On the Complexity of Finding Good Proofs for Description Logic Entailments

Christian Alrabbaa, Franz Baader, Stefan Borgwardt,
Patrick Koopmann, and Alisa Kovtunova

Institute of Theoretical Computer Science, TU Dresden, Germany

Abstract. If a Description Logic (DL) system derives a consequence, then one can in principle explain such an entailment by presenting a proof of the consequence in an appropriate calculus. Intuitively, good proofs are ones that are simple enough to be comprehensible to a user of the system. In recent work, we have introduced a general framework in which proofs are represented as labeled, directed hypergraphs, and have investigated the complexity of finding *small* proofs. However, large size is not the only reason that can make a proof complex. In the present paper, we introduce other measures for the complexity of a proof, and analyze the computational complexity of deciding whether a given consequence has a proof of complexity at most q. This problem can be NP-complete even for $\mathcal{EL}$, but we also identify measures where it can be decided in polynomial time.

1 Introduction

The first work on explaining DL entailments is probably the PhD thesis of McGuinness [18], where the results obtained by the structural subsumption algorithm of the CLASSIC system [8] are translated into proofs in a formal calculus. The thesis also investigates how to create shorter, better understandable proofs by pruning away unimportant parts. In [6], proofs of subsumptions generated by a tableau-based system are translated into sequent proofs. The authors then investigate how to make the sequent proofs shorter. More recent work on explaining DL entailment was often focused on computing so-called justifications, i.e., minimal subsets of the knowledge base (KB) from which the consequence in question follows (see, e.g., [4, 12, 24]). The basic assumption is here that, whereas KBs may be very large and have many consequences, a single consequence often follows from a small subset of the KB by an easy derivation. While this is true in certain applications [5], in general it may be quite hard for a user to see without help why a consequence follows from a given justification [14]. On the one hand, this has triggered research into assessing the complexity of a justification, i.e., how hard it is to derive the given consequence from the justification [13, 19]. On the other hand, it has rekindled the interest in generating proofs appropriate for human consumption. For example, the explanation plugin for the ontology editor Protégé described in [15] cannot only produce justfications, but can also display proofs, provided that proofs of an appropriate form are returned by the employed reasoner, an assumption that is, e.g., satisfied by the reasoner ELK [16]. While these proofs are represented in the formal syntax employed by Protégé, the work reported in [19, 22, 23] uses ontology verbalization techniques to translate proofs into natural language text.

Recently, we have investigated the complexity of deciding whether proofs of a certain size exist, which is measured in the number of steps involved in the proof [1]. Instead of concentrating on a specific DL and proof calculus or reasoner for this DL, we used a general framework in which proofs are represented as labeled, directed, acyclic hypergraphs whose vertices are labeled with axioms and hyperedges correspond to sound derivation steps. Note that such proofs need not be tree-shaped since a single axiom can be reused in multiple inference steps, thereby reducing the overall size of the proof. To abstract away from implementation details, we assume that a reasoner (called *deriver* to distinguish it from an actual implemented system) generates a so-called derivation structure, which consists of possible proof steps, from which actual proofs can be constructed. For example, if we consider the consequence-based reasoning approaches for the DLs $\mathcal{EL}$ and $\mathcal{ELI}$ described in [3], then we could obtain a derivation structure for a given KB and consequence by collecting all (finitely many) instantiated classification rules. The $\mathcal{EL}$ reasoner ELK actually returns such a derivation structure, which however only contains the rule instances that have actually been used during the reasoning process.

In this paper, we extend the results of [1] by considering more measures of *proof complexity* (other than size), with the ultimate goal of finding proofs that are easy to understand for users. In a recent user study [7], for example, it was observed that proof understandability can be influenced by the structure (e.g. linear vs. several independent reasoning threads), the complexity of the individual reasoning steps (e.g. whether a combination of quantifiers is involved), the representation of the domain (e.g. abstract concept and role names vs. long domain-specific names), and the overall format (e.g. graphical vs. textual).

In this paper, we consider a few possible approaches to defining proof complexity, where we focus on the overall structure of the proof. For this reason, we are not interested in the complexity/weight of individual axioms or inference steps, which has already been investigated in the literature [13, 20]. For most of the paper, we use the *size* of an axiom (or group of axioms) as a proxy for their complexity, but it is easy to substitute other measures from the literature.

We now list a few approaches for defining proof complexity, which we discuss throughout this paper.

1. A proof is at least as hard to understand as its hardest axiom.
2. The complexity of a proof is the sum (product, average, . . .) of all weights of its axioms.
3. As a special case of the previous item, the number of axioms constituting the proof defines its complexity.

4. The hardness of a proof corresponds to its *(weighted) tree size*, i.e., the number (weighted sum) of vertices in its tree unraveling [1].
5. A proof is as hard to understand as the hardest inference step.
6. The hardness of a proof is the sum (product, average, ...) of inference step weights.
7. Proof complexity is determined by the number of TBox axioms that are used, i.e., the size of a corresponding justification.
8. A proof is more complex if there are more inference steps between the TBox axioms and the conclusion. We could compute the *maximal distance* (*depth*), or the maximal sum of weights along such a path.
9. A linear proof is easier to follow, hence we could measure the *pathwidth* of a proof (when viewing a proof as an undirected graph obtained by replacing each directed hyperedge with several undirected edges between the conclusion and each premise).
10. We could consider a similary adapted notion of (undirected) *treewidth* as a measure for the complexity of a proof, which reflects its degree of connectivity.
11. Multiple adjacent reasoning steps using the same rule schema are easier to understand than when these steps are interspersed with other types of inference rules.
12. For the presentation to a user, it may also be relevant whether a proof is planar, i.e., it can be drawn without intersecting (hyper)edges.

There are of course many more ways in which "proof complexity" could be measured. This paper takes a few steps towards classifying the complexity of finding "best" proofs w.r.t. any such measure. Before we can introduce this problem formally, we recall the basic definitions from [1].

2 Preliminaries

We assume a basic familiarity with description logics [3]. Most of our theoretical discussions apply to an arbitrary (description) logic; from now on, we fix one such logic $\mathcal{L}$, where we focus on the terminological part, i.e., the TBox. Following the notation from [1], $S_{\mathcal{L}}$ denotes all TBox axioms that can be formulated in $\mathcal{L}$. We assume that the *size* $|\alpha|$ of such an axiom α is defined in some way, e.g. by the number of symbols in α. Given a TBox $\mathcal{T}$ and axiom α, a *justification* for α in $\mathcal{T}$ is a minimal subset $\mathcal{J} \subseteq \mathcal{T}$ such that $\mathcal{J} \models \alpha$. Already for $\mathcal{EL}$-ontologies, for which finding a single justification is possible in polynomial time, there may be still exponentially many justifications, and finding a justification of size $\leq n$ is NP-complete [4].

While justifications can pinpoint the reasons for an entailment in a TBox, we are interested in more detailed proofs such as, for example, the one in Figure 1.

2.1 Hypergraphs

As in [1], we view proofs as directed hypergraphs.

Definition 1 (Hypergraph). *A* (finite, directed, labeled) hypergraph *[21] is a triple $H = (V, E, \ell)$, where*

- *V is a finite set of* vertices*,*
- *E is a set of* hyperedges (S, d) *with* source vertices $S \subseteq V$ *and* target vertex $d \in V$*, and*
- *$\ell\colon V \to S_{\mathcal{L}}$ is a* labeling function *that assigns axioms to vertices.*

The *size* of H, denoted $|H|$, is measured by the size of the labels of its hyperedges:

$$|H| := \sum_{(S,d)\in E} |(S, d)|, \text{ where } |(S, d)| := |\ell(d)| + \sum_{v\in S} |\ell(v)|.$$

A vertex $v \in V$ is called a *leaf* if it has no incoming hyperedges, i.e., there is no $(S, v) \in E$, and v is a *sink* if it has no outgoing hyperedges, i.e., there is no $(S, d) \in E$ such that $v \in S$. We denote the set of all leafs and the set of all sinks in H as *leaf*(H) and *sink*(H), respectively.

A hypergraph $H' = (V', E', \ell')$ is called a *subgraph* of $H = (V, E, \ell)$ if $V' \subseteq V$, $E' \subseteq E$ and $\ell' = \ell|_{V'}$. In this case, we also say that H *contains* H' and write $H' \subseteq H$. Given two hypergraphs $H_1 = (V_1, E_1, \ell_1)$ and $H_2 = (V_2, E_2, \ell_2)$ s.t. $\ell_1(v) = \ell_2(v)$ for every $v \in V_1 \cap V_2$, their *union* is defined as $H_1 \cup H_2 := (V_1 \cup V_2, E_1 \cup E_2, \ell_1 \cup \ell_2)$.

Definition 2 (Cycle, Tree). *Given a hypergraph $H = (V, E, \ell)$ and $s, t \in V$, a* path P *of length $q > 0$ in H from s to t is a sequence of vertices and hyperedges*

$$P = (d_0, (S_1, d_1), d_1, (S_2, d_2), \ldots, d_{q-1}, (S_q, d_q), d_q),$$

where $d_0 = s$, $d_q = t$, and $d_{j-1} \in S_j$ for all j, $1 \le j \le q$. If there is such a path in H, we say that t is reachable *from s in H. If $t = s$, then P is called a* cycle*. The hypergraph H is* acyclic *if it does not contain a cycle. The hypergraph H is* connected *if every vertex is connected to every other vertex by a series of paths and reverse paths.*

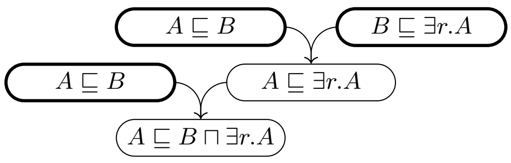
A tree $H = (V, E, \ell)$ *with* root $t \in V$ *is a hypergraph in which t is a sink and is reachable from every vertex $v \in V \setminus \{t\}$ by exactly one path.*

In particular, the root is the only sink in a tree, and all trees are acyclic and connected.

Definition 3 (Homomorphism). *Let $H = (V, E, \ell)$, $H' = (V', E', \ell')$ be two hypergraphs. A* homomorphism *from H to H', denoted $h\colon H \to H'$, is a mapping $h\colon V \to V'$ s.t. for all $(S, d) \in E$, one has $h(S, d) := (\{h(v) \mid v \in S\}, h(d)) \in E'$ and, for all $v \in V$, it holds that $\ell'(h(v)) = \ell(v)$. Such an h is an* isomorphism *if it is a bijection and, its inverse, $h^{-}: H' \to H$, is also a homomorphism.*

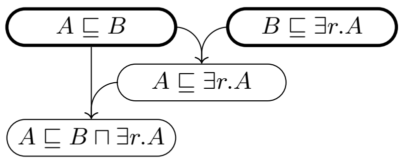
$$\dfrac{A \sqsubseteq B \qquad \dfrac{A \sqsubseteq B \qquad B \sqsubseteq \exists r.A}{A \sqsubseteq \exists r.A}}{A \sqsubseteq B \sqcap \exists r.A}$$

Fig. 1. A proof in the classical representation



$A \sqsubseteq B$
$B \sqsubseteq \exists r.A$
$A \sqsubseteq B$
$A \sqsubseteq \exists r.A$
$A \sqsubseteq B \sqcap \exists r.A$

Fig. 2. A tree-shaped proof



$A \sqsubseteq B$
$B \sqsubseteq \exists r.A$
$A \sqsubseteq \exists r.A$
$A \sqsubseteq B \sqcap \exists r.A$

Fig. 3. A proof with a reused vertex

2.2 Proofs

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

Definition 4 (Derivation Structure). *A* derivation structure $\mathcal{D} = (V, E, \ell)$ *over a TBox $\mathcal{T}$ 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 now define proofs as special derivation structures that derive a goal axiom.

Definition 5 (Proof). *Given an axiom η and a TBox $\mathcal{T}$, a* proof for $\mathcal{T} \models \eta$ *is a derivation structure $\mathcal{P} = (V, E, \ell)$ over $\mathcal{T}$ that*

- *contains exactly one sink $v_\eta \in V$, which is labeled by η, and*
- *is acyclic.*

A tree proof *is a proof that is a tree. A* subproof *of a hypergraph H is a subgraph of H that is a proof.*

Figures 2 and 3 both prove $A \sqsubseteq B \sqcap \exists r.A$ from $\mathcal{T} = \{A \sqsubseteq B,\ B \sqsubseteq \exists r.A\}$. The first proof is presented in hypergraph notation (cf. Figure 1) as a tree and the second one as a hypergraph without axiom label repetition, where the TBox axioms are marked with a thick border. Both of them can be seen as proofs in the sense of Definition 5 and they use the same inference steps, but have different numbers of vertices.

Two useful operations are the following of removal and replacement of subproofs above a vertex. 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 $\mathcal{P}_v = (V_v, E_v, \ell_v) \subseteq \mathcal{P}$, where V_v contains all vertices in V that have a path to v in $\mathcal{P}$. Dually, we denote by $\mathcal{P}^{-v} = (V^{-v}, E^{-v}, \ell^{-v})$ the largest subgraph of $\mathcal{P}$ such that

- $\mathit{leaf}(\mathcal{P}^{-v}) = \mathit{leaf}(\mathcal{P}) \cup \{v\}$ and
- $\mathit{sink}(\mathcal{P}^{-v}) = \mathit{sink}(\mathcal{P})$.

Intuitively, we obtain $\mathcal{P}^{-v}$ by removing every vertex from $\mathcal{P}$ from which every path to the sink goes through v. $\mathcal{P}^{-v}$ need not be a proof since v is now a leaf, but $\ell(v)$ may not be an axiom from $\mathcal{T}$. Finally, for a proof $\mathcal{P}' = (V', E', \ell')$ with sink v such that $V \cap V' = \{v\}$ and $\ell(v) = \ell'(v)$, we define the proof $\mathcal{P}[v \mapsto \mathcal{P}']$ as

$\mathcal{P}^{-v} \cup \mathcal{P}'$. The intuitive idea here is that v, as every vertex in V, is labeled with an axiom that has its own proof above it. And $\mathcal{P}^{-v}$ is obtained by removing the proof above v; vertices are kept only if they are used to prove other vertices in $\mathcal{P}$. Then, $\mathcal{P}[v \mapsto \mathcal{P}']$ is the result of "replacing" the proof above v by $\mathcal{P}'$. Moreover, since $\mathcal{P}$ and $\mathcal{P}'$ have only one vertex in common, it is guaranteed that the resulted hypergraph $\mathcal{P}[v \mapsto \mathcal{P}']$ is acyclic. Additionally, unravelling $\mathcal{P}$ into a tree can be seen as the result of recursively applying $\mathcal{P}[v \mapsto \mathcal{P}_v]$ to every vertex v in $\mathcal{P}$.

2.3 Derivers

As in [1], we rely on a *deriver* (imagine a DL reasoner) that, given a TBox $\mathcal{T}$ and a goal axiom η, produces a derivation structure $\mathcal{D}$ that contains "all possible proofs". This structure describes all instances of inference rules that are relevant for constructing proofs for η, without us having to know how inferences are performed. The quest for a good proof thus becomes a search for a good proof expressible by the inference steps in $\mathcal{D}$. Since reasoners may not be complete for proving arbitrary axioms of $\mathcal{L}$, we restrict η to a subset $C_\mathcal{L} \subseteq S_\mathcal{L}$ of supported consequences.

Definition 6 (Deriver). *A* deriver $\mathfrak{D}$ *is given by a set $C_\mathcal{L} \subseteq S_\mathcal{L}$ and a function that assigns derivation structures to pairs $(\mathcal{T}, \eta)$ of TBoxes $\mathcal{T} \subseteq S_\mathcal{L}$ and axioms $\eta \in C_\mathcal{L}$, such that $\mathcal{T} \models \eta$ iff $\mathfrak{D}(\mathcal{T}, \eta)$ contains a proof for $\mathcal{T} \models \eta$. A 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 $p(|\mathcal{T}| + |\eta|)$.*

We assume w.l.o.g. that $\mathfrak{D}(\mathcal{T}, \eta)$ does not contain two vertices with the same label; this does not affect our results.

Elk [16] can be seen as a polynomial deriver for $\mathcal{EL}$ in this sense, because it allows to instantiate the inference rules with only a polynomial number of $\mathcal{EL}$-concepts, namely the subconcepts appearing in $\mathcal{T}$ or η. The derivation structure Elk$(\mathcal{T}, \eta)$ contains all allowed instances of these rules, where each axiom is represented by a unique vertex. Since the number of premises in each rule is bounded by 2, the size of this structure is polynomial. Elk is complete only for goal axioms of the form $C \sqsubseteq D$, where D is a concept name and C is a subconcept from $\mathcal{T}$. To prove other kinds of entailments $\mathcal{T} \models \eta$, one first has to adapt $\mathcal{T}$ and η, thus leading to a different derivation structure Elk$(\mathcal{T}, \eta)$.

There are consequence-based derivers for other DLs, but they may not be polynomial [9]. In [1], we described a polynomial deriver for expressive description logics that is based on forgetting [17] and may take double-exponential time to compute the polynomial derivation structure $\mathfrak{D}(\mathcal{T}, \eta)$. In [1], we also investigated exponential derivers, but here we focus on the case of polynomial derivers.

3 Measuring Proof Complexity

Taking into consideration the variety of examples in the introduction, we define a general class of complexity measures for proofs. Our goal is to find proofs that minimize these measures, i.e., lower numbers are better.

Definition 7. *A* (complexity) 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* Ψ-measure *if, for every* $\mathcal{P} \in \mathrm{P}_{\mathcal{L}}$,

[P] $\mathfrak{m}(\mathcal{P})$ *is computable in* ***p****olynomial time in the size of* $\mathcal{P}$*,*
[SI] *every* ***s****ubproof of a homomorphic* ***i****mage of* $\mathcal{P}$ *weighs no more than* $\mathcal{P}$*, i.e.,* $\mathfrak{m}(\mathcal{P}'') \leq \mathfrak{m}(\mathcal{P})$ *for any homomorphism* $h\colon \mathcal{P} \to \mathcal{P}'$ *and* $\mathcal{P}'' \subseteq h(\mathcal{P})$ *such that* $\mathcal{P}'' \in \mathrm{P}_{\mathcal{L}}$*.*

Intuitively, a Ψ-measure $\mathfrak{m}$ does not increase when the proof gets smaller, either when parts of the proof are removed (i.e., for a subproof) or when parts are merged (i.e., for a homomorphic image). We cannot replace the property **[SI]** by two separate properties for each of these cases since the image of a proof $\mathcal{P}$ may not be acyclic, but has relevant acyclic subgraphs that should also not have larger weight than $\mathcal{P}$. Due to **[SI]**, there always exists a proof $\mathcal{P}$ for $\mathcal{T} \models \eta$ that is minimal w.r.t. $\mathfrak{m}$ as well as *non-redundant*, i.e., there is no subproof $\mathcal{P}' \subset \mathcal{P}$ for $\mathcal{T} \models \eta$. Moreover, two isomorphic proofs have the same weight w.r.t. $\mathfrak{m}$.

In Table 1, we provide examples of how the complexity measures from the introduction can be defined formally, where by $P \in \mathcal{P}$ we denote that P is a path in $\mathcal{P}$, and by $|P|$ the length of P. Since $\mathcal{P}$ is acyclic, there is always at least one such path. In this table, we use the size of an axiom or edge as a proxy for its complexity; other values from $\mathbb{Q}_{\geq 0}$ could be used instead. The table covers all measures from the introduction except for Items 4 and 9–12. *Tree size* (Item 4) is defined recursively; we discuss this measure in the context of a more general family of "local" measures later in Section 4.

Lemma 8. Tree size *and all measures in Table 1 except for* depth *and* worst path *are* Ψ*-measures. Indeed,* depth*,* worst path *and* treewidth *do not satisfy* **[SI]**.

The remaining items (9–12) are not Ψ-measures: the problems behind Items 9 and 10, i.e., to determine the pathwidth or treewidth for proofs which are graphs are NP-hard [2]. Moreover, note that treewidth is not interesting in our context, since we can always obtain a tree-shaped proof (with minimal treewidth 1) by unraveling. A similar argument applies to Item 12 (*planarity*) since every tree is planar. We will discuss Item 11 separately in Section 5.

We now formally define our main problem: finding optimal admissible proofs w.r.t. a measure $\mathfrak{m}$ and $\mathfrak{D}(\mathcal{T}, \eta)$.

Definition 9 (Optimal Proof). *Let* $\mathfrak{D}$ *be a deriver and* $\mathfrak{m}$ *be a measure. Given a TBox* $\mathcal{T}$ *and an axiom* $\eta \in C_{\mathcal{L}}$*, 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*

Table 1. Formal definitions of some measures $\mathfrak{m}$ for proofs $\mathcal{P} = (V, E, \ell)$.

Name	$\mathfrak{m}(\mathcal{P})$
Hardest axiom	$\max_{v \in V} \lvert\ell(v)\rvert$
Hardest inference step	$\max_{(S,d) \in E} \lvert(S, d)\rvert$
Sum of axiom complexities	$\sum_{v \in V} \lvert\ell(v)\rvert$
Sum of edge complexities	$\sum_{(S,d) \in E} \lvert(S, d)\rvert$
Vertex size	$\lvert V\rvert$
Justification size	$\lvert\{v \in V \mid \ell(v) \in \mathcal{T}\}\rvert$
Depth	$\max_{P \in \mathcal{P}} \lvert P\rvert$
Worst Path	$\max_{P \in \mathcal{P}} \sum_{(S,d) \in P} \lvert(S, d)\rvert$

decision problem, denoted $\mathsf{OP}(\mathfrak{D}, \mathfrak{m})$*, is to decide, given* $\mathcal{T}$ *and* η *as above and* $q \in \mathbb{Q}$*, whether there is an admissible proof* $\mathcal{P}$ *w.r.t.* $\mathfrak{D}(\mathcal{T}, \eta)$ *with* $\mathfrak{m}(\mathcal{P}) \leq q$.

If $\mathfrak{D}$ *is a polynomial deriver, we add the superscript* poly *to* OP*. If* $\mathfrak{m}$ *is a* Ψ*-measure, we add a subscript* Ψ*, e.g.* $\mathsf{OP}^{\mathsf{poly}}_{\Psi}(\mathfrak{D}, \mathfrak{m})$.

For the complexity analysis, we assume the deriver to be given in the form of a method that allows to access elements of $\mathfrak{D}(\mathcal{T}, \eta)$. The complexity of this method is not our concern here, we simply assume in the following that elements of $\mathfrak{D}(\mathcal{T}, \eta)$ can be accessed in constant time. Since all of our complexity bounds are at least P, our results remain the same for the derivers whose derivation structure can be constructed in polynomial time even if we take the complexity of computing the derivation structure into account.

We can show that if $\mathcal{P}$ is optimal w.r.t. a Ψ-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 $\mathsf{OP}_{\Psi}(\mathfrak{D}, \mathfrak{m})$ we can focus on proofs that are subgraphs of $\mathfrak{D}(\mathcal{T}, \eta)$. This is shown by the following lemma, which generalizes a result from [1]. The full proof can be found in the appendix.

Lemma 10. *For any* Ψ*-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* $q \in \mathbb{Q}$*, then there exists a subproof* $\mathcal{Q}$ *of* $\mathfrak{D}(\mathcal{T}, \eta)$ *for* $\mathcal{T} \models \eta$ *with* $\mathfrak{m}(\mathcal{Q}) \leq q$.

Proof (Sketch). Let $\mathcal{P}$ be any such proof. Then there is a homomorphism $h \colon \mathcal{P} \to \mathfrak{D}(\mathcal{T}, \eta)$. If $h(\mathcal{P})$ is a proof, then the claim immediately follows from **[SI]**. Otherwise, $h(\mathcal{P})$ contains cycles, which must be due to different vertices in $\mathcal{P}$ with the same label that are connected by a path. We can iteratively remove such paths to obtain another proof $\mathcal{P}^*$ that can be homomorphically mapped to a subproof $\mathcal{Q}$ of $h(\mathcal{P}) \subseteq \mathfrak{D}(\mathcal{T}, \eta)$, and therefore $\mathfrak{m}(\mathcal{Q}) \leq \mathfrak{m}(\mathcal{P}) \leq q$ by **[SI]**. □

In particular, this shows that an optimal proof always exists (recall that derivation structures are always finite).

Corollary 11. *If* $\mathcal{T} \models \eta$*, then, for every deriver* $\mathfrak{D}$ *and* Ψ*-measure* $\mathfrak{m}$*, there is an optimal proof for* $\mathcal{T} \models \eta$ *w.r.t.* $\mathfrak{D}$ *and* $\mathfrak{m}$.

We can now show the following generic upper bound for our decision problem.

Theorem 12. $\mathsf{OP}^{\mathsf{poly}}_{\Psi}(\mathfrak{D}, \mathfrak{m})$ *is in* NP.

Proof. We start by guessing a subgraph $\mathcal{P}$ of $\mathfrak{D}(\mathcal{T}, \eta)$. This can be done in non-deterministic polynomial time in the size of $\mathfrak{D}(\mathcal{T}, \eta)$, which is polynomial in the size of $\mathcal{T}$ and η. This subgraph $\mathcal{P}$ is necessarily sound, and checking the remaining properties of Definitions 4 and 5 and that $\mathfrak{m}(\mathcal{P}) \leq q$ can be done in polynomial time in the size of $\mathcal{P}$, in particular due to **[P]** in Definition 7. □

Note that, one can similarly show an upper bound of NP for *pathwidth*, since checking $\mathfrak{m}(\mathcal{P}) \leq q$ is in NP [2]. (First guessing $\mathcal{P}$ and then verifying $\mathfrak{m}(\mathcal{P}) \leq q$ can be performed by a polynomially bounded sequence of non-deterministic steps.)

Matching lower bounds hold for $\mathsf{OP}^{\mathsf{poly}}_{\Psi}(\mathfrak{D}, \mathfrak{m})$ with a fixed $\mathfrak{D}$ and $\mathfrak{m}$ being *vertex size* [1] or *justification size* [4]. As we already noted, all the complexity measures from Table 1, including *vertex size* and *justification size*, are Ψ-measures.

Theorem 13 ([1,4]). *For* $\mathfrak{m}$ *being* vertex size *or* justification size*, there is a polynomial deriver* $\mathfrak{D}$ *such that* $\mathsf{OP}^{\mathsf{poly}}_{\Psi}(\mathfrak{D}, \mathfrak{m})$ *is* NP*-complete.*

4 Local Measures

In this section, we consider a property that allows us to reduce the upper bound from NP to P. This also generalizes a result from [1]. Recall that intuitively, $\mathcal{P}[v \mapsto \mathcal{P}']$ denotes the result of replacing in $\mathcal{P}$ the proof above v by $\mathcal{P}'$.

Definition 14. *Let* $\mathcal{P} = (V, E, \ell)$ *be a proof and* $v \in V$*. We say that a* Ψ*-measure* $\mathfrak{m}$ *is* local for v in $\mathcal{P}$ *if for every proof* $\mathcal{P}'_v = (V', E', \ell')$ *with sink* v *such that* $V \cap V' = \{v\}$*,* $\ell(v) = \ell'(v)$ *and* $\mathfrak{m}(\mathcal{P}'_v) \bowtie \mathfrak{m}(\mathcal{P}_v)$*, we have* $\mathfrak{m}(\mathcal{P}[v \mapsto \mathcal{P}'_v]) \bowtie \mathfrak{m}(\mathcal{P})$*, for any choice of relation* $\bowtie \in \{<, \leq, =, \geq, >\}$*.*

A Ψ*-measure* $\mathfrak{m}$ *is* local *if it is local for every vertex in every proof.*

Intuitively, if the measure is local, then we can replace subproofs by easier or harder alternative proofs, and the weight of the full proof will change accordingly. Measures that can be defined in a monotone, recursive way on the graph structure usually have this property:

- the measures *hardest axiom*, *hardest inference step*, from Table 1;
- *tree size* defined for an aylic $H = (V, E, \ell)$ by $\mathfrak{m}(H) := \sum_{v \in sink(H)} \mathfrak{m}(H, v)$, where $\mathfrak{m}(H, v) := 1 + \sum_{(S,v) \in E} \sum_{w \in S} \mathfrak{m}(H, w)$;
- *weighted tree size*, defined similarly by $\mathfrak{m}(H) := \sum_{v \in sink(H)} \mathfrak{m}(H, v)$ with $\mathfrak{m}(H, v) := |\ell(v)| + \sum_{(S,v) \in E} \sum_{w \in S} \mathfrak{m}(H, w)$.

As the names of the last two measures indicate, local measures cannot distinguish between hypergraphs and their unravelings into trees, since unraveling of a proof $\mathcal{P}$ can be seen as recursively computing $\mathcal{P}[v \mapsto \mathcal{P}_v]$ for all vertices v in $\mathcal{P}$. That is

Algorithm 1: A Dijkstra-like algorithm

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

the reason why they 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 ELK-proof plugin for Protégé [15].

Algorithm 1 describes a Dijkstra-like approach that is inspired by the algorithm in [10] for finding minimal hyperpaths w.r.t. so-called *additive weighting functions*, which represent a subclass of local Ψ-measures. It progressively defines 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. Whenever all source vertices of a hyperedge (S, d) have been processed in this way, a new proof for $\ell(d)$ is constructed by joining the proofs for the source vertices using the new hyperedge, and this is compared to the best previously known proof for $\ell(d)$. All proofs constructed in this way are tree proofs, but this restriction does not matter since $\mathfrak{m}$ is local. For Line 18, 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 15. *For any local Ψ-measure $\mathfrak{m}$ and polynomial deriver $\mathfrak{D}$, Algorithm 1 computes an optimal proof in time polynomial in the size of $\mathcal{T}$ and η.*

Proof (Sketch). The main argument is that the algorithm is monotone in the sense that the smallest weight $\min\{\mathfrak{m}(\mathcal{P}(w)) \mid w \in Q\}$ can never decrease. Hence, easier proofs (according to $\mathfrak{m}$) will be constructed before harder ones. This shows that each vertex v will be removed at most once from Q in Line 17, and hence the algorithm terminates in polynomial time. Moreover, one can use the above fact to show that whenever v is chosen in Line 9, then $\mathcal{P}(v)$ is a minimal proof for $\ell(v)$ in $\mathfrak{D}(\mathcal{T}, \eta)$. This step uses an inductive argument that is based on the locality property of the measure $\mathfrak{m}$. Finally, by Lemma 10 we do not need to consider proofs that are not contained in $\mathfrak{D}(\mathcal{T}, \eta)$. $\square$

Since we can actually compute an optimal proof in polynomial time, the following complexity result follows.

Theorem 16. *For any local Ψ-measure $\mathfrak{m}$, $\mathsf{OP}_{\Psi}^{\mathsf{poly}}(\mathfrak{D}, \mathfrak{m})$ is in* P.

A matching lower bound was shown in [1] for the problem of finding an admissible *tree proof* of minimal *vertex size*. This is equivalent to finding an arbitrary proof of minimal *tree size*, since a tree proof (with the same *tree size*) can always be obtained by unravelling, and for tree proofs the measures *tree size* and *vertex size* coincide. Since *tree size* is a local Ψ-measure, we obtain the following.

Theorem 17 ([1]). *For $\mathfrak{m}$ being* tree size*, there is a polynomial deriver $\mathfrak{D}$ such that $\mathsf{OP}_{\Psi}^{\mathsf{poly}}(\mathfrak{D}, \mathfrak{m})$ is* P*-complete.*

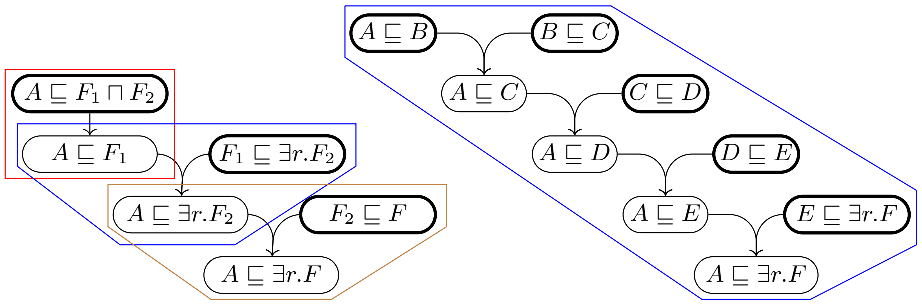
5 Chunks

In this section, we discuss Item 11 from the introduction, which is part of ongoing work. To define such a measure, we need to extend the definition of hypergraphs by a function ρ that assigns to each hyperedge (S, d) a label, the idea being that $\rho(S, d)$ is the name of the inference rule that was instantiated to obtain the inference step $\frac{\{\ell(s) | s \in S\}}{\ell(d)}$. Based on this, we define the notion of *chunks*.

Definition 18. *Given a proof $\mathcal{P}$, a* chunk *of $\mathcal{P}$ is a maximal connected subgraph $H \subseteq \mathcal{P}$ such that $\rho(e) = \rho(e')$ for all hyperedges e, e' in H'.*

Every proof has a unique decomposition into chunks that do not share hyperedges. A proof can thus be measured by the *chunk count*, which is the number of its chunks. This measure follows the idea that proofs with less alternations between inference rules are easier to understand. Consider for example proofs $\mathcal{P}_1$ and $\mathcal{P}_2$ depicted in Figure 4. Both are proofs for the same entailment $A \sqsubseteq \exists r.F$, and $\mathcal{P}_1$ has three chunks, while $\mathcal{P}_2$ has only one (all inference steps follow the same rule schema), hence $\mathcal{P}_2$ is easier to understand. Of course one can consider also more complicated measures that also take the size of the chunks into account.

Clearly, the measure satsifies **[P]**. However, we need extend the notion of homomorphism to take into account the edge labeling function ρ. Nevertheless, we conjecture that we can extend the NP upper bound in Section 3 to cover this measure by carefully adapting Condition **[SI]**. Unfortunately, the *chunk count* is not local, because it is not invariant under unraveling (which may copy a single chunk into multiple ones).



$A \sqsubseteq F_1 \sqcap F_2$
$A \sqsubseteq F_1$
$F_1 \sqsubseteq \exists r.F_2$
$A \sqsubseteq \exists r.F_2$
$F_2 \sqsubseteq F$
$A \sqsubseteq \exists r.F$
$A \sqsubseteq B$
$B \sqsubseteq C$
$A \sqsubseteq C$
$C \sqsubseteq D$
$A \sqsubseteq D$
$D \sqsubseteq E$
$A \sqsubseteq E$
$E \sqsubseteq \exists r.F$
$A \sqsubseteq \exists r.F$

Fig. 4. Two proofs $\mathcal{P}_1$ (on the left) and $\mathcal{P}_2$ of the same entailment $A \sqsubseteq \exists r.F$.

6 Conclusion

We have investigated the complexity of finding optimal proofs for description logic entailments w.r.t. a variety of measures. Obviously, size is not enough to evaluate the difficulties users face when trying to understand proofs, and existing measures for justification complexity (e.g. [13]) cannot capture the structural aspects of this problem. We identified two classes of structural measures, Ψ-measures and local Ψ-measures, and showed that finding optimal proofs w.r.t. such measures is in NP and in P, respectively, for polynomial derivers. In the future, we will investigate more measures, try to find more tractable subclasses, and identify properties of measures that make them P-hard or NP-hard. Most importantly, we will evaluate which measures are most relevant for practical purposes, to actually help users understand description logic proofs.

Acknowledgements This work was supported by the DFG in grant 389792660 as part of TRR 248 (`https://perspicuous-computing.science`), and QuantLA, GRK 1763 (`https://lat.inf.tu-dresden.de/quantla`). We also want to thank the anonymous reviewers for their many helpful comments.

References

1. Christian Alrabbaa, Franz Baader, Stefan Borgwardt, Patrick Koopmann, and Alisa Kovtunova. Finding small proofs for description logic entailments: Theory and practice. In Elvira Albert and Laura Kovacs, editors, *LPAR-23: 23rd International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, volume 73 of *EPiC Series in Computing*, pages 32–67. EasyChair, 2020. `doi:10.29007/nhpp`.
2. Stefan Arnborg, Derek G. Corneil, and Andrzej Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM J. Algebraic Discrete Methods*, 8(2):277–284, April 1987. `doi:10.1137/0608024`.
3. Franz Baader, Ian Horrocks, Carsten Lutz, and Uli Sattler. *An Introduction to Description Logic*. Cambridge University Press, 2017. `doi:10.1017/9781139025355`.

4. Franz Baader, Rafael Peñaloza, and Boontawee Suntisrivaraporn. Pinpointing in the description logic $\mathcal{EL}^+$. In *KI 2007: Advances in Artificial Intelligence, 30th Annual German Conference on AI, KI 2007, Osnabrück, Germany, September 10-13, 2007, Proceedings*, pages 52–67, 2007. `doi:10.1007/978-3-540-74565-5_7`.
5. Franz Baader and Boontawee Suntisrivaraporn. 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*, volume 410 of *CEUR-WS*, 2008. URL: `http://ceur-ws.org/Vol-410/Paper01.pdf`.
6. Alexander Borgida, Enrico Franconi, and Ian Horrocks. Explaining $\mathcal{ALC}$ subsumption. In *ECAI 2000, Proceedings of the 14th European Conference on Artificial Intelligence, Berlin, Germany, August 20-25, 2000*, pages 209–213, 2000. URL: `http://www.frontiersinai.com/ecai/ecai2000/pdf/p0209.pdf`.
7. Stefan Borgwardt, Anke Hirsch, Alisa Kovtunova, and Frederik Wiehr. In the eye of the beholder: Which proofs are best? In *Proc. of the 33th Int. Workshop on Description Logics (DL'20)*, 2020.
8. Ronald J. Brachman, Deborah L. McGuinness, Peter F. Patel-Schneider, Lori Alperin Resnick, and Alexander Borgida. Living with CLASSIC: When and how to use a KL-ONE-like language. In John F. Sowa, editor, *Principles of Semantic Networks*, pages 401–456. Morgan Kaufmann, 1991.
9. David Tena Cucala, Bernardo Cuenca Grau, and Ian Horrocks. 15 years of consequence-based reasoning. In *Description Logic, Theory Combination, and All That - Essays Dedicated to Franz Baader on the Occasion of His 60th Birthday*, pages 573–587, 2019. `doi:10.1007/978-3-030-22102-7_27`.
10. Giorgio Gallo, Giustino Longo, and Stefano Pallottino. Directed hypergraphs and applications. *Discrete Applied Mathematics*, 42(2):177–201, 1993. `doi:10.1016/0166-218X(93)90045-P`.
11. Petr Hlinený, Sang-il Oum, Detlef Seese, and Georg Gottlob. Width parameters beyond tree-width and their applications. *Comput. J.*, 51(3):326–362, 2008. URL: `https://doi.org/10.1093/comjnl/bxm052`, `doi:10.1093/comjnl/bxm052`.
12. Matthew Horridge. *Justification Based Explanation in Ontologies.* PhD thesis, University of Manchester, UK, 2011. URL: `https://www.research.manchester.ac.uk/portal/files/54511395/FULL_TEXT.PDF`.
13. Matthew Horridge, Samantha Bail, Bijan Parsia, and Uli Sattler. Toward cognitive support for OWL justifications. *Knowledge-Based Systems*, 53:66–79, 2013. `doi:10.1016/j.knosys.2013.08.021`.
14. Matthew Horridge, Bijan Parsia, and Ulrike Sattler. 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*, pages 354–369, 2010. `doi:10.1007/978-3-642-17746-0_23`.
15. Yevgeny Kazakov, Pavel Klinov, and Alexander Stupnikov. Towards reusable explanation services in protege. In Alessandro Artale, Birte Glimm, and Roman Kontchakov, editors, *Proc. of the 30th Int. Workshop on Description Logics (DL'17)*, volume 1879 of *CEUR Workshop Proceedings*, 2017. URL: `http://www.ceur-ws.org/Vol-1879/paper31.pdf`.
16. Yevgeny Kazakov, Markus Krötzsch, and Frantisek Simancik. The incredible ELK – from polynomial procedures to efficient reasoning with $\mathcal{EL}$ ontologies. *J. Autom. Reasoning*, 53(1):1–61, 2014. `doi:10.1007/s10817-013-9296-3`.
17. Carsten Lutz and Frank Wolter. Foundations for uniform interpolation and forgetting in expressive description logics. In *Proceedings of IJCAI 2011*, pages 989–995. IJCAI/AAAI, 2011. `doi:10.5591/978-1-57735-516-8/IJCAI11-170`.

18. Deborah L. McGuinness. *Explaining Reasoning in Description Logics*. PhD thesis, Rutgers University, NJ, USA, 1996. `doi:10.7282/t3-q0c6-5305`.
19. Tu Anh Thi Nguyen, Richard Power, Paul Piwek, and Sandra Williams. 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.*, pages 1–12, 2012. URL: `http://www.ida.liu.se/~patla/conferences/WoDOOM12/papers/paper4.pdf`.
20. Tu Anh Thi Nguyen, Richard Power, Paul Piwek, and Sandra Williams. 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*, pages 109–123, 2013. `doi:10.1007/978-3-642-38288-8_8`.
21. Lars Relund Nielsen, Kim Allan Andersen, and Daniele Pretolani. Finding the K shortest hyperpaths. *Computers & OR*, 32:1477–1497, 2005. `doi:10.1016/j.cor.2003.11.014`.
22. Marvin R. G. Schiller and Birte Glimm. Towards explicative inference for OWL. In *Informal Proceedings of the 26th International Workshop on Description Logics, Ulm, Germany, July 23 - 26, 2013*, pages 930–941, 2013. URL: `http://ceur-ws.org/Vol-1014/paper_36.pdf`.
23. Marvin R. G. Schiller, Florian Schiller, and Birte Glimm. 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. URL: `http://ceur-ws.org/Vol-1879/paper43.pdf`.
24. Stefan Schlobach and Ronald Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In Georg Gottlob and Toby Walsh, editors, *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI 2003)*, pages 355–362, Acapulco, Mexico, 2003. Morgan Kaufmann. URL: `http://ijcai.org/Proceedings/03/Papers/053.pdf`.

A Proofs

Lemma 8. Tree size *and all measures in Table 1 except for* depth *and* worst path *are* Ψ*-measures. Indeed,* depth*,* worst path *and* treewidth *do not satisfy* **[SI]***.*

Proof. It should be easy to see that, given a proof $\mathcal{P}$, all the measures in Table 1 can be computed according to the formulas in polynomial time in the size of $\mathcal{P}$. *Tree size* also satisfies the property **[P]**, see [1].

Since a homomorphism preserves edges and vertex labels, weights corresponding to the *hardest*, *sum* and *size* measures of subproofs of a homomorphic image is no greater than the weight of the proof.

In order to see why **[SI]** does not hold for *depth* and *worst path*, we use the following example in Figure 5. For simplicity we omit vertex labeling; we assume only $\ell(v) = \ell(v')$. Clearly, the graph on the left has *depth* 4, while its homomorphic image and a subproof obtained by omitting the cycle on the right have 5. A similar argument can be applied to *worst path* since *depth* is its special case (when the weight of an edge is always equal to 1).

The intuition behind why **[SI]** does not hold for *treewidth* is as follows. A connected graph G has treewidth 1 if and only if it is a tree. However, its homomorphic image of a tree is not necessary a tree. One can construct a tree proof and a homomorphism s.t. there is a subproof in the homomorphic image which is not a tree and, thus, has a *treewidth* greater than 1.

Despite the strong connection between *treewidth* and *pathwidth*, i.e., for any graph, its *pathwidth* is no smaller than its *treewidth*, we cannot conjecture a similar result for *pathwidth.* Cycles (if any) created by a homomorphism which also could increase *pathwidth* are not part of subrpoofs for **[SI]**. □

Lemma 10. *For any* Ψ*-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* $q \in \mathbb{Q}$*, then there exists a subproof* $\mathcal{Q}$ *of* $\mathfrak{D}(\mathcal{T},\eta)$ *for* $\mathcal{T} \models \eta$ *with* $\mathfrak{m}(\mathcal{Q}) \leq q$*.*

Proof. Let $\mathcal{P}$ be such a proof with associated homomorphism $h\colon \mathcal{P} \to \mathfrak{D}(\mathcal{T},\eta)$. If $h(\mathcal{P})$ is acyclic, then by **[SI]** in Definition 7 we have $\mathfrak{m}(h(\mathcal{P})) \leq \mathfrak{m}(\mathcal{P}) \leq q$. Since

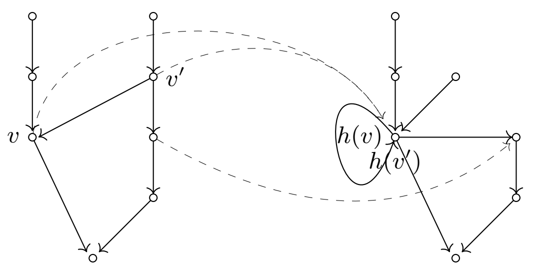


Fig. 5. A homomorphism that increases *depth* and *worst path* of the graph

$\mathcal{P}$ has a unique sink v_η, it must be mapped to a unique sink $h(v_\eta)$ in $h(\mathcal{P})$, and thus $h(\mathcal{P})$ is the desired subproof of $\mathfrak{D}(\mathcal{T}, \eta)$.

If $h(\mathcal{P})$ is not acyclic, our goal is to find another admissible proof $\mathcal{P}^*$ w.r.t. $\mathfrak{D}(\mathcal{T}, \eta)$ that uses a subset of the vertices of $\mathcal{P}$ such that $h(\mathcal{P}^*) \subseteq h(\mathcal{P})$ is acyclic, and therefore satisfies the requirements of the lemma by **[SI]** in Definition 7. For this purpose, consider an arbitrary cycle in $h(\mathcal{P})$, which must be due to two vertices v, v' in $\mathcal{P}$ such that $h(v) = h(v')$ and there is a path between v and v' (or due to multiple such pairs of vertices). Since $\mathcal{P}$ is acyclic, we can assume that there is a path from v to v', but no path from v' to v. We now consider the two subproofs $\mathcal{P}_v$ and $\mathcal{P}_{v'}$. As there is a path from v to v', we have $\mathcal{P}_v \subset \mathcal{P}_{v'}$. Since $h(v) = h(v')$, both vertices are labeled with the same axiom. The idea of the following construction is to remove $\mathcal{P}_{v'}$ from $\mathcal{P}$ and replace it with $\mathcal{P}_v$, which effectively removes all paths from v to v'.

More formally, we first consider the hypergraph $H = \mathcal{P}^{-v'} \cup \mathcal{P}_v$ and then, in the hyperedges (S, d) in H that still contain $v' \in S$, we replace v' by v, effectively merging the two vertices, remove v' from the set of vertices, and thus obtain a hypergraph $\mathcal{P}'$. If there was no such hyperedge, then v' was the sink of $\mathcal{P}$, i.e., $\ell(v') = \eta$, and v will now be the new sink in $\mathcal{P}'$ with $\ell(v) = \ell(v') = \eta$. We now show that $\mathcal{P}'$ is also an admissible proof w.r.t. $\mathfrak{D}(\mathcal{T}, \eta)$. Our construction does not produce new leafs, and hence $\mathcal{P}'$ is still grounded. Clearly, all remaining edges are sound since they were already sound in $\mathcal{P}$. Moreover, $\mathcal{P}'$ is acyclic since all cycles in $\mathcal{P}'$ can be traced back to paths in $\mathcal{P}$ that involve both v and v'; but we have assumed that there are no paths from v' to v, and have destroyed all paths from v to v'. As argued above, we have also kept the property that there is exactly one sink, which is labeled with η. Observe that h is also a homomorphism from $\mathcal{P}'$ to $\mathfrak{D}(\mathcal{T}, \eta)$ (when restricted to the vertex set of $\mathcal{P}'$), because $h(v) = h(v')$, and moreover $h(\mathcal{P}') \subseteq h(\mathcal{P})$.

The resulting proof $\mathcal{P}'$ has less vertices than $\mathcal{P}$ since we have removed at least v'. This means that, after finitely many such operations, we can obtain from $\mathcal{P}$ the desired proof $\mathcal{P}^*$ such that $h(\mathcal{P}^*) \subseteq h(\mathcal{P})$ is acyclic. Since $h(\mathcal{P}^*)$ also has a unique sink labeled by η, it is a subproof of $\mathfrak{D}(\mathcal{T}, \eta)$ and $\mathfrak{m}(h(\mathcal{P}^*)) \leq \mathfrak{m}(\mathcal{P}) \leq q$ by **[SI]** in Definition 7. □

Corollary 11. *If $\mathcal{T} \models \eta$, then, for every deriver $\mathfrak{D}$ and Ψ-measure $\mathfrak{m}$, there is an optimal proof for $\mathcal{T} \models \eta$ w.r.t. $\mathfrak{D}$ and $\mathfrak{m}$.*

Proof. By Definition 6, 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 weights of these proofs always has a minimum. Finally, if there were an admissible proof weighing less than this minimum, it would contradict Lemma 10. □

Lemma 15. *For any local Ψ-measure $\mathfrak{m}$ and polynomial deriver $\mathfrak{D}$, Algorithm 1 computes an optimal proof in time polynomial in the size of $\mathcal{T}$ and η.*

Proof. We can show the following facts about this algorithm.

(I) *Whenever $\mathcal{P}(v)$ is defined, then it is a proof for $\ell(v)$ contained in $\mathfrak{D}(\mathcal{T},\eta)$.*

We prove this by induction on the order in which the hypergraphs $\mathcal{P}(v)$ are constructed by Algorithm 1. The ones in Line 4 consist of a single leaf v, which is labeled by a TBox axiom, and hence are sound, grounded, acyclic, and have the single sink v. Similarly, $\mathcal{P}(v)$ in Line 6 is always a proof since it consists of a single edge from $\mathfrak{D}(\mathcal{T},\eta)$, has no leafs, and has v as the only sink.

Consider now the hypergraph $\mathcal{P}$ constructed in Line 14 as a possible candidate for $\mathcal{P}(v)$ (where $v = d$). At this point, all $\mathcal{P}(s)$, $s \in S$, are already defined since the counter $k(e)$ can only reach $|S|$ if each $s \in S$ has already been chosen in Line 9, and thus $\mathcal{P}(s)$ must have been defined. Hence, by induction, each $\mathcal{P}(s)$ is a proof for $\ell(s)$ contained in $\mathfrak{D}(\mathcal{T},\eta)$, and, because we assume that $\mathfrak{D}(\mathcal{T},\eta)$ contains no two vertices with the same label, must have s as sink. This shows that the hypergraph $\mathcal{P}$ constructed in Line 14 is sound, grounded, and has a single sink, namely v. Finally, $\mathcal{P}(v)$ is only updated to $\mathcal{P}$ in Line 17 if $\mathcal{P}$ is acyclic and therefore it is a proof.

(II) *If vertex v is chosen before vertex w in Line 9, then $\mathfrak{m}(\mathcal{P}(v)) \leq \mathfrak{m}(\mathcal{P}(w))$.*

We show that after choosing v in Line 9 the algorithm cannot produce a new proof $\mathcal{P}$ in Line 14 with $\mathfrak{m}(\mathcal{P}) < \mathfrak{m}(\mathcal{P}(v))$, and thus the smallest weight $\min\{\mathfrak{m}(\mathcal{P}(w)) \mid w \in Q\}$ can never decrease. Consider the proof $\mathcal{P}$ from Line 15. Since $v \in S$, we have $\mathcal{P}(v) \subset \mathcal{P}$, and therefore $\mathfrak{m}(\mathcal{P}(v)) \leq \mathfrak{m}(\mathcal{P})$ by **[SI]**.

(III) *Algorithm 1 terminates in polynomial time.*

Item (II) implies that each vertex $v \in V$ can be removed from Q at most once: in order for v to be added again to Q in Line 17, there would need to exist a proof other than $\mathcal{P}(v)$ with the same sink v but a smaller weight, but according to (II), after choosing v in Line 9, the algorithm does not construct any proofs with a weight smaller than $\mathfrak{m}(\mathcal{P}(v))$ (for any sink). Therefore, in Line 14, during the complete run of the algorithm, each edge $(S,d) \in E$ will be used at most once. Moreover, all primitive operations in the algorithm can be done in polynomial time, such as checking acyclicity of hypergraphs in Line 15 or finding the minimal value $\mathfrak{m}(\mathcal{P}(v))$ in Line 9. It follows that Algorithm 1 terminates in time polynomial in the size of $\mathfrak{D}(\mathcal{T},\eta)$, which is polynomial in the size of $\mathcal{T}$ and η.

(IV) *Every vertex $v \in V$ that is the sink of a proof $\mathcal{P}$ contained in $\mathfrak{D}(\mathcal{T},\eta)$ is added to Q at some point.*

We prove this by induction on the structure of $\mathcal{P}$. If $\mathcal{P}$ contains only v, then either $\ell(v) \in \mathcal{T}$, and hence v is added to Q in Line 4, or otherwise there is an edge $(\emptyset, v)$ in $\mathcal{P}$ (and hence in E), in which case v is added to Q in Line 6.

If $\mathcal{P}$ has more than one vertex, then it must contain at least one edge $e = (S, v) \in E$, where each $s \in S$ is the sink of a subproof of $\mathcal{P}$ in $\mathfrak{D}(\mathcal{T},\eta)$. By induction we know that each $s \in S$ is added to Q at some point during the algorithm. By Item (III), they must also be removed from Q at some point afterwards, and hence eventually $k(e)$ reaches $|S|$ in Line 13. If $\mathcal{P}(v)$ was already defined at this point, then v had already been added to Q earlier. Otherwise, v is now added to Q in Line 17.

(V) *When the algorithm terminates and $\mathcal{P}(v)$ is defined, then $\mathfrak{m}(\mathcal{P}(v))$ is minimal among all proofs for $\ell(v)$ contained in $\mathfrak{D}(\mathcal{T},\eta)$.*

By (I), $\mathcal{P}(v)$ is a proof of this form. Assume to the contrary that there is a proof $\mathcal{P}$ for $\ell(v)$ contained in $\mathfrak{D}(\mathcal{T},\eta)$ such that $\mathfrak{m}(\mathcal{P}) < \mathfrak{m}(\mathcal{P}(v))$. Then $\mathcal{P}$ and $\mathcal{P}(v)$ must both have the sink v, because we assume that $\mathfrak{D}(\mathcal{T},\eta)$ contains no two vertices with the same label. Assume moreover that

i) $\mathcal{P}$ is an optimal proof for $\ell(v)$, that is, $\mathfrak{m}(\mathcal{P}) \leq \mathfrak{m}(\mathcal{P}')$ for every other proof $\mathcal{P}'$ for $\ell(v)$ in $\mathfrak{D}(\mathcal{T},\eta)$ (cf. Corollary 11), and

ii) among all other vertices $v' \in V$ for which there exists a proof $\mathcal{P}'$ for $\ell(v')$ in $\mathfrak{D}(\mathcal{T},\eta)$ such that $\mathfrak{m}(\mathcal{P}') < \mathfrak{m}(\mathcal{P}(v'))$, we also have $\mathfrak{m}(\mathcal{P}) \leq \mathfrak{m}(\mathcal{P}')$ and whenever $\mathfrak{m}(\mathcal{P}) = \mathfrak{m}(\mathcal{P}')$, then $|\mathcal{P}| \leq |\mathcal{P}'|$.

Since $\mathcal{P}$ is optimal, we know that it has a unique last inference step (S,v). We show that, for every vertex $w \in S$, an optimal proof was assigned to $\mathcal{P}(w)$ before v was chosen in Line 9. We first show that for every $w \in S$, the subproof $\mathcal{P}_w$ must be similarly minimal.

If this were not the case, then there would be a proof $\mathcal{P}'_w$ for $\ell(w)$ in $\mathfrak{D}(\mathcal{T},\eta)$ such that $\mathfrak{m}(\mathcal{P}'_w) < \mathfrak{m}(\mathcal{P}_w)$. But then, by locality, we have $\mathfrak{m}(\mathcal{P}[w \mapsto \mathcal{P}''_w]) < \mathfrak{m}(\mathcal{P})$, where $\mathcal{P}''_w$ is constructed from $\mathcal{P}'_w$ by renaming all vertices except w. Since $\mathcal{P}[w \mapsto \mathcal{P}''_w]$ is an admissible proof for $\ell(v)$ w.r.t. $\mathfrak{D}(\mathcal{T},\eta)$, by Lemma 10 we know that $\mathfrak{D}(\mathcal{T},\eta)$ contains a proof $\mathcal{P}'$ for $\ell(v)$ with $\mathfrak{m}(\mathcal{P}') \leq \mathfrak{m}(\mathcal{P}[w \mapsto \mathcal{P}''_w]) < \mathfrak{m}(\mathcal{P})$, contradicting the optimality of $\mathcal{P}$. Hence, $\mathcal{P}_w$ must be optimal among all proofs for $\ell(w)$ in $\mathfrak{D}(\mathcal{T},\eta)$.

Moreover, by **[SI]**, we have $\mathfrak{m}(\mathcal{P}_w) \leq \mathfrak{m}(\mathcal{P})$ and $|\mathcal{P}_w| < |\mathcal{P}|$. By Assumption ii) and (IV), this means that $\mathcal{P}(w)$ is also optimal w.r.t. $\mathfrak{m}$ among all proofs for $\ell(w)$ in $\mathfrak{D}(\mathcal{T},\eta)$.

Since both $\mathcal{P}_w$ and $\mathcal{P}(w)$ are optimal in this sense, we obtain that

$$\mathfrak{m}(\mathcal{P}(w)) = \mathfrak{m}(\mathcal{P}_w) \leq \mathfrak{m}(\mathcal{P}) < \mathfrak{m}(\mathcal{P}(v)),$$

which by (II) means that w must have been chosen (in Line 9) before v. To summarize, for every vertex $w \in S$, we know that an optimal proof for $\ell(w)$ with weight $\mathfrak{m}(\mathcal{P}_w)$ has already been assigned to $\mathcal{P}(w)$ before v is chosen in Line 9, and moreover each $w \in S$ was chosen before v. But then, for one of these vertices w (the last one to be processed), a proof $\mathcal{P}'$ is constructed from the subproofs $\mathcal{P}(w)$ and the edge (S,v) in Line 14. If we consider the variant $\mathcal{P}''$ of $\mathcal{P}'$ where each of the subproofs $\mathcal{P}(w)$ is renamed such that it shares only the vertex w with the rest of the proof, then by locality we have $\mathfrak{m}(\mathcal{P}'') = \mathfrak{m}(\mathcal{P}')$. Moreover, since $\mathfrak{m}(\mathcal{P}(w)) = \mathfrak{m}(\mathcal{P}_w)$ holds for all $w \in S$, locality also yields that $\mathfrak{m}(\mathcal{P}'') = \mathfrak{m}(\mathcal{P})$. Since $\mathcal{P}'$ was constructed as a candidate for $\mathcal{P}(v)$ by the algorithm, we know that $\mathfrak{m}(\mathcal{P}(v)) \leq \mathfrak{m}(\mathcal{P}') = \mathfrak{m}(\mathcal{P})$, which contradicts our assumption that $\mathfrak{m}(\mathcal{P}) < \mathfrak{m}(\mathcal{P}(v))$. We obtain that $\mathcal{P}(v)$ must be optimal.

Since by Lemma 10 the derivation structure $\mathfrak{D}(\mathcal{T}, \eta)$ contains an optimal proof for $\mathcal{T} \models \eta$, Items (III)–(V) show that Algorithm 1 returns such a proof in Line 18. $\square$