The Shape of $\mathcal{EL}$ Proofs: A Tale of Three Calculi

Christian Alrabbaa, Stefan Borgwardt, Philipp Herrmann and Markus Krötzsch

Institute of Theoretical Computer Science, Technische Universität Dresden, 01062 Dresden, Germany

Abstract

Consequence-based reasoning can be used to construct proofs that explain entailments of description logic (DL) ontologies. In the literature, one can find multiple consequence-based calculi for reasoning in the $\mathcal{EL}$ family of DLs, each of which gives rise to proofs of different shapes. Here, we study three such calculi and the proofs they produce on a benchmark based on the OWL Reasoner Evaluation. The calculi are implemented using a translation into existential rules with stratified negation, which had already been demonstrated to be effective for the calculus of the Elk reasoner. We then use the rule engine Nemo to evaluate the rules and obtain traces of the rule execution. After translating these traces back into DL proofs, we compare them on several metrics that reflect different aspects of their complexity.

Keywords

Explanations, Proofs, Rule Engine, Nemo

1. Introduction

Consequence-based reasoning for DLs has a long tradition, starting from the first reasoning algorithms for $\mathcal{EL}$ with general concept inclusions [1, 2], all the way up to expressive DLs like $\mathcal{ALCHOIQ}$ [3, 4]. Due to its favorable computational properties, it is employed in several reasoners, such as Elk, Konclude, and Sequoia [5, 6, 7]. Another advantage of consequence-based approaches is the possibility to extract proofs consisting of step-wise derivations, in order to explain consequences of the ontology to an ontology engineer. In principle, the relatively simple rules of a reasoning calculus can immediately be used to construct proofs. However, in practice, this would have to be implemented by each consequence-based reasoner, and is so far only supported by Elk [8]. Several approaches have been developed to extract proofs from reasoners in a black-box fashion [9, 10, 11], which do not use a fixed set of rules and often lead to more complicated proof steps, but smaller proofs overall [12]. Our main research question is how proofs resulting from consequence-based calculi differ based on the shape of the rules. Specifically, we explore which calculi may be more appropriate in certain scenarios—for instance, producing proofs suitable for specific types of visualizations, or ones in which information is introduced and resolved locally, resulting in a reasoning structure that is easier to follow. We investigate this on three calculi for the $\mathcal{EL}$ family of description logics [2, 5, 13].

We follow an approach to encode DL reasoning into (an extension of) Datalog rules [14, 15, 16]. To execute these rules, we use Nemo, a Datalog-based rule engine that is easy to use and

DL 2025: 38th International Workshop on Description Logics, September 3–6, 2025, Opole, Poland

christian.alrabbaa@tu-dresden.de (C. Alrabbaa); stefan.borgwardt@tu-dresden.de (S. Borgwardt); philipp.herrmann1@tu-dresden.de (P. Herrmann); markus.kroetzsch@tu-dresden.de (M. Krötzsch)
0000-0002-2925-1765 (C. Alrabbaa); 0000-0003-0924-8478 (S. Borgwardt); 0000-0002-9172-2601 (M. Krötzsch)

offers many advanced features, such as datatypes, existential rules, aggregates, and stratified negation [16]. This has already been demonstrated to be effective for the reasoning calculus of Elk, including pre-processing of the ontology in OWL format,[1] which essentially implements an $\mathcal{EL}_\bot^+$ reasoner by existential rules with stratified negation [16]. The advantage of this approach is that one can relatively quickly implement new reasoning procedures with low effort.

In this paper, we also consider two other calculi for $\mathcal{EL}$-based logics from the literature: the original $\mathcal{EL}^{++}$ calculus, denoted by Envelope [2] (restricted to $\mathcal{EL}_\bot^+$, i.e., without nominals and concrete domains), and the $\mathcal{EL}$ calculus Textbook [13] (extended to $\mathcal{ELH}_\bot$, i.e., with role hierarchies and $\bot$). The modifications to the calculi allow us to compare them on a dataset of $\mathcal{ELH}_\bot$ reasoning tasks extracted from the 2015 OWL Reasoner Evaluation (ORE) [17]. After encoding them into rules as for the Elk calculus, we use the tracing capabilities of Nemo to compute proofs for the derived consequences. However, since Nemo traces are based on translated rules and may include statements not expressible as DL axioms (e.g., side conditions), we must still transform them to obtain DL proofs suitable for inspection by ontology engineers. Finally, we compare the shape of the resulting proofs using several measures, such as the *size*, *depth*, *directed cutwidth* [18], and cognitive complexity of inference steps [19].

2. Calculi and Proofs

Let $\mathsf{N_C}$ and $\mathsf{N_R}$ be two disjoint, countably infinite sets of *concept-* and *role names*, respectively. $\mathcal{EL}_\bot^+$ *concepts* are defined by the grammar $C, D ::= \top \mid \bot \mid A \mid C \sqcap D \mid \exists r.C$, where $A \in \mathsf{N_C}$ and $r \in \mathsf{N_R}$. $\mathcal{EL}_\bot^+$ *axioms* are either *concept inclusions* of the form $C \sqsubseteq D$ or *complex role inclusions* of the form $r_1 \circ \cdots \circ r_n \sqsubseteq r$, where $r_1, \ldots, r_n, r \in \mathsf{N_R}$. An $\mathcal{EL}_\bot^+$ *TBox* is a finite set of $\mathcal{EL}_\bot^+$ axioms. $\mathcal{ELH}_\bot$ is the fragment of $\mathcal{EL}_\bot^+$ that only allows *simple role inclusions* of the form $r \sqsubseteq s$. We assume the reader to be familiar with the semantics of these logics, in particular the definition of *entailment* of an axiom α from a TBox $\mathcal{T}$, written $\mathcal{T} \models \alpha$ [13].

Calculi. We consider three inference calculi for fragments of $\mathcal{EL}_\bot^+$ that are tailored towards *classification*, i.e., computing all entailments of the form $\mathcal{T} \models A \sqsubseteq B$ for $A, B \in \mathsf{N_C}$.

Figure 1 shows the inference rules used in the reasoner Elk for $\mathcal{EL}_\bot^+$ TBoxes. Side conditions are marked in gray, where "C occurs negatively" means that C occurs within a concept on the left-hand side of some axiom in $\mathcal{T}$, and $\sqsubseteq_\mathcal{T}^*$ denotes the precomputed *role hierarchy*, i.e., the transitive closure over simple role inclusions. Furthermore, complex role inclusions are assumed to be in the normal form $r_1 \circ r_2 \sqsubseteq r$, using only binary role composition. Axioms with longer role compositions can be normalized with the help of fresh role names. Then, $\mathcal{T} \models A \sqsubseteq B$ holds iff either $A \sqsubseteq B$ or $A \sqsubseteq \bot$ can be derived by these rules from $\mathsf{init}(A)$ [5].

Figure 2 shows the Textbook classification rules for $\mathcal{EL}$ from [13] with a slight modification to rule CR5 and the addition of a variant of $\mathsf{R}_\bot$ from Elk, in order to support $\mathcal{ELH}_\bot$. Here, all input and derived concept inclusions are required to be of one of the forms $A \sqsubseteq B$, $A_1 \sqcap A_2 \sqsubseteq B$, $A \sqsubseteq \exists r.B$ or $\exists r.A \sqsubseteq B$, where A, A_1, A_2, B are concept names from $\mathcal{T}$, $\top$ or $\bot$, which is again without loss of generality. This calculus is correct in the same sense as the Elk calculus above, but, instead of the $\mathsf{init}(A)$ statement, it is initialized with all axioms of the input TBox $\mathcal{T}$.

[1]https://www.w3.org/TR/owl2-overview/

$$\mathsf{R}_0\ \frac{\mathsf{init}(C)}{C \sqsubseteq C} \qquad \mathsf{R}_\top\ \frac{\mathsf{init}(C)}{C \sqsubseteq \top} : \top \text{ occurs negatively in } \mathcal{T} \qquad \mathsf{R}_\sqsubseteq\ \frac{C \sqsubseteq D}{C \sqsubseteq E} : D \sqsubseteq E \in \mathcal{T}$$

$$\mathsf{R}_\sqcap^-\ \frac{C \sqsubseteq D_1 \sqcap D_2}{C \sqsubseteq D_1 \quad C \sqsubseteq D_2} \qquad \mathsf{R}_\sqcap^+\ \frac{C \sqsubseteq D_1 \qquad C \sqsubseteq D_2}{C \sqsubseteq D_1 \sqcap D_2} : D_1 \sqcap D_2 \text{ occurs negatively in } \mathcal{T}$$

$$\mathsf{R}_\exists^-\ \frac{C \sqsubseteq \exists r.E}{C \xrightarrow{r} E} \qquad \mathsf{R}_\exists^+\ \frac{C \xrightarrow{r} D \qquad D \sqsubseteq E}{C \sqsubseteq \exists s.E} : \begin{array}{l} r \sqsubseteq^*_\mathcal{T} s \\ \exists s.E \text{ occurs negatively in } \mathcal{T} \end{array}$$

$$\mathsf{R}_\rightsquigarrow\ \frac{C \xrightarrow{r} E}{\mathsf{init}(E)} \qquad \mathsf{R}_\bot\ \frac{C \xrightarrow{r} E \qquad E \sqsubseteq \bot}{C \sqsubseteq \bot} \qquad \mathsf{R}_\circ\ \frac{C \xrightarrow{r_1} D \qquad D \xrightarrow{r_2} E}{C \xrightarrow{s} E} : \begin{array}{l} r_1 \sqsubseteq^*_\mathcal{T} s_1 \\ r_2 \sqsubseteq^*_\mathcal{T} s_2 \\ s_1 \circ s_2 \sqsubseteq s \in \mathcal{T} \end{array}$$

Figure 1: Optimized Elk calculus [5].

$$\mathsf{CR1}\ \frac{}{C \sqsubseteq C} \qquad \mathsf{CR2}\ \frac{}{C \sqsubseteq \top} \qquad \mathsf{CR4}\ \frac{C \sqsubseteq D_1 \qquad C \sqsubseteq D_2 \qquad D_1 \sqcap D_2 \sqsubseteq E}{C \sqsubseteq E}$$

$$\mathsf{CR3}\ \frac{C \sqsubseteq D \qquad D \sqsubseteq E}{C \sqsubseteq E} \qquad \mathsf{CR5'}\ \frac{C \sqsubseteq \exists r.D_1 \quad D_1 \sqsubseteq D_2 \quad \exists s.D_2 \sqsubseteq E}{C \sqsubseteq E} : r \sqsubseteq^*_\mathcal{T} s$$

$$\mathsf{R}'_\bot\ \frac{C \sqsubseteq \exists r.E \qquad E \sqsubseteq \bot}{C \sqsubseteq \bot}$$

Figure 2: Textbook calculus [13] with a modified CR5 and added variant of $\mathsf{R}_\bot$.

$$\mathrm{CR1}\ \frac{C \sqsubseteq D}{C \sqsubseteq E} : D \sqsubseteq E \in \mathcal{T} \qquad \mathrm{CR2}\ \frac{C \sqsubseteq D_1 \quad C \sqsubseteq D_2}{C \sqsubseteq E} : D_1 \sqcap D_2 \sqsubseteq E \in \mathcal{T}$$

$$\mathrm{CR3}\ \frac{C \sqsubseteq D}{C \sqsubseteq \exists r.E} : D \sqsubseteq \exists r.E \in \mathcal{T} \qquad \mathrm{CR4}\ \frac{C \sqsubseteq \exists r.D_1 \quad D_1 \sqsubseteq D_2}{C \sqsubseteq E} : \exists r.D_2 \sqsubseteq E \in \mathcal{T}$$

$$\mathrm{CR5}\ \frac{C \sqsubseteq \exists r.D \quad D \sqsubseteq \bot}{C \sqsubseteq \bot} \qquad \mathrm{CR10}\ \frac{C \sqsubseteq \exists r.E}{C \sqsubseteq \exists s.E} : r \sqsubseteq s \in \mathcal{T}$$

$$\mathrm{CR11}\ \frac{C \sqsubseteq \exists r_1.D \quad D \sqsubseteq \exists r_2.E}{C \sqsubseteq \exists s.E} : r_1 \circ r_2 \sqsubseteq s \in \mathcal{T}$$

Figure 3: Envelope calculus [2], restricted to $\mathcal{EL}^+_\bot$.

Figure 3 shows the Envelope rules for $\mathcal{EL}^+_\bot$ from [2], omitting rules CR6–CR9 for nominals and concrete domains. For a consistent representation, we translate the statements "$D \in S(C)$" and "$(C, D) \in R(r)$" from the original paper into $C \sqsubseteq D$ and $C \sqsubseteq \exists r.D$, respectively. Note that C and D must be concept names from $\mathcal{T}$, $\top$ or $\bot$. This calculus requires the concept and role inclusions in $\mathcal{T}$ to be in normal form, and is correct in the same sense as for the other calculi, but stars only from the tautologies $C \sqsubseteq C$ and $C \sqsubseteq \top$ for all concept names C from $\mathcal{T}$.

Proofs. Following the notion introduced in [11], a *proof* of $\mathcal{T} \models A \sqsubseteq B$, where A and B are concept names, is a finite, acyclic, directed *hypergraph*, where each vertex v is labeled with an

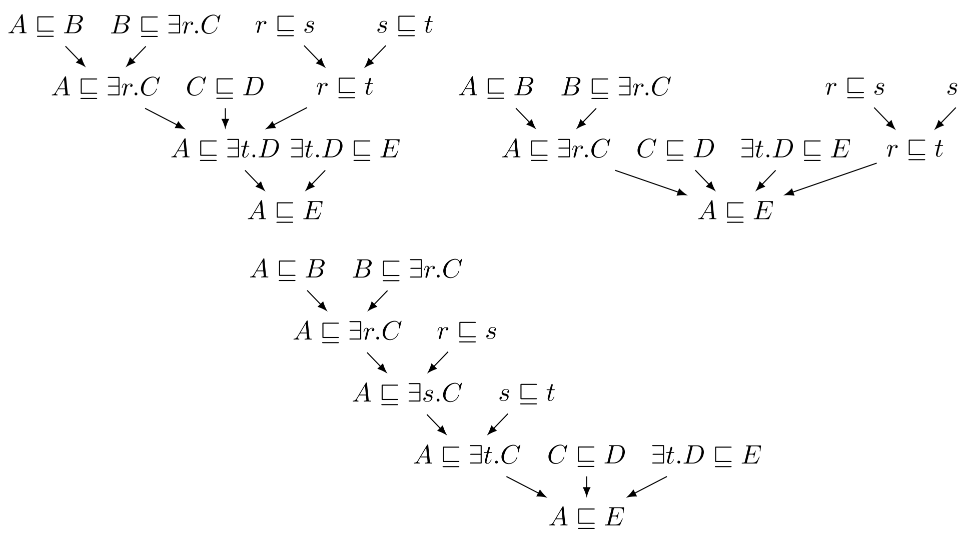
A ⊑ B B ⊑ ∃r.C r ⊑ s s ⊑ t
A ⊑ ∃r.C C ⊑ D r ⊑ t
A ⊑ ∃t.D ∃t.D ⊑ E
A ⊑ E
A ⊑ B B ⊑ ∃r.C r ⊑ s s ⊑ t
A ⊑ ∃r.C C ⊑ D ∃t.D ⊑ E r ⊑ t
A ⊑ E
A ⊑ B B ⊑ ∃r.C
A ⊑ ∃r.C r ⊑ s
A ⊑ ∃s.C s ⊑ t
A ⊑ ∃t.C C ⊑ D ∃t.D ⊑ E
A ⊑ E

Figure 4: Example proofs based on Elk (top left), Textbook (top right), and Envelope (bottom)

axiom $\ell(v)$. Hyperedges represent sound inference steps and are of the form (S, d), where S is a set of vertices and d is a vertex s.t. $\{\ell(v) \mid v \in S\} \models \ell(d)$; note that the direction of the edge is from the vertices in S to the vertex d. The leaf vertices (without incoming hyperedges) of a proof are labeled with axioms from $\mathcal{T}$, and the unique root (without outgoing hyperedges) is labeled with $A \sqsubseteq B$. Moreover, to obtain smaller proofs, they are also required to be *non-redundant*, which means that the same axiom cannot have multiple subproofs: no two vertices can have the same label, every vertex can have at most one incoming hyperedge, and a vertex has no incoming hyperedge iff it is labeled by an axiom from $\mathcal{T}$. Usually, hyperedges are additionally distinguished by a label, which corresponds to the rule name from a calculus. However, for simplicity, we dispense with hyperedges altogether and simply view proofs as directed acyclic graphs (with edges pointing towards the root) without rule names. The children of a vertex v are then the premises of the (unique) inference step that was used to derive $\ell(v)$. The label of a vertex without predecessors must then be either from $\mathcal{T}$ or a tautology that can be derived using a rule without premises. Moreover, since most proof visualizations [20, 21] are based on tree-shaped proofs, we only consider the tree-shaped unravelings of these directed acyclic graphs in the following (see Figure 4). Thus, there can be multiple vertices with the same label, but they must have isomorphic subproofs (subtrees).

Consider, for example, the TBox $\mathcal{T} = \{A \sqsubseteq B,\ B \sqsubseteq \exists r.C,\ C \sqsubseteq D,\ \exists t.D \sqsubseteq E,\ r \sqsubseteq s,\ s \sqsubseteq t\}$ and the entailment $\mathcal{T} \models A \sqsubseteq E$. Since proofs are to be inspected by ontology engineers, they are restricted to DL axioms. This means, however, that they cannot include side conditions and statements that are not DL axioms (e.g., $\mathsf{init}(C)$). Thus, we have to adapt the calculus rules before we can use them as inference steps in proofs.

We treat side conditions of the form $C \sqsubseteq D \in \mathcal{T}$ as ordinary premises $C \sqsubseteq D$, since they are DL axioms. However, this does not mean that $\mathsf{R}_{\sqsubseteq}$ from Elk is now equivalent to CR3 from Textbook, since the second premise in the former is still restricted to axioms from $\mathcal{T}$.

Similarly, we treat side conditions $r \sqsubseteq^*_{\mathcal{T}} s$ as premises $r \sqsubseteq s$. However, to obtain a valid proof, we also have to derive these axioms from the TBox. For both the Elk and the Textbook calculus, we therefore add the following two rules:

$$\frac{}{t \sqsubseteq t} : t \in \mathsf{N_R} \text{ occurs negatively in } \mathcal{T} \qquad \frac{s \sqsubseteq t}{r \sqsubseteq t} : r \sqsubseteq s \in \mathcal{T}$$

Next, we replace statements of the form $C \xrightarrow{r} E$ in the Elk calculus by $C \sqsubseteq \exists r.E$, which preserves the soundness of the inference rules, and is similar to the treatment of "$(C, D) \in R(r)$" in the Envelope calculus. We also omit $\mathsf{init}(C)$ statements, meaning that R_0 and R_1 from Elk become similar to CR1 and CR2 from Textbook, but they may only appear in a proof if $\mathsf{init}(C)$ was actually derived by the original rules. We also add CR1 and CR2 to the Envelope calculus to express the initialization step, which does not correspond to explicit rules in [2]. For the Elk and Envelope calculi, which are not explicitly initialized with the TBox axioms, this results in proofs such as the following:

$$\frac{\frac{}{A \sqsubseteq A} \qquad A \sqsubseteq B}{A \sqsubseteq B}$$

However, this violates non-redundancy since $A \sqsubseteq B$ has multiple non-isomorphic proofs. In such cases, we keep only a subproof of minimal size (in the example, only the leaf $A \sqsubseteq B$). This also applies to other inference rules, for example, $\mathsf{R}_{\exists^-}$, which now trivially derives $C \sqsubseteq \exists r.E$ from $C \sqsubseteq \exists r.E$. Thus, due to the above simplification, this rule will never appear in a proof.

We also omit the side conditions of the form "X occurs negatively in $\mathcal{T}$". To represent them in our proofs, we could add the axiom in which X occurs negatively as a premise; however, this axiom is not logically necessary for the inference step, and may be confusing because it has no connection to the inference other than the occurrence of X. For example, consider the following application of $\mathsf{R}^+_\exists$, which also requires that $\exists t.D$ occurs negatively in $\mathcal{T}$:

$$\frac{A \sqsubseteq \exists r.C \qquad C \sqsubseteq D \qquad r \sqsubseteq t}{A \sqsubseteq \exists t.D}$$

We could add $\exists t.D \sqsubseteq E$ as a premise to express the side condition, but this axiom is not required to derive $A \sqsubseteq \exists t.D$. Moreover, since $\exists t.D \sqsubseteq E$ is also used in the subsequent application of $\mathsf{R}_\sqsubseteq$, having it appear in two consecutive steps—while only being necessary for one—may be unnecessarily confusing.

The proofs resulting from all these considerations can be seen in Figure 4, where we draw them as trees to emphasize their shape, which we analyze in the following. Despite the transformations and simplifications we applied, the three calculi can still result in substantially different proofs, even for such a simple entailment.

3. Measures

We evaluate the shape of proofs by several measures from the literature, which reflect different aspects of how ontology engineers can inspect a proof in different representations (see Figure 5).

Size. The *size* of a proof [11] is the number of occurrences of axioms in the proof tree. For example, if ontology engineers need to inspect the whole proof because they are unfamiliar

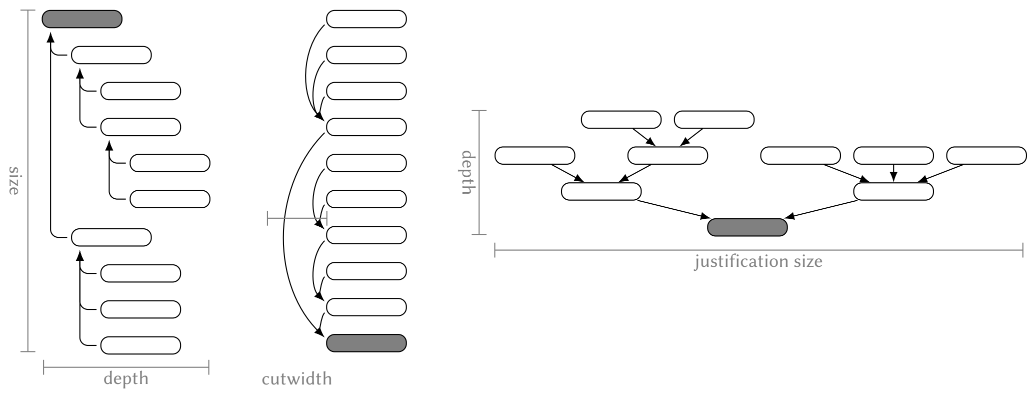

Figure 5: Different visual representations of a proof: nested list (left), linear (middle), proof tree (right).

with the ontology, the size of the proof influences how much time they will need to inspect it. The size is proportional to the height of a linear representation of the proof or a nested list visualization as in a file browser, e.g., in the *proof explanation plugin* for Protégé [20].

Depth. The *depth* of a proof [22] is the length of the longest path from a leaf to the root. This is related to the task of finding an error in the ontology based on an erroneous inference. In this case, ontology engineers would rather inspect the proof only along the "most suspicious-looking" branch (or a few such branches), and thus the time they may need in the worst case depends on the length of the longest branch. The depth is also proportional to the height of a more traditional proof-tree representation, as used in Evonne [21].

The width of such a tree visualization, assuming that adjacent subtrees are *non-overlapping* in the sense that nodes from one subtree are not positioned vertically above nodes from another subtree, can be expressed as a function of the number of leafs in the proof, which is also called its *justification size* [23]. However, we do not consider the justification size in this paper, since it does not depend on the reasoning calculus.

Cutwidth. The *(directed) cutwidth* of a graph tries to measure how linear it is [18]. Given a linear vertical arrangement of a proof's nodes [21], the cutwidth is the maximal number of edges that would be affected when cutting the graph horizontally between any two consecutive nodes (see Figure 5). The cutwidth of the graph is then the minimum of the cutwidths of all possible such linear arrangements. Hence, it is proportional to the maximal number of intermediate axioms an ontology engineer needs to keep in memory when reading the proof in such an (optimal) linear representation from top to bottom.

Bushiness Score. We developed another score to measure how "bushy" (non-linear) a proof is. It is computed as the ratio between the size of the proof and its depth ($+1$ for the root). Hence, it can be interpreted as the average number of vertices per level (see Figure 5). A completely linear proof in which all inference steps have only one premise results in a bushiness score of 1 (not bushy at all), and the full binary tree with five levels gets a score of $\frac{31}{5} = 6.2$ (very bushy).

Step Complexity. Finally, we consider a measure based on the cognitive complexity of inference steps, which has been proposed in the context of justifications, and has been evaluated in user studies [19]. It reflects multiple aspects of the syntactical structure of the argument, such as the depth of involved concepts, how many constructors and axiom types are used, or whether it uses the triviality of a concept name (i.e., being equivalent to $\bot$ or $\top$). Here, we consider the average of the step complexities of the individual inference steps in the proof.

4. Directed Cutwidth on Trees

Computing cutwidth is NP-complete in general [18], and the general-purpose implementations we tried could not compute the metric for our complete evaluation set in a feasible time. The problem becomes polynomial on trees [24], but the algorithm is complicated, and we are not aware of any implementation. *Directed* cutwidth on trees, however, admits a far simpler algorithm, which we now introduce and prove to be correct, since we could not find any publication that establishes this result.

Consider a tree $T = \langle V, E\rangle$ with vertices V and directed edges E pointing towards the leafs.[2] A *serialization* of T is a word $S \in V^*$ that contains each vertex $v \in V$ at a unique position $v_S \in \{1, \dots, |V|\}$ such that $\langle v, w\rangle \in E$ implies $v_S < w_S$. For $i \in \{1, \dots, |V| - 1\}$, the number of edges cut in the ith gap between vertices in S is $\mathsf{cut}(i) = |\{\langle v, w\rangle \in E \mid v_S \leq i < w_S\}|$. The *cutwidth* $\mathsf{cw}(S)$ of S is $\max_{1 \leq i < |V|} \mathsf{cut}(i)$, and the *directed cutwidth* $\mathsf{cw}(T)$ of T is the minimal cutwidth of any serialization of T. The out-degree of any vertex in T is a lower bound for its directed cutwidth, though it can be higher (e.g., for a full binary tree T, $\mathsf{cw}(T)$ is depth $+ 1$). We omit *directed* below, since we consider no other kind of cutwidth on trees.

Definition 1. *The* standard serialization $S(T)$ *of a tree* $T = \langle V, E\rangle$ *is defined inductively. If* $V = \{v\}$*, then* $S(T) := v$*. If* $|V| > 1$*, then let* r *be the root of* T *and let* $C_1, \dots, C_\ell$ *be its direct subtrees, ordered such that* $i < j$ *implies* $\mathsf{cw}(S(C_i)) \leq \mathsf{cw}(S(C_j))$*; then* $S(T) := rS(C_1)\cdots S(C_\ell)$*.*

For non-singleton trees T, each of the sub-sequences $S(C_i)$ has $\ell - i$ "overarching" edges from the root to the roots of later C_j, $j > i$ (see also Figure 5). Therefore, if the root of T has ℓ children, $\mathsf{cw}(S(T)) = \max(\{\ell\} \cup \{\mathsf{cw}(S(C_i)) + \ell - i \mid 1 \leq i \leq \ell\})$. It is easy to compute $S(T)$ and $\mathsf{cw}(S(T))$ in a bottom-up fashion. To do this in small steps, after computing $S(C_i)$ and $\mathsf{cw}(S(C_i))$ for all child trees C_i of a vertex r, we can add the children C_i to r one by one, in an order of non-decreasing cutwidth. The following lemma is the key to showing that this recursive procedure yields, for each partial subtree (with only some child trees added yet), the exact cutwidth. Its proof can be found in the extended version of the paper [25].

Lemma 1. *For* $i \in \{1, 2\}$*, let* T_i *be a tree with root* r_i *and cutwidth* $w_i = \mathsf{cw}(T_i) = \mathsf{cw}(S_i)$ *for an optimal serialization* S_i*. Assume that* $C_1, \dots, C_\ell$ *are the direct child trees below* r_1 *in an order of non-decreasing cutwidth, and that* $w_2 \geq \mathsf{cw}(C_i)$ *for all* $1 \leq i \leq \ell$*. Then the tree* T *obtained from* T_1 *by adding* T_2 *as an additional direct child tree below* r_1 *has cutwidth* $\mathsf{cw}(T) = \max(w_1 + 1, w_2)$*, and an optimal serialization is* $S_1 S_2$*.*

Now we can perform an easy induction over the structure of T to show that the recursive computation of $\mathsf{cw}(S(T))$ produces $\mathsf{cw}(T)$. The induction base are single-node trees (leafs), and the step is Lemma 1.

Theorem 1. *For trees* T*,* $\mathsf{cw}(T) = \mathsf{cw}(S(T))$*, which can thus be computed in polynomial time.*

[2]This edge direction is more intuitive for this section, but our results are readily applied to proof trees with edges oriented the other way (see Figures 4 and 5). Indeed, directed cutwidth is the same for the dual ordering.

5. Implementation

We now describe the encoding of the calculi into existential rules with stratified negation in Nemo syntax, and the subsequent translation of rule-based reasoning traces into DL proofs. We have implemented this in the DL explanation library Evee.[3]

5.1. Existential Rules

The implementation of the Elk calculus in existential rules with stratified negation is split into three stages [16]. The first stage consists of RDF import and normalization. Here, the input OWL ontology is translated into facts over several predicates with the prefix `nf`. These facts serve as the input for the calculus, e.g., `nf:subClassOf(A,B)` for $A \sqsubseteq B \in \mathcal{T}$.[4] The normalization takes advantage of the OWL RDF encoding, which already contains a blank node for each complex concept that can be used as an auxiliary concept name identifying that concept. For example `nf:exists(?C,?R,?D)` states that `?C` is an auxiliary concept name representing the existential restriction with role `?R` and filler concept `?D`. All such auxiliary concept names are marked by the predicate `auxClass`, which allows identifying the named concepts occurring in the input ontology using the rule `nf:isMainClass(?X) :- class(?X), ~auxClass(?X) .` Auxiliary role names are treated similarly. Since there can be multiple blank nodes denoting the same complex concept, there are additional existential rules that create unique representatives for these concepts, in order to avoid redundant computations. Additionally, the normalization stage precomputes the role hierarchy $\sqsubseteq^*_{\mathcal{T}}$ in the predicate `nf:subProp`.

The second stage consists of the actual inference rules of the Elk calculus, which derive additional facts over predicates with the prefix `inf`. Due to the normalization, the implementation of these rules is straightforward and close to the original calculus, for example for the rule $\mathsf{R}_{\sqsubseteq}$:

```
inf:subClassOf(?C,?E) :- inf:subClassOf(?C,?D), nf:subClassOf(?D,?E) .
```

In the third stage, the interesting entailments, namely subsumptions between concept names, are extracted with the help of the `inf:subClassOf` and `nf:isMainClass` predicates.

Modifications. We slightly adapted these rules for our purposes. All modifications are confined to the normalization and extraction; the inference rules of the calculus are not affected.

To the normalization stage, we added an inference rule that derives $\bot \sqsubseteq A$ for each concept name A in the input, which ensures that $C \sqsubseteq A$ is inferred for any concept C with $\mathcal{T} \models C \sqsubseteq \bot$. Such reasoning with the $\bot$-concept is handled outside the calculus of Kazakov et al. [5], we explicitly included it to obtain a complete reasoner for classification.

Next, we updated the precomputation of the role hierarchy, which is recursively computed in a top-down manner, starting with the largest role in a set of connected role inclusions, rather than in a bottom-up fashion (as in the original rules).

```
directSubProp(?R,?S) :- TRIPLE(?R,rdfs:subPropertyOf,?S) .
nf:subProp(?R,?R) :- nf:exists(?C,?R,?D), nf:isSubClass(?C) .
nf:subProp(?R,?T) :- directSubProp(?R,?S), nf:subProp(?S,?T) .
```

[3]See https://github.com/de-tu-dresden-inf-lat/evee/tree/nemo-extractor/evee-nemo-proof-extractor
[4]More precisely, concept names are identified by IRIs, e.g., `<http://example.org/iriA>`, but we omit them here.

For example, starting from $r \sqsubseteq s$, $s \sqsubseteq t$, $t \sqsubseteq u$, we first derive $s \sqsubseteq u$ and then $r \sqsubseteq u$. This modification is necessary, because the original implementation did not compute the role hierarchy correctly. It started from roles r occurring negatively in $\mathcal{T}$ (using `nf:isSubClass` as above), but only computed the role hierarchy *above* r, the opposite of what is needed by $\mathsf{R}^{+}_{\exists}$.

We also added support for transitivity and domain axioms by introducing two rules that translate $\mathsf{domain}(r, C)$ into $\exists r.\top \sqsubseteq C$ and $\mathsf{trans}(r)$ into $r \circ r \sqsubseteq r$. Moreover, we support the current OWL 2 encoding of role chains, i.e., `owl:propertyChainAxiom`, including role chains with more than two role names. Finally, we added the possibility to prove equivalence axioms $C \equiv D$ directly from $C \sqsubseteq D$ and $D \sqsubseteq C$.

Other Calculi. The same normalization is also utilized in the implementation of the Textbook and Envelope calculi, but for those we added the additional normalization predicates `nf:subClassConj(?C,?D,?E)` for $C \sqcap D \sqsubseteq E$, `nf:subClassEx(?R,?C,?D)` for $\exists r.C \sqsubseteq D$, and `nf:supClassEx(?C,?R,?D)` for $C \sqsubseteq \exists r.D$, according to the normal forms expected by these calculi. The main inference rules of both calculi are implemented as expected, for example

```
inf:subClassOf(?C,?F) :- inf:supClassEx(?C,?R,?D),
  inf:subClassOf(?D,?E), nf:subProp(?R,?S), inf:subClassEx(?S,?E,?F) .
```

for $\mathsf{CR5}'$ from the Textbook calculus, and

```
R(?C,?R,?E) :- S(?C,?D), nf:supClassEx(?D,?R,?E)
```

for CR3 from the Envelope calculus, where we used the original names R and S [2].

5.2. Translation to OWL Proofs

Nemo allows us to obtain a *trace* of how a particular entailment, e.g., `inf:subClassOf(A,B)`, was obtained using the rules. Such a trace is similar to a DL proof, but the vertices are labeled with ground facts instead of DL axioms. In the Java class `NemoProofParser`, we implemented a translation from such traces into valid DL proofs, which reads a Nemo trace in JSON format into an `IProof<String>` object, where each `String` represents a fact, and outputs an `IProof<OWLAxiom>` object[5] that can be further processed or again saved in JSON format.

The abstract class `AbstractAtomParser`, instantiated for each of the three calculi, translates facts into DL axioms, e.g., `inf:subClassOf(A,B)` into $A \sqsubseteq B$ and `nf:supClassEx(A,r,B)` into $A \sqsubseteq \exists r.B$. Any fact that does not have a corresponding DL axiom, e.g., `inf:init(A)`, is translated into $\bot \sqsubseteq \top$, which indicates that the atom is skipped and will not appear in the final proof (see Section 2). In particular, traces contain many normalization steps that use various auxiliary predicates, which do not show up in the final DL proof.

Auxiliary concept names that are represented by blank nodes (either from the RDF encoding or from existential rules) need special treatment, since they would not make sense in the translated proof. Instead, we replace them by the concepts they denote. Since a fact containing a blank node, such as `inf:subClassOf(A,_:61)`, does not contain information about the complex concept the blank node identifies, we need to collect other relevant facts from the trace.

[5] `OWLAxiom` is part of the Java OWL API [26].

For example, `nf:exists(_:61,t,D)` encodes that `_:16` stands for $\exists t.D$. If `D` is also a blank node, then we have to recursively consider more facts until we can construct the final concept.

However, this approach does not work is for complex role inclusions with auxiliary role names. For example, if $r \circ s \circ t \sqsubseteq u$ is normalized into `nf:subPropChain(r,s,_:5)` and `nf:subPropChain(_:5,t,u)`, we cannot simply replace `_:5` by $r \circ s$, as this would yield $r{\circ}s \sqsubseteq r{\circ}s$, which is not expressible in OWL due to $r{\circ}s$ on the right-hand side. Still, this axiom is a tautology, and can be omitted without affecting the correctness of the inference. During reasoning, auxiliary roles can appear in existential restrictions, e.g., in `inf:supClassEx(C,_:5,D)`, which we translate into multiple nested existential restrictions: $C \sqsubseteq \exists r.\exists s.D$.

Lastly, after translating the inference steps in the described manner, we minimize the size of the resulting OWL proof using the `MinimalProofExtractor` class, which eliminates redundant inferences and unnecessary tautologies [22]. In particular, this removes all occurrences of $\bot \sqsubseteq \top$. As a result, we obtain a correct and complete DL proof from the NEMO reasoning trace.

6. Evaluation

We compared the proofs resulting from the three calculi on a benchmark of 1,573 reasoning tasks that were extracted from the ORE 2015 ontologies [17, 11]. Each reasoning task consists of a justification and the entailed axiom, which avoids the overhead of having to deal with a large ontology and lets us focus on the structure of the inference steps used to prove the entailments. The benchmark covers all types of entailments that hold in the ORE ontologies, modulo renaming of concept and role names (and some timeouts). The resulting proofs and measurements can be found on Zenodo [27]. The extended version of the paper [25] contains detailed graphs for pairwise comparisons of the calculi.

From the structure of the calculi, we expected proofs of the TEXTBOOK calculus to be less linear and shallower, i.e., have larger directed cutwidth, larger bushiness score and smaller depth, because it does not restrict any premises of the inference rules to be from the input TBox (see also Figure 5), and therefore allows for more balanced proof trees. This was confirmed in the experiments, with the directed cutwidth of TEXTBOOK proofs being higher in 1,486 and 1,381 cases compared to ELK and ENVELOPE, respectively, and never lower. Similarly, the bushiness score is higher for 1,534 and 1,512 proofs, and lower in only 12 and 23 cases, respectively. Conversely, the depth is lower for 1,503 proofs, and higher in only 8 cases, compared to both other calculi. We conjecture that the depth of the TEXTBOOK proofs is approximately logarithmic in the depth of the corresponding ELK or ENVELOPE proofs, and that the relationship for directed cutwidth and bushiness score is exponential. On the same three measures, the ELK and ENVELOPE proofs behave very similarly, with ENVELOPE proofs having slightly higher directed cutwidth and bushiness score, but nearly identical depth, compared to the ELK proofs.

We also compared directed cutwidth and bushiness score, since they both try to measure how linear proofs are. We observe that there is indeed a correlation between them; however, whereas directed cutwidth is always a natural number, bushiness score allows a more fine-grained comparison of proofs, and also tends to increase faster than the directed cutwidth.

For the remaining measures of size and average step complexity, the ELK calculus is the outlier, with both lower size (in 1,025 and 584 cases compared to ENVELOPE and TEXTBOOK, respectively) and lower average step complexity (1,281 and 1,062 proofs, respectively). Here, Envelope and Textbook obtain very similar results, with slightly lower values for Textbook.

There is no clear winner across all the measures we considered. However, depending on specific use cases, proofs generated using certain calculi may be more preferable. Overall, proofs generated with the Elk calculus seem to be the better choice for providing explanations of entailments, since they are smaller and rely on simpler inferences. Additionally, Elk proofs have lower cutwidth and bushiness scores, indicating that axioms are used as soon as possible in the proof, which can be an advantage when reading the proofs in a linear format (see Figure 5). Conversely, the low depth of Textbook proofs may be a better option for other types of visualizations, since it can reduce the amount of vertical or horizontal scrolling required, which also allows users to inspect the proof in a more linear manner. Our experiment do not show a specific advantage for proofs generated using the Envelope calculus over the other two.

6.1. Limitations

The benchmarks are not fully representative of general $\mathcal{EL}_{\bot}^{+}$ proofs, mainly for two reasons. First, the benchmark does not contain all possible justifications and entailments from the ORE 2015 ontologies, since for some cases it was too costly to compute all justifications [11]. Second, Nemo always computes only one trace, even if there are different combinations of inference steps that result in the same axiom. Since this is done by a specific algorithm in Nemo, there is a systematic bias in the resulting proofs. For example, for the Textbook calculus, instead of very bushy proofs, Nemo could also have returned more linear proofs, as for the other calculi. In this case, Nemo traces seem to be biased towards smaller depth, which allows us to observe the differences between the calculi and confirms our initial intuitions. In future work, we plan to reevaluate this once Nemo is extended to consider all possible traces rather than just one.

7. Conclusion

We compared the proofs obtained from three reasoning calculi for the $\mathcal{EL}$ family of DLs. This was facilitated by Nemo, a powerful rule engine that allowed us to quickly implement the calculi without having to develop a dedicated reasoner. As expected, the Textbook calculus [13] indeed produces more bushy and more shallow proofs, and it turns out that the Elk calculus [5] generally yields smaller proofs whose inference steps are on average less complex [19]. This enables us to choose specific calculi for different purposes, e.g., to show proofs in a visualization format where the screen space is restricted either horizontally or vertically, or when the goal is to be able to understand individual inference steps more quickly. In the future, we want to apply this method to consequence-based calculi for more expressive logics, e.g., $\mathcal{ALC}$ and beyond.

Acknowledgments

This work was supported by DFG in grant 389792660 (TRR 248: CPEC, https://perspicuous-computing.science), and by BMFTR and DAAD in project 57616814 (SECAI, School of Embedded and Composite AI).

References

[1] S. Brandt, Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and—what else?, in: R. L. de Mántaras, L. Saitta (Eds.), Proceedings of the 16th European Conference on Artificial Intelligence (ECAI'04), IOS Press, 2004, pp. 298–302. URL: https://www.frontiersinai.com/ecai/ecai2004/ecai04/p0298.html.

[2] F. Baader, S. Brandt, C. Lutz, Pushing the EL envelope, in: L. P. Kaelbling, A. Saffiotti (Eds.), Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI'05), Professional Book Center, 2005, pp. 364–369. URL: http://ijcai.org/Proceedings/05/Papers/0372.pdf.

[3] D. Tena Cucala, B. Cuenca Grau, I. Horrocks, Consequence-based reasoning for description logics with disjunction, inverse roles, number restrictions, and nominals, in: J. Lang (Ed.), Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI'18), ijcai.org, 2018, pp. 1970–1976. doi:10.24963/IJCAI.2018/272.

[4] D. Tena Cucala, B. Cuenca Grau, I. Horrocks, 15 years of consequence-based reasoning, in: C. Lutz, U. Sattler, C. Tinelli, A. Turhan, F. Wolter (Eds.), Description Logic, Theory Combination, and All That: Essays Dedicated to Franz Baader on the Occasion of His 60th Birthday, volume 11560 of *LNCS*, Springer, 2019, pp. 573–587. doi:10.1007/978-3-030-22102-7_27.

[5] Y. Kazakov, M. Krötzsch, F. Simančík, The incredible ELK: From polynomial procedures to efficient reasoning with $\mathcal{EL}$ ontologies, J. Autom. Reason. 53 (2014) 1–61. doi:10.1007/s10817-013-9296-3.

[6] A. Steigmiller, T. Liebig, B. Glimm, Konclude: System description, J. Web Semant. 27-28 (2014) 78–85. doi:10.1016/J.WEBSEM.2014.06.003.

[7] A. Bate, B. Motik, B. Cuenca Grau, D. Tena Cucala, F. Simančík, I. Horrocks, Consequence-based reasoning for description logics with disjunctions and number restrictions, J. Artif. Intell. Res. 63 (2018) 625–690. doi:10.1613/JAIR.1.11257.

[8] Y. Kazakov, P. Klinov, Goal-directed tracing of inferences in EL ontologies, in: P. Mika, T. Tudorache, A. Bernstein, C. Welty, C. A. Knoblock, D. Vrandečić, P. Groth, N. F. Noy, K. Janowicz, C. A. Goble (Eds.), Proceedings of the 13th International Semantic Web Conference (ISWC'14), volume 8797 of *LNCS*, Springer, 2014, pp. 196–211. doi:10.1007/978-3-319-11915-1_13.

[9] M. Horridge, B. Parsia, U. Sattler, Justification oriented proofs in OWL, in: P. F. Patel-Schneider, Y. Pan, P. Hitzler, P. Mika, L. Zhang, J. Z. Pan, I. Horrocks, B. Glimm (Eds.), Proceedings of the 9th International Semantic Web Conference (ISWC'10), volume 6496 of *LNCS*, Springer, 2010, pp. 354–369. doi:10.1007/978-3-642-17746-0_23.

[10] S. Schlobach, Explaining subsumption by optimal interpolation, in: J. J. Alferes, J. A. Leite (Eds.), Proceedings of the 9th European Conference on Logics in Artificial Intelligence (JELIA'04), volume 3229 of *LNCS*, Springer, 2004, pp. 413–425. doi:10.1007/978-3-540-30227-8_35.

[11] C. Alrabbaa, F. Baader, S. Borgwardt, P. Koopmann, A. Kovtunova, Finding small proofs for description logic entailments: Theory and practice, in: E. Albert, L. Kovács (Eds.), Proceedings of the 23rd International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR'20), volume 73 of *EPiC Series in Computing*, EasyChair,

2020, pp. 32–67. doi:10.29007/nhpp.

[12] C. Alrabbaa, S. Borgwardt, T. Friese, A. Hirsch, N. Knieriemen, P. Koopmann, A. Kovtunova, A. Krüger, A. Popovic, I. S. R. Siahaan, Explaining reasoning results for OWL ontologies with Evee, in: P. Marquis, M. Ortiz, M. Pagnucco (Eds.), Proceedings of the 21st International Conference on Principles of Knowledge Representation and Reasoning (KR 2024), 2024. doi:10.24963/KR.2024/67.

[13] F. Baader, I. Horrocks, C. Lutz, U. Sattler, An Introduction to Description Logic, Cambridge University Press, 2017. URL: http://dltextbook.org/. doi:10.1017/9781139025355.

[14] M. Krötzsch, Efficient rule-based inferencing for OWL EL, in: T. Walsh (Ed.), Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI'11), IJCAI/AAAI, 2011, pp. 2668–2673. doi:10.5591/978-1-57735-516-8/IJCAI11-444.

[15] D. Carral, I. Dragoste, M. Krötzsch, Reasoner = logical calculus + rule engine, Künstliche Intell. 34 (2020) 453–463. doi:10.1007/S13218-020-00667-6.

[16] A. Ivliev, L. Gerlach, S. Meusel, J. Steinberg, M. Krötzsch, Nemo: Your friendly and versatile rule reasoning toolkit, in: P. Marquis, M. Ortiz, M. Pagnucco (Eds.), Proceedings of the 21st International Conference on Principles of Knowledge Representation and Reasoning (KR'24), 2024. doi:10.24963/KR.2024/70.

[17] B. Parsia, N. Matentzoglu, R. S. Gonçalves, B. Glimm, A. Steigmiller, The OWL reasoner evaluation (ORE) 2015 competition report, J. Autom. Reason. 59 (2017) 455–482. doi:10.1007/S10817-017-9406-8.

[18] H. L. Bodlaender, M. R. Fellows, D. M. Thilikos, Derivation of algorithms for cutwidth and related graph layout parameters, J. Comput. Syst. Sci. 75 (2009) 231–244. doi:10.1016/J.JCSS.2008.10.003.

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

[20] Y. Kazakov, P. Klinov, A. Stupnikov, Towards reusable explanation services in Protege, in: A. Artale, B. Glimm, R. Kontchakov (Eds.), Proceedings of the 30th International Workshop on Description Logics (DL'17), volume 1879 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2017. URL: http://ceur-ws.org/Vol-1879/paper31.pdf.

[21] J. Méndez, C. Alrabbaa, P. Koopmann, R. Langner, F. Baader, R. Dachselt, Evonne: A visual tool for explaining reasoning with OWL ontologies and supporting interactive debugging, Computer Graphics Forum (2023). doi:https://doi.org/10.1111/cgf.14730.

[22] C. Alrabbaa, F. Baader, S. Borgwardt, P. Koopmann, A. Kovtunova, Finding good proofs for description logic entailments using recursive quality measures, in: A. Platzer, G. Sutcliffe (Eds.), Proceedings of the 28th International Conference on Automated Deduction (CADE'21), volume 12699 of *LNCS*, Springer, 2021, pp. 291–308. doi:10.1007/978-3-030-79876-5_17.

[23] C. Alrabbaa, F. Baader, S. Borgwardt, P. Koopmann, A. Kovtunova, On the complexity of finding good proofs for description logic entailments, in: S. Borgwardt, T. Meyer (Eds.), Proceedings of the 33rd International Workshop on Description Logics (DL 2020), volume 2663 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2020. URL: http://ceur-ws.org/Vol-2663/paper-1.pdf.

[24] M. Yannakakis, A polynomial algorithm for the min-cut linear arrangement of trees, J. ACM 32 (1985) 950–988. doi:10.1145/4221.4228.

[25] C. Alrabbaa, S. Borgwardt, P. Herrmann, M. Krötzsch, The shape of $\mathcal{EL}$ proofs: A tale of three calculi (extended version), 2025. doi:10.48550/arXiv.2507.21851.

[26] M. Horridge, S. Bechhofer, The OWL API: A Java API for OWL ontologies, Semantic Web 2 (2011) 11–21. doi:10.3233/SW-2011-0025.

[27] C. Alrabbaa, S. Borgwardt, P. Herrmann, M. Krötzsch, The shape of EL proofs: A tale of three calculi - DL25 - Resources, 2025. doi:10.5281/zenodo.16320822.