the Lymphography benchmark here: the mean measured running time over three runs with one worker thread is 51.8 seconds, with two worker threads 30.8 seconds, with four worker threads 21.5 seconds and with 8 worker threads 19.6 seconds. Thus, the running times decrease when the number of threads is increased. This effect is strongest when going from 1 to 2 threads.

These results indicate that our parallelization scheme is effective in making use of multiple threads, and enables our implementation to find larger concepts in practical time frames.

7 Conclusion

Our theoretical investigation has shown how to extend bounded fitting to handle inverse roles, qualified number restrictions, and feature comparisons, to preserve theoretical guarantees about its generalization abilities. Our practical evaluation confirms our findings and shows that bounded fitting is a promising approach for description logic concept learning, also for expressive description logics.

We believe that new benchmarks are needed to better evaluate single features of the implemented systems. We made a first step into this direction by providing a new benchmark using which we can test the ability of concept learners to deal with qualified number restrictions. We would like to create similar benchmarks for inverse roles and feature comparisons.

Ethical Statement

There are no ethical issues.

Acknowledgments

Jean Christoph Jung and Tom Voellmer were supported by DFG grant JU 3197/1-1.

References

[Baader *et al.*, 2017] Franz Baader, Ian Horrocks, Carsten Lutz, and Ulrike Sattler. *An Introduction to Description Logics*. Cambridge University Press, 2017.

[Blumer *et al.*, 1987] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Occam's razor. *Information Processing Letters*, 24(6):377–380, 1987.

[Blumer *et al.*, 1989] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM (JACM)*, 36(4):929–965, 1989.

[Bühmann *et al.*, 2018] Lorenz Bühmann, Jens Lehmann, Patrick Westphal, and Simon Bin. DL-Learner – structured machine learning on semantic web data. In *Companion Proceedings of the The Web Conference 2018*, pages 467–471, 2018.

[Cook, 1971] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, page 151–158. Association for Computing Machinery, 1971.

[de Rijke, 2000] Maarten de Rijke. A note on graded modal logic. *Stud Logica*, 64(2):271–283, 2000.

[Demir and Ngonga Ngomo, 2023] Caglar Demir and Axel-Cyrille Ngonga Ngomo. Neuro-symbolic class expression learning. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, (IJCAI-23)*, pages 3624–3632, 2023.

[Demir *et al.*, 2025] Caglar Demir, Moshood Yekini, Michael Röder, Yasir Mahmood, and Axel-Cyrille Ngonga Ngomo. Tree-based OWL class expression learner over large graphs. In *Machine Learning and Knowledge Discovery in Databases. Research Track - European Conference ECML PKDD 2025, Proceedings, Part III*, pages 495–511. Springer, 2025.

[Funk *et al.*, 2019] Maurice Funk, Jean Christoph Jung, Carsten Lutz, Hadrien Pulcini, and Frank Wolter. Learning description logic concepts: When can positive and negative examples be separated? In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19)*, pages 1682–1688, 2019.

[Funk *et al.*, 2025] Maurice Funk, Jean Christoph Jung, and Tom Voellmer. SAT-based bounded fitting for the description logic $\mathcal{ALC}$. In *Proceedings of the 24th International Semantic Web Conference (ISWC)*, 2025.

[Funk *et al.*, 2026] Maurice Funk, Jean Christoph Jung, and Tom Voellmer. Bounded fitting for expressive description logics. *CoRR*, abs/2605.07452, 2026.

[Grohe *et al.*, 2021] Martin Grohe, Kristian Kersting, Martin Mladenov, and Pascal Schweitzer. Color refinement and its applications. In *An Introduction to Lifted Probabilistic Inference*, page 349–372. The MIT Press, August 2021.

[Heindorf *et al.*, 2022] Stefan Heindorf, Lukas Blübaum, Nick Düsterhus, Till Werner, Varun Nandkumar Golani, Caglar Demir, and Axel-Cyrille Ngonga Ngomo. EvoLearner: Learning description logics with evolutionary algorithms. In *WWW '22: The ACM Web Conference 2022*, pages 818–828. ACM, 2022.

[Jung *et al.*, 2022] Jean Christoph Jung, Carsten Lutz, Hadrien Pulcini, and Frank Wolter. Logical separability of labeled data examples under ontologies. *Artificial Intelligence*, 313:103785, 2022.

[Kearns and Valiant, 1994] Michael J. Kearns and Leslie G. Valiant. Cryptographic limitations on learning Boolean formulae and finite automata. *Journal of the ACM (JACM)*, 41(1):67–95, 1994.

[Kouagou *et al.*, 2023] N'Dah Jean Kouagou, Stefan Heindorf, Caglar Demir, and Axel-Cyrille Ngonga Ngomo. Neural class expression synthesis in *ALCHIQ(D)*. In *Machine Learning and Knowledge Discovery in Databases: Research Track - European Conference, ECML PKDD 2023*, pages 196–212. Springer, 2023.

[Kouagou *et al.*, 2024] N'Dah Jean Kouagou, Stefan Heindorf, Caglar Demir, and Axel-Cyrille Ngonga Ngomo. ROCES: robust class expression synthesis in description logics via iterative sampling. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI-24)*, pages 4335–4343, 2024.

[Lehmann and Hitzler, 2010] Jens Lehmann and Pascal Hitzler. Concept learning in description logics using refinement operators. *Machine Learning*, 78:203–250, 2010.

[Lehmann, 2010] Jens Lehmann. *Learning OWL Class Expressions*, volume 6 of *Studies on the Semantic Web*. IOS Press, 2010.

[Martins, 2019] Denis Mayr Lima Martins. Reverse engineering database queries from examples: State-of-the-art, challenges, and research opportunities. *Information Systems*, 83:89–100, 2019.

[Neider and Gavran, 2018] Daniel Neider and Ivan Gavran. Learning linear temporal properties. In *Proceedings of 18th Conference on Formal Methods in Computer Aided Design (FMCAD)*, pages 1–10. IEEE, 2018.

[Pommellet *et al.*, 2024] Adrien Pommellet, Daniel Stan, and Simon Scatton. SAT-based learning of computation tree logic. In *Proceedings of 12th International Joint Conference on Automated Reasoning (IJCAR)*, pages 366–385. Springer, 2024.

[Riener, 2019] Heinz Riener. Exact synthesis of LTL properties from traces. In *Forum for Specification and Design Languages (FDL 2019)*, pages 1–6. IEEE, 2019.

[Rizzo *et al.*, 2020] Giuseppe Rizzo, Nicola Fanizzi, and Claudia d'Amato. Class expression induction as concept

space exploration: From DL-FOIL to DL-FOCL. *Future Gener. Comput. Syst.*, 108:256–272, 2020.

[Sinz, 2005] Carsten Sinz. Towards an optimal CNF encoding of boolean cardinality constraints. In *Principles and Practice of Constraint Programming - CP 2005, 11th International Conference (CP 2005), Proceedings*, pages 827–831. Springer, 2005.

[Suchanek *et al.*, 2024] Fabian M. Suchanek, Mehwish Alam, Thomas Bonald, Lihu Chen, Pierre-Henri Paris, and Jules Soria. YAGO 4.5: A large and clean knowledge base with a rich taxonomy. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2024)*, pages 131–140. ACM, 2024.

[ten Cate *et al.*, 2023a] Balder ten Cate, Maurice Funk, Jean Christoph Jung, and Carsten Lutz. Fitting algorithms for conjunctive queries. *SIGMOD Rec.*, 52(4):6–18, 2023.

[ten Cate *et al.*, 2023b] Balder ten Cate, Maurice Funk, Jean Christoph Jung, and Carsten Lutz. SAT-based PAC learning of description logic concepts. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence (IJCAI 2023)*, pages 3347–3355, 2023.

[Tran *et al.*, 2017] An C. Tran, Jens Dietrich, Hans W. Guesgen, and Stephen Marsland. Parallel symmetric class expression learning. *Journal of Machine Learning Research*, 18:64:1–64:34, 2017.

[Valiant, 1984] Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[W3C, 2012] W3C. OWL 2 web ontology language overview. https://www.w3.org/TR/owl2-overview/, 2012. Accessed: July 24, 2025.

[Westphal *et al.*, 2019] Patrick Westphal, Lorenz Bühmann, Simon Bin, Hajira Jabeen, and Jens Lehmann. SML-bench - A benchmarking framework for structured machine learning. *Semantic Web*, 10(2):231–245, 2019.

[Zloof, 1975] Moshé M. Zloof. Query by example. In *American Federation of Information Processing Societies: 1975 National Computer Conference*, volume 44 of *AFIPS Conference Proceedings*, pages 431–438. AFIPS Press, 1975.