Finding Good Proofs for Description Logic Entailments Using Recursive Quality Measures

Christian Alrabbaa[0000−0002−2925−1765], Franz Baader[0000−0002−4049−221X],
Stefan Borgwardt[0000−0003−0924−8478], Patrick Koopmann[0000−0001−5999−2583],
and Alisa Kovtunova[0000−0001−9936−0943]

Theoretical Computer Science, TU Dresden, Dresden, Germany

Abstract. Logic-based approaches to AI have the advantage that their behavior can in principle be explained to a user. If, for instance, a Description Logic reasoner derives a consequence that triggers some action of the overall system, then one can explain such an entailment by presenting a proof of the consequence in an appropriate calculus. How comprehensible such a proof is depends not only on the employed calculus, but also on the properties of the particular proof, such as its overall size, its depth, the complexity of the employed sentences and proof steps, etc. For this reason, we want to determine the complexity of generating proofs that are below a certain threshold w.r.t. a given measure of proof quality. Rather than investigating this problem for a fixed proof calculus and a fixed measure, we aim for general results that hold for wide classes of calculi and measures. In previous work, we first restricted the attention to a setting where proof size is used to measure the quality of a proof. We then extended the approach to a more general setting, but important measures such as proof depth were not covered. In the present paper, we provide results for a class of measures called recursive, which yields lower complexities and also encompasses proof depth. In addition, we close some gaps left open in our previous work, thus providing a comprehensive picture of the complexity landscape.

1 Introduction

Explainability has developed into a major issue in Artificial Intelligence, particularly in the context of sub-symbolic approaches based on Machine Learning [6]. In contrast, results produced by symbolic approaches based on logical reasoning are "explainable by design" since a derived consequence can be formally justified by showing a proof for it. In practice, things are not that easy since proofs may be very long, and even single proof steps or stated sentences may be hard to comprehend for a user that is not an expert in logic. For this reason, there has been considerable work in the Automated Deduction and Logic in AI communities on how to produce "good" proofs for certain purposes, both for full first-order logic, but also for decidable logics such a Description Logics (DLs) [9]. We mention here only a few approaches, and refer the reader to the introduction of our previous work [2] for a more detailed review.

First, there is work that transforms proofs that are produced by an automated reasoning system into ones in a calculus that is deemed to be more appropriate for human consumption [11, 22, 23]. Second, abstraction techniques are used to reduce the size of proofs by introducing definitions, lemmas, and more abstract deduction rules [16, 17]. Justification-based explanations for DLs [10, 14, 28] can be seen as a radical abstraction technique where the abstracted proof consists of a single proof step, from a minimal set of stated sentences that implies a certain consequence directly to this consequence. Finally, instead of presenting proofs in a formal, logical syntax, one can also try to increase readability by translating them into natural language text [12, 25–27] or visualizing them [5].

The purpose of this work is of a more (complexity) theoretic nature. We want to investigate how hard it is to find good proofs, where the quality of a proof is described by a measure $\mathfrak{m}$ that assigns non-negative rational numbers to proofs. More precisely, as usual we investigate the complexity of the corresponding decision problem, i.e., the problem of deciding whether there is a proof $\mathcal{P}$ with $\mathfrak{m}(\mathcal{P}) \leq q$ for a given rational number q. In order to abstract from specific logics and proof calculi, we develop a general framework in which proofs are represented as labeled, directed hypergraphs, whose hyperedges correspond to single sound derivation steps. To separate the complexity of generating good proofs from the complexity of reasoning in the underlying logic, we introduce the notion of a *deriver*, which generates a so-called *derivation structure*. This structure consists of possible proof steps, from which all proofs of the given consequence can be constructed. Basically, such a derivation structure can be seen as consisting of all relevant instantiations of the rules of a calculus that can be used to derive the consequence. We restrict the attention to decidable logics and consider derivers that produce derivation structures of polynomial or exponential size. Examples of such derivers are consequence-based reasoners for the DLs $\mathcal{EL}$ [7, 21] and $\mathcal{ELI}$ [9, 18], respectively. In our complexity results, the derivation structure is assumed to be already computed by the deriver,[1] i.e., the complexity of this step is not assumed to be part of the complexity of computing good proofs. Our complexity results investigate the problem along the following orthogonal dimensions: we distinguish between (i) polynomial and exponential derivers; and (ii) whether the threshold value q is encoded in unary or binary. The obtained complexity upper bounds hold for all instances of a considered setting, whereas the lower bounds mean that there is an instance (usually based on $\mathcal{EL}$ or $\mathcal{ELI}$) for which this lower bound can be proved.

In our first work in this direction [2], we focused our attention on *size* as the measure of proof quality. We could show that the above decision problem is NP-complete even for polynomial derivers and unary coding of numbers. For exponential derivers, the complexity depends on the coding of numbers: NP-complete (NExpTime-complete) for unary (binary) coding. For the related measure *tree size* (which assumes that the proof hypergraphs are tree-shaped, i.e. cannot reuse already derived consequences), the complexity turned out to

---

[1] The highly efficient reasoner ELK [21] for (an extension of) $\mathcal{EL}$ actually produces a derivation structure, and thus is a deriver in our sense.

Table 1. Overview over existing and new complexity results for deciding the existence of good proofs, w.r.t. polynomial/exponential derivers and unary/binary encoding of the bound q (known results in gray).

Measure	polynomial unary	polynomial binary	exponential unary	exponential binary
Size	NP [2]	NP [2]	NExpTime [2]	NExpTime [2]
Monotone recursive Φ-measures	$\leq$ P	$\leq$ P [Th.12]	$\leq$ ExpTime	$\leq$ ExpTime [Th.12]
Tree size	P [2]	P	NP [2]	PSpace [Th.17,18]
Depth	P [Th.14]	P	PSpace [Th.16]	ExpTime [Th.14]
Logarithmic depth	P [Cor.15]	P	ExpTime [Cor.15]	ExpTime

be considerably lower, due to the fact that a Dijkstra-like greedy algorithm can be applied. In [3], we generalized the results by introducing a class of measures called *Ψ-measures*, which contains both size and tree size and for which the same complexity upper bounds as for size could be shown for polynomial derivers. We also lifted the better upper bounds for tree size (for polynomial derivers) to *local Ψ-measures*, a natural class of proof measures. In this paper, we extend this line of research by providing a more general notion of measures, *monotone recursive Φ-measures*, which now also allow to measure the *depth* of a proof. We think that depth is an important measure since it measures how much of the proof tree a (human or automated) proof checker needs to keep in memory at the same time. We analyze these measures not only for polynomial derivers, but this time also consider exponential derivers, thus giving insights on how our complexity results transfer to more expressive logics. In addition to upper bounds for the general class of monotone recursive Φ-measures, we show improved bounds for the specific measures considering depth and tree size, in the latter case improving results from [2]. Overall, we thus obtain a comprehensive picture of the complexity landscape for the problem of finding good proofs for DL and other entailments (see Table 1).

An extended version of this paper with detailed proofs can be found at [4].

2 Preliminaries

Most of our theoretical discussion applies to arbitrary *logics* $\mathcal{L} = (\mathcal{S}_\mathcal{L}, \models_\mathcal{L})$ that consist of a set $\mathcal{S}_\mathcal{L}$ of *$\mathcal{L}$-sentences* and a *consequence relation* $\models_\mathcal{L} \subseteq P(\mathcal{S}_\mathcal{L}) \times \mathcal{S}_\mathcal{L}$ between *$\mathcal{L}$-theories*, i.e. subsets of $\mathcal{L}$-sentences, and single $\mathcal{L}$-sentences. We assume that $\models_\mathcal{L}$ has a semantic definition, i.e. for some definition of "model", $\mathcal{T} \models_\mathcal{L} \eta$ holds iff every model of all elements in $\mathcal{T}$ is also a model of η. We also assume that the *size* $|\eta|$ of an $\mathcal{L}$-sentence η is defined in some way, e.g. by the number of symbols in η. Since $\mathcal{L}$ is usually fixed, we drop the prefix "$\mathcal{L}$-" from now on. For example, $\mathcal{L}$ could be *first-order logic*. However, we are mainly interested in proofs for DLs, which can be seen as decidable fragments of first-order logic [9]. In particular, we use specific DLs to show our hardness results.

$$\mathsf{R}_0 \ \overline{C \sqsubseteq C} \qquad \mathsf{R}_\top \ \overline{C \sqsubseteq \top} \qquad \mathsf{R}_\sqsubseteq \ \frac{C \sqsubseteq D}{C \sqsubseteq E} : D \sqsubseteq E \in \mathcal{T} \qquad \mathsf{R}^-_{\sqcap,1} \ \frac{C \sqsubseteq D \sqcap E}{C \sqsubseteq D}$$

$$\mathsf{R}^-_{\sqcap,2} \ \frac{C \sqsubseteq D \sqcap E}{C \sqsubseteq E} \qquad \mathsf{R}^+_\sqcap \ \frac{C \sqsubseteq D \quad C \sqsubseteq E}{C \sqsubseteq D \sqcap E} \qquad \mathsf{R}_\exists \ \frac{C \sqsubseteq \exists r.D \quad D \sqsubseteq E}{C \sqsubseteq \exists r.E}$$

Fig. 1. The inference rules for $\mathcal{EL}$ used in ELK [21].

The syntax of DLs is based on disjoint, countably infinite sets $\mathsf{N_C}$ and $\mathsf{N_R}$ of *concept names* $A, B, \ldots$ and *role names* $r, s, \ldots$, respectively. Sentences of the DL $\mathcal{EL}$, called *general concept inclusions (GCIs)*, are of the form $C \sqsubseteq D$, where C and D are $\mathcal{EL}$-*concepts*, which are built from concept names by applying the constructors $\top$ (*top*), $C \sqcap D$ (*conjunction*), and $\exists r.C$ (*existential restriction* for a role name r). The DL $\mathcal{ELI}$ extends $\mathcal{EL}$ by the role constructor r^- (*inverse role*). In DLs, finite theories are called *TBoxes* or *ontologies*.

The semantics of DLs is based on first-order interpretations; for details, see [9]. In Figure 1, we depict a simplified version of the inference rules for $\mathcal{EL}$ from [21]. For example, $\{A \sqsubseteq \exists r.B, \ B \sqsubseteq C, \ \exists r.C \sqsubseteq D\} \models A \sqsubseteq D$ is a valid inference in $\mathcal{EL}$. Deciding consequences in $\mathcal{EL}$ is P-complete [7], and in $\mathcal{ELI}$ it is ExpTime-complete [8].

2.1 Proofs

We formalize proofs as (labeled, directed) *hypergraphs* (see Figures 2, 3), which are tuples (V, E, ℓ) consisting of a finite set V of *vertices*, a finite set E of *(hyper)edges* of the form (S, d) with $S \subseteq V$ and $d \in V$, and a *vertex labeling function* $\ell \colon V \to \mathcal{S}_\mathcal{L}$. Full definitions of such hypergraphs, as well as related notions such as *trees*, *unravelings*, *homomorphisms*, *cycles* can be found in the extended version [4]. For example, there is a homomorphism from Figure 3 to Figure 2, but not vice versa, and Figure 3 is the tree unraveling of Figure 2.
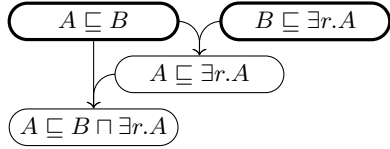

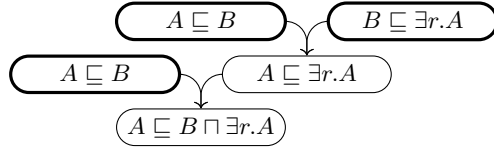**Fig. 2.** An acyclic hypergraph/proof   **Fig. 3.** A tree hypergraph/proof

The following definition formalizes basic requirements for hyperedges to be considered valid inference steps from a given finite theory.

Definition 1 (Derivation Structure). *A derivation structure* $\mathcal{D} = (V, E, \ell)$ *over a finite theory* $\mathcal{T}$ *is a hypergraph that is*

  − grounded, *i.e. every leaf v in $\mathcal{D}$ is labeled by $\ell(v) \in \mathcal{T}$; and*

– sound, *i.e. for every $(S, d) \in E$, the entailment $\{\ell(s) \mid s \in S\} \models \ell(d)$ holds.*

We define proofs as special derivation structures that derive a conclusion.

Definition 2 (Proof). *Given a conclusion η and a finite theory $\mathcal{T}$, a proof for $\mathcal{T} \models \eta$ is a derivation structure $\mathcal{P} = (V, E, \ell)$ over $\mathcal{T}$ such that*

– *$\mathcal{P}$ contains exactly one sink $v_\eta \in V$, which is labeled by η,*
– *$\mathcal{P}$ is acyclic, and*
– *every vertex has at most one incoming edge, i.e. there is no vertex $w \in V$ s.t. there are $(S_1, w), (S_2, w) \in E$ with $S_1 \neq S_2$.*

A tree proof *is a proof that is a tree. A* subproof S *of a hypergraph H is a subgraph of H that is a proof s.t. the leaves of S are a subset of the leaves of H.*

The hypergraphs in Figures 2 and 3 can be seen as proofs in the sense of Definition 2, where the sentences of the theory are marked with a thick border. Both proofs use the same inference steps, but have different numbers of vertices. They both prove $A \sqsubseteq B \sqcap \exists r.A$ from $\mathcal{T} = \{A \sqsubseteq B,\ B \sqsubseteq \exists r.A\}$. The second proof is a tree and the first one a hypergraph without label repetition.

Lemma 3. *Let $\mathcal{P} = (V, E, \ell)$ be a proof for $\mathcal{T} \models \eta$. Then*

1. *all paths in $\mathcal{P}$ are finite and all longest paths in $\mathcal{P}$ have v_η as the target; and*
2. *$\mathcal{T} \models \eta$.*

Given a proof $\mathcal{P} = (V, E, \ell)$ and a vertex $v \in V$, the *subproof of $\mathcal{P}$ with sink v* is the largest subgraph $\mathcal{P}_v = (V_v, E_v, \ell_v)$ of $\mathcal{P}$ where V_v contains all vertices in V that have a path to v in $\mathcal{P}$.

2.2 Derivers

In practice, proofs and derivation structures are constructed by a reasoning system, and in theoretical investigations, it is common to define proofs by means of a calculus. To abstract from these details, we use the concept of a *deriver* as in [2], which is a function that, given a theory $\mathcal{T}$ and a conclusion η, produces the corresponding derivation structure in which we can look for an optimal proof. However, in practice, it would be inefficient and unnecessary to compute the entire derivation structure beforehand when looking for an optimal proof. Instead, we allow to access elements in a derivation structure using an oracle, which we can ask whether given inferences are a part of the current derivation structure. Similar functionality exists for example for the DL reasoner ELK [19], and may correspond to checking whether the inference is an instance of a rule in the calculus. Since reasoners may not be complete for proving arbitrary sentences of $\mathcal{L}$, we restrict the conclusion η to a subset $C_\mathcal{L} \subseteq \mathcal{S}_\mathcal{L}$ of supported consequences.

Definition 4 (Deriver). *A* deriver $\mathfrak{D}$ *is given by a set $C_\mathcal{L} \subseteq \mathcal{S}_\mathcal{L}$ and a function that assigns derivation structures to pairs $(\mathcal{T}, \eta)$ of finite theories $\mathcal{T} \subseteq \mathcal{S}_\mathcal{L}$ and sentences $\eta \in C_\mathcal{L}$, such that $\mathcal{T} \models \eta$ iff $\mathfrak{D}(\mathcal{T}, \eta)$ contains a proof for $\mathcal{T} \models \eta$. A*

$$\mathsf{CR1} \ \frac{}{K \sqsubseteq A} \ \text{if } A \in K \text{ and } K \text{ appears in } \mathcal{T}'$$

$$\mathsf{CR2} \ \frac{M \sqsubseteq A \text{ for all } A \in K, \ K \sqsubseteq C}{M \sqsubseteq C} \ \text{if } M \text{ appears in } \mathcal{T}'$$

$$\mathsf{CR3} \ \frac{M \sqsubseteq \exists r.L \quad L \sqsubseteq \forall r^-.A}{M \sqsubseteq A} \qquad \mathsf{CR4} \ \frac{L \sqsubseteq \exists r.M \quad L \sqsubseteq \forall r.A}{L \sqsubseteq \exists r.(M \sqcap A)}$$

Fig. 4. The inference rules for $\mathcal{ELI}$ [9]. Given a finite theory $\mathcal{T}$ in a certain normal form, the rules produce a saturated theory $\mathcal{T}'$. Here, K, L, M are conjunctions of concept names, A is a concept name, C is an $\mathcal{ELI}$ concept of the form A, $\exists r.M$, or $\forall r.A$, and r is a role name or the inverse of a role name. In this calculus conjunctions are implicitly viewed as sets, i.e. the order and multiplicity of conjuncts is ignored.

proof $\mathcal{P}$ for $\mathcal{T} \models \eta$ is called admissible w.r.t. $\mathfrak{D}(\mathcal{T}, \eta)$ *if there is a homomorphism* $h \colon \mathcal{P} \to \mathfrak{D}(\mathcal{T}, \eta)$. *We call $\mathfrak{D}$ a* polynomial deriver *if there exists a polynomial $p(x)$ such that the size of $\mathfrak{D}(\mathcal{T}, \eta)$ is bounded by $p(|\mathcal{T}| + |\eta|)$.* Exponential derivers *are defined similarly by the restriction $|\mathfrak{D}(\mathcal{T}, \eta)| \leq 2^{p(|\mathcal{T}| + |\eta|)}$.*

ELK is an example of a polynomial deriver, that is, for a given $\mathcal{EL}$ theory $\mathcal{T}$ and $\mathcal{EL}$ sentence η, $\text{ELK}(\mathcal{T}, \eta)$ contains all allowed instances of the rules shown in Figure 1. As an example for an exponential deriver we use ELI, which uses the rules from Figure 4 and is complete for $\mathcal{ELI}$ theories and conclusions of the form $A \sqsubseteq B$, $A, B \in \mathsf{N_C}$. The oracle access for a deriver $\mathfrak{D}$ works as follows. Let $\mathcal{D} = (V, E, \ell) := \mathfrak{D}(\mathcal{T}, \eta)$ and $V = \{v_1, \dots, v_m\}$. $\mathcal{D}$ is accessed using the following two functions, where $i, i_1, \dots, i_l$ are indices of vertices and α is a sentence:

$$[\mathcal{D}](i_1, \dots, i_l, i) := \begin{cases} \mathsf{true} & \text{if } (\{v_{i_1}, \dots, v_{i_l}\}, v_i) \in E, \\ \mathsf{false} & \text{otherwise;} \end{cases}$$

$$[\mathcal{D}](i, \alpha) := \begin{cases} \mathsf{true} & \text{if } \ell(v_i) = \alpha, \\ \mathsf{false} & \text{otherwise.} \end{cases}$$

In this paper, we focus on polynomial and exponential derivers, for which we further make the following technical assumptions: 1) $\mathfrak{D}(\mathcal{T}, \eta)$ does not contain two vertices with the same label; 2) the number of premises in an inference is polynomially bounded by $|\mathcal{T}|$ and $|\eta|$; and 3) the size of each label is polynomially bounded by $|\mathcal{T}|$ and $|\eta|$. While 1) is without loss of generality, 2) and 3) are not. If a deriver does not satisfy 2), we may be able to fix this by splitting inference steps. Assumption 3) would not work for derivers with higher complexity, but is required in our setting to avoid trivial complexity results for exponential derivers. We furthermore assume that for polynomial and exponential derivers, the polynomial p from Definition 4 bounding the size of derivation structures is known.

3   Measuring Proofs

To formally study quality measures for proofs, we developed the following definition, which will be instantiated with concrete measures later. Our goal is to find proofs that minimize these measures, i.e. lower numbers are better.

Definition 5 (Φ-Measure). *A* (quality) measure *is a function* $\mathfrak{m}\colon \mathrm{P}_{\mathcal{L}} \to \mathbb{Q}_{\geq 0}$, *where* $\mathrm{P}_{\mathcal{L}}$ *is the set of all proofs over* $\mathcal{L}$ *and* $\mathbb{Q}_{\geq 0}$ *is the set of non-negative rational numbers. We call* $\mathfrak{m}$ *a* Φ-measure *if, for every* $\mathcal{P} \in \mathrm{P}_{\mathcal{L}}$, *the following hold.*

- **[P]** $\mathfrak{m}(\mathcal{P})$ *is computable in **p**olynomial time in the size of* $\mathcal{P}$.
- **[HI]** *Let* $h\colon \mathcal{P} \to H$ *be any homomorphism, and* $\mathcal{P}'$ *be any subproof of the **h**omomorphic **i**mage* $h(\mathcal{P})$ *that is minimal (w.r.t.* $\mathfrak{m}$*) among all such subproofs having the same sink. Then* $\mathfrak{m}(\mathcal{P}') \leq \mathfrak{m}(\mathcal{P})$.

Intuitively, a Φ-measure $\mathfrak{m}$ does not increase when the proof gets smaller, either when parts of the proof are removed (to obtain a subproof) or when parts are merged (in a homomorphic image). For example, $\mathfrak{m}_{\mathsf{size}}((V, E, \ell)) := |V|$ is a Φ-measure, called the *size* of a proof, and we have already investigated the complexity of the following deicision problem for $\mathfrak{m}_{\mathsf{size}}$ in [2].

Definition 6 (Optimal Proof). *Let* $\mathfrak{D}$ *be a deriver and* $\mathfrak{m}$ *be a measure. Given a finite theory* $\mathcal{T}$ *and a sentence* $\eta \in C_{\mathcal{L}}$ *s.t.* $\mathcal{T} \models \eta$, *an admissible proof* $\mathcal{P}$ *w.r.t.* $\mathfrak{D}(\mathcal{T}, \eta)$ *is called* optimal *w.r.t.* $\mathfrak{m}$ *if* $\mathfrak{m}(\mathcal{P})$ *is minimal among all such proofs. The associated decision problem, denoted* $\mathsf{OP}(\mathfrak{D}, \mathfrak{m})$, *is to decide, given* $\mathcal{T}$ *and* η *as above and* $q \in \mathbb{Q}_{\geq 0}$, *whether there is an admissible proof* $\mathcal{P}$ *w.r.t.* $\mathfrak{D}(\mathcal{T}, \eta)$ *with* $\mathfrak{m}(\mathcal{P}) \leq q$.

For our complexity analysis, we distinguish the encoding of q with a subscript (unary/binary), e.g. $\mathsf{OP}_{\mathsf{unary}}(\mathfrak{D}, \mathfrak{m})$.

   We first show that if $\mathcal{P}$ is optimal w.r.t. a Φ-measure $\mathfrak{m}$ and $\mathfrak{D}(\mathcal{T}, \eta)$, then the homomorphic image of $\mathcal{P}$ in $\mathfrak{D}(\mathcal{T}, \eta)$ is also a proof. Thus, to decide $\mathsf{OP}(\mathfrak{D}, \mathfrak{m})$ we can restrict our search to proofs that are subgraphs of $\mathfrak{D}(\mathcal{T}, \eta)$.

Lemma 7. *For any deriver* $\mathfrak{D}$ *and* Φ-measure $\mathfrak{m}$, *if there is an admissible proof* $\mathcal{P}$ *w.r.t.* $\mathfrak{D}(\mathcal{T}, \eta)$ *with* $\mathfrak{m}(\mathcal{P}) \leq q$ *for some* $q \in \mathbb{Q}_{\geq 0}$, *then there exists a subproof* $\mathcal{Q}$ *of* $\mathfrak{D}(\mathcal{T}, \eta)$ *for* $\mathcal{T} \models \eta$ *with* $\mathfrak{m}(\mathcal{Q}) \leq q$.

   In particular, this shows that an optimal proof always exists.

Corollary 8. *For any deriver* $\mathfrak{D}$ *and* Φ-measure $\mathfrak{m}$, *if* $\mathcal{T} \models \eta$, *then there is an optimal proof for* $\mathcal{T} \models \eta$ *w.r.t.* $\mathfrak{D}$ *and* $\mathfrak{m}$.

Proof. By Definition 4, the derivation structure $\mathfrak{D}(\mathcal{T}, \eta)$ contains at least one proof for $\mathcal{T} \models \eta$. Since $\mathfrak{D}(\mathcal{T}, \eta)$ is finite, there are finitely many proofs for $\mathcal{T} \models \eta$ contained in $\mathfrak{D}(\mathcal{T}, \eta)$. The finite set of all $\mathfrak{m}$-weights of these proofs always has a minimum. Finally, if there were an admissible proof weighing less than this minimum, it would contradict Lemma 7. $\qquad\square$

3.1   Monotone Recursive Measures

Since the complexity of $\mathsf{OP}(\mathfrak{D}, \mathfrak{m})$ for Φ-measures in general is quite high [2], in this paper we focus on a subclass of measures that can be evaluated recursively.

Definition 9. *A Φ-measure $\mathfrak{m}$ is* recursive *if there exist*

- *a leaf function* $\mathsf{leaf}_{\mathfrak{m}} : \mathcal{S}_{\mathcal{L}} \to \mathbb{Q}_{\geq 0}$ *and*
- *a partial edge function* $\mathsf{edge}_{\mathfrak{m}}$*, which maps (i) the labels $(\mathcal{S}, \alpha)$ of a hyperedge and (ii) a finite multiset $\mathcal{Q}$ of already computed intermediate weights in $\mathbb{Q}_{\geq 0}$ to a combined weight* $\mathsf{edge}_{\mathfrak{m}}\big((\mathcal{S}, \alpha), \mathcal{Q}\big)$

such that, for any proof $\mathcal{P} = (V, E, \ell)$ with sink v, we have

$$\mathfrak{m}(\mathcal{P}) = \begin{cases} \mathsf{leaf}_{\mathfrak{m}}(\ell(v)) & \text{if } V = \{v\}, \\ \mathsf{edge}_{\mathfrak{m}}\big(\ell(S, v), \{\mathfrak{m}(\mathcal{P}_w) \mid w \in S\}\big) & \text{if } (S, v) \in E. \end{cases}$$

Such a measure is monotone *if, for any multiset $\mathcal{Q}$, whenever $q \in \mathcal{Q}$ and $\mathcal{Q}' = (\mathcal{Q} \setminus \{q\}) \cup \{q'\}$ with $q' \leq q$ and both $\mathsf{edge}_{\mathfrak{m}}\big((\mathcal{S}, \alpha), \mathcal{Q}'\big)$ and $\mathsf{edge}_{\mathfrak{m}}\big((\mathcal{S}, \alpha), \mathcal{Q}\big)$ are defined, then $\mathsf{edge}_{\mathfrak{m}}\big((\mathcal{S}, \alpha), \mathcal{Q}'\big) \leq \mathsf{edge}_{\mathfrak{m}}\big((\mathcal{S}, \alpha), \mathcal{Q}\big)$.*

Intuitively, a recursive measure $\mathfrak{m}$ can be computed in a bottom-up fashion starting with the weights of the leaves given by $\mathsf{leaf}_{\mathfrak{m}}$. The function $\mathsf{edge}_{\mathfrak{m}}$ is used to recursively combine the weights of the direct subproofs into a weight for the full proof. This function is well-defined since in a proof every vertex has at most one incoming edge. We require $\mathsf{edge}_{\mathfrak{m}}$ to be defined only for inputs $\big((\mathcal{S}, \alpha), \mathcal{Q}\big)$ that actually correspond to a valid proof in $\mathcal{L}$, i.e. where $\mathcal{S} \models_{\mathcal{L}} \alpha$ and $\mathcal{Q}$ consists of the weights of some proofs for the sentences in $\mathcal{S}$. For example, if $\mathfrak{m}$ always yields natural numbers, we obviously do not need $\mathsf{edge}_{\mathfrak{m}}$ to be defined for multisets containing fractional numbers.

In this paper, we are particularly interested in the following monotone recursive Φ-measures.

- The *depth* $\mathfrak{m}_{\mathsf{depth}}$ of a proof is defined by

$$\mathsf{leaf}_{\mathfrak{m}_{\mathsf{depth}}}(\alpha) := 0 \text{ and } \mathsf{edge}_{\mathfrak{m}_{\mathsf{depth}}}\big((\mathcal{S}, \alpha), \mathcal{Q}\big) := 1 + \max \mathcal{Q}.$$

- The *tree size* $\mathfrak{m}_{\mathsf{tree}}$ is given by

$$\mathsf{leaf}_{\mathfrak{m}_{\mathsf{tree}}}(\alpha) := 1 \text{ and } \mathsf{edge}_{\mathfrak{m}_{\mathsf{tree}}}\big((\mathcal{S}, \alpha), \mathcal{Q}\big) := 1 + \sum \mathcal{Q}.$$

What distinguishes *tree size* from *size* is that vertices are counted multiple times if they are used in several subproofs. The name *tree size* is inspired by the fact that it can be interpreted as the *size* of the tree unraveling of a given proof (cf. Figures 2 and 3). In fact, we show in the extended version [4] that all recursive Φ-measures are invariant under unraveling. This indicates that *tree size*, *depth* and other monotone recursive Φ-measures are especially well-suited for cases where proofs are presented to users in the form of trees. This is for example the case for the proof plugin for Protégé [20].

Lemma 10. Depth *and* tree size *are monotone recursive Φ-measures.*

Algorithm 1: A Dijkstra-like algorithm

Input: A derivation structure $\mathfrak{D}(\mathcal{T}, \eta) = (V, E, \ell)$, a monotone recursive
Φ-measure $\mathfrak{m}$

Output: An optimal proof of $\mathcal{T} \models \eta$ w.r.t. $\mathfrak{D}(\mathcal{T}, \eta)$ and $\mathfrak{m}$

1 $Q := \emptyset$
2 **foreach** $e \in E$ **do** $k(e) := 0$
3 **foreach** $v \in V$ **do**
4   **if** $\ell(v) \in \mathcal{T}$ **then**
5 $\mathcal{P}(v) := (\{v\}, \emptyset, \ell|_{\{v\}}); Q := Q \cup \{v\}$ `// `$\ell(v)$` is in the theory`
6   **else if** $(\emptyset, v) \in E$ **then**
7 $\mathcal{P}(v) := (\{v\}, \{(\emptyset, v)\}, \ell|_{\{v\}}); Q := Q \cup \{v\}$ `// `$\ell(v)$` is a tautology`
8   **else**
9 $\mathcal{P}(v) :=$ undefined
10 **while** $Q \neq \emptyset$ **do**
11   choose $v \in Q$ with minimal $\mathfrak{m}(\mathcal{P}(v))$ `// `$\mathcal{P}(v)$` is optimal for `$\ell(v)$
12 $Q := Q \setminus \{v\}$
13   **foreach** $e = (S, d) \in E$ with $v \in S$ **do**
14 $k(e) := k(e) + 1$
15   **if** $k(e) = |S|$ **then**   `// all source vertices have been reached`
16 $\mathcal{P} := (S \cup \{d\}, e, \ell_{S \cup \{d\}}) \cup \bigcup_{s \in S} \mathcal{P}(s)$ `// construct new proof`
17   **if** $\mathcal{P}$ is acyclic **then**
18   **if** $\mathcal{P}(d)$ is undefined or $\mathfrak{m}(\mathcal{P}(d)) > \mathfrak{m}(\mathcal{P})$ **then**
19 $\mathcal{P}(d) := \mathcal{P}; Q := Q \cup \{d\}$ `// `$\mathcal{P}$` is better for `$\ell(d)$
20 **return** $\mathcal{P}(v_\eta)$, where $\ell(v_\eta) = \eta$

---

4 Complexity Results

We investigate the decision problem OP for monotone recursive Φ-measures. We first show upper bounds for the general case, and then consider measures for *depth* and *tree size*, for which we obtain even lower bounds. An artificial modification of the *depth* measure gives a lower bound matching the general upper bound even if unary encoding is used for the threshold q.

4.1 The General Case

Algorithm 1 describes a Dijkstra-like approach that is inspired by the algorithm in [13] for finding minimal hyperpaths w.r.t. so-called *additive weighting functions*, which represent a subclass of monotone recursive Φ-measures. The algorithm progressively discovers proofs $\mathcal{P}(v)$ for $\ell(v)$ that are contained in $\mathfrak{D}(\mathcal{T}, \eta)$. If it reaches a new vertex v in this process, this vertex is added to the set Q. In each step, a vertex with minimal weight $\mathfrak{m}(\mathcal{P}(v))$ is chosen and removed from Q. For each hyperedge $e = (S, d) \in E$, a counter $k(e)$ is maintained that is increased whenever a vertex $v \in S$ is chosen. Once this counter reaches $|S|$, we know that all source vertices of e have been processed. The algorithm then constructs a new proof $\mathcal{P}$ for $\ell(d)$ by joining the proofs for the source vertices using the

current hyperedge e. This proof $\mathcal{P}$ is then compared to the best previously known proof $\mathcal{P}(d)$ for $\ell(d)$ and $\mathcal{P}(d)$ is updated accordingly. For Line 20, recall that we assumed $\mathfrak{D}(\mathcal{T}, \eta)$ to contain no two vertices with the same label, and hence it contains a unique vertex v_η with label η.

Lemma 11. *For any monotone recursive Φ-measure $\mathfrak{m}$ and deriver $\mathfrak{D}$, Algorithm 1 computes an optimal proof in time polynomial in the size of $\mathfrak{D}(\mathcal{T}, \eta)$.*

Since we can actually compute an optimal proof in polynomial time in the size of the whole derivation structure, it is irrelevant how the upper bound q in the decision problem OP is encoded, and hence the following results follow.

Theorem 12. *For any monotone recursive Φ-measure $\mathfrak{m}$ and polynomial deriver $\mathfrak{D}$, $\mathsf{OP}_{\mathsf{binary}}(\mathfrak{D}, \mathfrak{m})$ is in* P. *It is in* ExpTime *for all exponential derivers $\mathfrak{D}$.*

4.2   Proof Depth

We now consider the measure $\mathfrak{m}_{\mathsf{depth}}$ in more detail. We can show lower bounds of P and ExpTime for polynomial and exponential derivers, respectively, although the latter only holds for upper bounds q encoded in binary.

Since our definition of $\mathsf{OP}(\mathfrak{D}, \mathfrak{m})$ requires that the input entailment $\mathcal{T} \models \eta$ already holds, we cannot use a straightforward reduction from the entailment problem in $\mathcal{EL}$ or $\mathcal{ELI}$, however. Instead, we show that ordinary proofs $\mathcal{P}$ for $\mathcal{T} \models \eta$ satisfy $\mathfrak{m}(\mathcal{P}) \leq q$ for some q, and then extend the TBox to $\mathcal{T}'$ in order to create an artificial proof $\mathcal{P}'$ with $\mathfrak{m}(\mathcal{P}') > q$. In this way, we ensure that $\mathcal{T}' \models \eta$ holds and can use q to distinguish the artificial from the original proofs.

For $\mathcal{ELI}$, we can use an observation from [9, Example 6.29] for this purpose.

Proposition 13 ([9]). *For every $q \in \mathbb{Q}_{\geq 0}$ and $\mathcal{ELI}$ sentence of the form $A \sqsubseteq B$, where $A, B \in \mathsf{N_C}$, one can construct in time polynomial in q an $\mathcal{ELI}$ theory $\mathcal{T}$ such $\mathcal{T} \models A \sqsubseteq B$, and every proof for $\mathcal{T} \models A \sqsubseteq B$ in* Eli *is of depth larger than 2^q.*

We can now reduce the entailment problems for $\mathcal{EL}$ and $\mathcal{ELI}$ to obtain the claimed lower bounds.

Theorem 14. *The problems $\mathsf{OP}_{\mathsf{unary}}(\text{Elk}, \mathfrak{m}_{\mathsf{depth}})$ and $\mathsf{OP}_{\mathsf{binary}}(\text{Eli}, \mathfrak{m}_{\mathsf{depth}})$ are* P-*hard and* ExpTime-*hard, respectively.*

Proof. For the P-hardness, we provide a LogSpace-reduction from the entailment problem of a GCI $A \sqsubseteq B$ with two concept names A, B from an $\mathcal{EL}$-theory $\mathcal{T}$, which is P-hard [9]. To reduce this problem to $\mathsf{OP}_{\mathsf{unary}}(\text{Elk}, \mathfrak{m}_{\mathsf{tree}})$, we need to find a theory $\mathcal{T}'$ and a number q such that $\mathcal{T}' \models A \sqsubseteq B$ holds, and moreover $\mathcal{T} \models A \sqsubseteq B$ holds iff $\text{Elk}(\mathcal{T}', A \sqsubseteq B)$ contains a proof of $\mathcal{T}' \models A \sqsubseteq B$ of depth $\leq q$ (cf. Lemma 7).

First, observe that, since proofs must be acyclic, the depth of any proof of $A \sqsubseteq B$ from $\mathcal{T}$ is bounded by $q := |\text{Elk}(\mathcal{T}, A \sqsubseteq B)|$, whose size in unary encoding is polynomial in the size of $\mathcal{T}$. We now construct

$$\mathcal{T}' := \mathcal{T} \cup \{A \sqsubseteq A_1, \ A_1 \sqsubseteq A_2, \ldots, A_{q+2} \sqsubseteq B\},$$

where $A_1, \ldots, A_q$ are concept names not occurring in $\mathcal{T}$. Clearly, we have $\mathcal{T}' \models A \sqsubseteq B$. Furthermore, the existence of an admissible proof for $\mathcal{T}' \models A \sqsubseteq B$ of depth at most q is equivalent to $\mathcal{T} \models A \sqsubseteq B$, since any proof that uses the new concept names must take $q + 1$ consecutive steps using rule $\mathsf{R}_{\sqsubseteq}$, i.e. must be of depth $q + 1$. Moreover, we can compute q (in binary representation) and output it in unary representation using a logarithmically space-bounded Turing machine, and similarly for $\mathcal{T}'$. Hence, the above construction constitutes the desired LOGSPACE-reduction.

For the remaining result, we can use similar arguments about the exponential deriver ELI, where entailment is EXPTIME-hard [9]:

- the minimal depth of a proof in an exponential derivation structure is at most exponential, and this exponential bound q can be computed in polynomial time using binary encoding;
- by Proposition 13, there is an $\mathcal{ELI}$ theory $\mathcal{T}$ of size polynomial in the size of the binary encoding of q such that $\mathcal{T} \models A \sqsubseteq B$ and any proof for $\mathcal{T} \models A \sqsubseteq B$ must have at least depth $q + 1$. □

To demonstrate that the generic upper bounds from Theorem 12 are tight even for unary encoding, we quickly consider the artificial measure $\mathfrak{m}_{\mathsf{log(depth)}}$ (*logarithmic depth*), which simply computes the (binary) logarithm of the depth of a given proof. This is also a monotone recursive Φ-measure, since the logarithmic depth contains exactly the same information as the depth itself. It is easy to obtain the following lower bounds from the previous results about $\mathfrak{m}_{\mathsf{depth}}$.

Corollary 15. $\mathsf{OP}_{\mathsf{unary}}(\textsc{Elk}, \mathfrak{m}_{\mathsf{log(depth)}})$ *is* P-*hard and* $\mathsf{OP}_{\mathsf{unary}}(\textsc{Eli}, \mathfrak{m}_{\mathsf{log(depth)}})$ *is* EXPTIME-*hard.*

Proof. For any deriver $\mathfrak{D}$, $\mathsf{OP}_{\mathsf{binary}}(\mathfrak{D}, \mathfrak{m}_{\mathsf{depth}})$ can be LOGSPACE-reduced to $\mathsf{OP}_{\mathsf{unary}}(\mathfrak{D}, \mathfrak{m}_{\mathsf{log(depth)}})$, because in order to find a proof of depth at most q (with q given in binary), one can equivalently look for a proof whose logarithmic depth is bounded by the value $\log q$. The unary encoding of $\log q$ has the same size as the binary encoding of q and can be computed in LOGSPACE by flipping all bits of the binary encoding of q to 1. □

We now return to $\mathfrak{m}_{\mathsf{depth}}$ and cover the remaining case of exponential derivers and unary encoding of the upper bound q.

Theorem 16. $\mathsf{OP}_{\mathsf{unary}}(\mathfrak{D}, \mathfrak{m}_{\mathsf{depth}})$ *is in* PSPACE *for any exponential deriver* $\mathfrak{D}$. *It is* PSPACE-*hard for the exponential deriver* $\mathfrak{D} = \textsc{Eli}$.

Proof. For the upper bound, we employ a depth-first guessing strategy: we guess a proof of depth at most q, where at each time point we only keep one branch of the proof in memory. As the length of this branch is bounded by q, and due to our assumptions on derivers, this procedure only requires polynomial space.

For the lower bound, we provide a reduction from the PSPACE-complete QBF problem (satisfiability of quantified Boolean formulas). Let $\mathsf{Q}_1 x_1 \mathsf{Q}_2 x_2 \ldots \mathsf{Q}_m x_m . \phi$ be a quantified Boolean formula, where for $i \in \{1, \ldots, m\}$, $\mathsf{Q}_i \in \{\exists, \forall\}$, and ϕ is

a formula over $\{x_1, \ldots, x_m\}$. We assume ϕ to be in negation normal form, that is, negation only occurs directly in front of a variable. We construct an $\mathcal{ELI}$ theory $\mathcal{T}$ and a number q, both of size polynomial in the size of the formula, such that $\mathcal{T} \models A \sqsubseteq B$ holds (cf. Definition 6) and $\mathcal{T}$ has a proof for $A \sqsubseteq B$ of depth q iff the QBF formula is valid. We use two roles r_1, r_2 to deal with the variable valuations, concept names $A_0, \ldots, A_m$ to count the quantifier nesting, and a concept name A_ψ for every sub-formula ψ of ϕ. In addition, we use the concept names A and B occurring in the conclusion, and two concept names B_1 and B_2.

The concept name A initializes the formula at quantifier nesting level 0:

$$A \sqsubseteq A_0$$

For every $i \in \{1, \ldots, m\}$, $\mathcal{T}$ contains the following sentence to select a truth valuation for x_i, increasing the nesting depth in each step.

$$A_{i-1} \sqsubseteq \exists r_1.(A_i \sqcap A_{x_i}) \tag{1}$$
$$A_{i-1} \sqsubseteq \exists r_2.(A_i \sqcap A_{\neg x_i}). \tag{2}$$

To ensure truth valuations are kept along the role-successors, we use the following sentences for every $l \in \{x_i, \neg x_i \mid 1 \le i \le m\}$:

$$A_l \sqsubseteq \forall r_1.A_l \qquad A_l \sqsubseteq \forall r_2.A_l \tag{3}$$

The following GCIs are now used to evaluate ϕ. For every conjunction $\psi = \psi_1 \wedge \psi_2$ occurring in ϕ, we use:

$$A_{\psi_1} \sqcap A_{\psi_2} \sqsubseteq A_\psi, \tag{4}$$

and for every disjunction $\psi = \psi_1 \vee \psi_2$, we use:

$$A_{\psi_1} \sqsubseteq A_\psi \qquad A_{\psi_2} \sqsubseteq A_\psi \tag{5}$$

Finally, the following GCIs are used to propagate the result of the evaluation back towards the start.

$$A_\phi \sqsubseteq B \tag{6}$$

$$A_i \sqcap B \sqsubseteq \forall r_1^-.B \qquad A_i \sqcap B \sqsubseteq \forall r_2^-.B \qquad\qquad \text{if } \mathsf{Q}_i = \exists \tag{7}$$

$$A_i \sqcap B \sqsubseteq \forall r_1^-.B_1 \qquad A_i \sqcap B \sqsubseteq \forall r_2^-.B_2 \qquad B_1 \sqcap B_2 \sqsubseteq B \qquad \text{if } \mathsf{Q}_i = \forall \tag{8}$$

One can now show that there exists a proof for $A \sqsubseteq B$ from $\mathcal{T}$ of depth at most q iff the QBF formula is valid, where q is polynomial and determined by the size and structure of ϕ. Finally, we can extend $\mathcal{T}$ with the sentences from Proposition 13 to ensure that $\mathcal{T} \models A \sqsubseteq B$ holds while retaining this equivalence. $\qquad\square$

4.3   The Tree Size Measure

The tree size measure was discussed already in [2], where tight bounds were provided for polynomial derivers and exponential derivers with unary encoding. For the case of exponential derivers with binary encoding, only an EXPTIME upper bound was provided, and the precise complexity left open. We improve this result by showing that $\mathsf{OP}_{\mathsf{binary}}(\mathfrak{D}, \mathfrak{m}_{\mathsf{tree}})$ can indeed be decided in PSPACE.
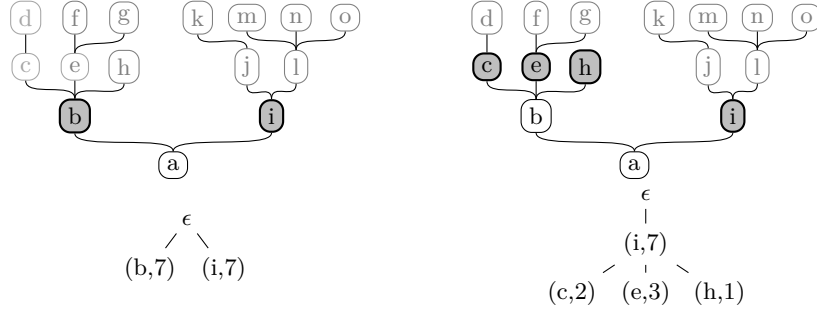
Fig. 5. Illustration of the argument used for Theorem 17. On the top, the partially guessed proof tree for two consecutive steps of the algorithm is shown, where the dark nodes are what is currently kept in memory. On the bottom, we see how the corresponding tuples are organized into a tree satisfying Conditions **S1**–**S6**.

Theorem 17. *For any exponential deriver $\mathfrak{D}$, $\mathsf{OP}_{\mathsf{binary}}(\mathfrak{D}, \mathfrak{m}_{\mathsf{tree}})$ is in* PSPACE.

Proof (sketch). We describe a non-deterministic procedure for $\mathsf{OP}_{\mathsf{binary}}(\mathfrak{D}, \mathfrak{m}_{\mathsf{tree}})$, in polynomial space. Let $\mathcal{T}$ be a theory, η the goal sentence, and q a rational number in binary encoding. By Lemma 7, it suffices to find a proof $\mathcal{P}$ for $\mathcal{T} \models \eta$ in $\mathfrak{D}(\mathcal{T}, \eta)$ with $\mathfrak{m}_{\mathsf{tree}}(\mathcal{P}) \leq q$. The procedure guesses such a proof starting from the conclusion, while keeping in memory a set S of tuples (η', q'), where η' is a sentence and $q' \leq q$ a rational number. Intuitively, such a tuple states: "We still need to guess a proof for η' of tree size at most q'."

1. Initialize $S := \{(\eta, q)\}$.
2. While $S \neq \emptyset$,
   (a) select from S a tuple (η', q') such that for all tuples $(\eta'', q'') \in S$ it holds that $q'' \geq q'$;
   (b) guess a hyperedge $(\{v_1, \ldots, v_m\}, v')$ in $\mathfrak{D}(\mathcal{T}, \eta)$ (using the oracle access described in Section 2.2) and m numbers $q_1, \ldots, q_m$, such that $\ell(v') = \eta'$ and $q_1 + \ldots + q_m + 1 \leq q'$; and
   (c) replace (η', q') in S by the tuples $(\ell(v_1), q_1), \ldots, (\ell(v_m), q_m)$.

There is a proof for $\mathcal{T} \models \eta$ of tree size at most q iff every step in the algorithm is successful. To show that it only requires polynomial space, we show that during the computation, the number of elements in S is always polynomially bounded. For this, we show that the elements in S can always be organized into a tree with the following properties:

S1 the root is labeled with ϵ,
S2 every other node is labeled with a distinct element from S,
S3 every node that is not the root or a leaf has at least 2 children,
S4 every node has at most p children, where p is the maximal number of premises in any inference in $\mathfrak{D}(\mathcal{T}, \eta)$, which we assumed to be polynomial in the input,
S5 every node (η', q') has at most 1 child (η'', q'') that is not a leaf and for this child it holds that $q'' < \frac{q'}{2}$,

S6 for every node labeled (η', q') with children labeled (η_1, q_1), ..., (η_m, q_m), we have $q_1 + \ldots + q_m < q'$.

We prove this by induction on the steps of the algorithm, where in each step, we either replace one tuple in the tree, or put the new tuples under the leaf with the currently smallest value (see Fig.5). By **S3** and because every number in S is bounded by q, we can show that the tree has depth at most $\log_2 q$, which with **S4** and **S5** implies that it has at most $p \cdot \log_2 q$ nodes. **S2** then implies that that $|S| \leq p \cdot \log_2 q$ is always satisfied, and thus that S is polynomially bounded. □

A corresponding lower bound can be found for the exponential deriver ELI by a reduction of the word problem for deterministic Turing machines with polynomial space bound.

Theorem 18. *For the exponential deriver* ELI, $\mathsf{OP}_{\mathsf{binary}}(\text{ELI}, \mathfrak{m}_{\mathsf{tree}})$ *is* PSPACE-*hard.*

Proof (sketch). Let $T = (Q, \Gamma, \not{b}, \Sigma, \delta, q_0, F)$ be a deterministic Turing machine, where Q is the set of states, Γ the tape alphabet, $\not{b} \in \Gamma$ the blank symbol, $\Sigma \subseteq \Gamma$ the input alphabet, $\delta : Q \times \Gamma \nrightarrow Q \times \Gamma \times \{-1, 0, +1\}$ the partial transition function, q_0 the initial state, and $F \subseteq Q$ the accepting states. We assume that T is polynomially space bounded, that is, there is a polynomial p such that on input words $w \in \Sigma^*$, T only accesses the first $p(|w|)$ cells of the tape. For a word w, we denote by $w[i]$ its ith letter. For some fixed word w, we construct a theory $\mathcal{T}$ using the following names, where $k = p(|w|)$:

- Start marks the inital and Accept an accepting configuration;
- to denote that we are in state $q \in Q$, we use a concept name S_q;
- for every $a \in \Gamma$ and $i \in \{0, \ldots, k\}$, we use a concept name A_i^a denoting that the letter a is on tape position i;
- for every $i \in \{0, \ldots, k\}$, we use the concept name P_i^+ to denote that the head is currently on position i, and P_i^- to denote that it is not;
- the role r is used to express the transitions between the configurations.

For convenience, we present the theory not in the required normal form, but aggregate conjunctions on the right. The following sentence describes the initial configuration.

$$\mathsf{Start} \sqsubseteq S_{q_0} \sqcap \prod_{i=0}^{|w|-1} A_i^{w[i]} \sqcap \prod_{i=|w|}^{k} A_i^{\not{b}} \sqcap P_0^+ \sqcap \prod_{i=1}^{k} P_i^- \tag{9}$$

The transition from one configuration to the next is encoded with the following sentences for every $i \in \{0, \ldots, k\}$ and every $(q, a) \in Q \times \Gamma$ with $\delta(q, a) = (q', b, d)$:

$$S_q \sqcap A_i^a \sqcap P_i^+ \sqsubseteq \exists r.S_{q'} \sqcap \forall r.A_i^b \sqcap \forall r.P_{i+d}^+ \sqcap \prod_{j \in \{0, \ldots, k\} \setminus \{i+d\}} \forall r.P_j^- \tag{10}$$

$$A_i^a \sqcap P_i^- \sqsubseteq \forall r.A_i^a \tag{11}$$

Finally, we use the following sentences to detect accepting configurations and propagate the information of acceptance back to the initial configuration

$$S_f \sqsubseteq \mathsf{Accept} \text{ for all } f \in F, \tag{12}$$

$$\mathsf{Accept} \sqsubseteq \forall r^-.\mathsf{Accept} \tag{13}$$

One can find a number q exponential in k and the size of T s.t. that there is a proof for $\mathcal{T} \models \mathsf{Start} \sqsubseteq \mathsf{Accept}$ with tree size at most q iff T accepts w. Using Proposition 13, we can extend $\mathcal{T}$ to a theory $\mathcal{T}'$ s.t. $\mathcal{T}' \models \mathsf{Start} \sqsubseteq \mathsf{Accept}$, while a proof of tree size q exists iff T accepts w (observe that $\mathfrak{m}_{\mathsf{tree}}(\mathcal{P}) \geq \mathfrak{m}_{\mathsf{depth}}(\mathcal{P})$ holds for all proofs $\mathcal{P}$).   □

5   Conclusion

We have investigated the complexity of finding optimal proofs w.r.t. quality measures that satisfy the property of being *monotone recursive*. Two important examples of this class of measures, *depth* and *tree size*, have been considered in detail in combination with exponential and polynomial derivers. The obtained results are promising: given a deriver, the search for an optimal proof for an entailment can be easier than producing all of the proofs by this deriver. The algorithms used to show the upper bounds can serve as building blocks for finding an optimal proof w.r.t. to a monotone recursive measure automatically.

   We conjecture that weighted versions of *tree size* and *depth*, where sentences or inference steps can have associated rational weights, are also monotone recursive, and the generic upper bounds established in this paper can be straightforwardly applied to them. However, a more thorough study is required here, since the complexity of the decision problem depends on the exact way in which the weights are employed. This step towards weighted measures is motivated by user studies [1, 15, 24], demonstrating that different types of sentences and logical inferences can be more or less difficult to understand.

References

1. Alharbi, E., Howse, J., Stapleton, G., Hamie, A., Touloumis, A.: The efficacy of OWL and DL on user understanding of axioms and their entailments. In: d'Amato, C., Fernández, M., Tamma, V.A.M., Lécué, F., Cudré-Mauroux, P., Sequeda, J.F., Lange, C., Heflin, J. (eds.) ISWC 2017 - 16th International Semantic Web Conference, Proceedings. Lecture Notes in Computer Science, vol. 10587, pp. 20–36. Springer (2017). `https://doi.org/10.1007/978-3-319-68288-4_2`
2. Alrabbaa, C., Baader, F., Borgwardt, S., Koopmann, P., Kovtunova, A.: Finding small proofs for description logic entailments: Theory and practice. In: Albert, E.,

Kovacs, L. (eds.) LPAR-23: 23rd International Conference on Logic for Programming, Artificial Intelligence and Reasoning. EPiC Series in Computing, vol. 73, pp. 32–67. EasyChair (2020). `https://doi.org/10.29007/nhpp`

3. Alrabbaa, C., Baader, F., Borgwardt, S., Koopmann, P., Kovtunova, A.: On the complexity of finding good proofs for description logic entailments. In: Borgwardt, S., Meyer, T. (eds.) Proceedings of the 33rd International Workshop on Description Logics (DL 2020). CEUR Workshop Proceedings, vol. 2663. CEUR-WS.org (2020), `http://ceur-ws.org/Vol-2663/paper-1.pdf`

4. Alrabbaa, C., Baader, F., Borgwardt, S., Koopmann, P., Kovtunova, A.: Finding good proofs for description logic entailments using recursive quality measures (extended technical report) (2021), `https://arxiv.org/abs/2104.13138`, arXiv:2104.13138 [cs.AI]

5. Alrabbaa, C., Baader, F., Dachselt, R., Flemisch, T., Koopmann, P.: Visualising proofs and the modular structure of ontologies to support ontology repair. In: DL 2020: International Workshop on Description Logics. CEUR Workshop Proceedings, vol. 2663. CEUR-WS.org (2020), `http://ceur-ws.org/Vol-2663/paper-2.pdf`

6. Arrieta, A.B., Diaz-Rodriguez, N., Ser, J.D., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., Herrera, F.: Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. Information Fusion **58**, 82–115 (2020). `https://doi.org/10.1016/j.inffus.2019.12.012`

7. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Kaelbling, L.P., Saffiotti, A. (eds.) Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI'05). pp. 364–369. Professional Book Center (2005), `http://ijcai.org/Proceedings/09/Papers/053.pdf`

8. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope further. In: Clark, K., Patel-Schneider, P.F. (eds.) Proc. of the 4th Workshop on OWL: Experiences and Directions. pp. 1–10 (2008), `http://webont.org/owled/2008dc/papers/owled2008dc_paper_3.pdf`

9. Baader, F., Horrocks, I., Lutz, C., Sattler, U.: An Introduction to Description Logic. Cambridge University Press (2017). `https://doi.org/10.1017/9781139025355`

10. Baader, F., Suntisrivaraporn, B.: Debugging SNOMED CT using axiom pinpointing in the description logic $\mathcal{EL}^+$. In: Proc. of the 3rd Conference on Knowledge Representation in Medicine (KR-MED'08): Representing and Sharing Knowledge Using SNOMED. CEUR-WS, vol. 410 (2008), `http://ceur-ws.org/Vol-410/Paper01.pdf`

11. Borgida, A., Franconi, E., Horrocks, I.: Explaining $\mathcal{ALC}$ subsumption. In: ECAI 2000, Proceedings of the 14th European Conference on Artificial Intelligence, Berlin, Germany, August 20-25, 2000. pp. 209–213 (2000), `http://www.frontiersinai.com/ecai/ecai2000/pdf/p0209.pdf`

12. Fiedler, A.: Natural language proof explanation. In: Mechanizing Mathematical Reasoning, Essays in Honor of Jörg H. Siekmann on the Occasion of His 60th Birthday. pp. 342–363 (2005). `https://doi.org/10.1007/978-3-540-32254-2_20`

13. Gallo, G., Longo, G., Pallottino, S.: Directed hypergraphs and applications. Discrete Applied Mathematics **42**(2), 177–201 (1993). `https://doi.org/10.1016/0166-218X(93)90045-P`

14. Horridge, M.: Justification Based Explanation in Ontologies. Ph.D. thesis, University of Manchester, UK (2011), `https://www.research.manchester.ac.uk/portal/files/54511395/FULL_TEXT.PDF`

15. Horridge, M., Bail, S., Parsia, B., Sattler, U.: Toward cognitive support for OWL justifications. Knowl. Based Syst. **53**, 66–79 (2013). `https://doi.org/10.1016/j.knosys.2013.08.021`

16. Horridge, M., Parsia, B., Sattler, U.: Justification oriented proofs in OWL. In: The Semantic Web - ISWC 2010 - 9th International Semantic Web Conference, ISWC 2010, Shanghai, China, November 7-11, 2010, Revised Selected Papers, Part I. pp. 354–369 (2010). `https://doi.org/10.1007/978-3-642-17746-0_23`

17. Huang, X.: Reconstruction proofs at the assertion level. In: Proceedings of the 12th International Conference on Automated Deduction. p. 738–752. CADE-12, Springer-Verlag (1994). `https://doi.org/10.1007/3-540-58156-1_53`

18. Kazakov, Y.: Consequence-driven reasoning for horn SHIQ ontologies. In: Boutilier, C. (ed.) IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009. pp. 2040–2045 (2009), `http://ijcai.org/Proceedings/09/Papers/336.pdf`

19. Kazakov, Y., Klinov, P.: Goal-directed tracing of inferences in $\mathcal{EL}$ ontologies. In: Mika, P., Tudorache, T., Bernstein, A., Welty, C., Knoblock, C.A., Vrandecic, D., Groth, P.T., Noy, N.F., Janowicz, K., Goble, C.A. (eds.) Proc. of the 13th International Semantic Web Conference (ISWC 2014). Lecture Notes in Computer Science, vol. 8797, pp. 196–211. Springer (2014). `https://doi.org/10.1007/978-3-319-11915-1_13`

20. Kazakov, Y., Klinov, P., Stupnikov, A.: Towards reusable explanation services in Protege. In: Artale, A., Glimm, B., Kontchakov, R. (eds.) Proc. of the 30th Int. Workshop on Description Logics (DL'17). CEUR Workshop Proceedings, vol. 1879 (2017), `http://www.ceur-ws.org/Vol-1879/paper31.pdf`

21. Kazakov, Y., Krötzsch, M., Simancik, F.: The incredible ELK – from polynomial procedures to efficient reasoning with $\mathcal{EL}$ ontologies. J. Autom. Reasoning **53**(1), 1–61 (2014). `https://doi.org/10.1007/s10817-013-9296-3`

22. Lingenfelder, C.: Structuring computer generated proofs. In: Proceedings of the 11th International Joint Conference on Artificial Intelligence. Detroit, MI, USA, August 1989. pp. 378–383 (1989), `http://ijcai.org/Proceedings/89-1/Papers/060.pdf`

23. McGuinness, D.L.: Explaining Reasoning in Description Logics. Ph.D. thesis, Rutgers University, NJ, USA (1996). `https://doi.org/10.7282/t3-q0c6-5305`

24. Nguyen, T.A.T., Power, R., Piwek, P., Williams, S.: Measuring the understandability of deduction rules for OWL. In: Proceedings of the First International Workshop on Debugging Ontologies and Ontology Mappings, WoDOOM 2012, Galway, Ireland, October 8, 2012. pp. 1–12 (2012), `http://www.ida.liu.se/~patla/conferences/WoDOOM12/papers/paper4.pdf`

25. Nguyen, T.A.T., Power, R., Piwek, P., Williams, S.: Predicting the understandability of OWL inferences. In: The Semantic Web: Semantics and Big Data, 10th International Conference, ESWC 2013, Montpellier, France, May 26-30, 2013. Proceedings. pp. 109–123 (2013). `https://doi.org/10.1007/978-3-642-38288-8_8`

26. Schiller, M.R.G., Glimm, B.: Towards explicative inference for OWL. In: Informal Proceedings of the 26th International Workshop on Description Logics, Ulm, Germany, July 23 - 26, 2013. pp. 930–941 (2013), `http://ceur-ws.org/Vol-1014/paper_36.pdf`

27. Schiller, M.R.G., Schiller, F., Glimm, B.: Testing the adequacy of automated explanations of EL subsumptions. In: Proceedings of the 30th International Workshop on Description Logics, Montpellier, France, July 18-21, 2017. (2017), `http://ceur-ws.org/Vol-1879/paper43.pdf`

28. Schlobach, S., Cornet, R.: Non-standard reasoning services for the debugging of description logic terminologies. In: Gottlob, G., Walsh, T. (eds.) Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI 2003). pp. 355–362. Morgan Kaufmann, Acapulco, Mexico (2003), `http://ijcai.org/Proceedings/03/Papers/053.pdf`