DOI: 10.1111/cgf.14730

COMPUTER GRAPHICS *forum*
Volume 0 (2023), number 0 pp. 1–15

EVONNE: A Visual Tool for Explaining Reasoning with OWL Ontologies and Supporting Interactive Debugging

J. Méndez,[1] C. Alrabbaa,[2] P. Koopmann,[2] R. Langner,[1] F. Baader[2] and R. Dachselt[1,3,4]

[1]Interactive Media Lab Dresden, TU Dresden, Dresden, Germany
julian.mendez2@tu-dresden.de, {langner, dachselt}@acm.org
[2]Institute of Theoretical Computer Science, TU Dresden, Dresden, Germany
{christian.alrabbaa, patrick.koopmann, franz.baader}@tu-dresden.de
[3]Cluster of Excellence Physics of Life, TU Dresden, Dresden, Germany
[4]Centre for Tactile Internet with Human-in-the-Loop (CeTI), TU Dresden, Dresden, Germany

Abstract

OWL is a powerful language to formalize terminologies in an ontology. Its main strength lies in its foundation on description logics, allowing systems to automatically deduce implicit information through logical reasoning. However, since ontologies are often complex, understanding the outcome of the reasoning process is not always straightforward. Unlike already existing tools for exploring ontologies, our visualization tool Evonne *is tailored towards explaining logical consequences. In addition, it supports the debugging of unwanted consequences and allows for an interactive comparison of the impact of removing statements from the ontology. Our visual approach combines (1) specialized views for the explanation of logical consequences and the structure of the ontology, (2) employing multiple layout modes for iteratively exploring explanations, (3) detailed explanations of specific reasoning steps, (4) cross-view highlighting and colour coding of the visualization components, (5) features for dealing with visual complexity and (6) comparison and exploration of possible fixes to the ontology. We evaluated* Evonne *in a qualitative study with 16 experts in logics, and their positive feedback confirms the value of our concepts for explaining reasoning and debugging ontologies.*

Keywords: interaction, visualization

1. Introduction

OWL ontologies as a means to formalize terminology have applications in medicine [IB14], biology [HSG15], the semantic web [Hor08] and many other areas. Realistic ontologies often provide formalizations of up to hundreds of thousands of concepts, examples including the medical ontology SNOMED CT [LdKLC13] and the Gene Ontology GO [Gen04]. Using a formalization in *OWL* (Web Ontology Language) [HKP*09] allows OWL reasoners to compute implicit information through automated reasoning. Due to the sheer size, and non-trivial nature of the interactions between the logical statements (*axioms*) in an ontology, the output of the automated reasoning process can be unexpected. For ontology engineers, it is thus vital to understand *why* an entailment inferred through logical reasoning follows from the ontology, and how it can be repaired in case it is wrong. Moreover, it is important to understand the role of the involved axioms within the greater context of the ontology if an engineer intends to repair it.

To this end, entailments can be explained using *justifications* (minimal sets of axioms responsible for an entailment [SC03, Hor11]), or *proofs* (series of reasoning steps to reach an entailment from a justification), which can be produced using different approaches [ABB*20]. Research in this area generally focuses on *computational complexity*, *soundness*, *completeness* and *efficiency* of computing such proofs [ABB*21, KKS14, KK15]. However, aspects like *interactivity* and *presentation* of proofs have received less attention. In this work, we address these visualization aspects with the goals of facilitating the understanding of entailments and supporting debugging by showing the *impact* of the potential fixes on the ontology.

J. Méndez and C. Alrabbaa contributed equally to this work.

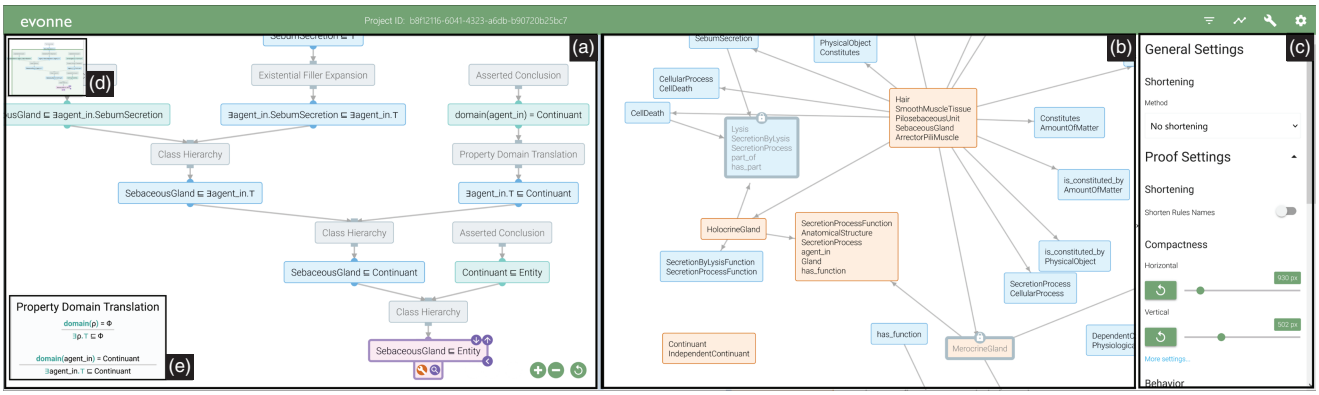

Figure 1: *Overview of* Evonne *in split view: (a) Proof View, (b) Ontology View, (c) sidebar where settings and diagnoses for debugging are shown, (d) minimap feature for the proof view and (e) Rule Explanation tooltip, showing a property domain translation.*

We present the latest version of our tool Evonne: a web application for explaining reasoning through visualizations of proofs and ontologies. Evonne uses specialized views for exploring both *explanations of logical entailments* (as proofs) and the *knowledge from which these entailments are obtained* (in the ontologies). In particular, we offer innovative layout modes for proofs that aim to help overcome the cognitive complexity of the tasks surrounding them, while also exploiting the connection with the ontology to visually support the understanding and repairing of ontologies. For instance, we introduce the *Magic Mode* (bidirectional layout), a specialized approach for exploring proof trees, which allows a true combination of the typical reading directions by granting users the flexibility to manipulate the structure of proofs while maintaining their semantic integrity. For the ontology content, we use a visualization based on the *atomic decomposition* (AD) [VHP*20], instead of using the typical *subsumption hierarchy* [DLSP18], and support debugging via exploration of *diagnoses* [ABD*20].

Some early concepts for Evonne were discussed in Refs. [ABD*20, FLAD20]. Here, we present (1) the design rationale for the proof layouts, diagnoses exploration, and visual encoding; (2) further concepts such as the linear and bidirectional layouts, as well as filtering options for diagnoses, hiding known parts of the proofs, *etc*.; (3) the finalized realization of our concepts to date including several settings that accommodate user preference and expertise and (4) an evaluation of our approaches in a qualitative user study with 16 logic experts, where we collected information about their perception of our concepts, as well as qualitative feedback about the value of our tool, and their user experience as a whole. The positive reception and assessment of our work confirms our assumptions about usefulness and usability, while the constructive feedback also reveals future areas of work to explore.

2. Background

In this section, we provide an overview of the related work on visualizations for ontology engineering tasks, recall notions from description logics (DLs) that are relevant to Evonne, and give an overview of the challenges involved in this work.

2.1. Visualization of ontologies

The current tool of choice for ontology visualization and editing is Protégé [Mus15] or its web-based alternative WebProtégé [HGN*19] that allows collaborative editing. By default, Protégé provides an indented list of the concepts formalized in the ontology, and additional textual views of the metadata for each. More specialized views are developed as plug-ins, which can be found through the *Protégé Plugin Library* https://protegewiki.stanford.edu/wiki/Protege_Plugin_Library. Most of the tools visualize the whole ontology using its concept or subsumption hierarchy. In the following, we refer to both plug-ins and other standalone tools indistinctly.

Fu *et al.* [FNS13] compared indented trees and graphs (the typical representations of ontologies), and found that the former are more familiar to novices, while the latter are more intuitive and controllable. OWLViz [Hor] and OntoGraf [Fal] visualize ontologies using *node-link diagrams*, but they struggle with large ontologies and have limited interaction possibilities. *Force-directed layouts* are used by, *e.g*. WebVOWL [WLA18] to make efficient use of the screen space, but the large size of ontologies, which continues growing, remains an unsolved issue. On the other hand, Jambalaya [SMS*01] and OWL-VisMod [GPNT12] use treemaps, but these are typically not easy to navigate and clutter quickly when relations between the ontology concepts are shown on superposed views. OWLEasyViz [CSM09] represents concepts as ellipses, allowing the user to reveal their sub-concepts by clicking on these ellipses. This approach, however, is costly in terms of interaction, and becomes disorienting as more elements come into view. Lastly, and most relevant to our setting, is the work on explanation services. In particular, there is the *proof explanation plug-in* [KKS17], which shows proofs of entailments as *indented trees*. In the case of large proofs, this visualization can lead to more cognitive load for users, *e.g*. when keeping track of related proof statements that occur on different branches.

For ontology editing, typical diagram editors for VOWL and UML notations are used, as is the case with crowd [BGCF20] and UnSHACLed [LDMH*21]. The same scalability issues mentioned earlier are present here, since always the whole ontology is loaded

into these editors. Furthermore, these are general editing tools, and do not provide guidance for users to find errors. In addition, there are tools for supporting other aspects of ontology engineering such as: (i) *debugging inconsistencies* (*e.g.* SWOOP [PSK05, KPSG06] and REPAIRTAB [TSP*08]), (ii) *inspecting information loss* (*e.g.* INFERENCE INSPECTOR [MVJS18] for authoring, and CHIMP [PSDB20] for quantifying the impact of changes) and (iii) *education and training* (*e.g.* SIVA [PPH18] a tool for simulating reasoning by showing a step-by-step application of a *tableau algorithm*). However, these tools mostly use *textual* visualizations.

In order to support exploration and understanding of the underlying structures (in our case, of proofs and ontologies), interaction plays a crucial role [LIRC12, EMJ*11, TS20]. In EVONNE, we use techniques for exploring trees [LPP*06, TM03, RBB02] and node-link diagrams [CC07, CGH*19, MB19] that are well-known individually, but had to be adapted to suit our constraints. We also employ a tailored version of general-use techniques such as *linking and brushing* across multiple coordinated views [BC87, Rob07] and *overview + detail* [CKB09, Mun14].

2.2. Notions from description logics

OWL is based on DLs, a family of logics targeted at describing concepts and relationships between them. In DL context, a *concept* is a syntactical entity that describes a class of objects (*e.g.* vehicles, genomes, *etc.*) by combining basic terms (concept and role *names*) using the logical connectives of the DL. A *DL ontology* is then a set of *axioms*, which express relations between concepts (*e.g.* containment, equivalence, disjointness). Different DLs differ in the set of operators that can be used to describe a concept or an axiom. The pizza ontology [RDH*] is a toy ontology commonly used for teaching OWL and DLs. In such an ontology, the following axiom in the DL $\mathcal{EL}$ states that every pizza has a topping and a base:

$$\mathsf{Pizza} \sqsubseteq \exists \mathsf{hasTopping}.\mathsf{PizzaTopping} \sqcap \exists \mathsf{hasBase}.\mathsf{PizzaBase}$$

Here, Pizza, PizzaTopping and PizzaBase are concept names describing sets of objects, and hasTopping and hasBase are role names describing relations between those objects. Details on the syntax and semantics of DLs and OWL can be found in Refs. [BHLS17, HKP*09], respectively. A DL *reasoner* uses automated deduction to compute information that logically follows from the axioms in an ontology [GHM*14, SLG14]. For instance, it could infer that a Margherita is a VegetarianPizza by using the definitions of Margherita, VegetarianPizza, and all involved ingredients in the pizza ontology. A typical deduction task is *classification*, which computes all logically entailed axioms of the form $A \sqsubseteq B$, where A and B are concept names, stating that every A is also a B (*e.g.* for $A =$ Margherita and $B =$ Vegetarian). Classification, thus, computes the implied *subsumption hierarchy* between the concepts defined in the ontology, which is the structure shown (using, *e.g.* indented trees, as mentioned in Section 2.1) in most tools that visualize ontologies.

Proofs and justifications: Currently, the most common way of explaining an entailed axiom α is to use *justifications*, which are minimal sets of axioms from the ontology that are sufficient for inferring α [SC03, Hor11]. In contrast, a *proof* for α is a hypertree describing logical inference steps, where leaves are labelled with axioms from the ontology, internal nodes with derived axioms, and the root corresponds to α [ABB*21]. The hyper-edges describe the *inference steps*, stating how one axiom is derived from other axioms. An example of such a proof can be seen in Figure 1(a). The advantage of using proofs rather than justifications is that they do not only show the ontological knowledge (axioms) that is used to infer the entailment, but also show in detail how these axioms are interacting with one another to create the entailment to be explained.

Diagnoses and repairs: While a proof explains why an axiom is entailed by an ontology, a *diagnosis* describes how to eliminate the entailment (in case it is erroneous). Specifically, for a given ontology $\mathcal{O}$ and an entailed axiom α, a *diagnosis for α in $\mathcal{O}$* is a minimal set of axioms such that removing them would break the entailment of α. The result of removing these axioms is then called a *repair* [MMV11]. EVONNE supports users in finding the right diagnosis. Outside of EVONNE, they may then choose to remove the axioms in the diagnosis, or to modify them appropriately.

Modules and atomic decompositions: Different to other tools for ontologies, EVONNE makes use of the *modular structure* of an ontology. *Modules* [GHKS07] are subsets of the ontology that can be used to organize its content. A module is specified by a *signature*—a set of (concept or role) names—and contains all axioms of the ontology that are needed for the entailment of axioms that only use those names. There exist standard tools for computing such modules automatically [HB11, KLWW08, KC20]. The set of all possible modules for an ontology is captured by the AD, which consists of a partitioning of the axioms in the ontology into *atoms*, together with an acyclic *dependency relation* between those atoms [VHP*20, HMP*14]. Every module of the ontology can be constructed by taking the union over a set of atoms and all atoms reachable from them via the dependency relation. Intuitively, the dependency relation expresses how the atoms influence each other in specifying knowledge about the different subsets of the terminology defined by the ontology, which can be used to visualize the impact of repair options [ABD*20].

When representing the AD graphically, atoms can be represented more concisely using signatures. We here use a technique first introduced by Alrabbaa *et al.* [ABD*20]. An atom that is not dependent on other atoms is also a module, and is assigned the signature containing all names in that module. If an atom depends on other atoms, we assign to it the names that occur in the corresponding module, excluding the names that occur in the atoms it depends on. Intuitively, its signature then refers to the new terminology that its axioms introduce. It is possible for the signature assigned to an atom in this way to be empty, in which case the atom does not introduce new terminology, but only connects the terminologies of other atoms.

2.3. Summary of challenges

The two previous sections show, respectively, the state of the art for visualization in ontology engineering tasks and the DL notions

that are used in our work to overcome major limitations of the existing work. In Evonne, we explain entailments using proofs, and visualize the relevant part of the ontology using its AD. In particular, we represent proofs as *hypertrees*, and ADs as *directed acyclic graphs* (DAG), using node-link diagrams. In a study computing proofs for real-life ontologies from the well-known BioPortal corpus [RMKM08], the largest proof observed had 140 nodes [ABB*20]. The largest AD we computed for our real-life ontology examples has around 1624 atoms, the largest atom having 2201 axioms. Visualizing ontologies is not a typical use case of ADs [Ves13], the only exception being the relatively simple Protégé plug-in DeMoSt [DVKP*11]. The other approaches for visualizing ontologies mostly make use of the subsumption hierarchy instead. However, as argued by Alrabbaa *et al.* [ABD*20], the AD may give deeper insights in the interactions of the relevant parts of the ontology.

Our work constitutes a next step in providing explanation services that are responsive to user preferences and to their domain expertise, considering the limited support for explanations using proofs for ontology entailments. One of the main challenges of adapting visualization techniques for the domain of formal logic is to ensure that the *semantics* of the visualized data are preserved in all visual transformations. Note that, when developing solutions for these use cases, one must avoid design choices that are too far away from the current work environments of logicians. For instance, developing an augmented reality solution would be unjustified given the still limited visualization support in traditional 2D environments. This is why Evonne remains a desktop application, but provides initial features for use in multi-display environments.

3. Evonne: Interaction and Visualization Design

With Evonne, we aim to support logicians and ontology engineers in understanding proofs, as well as in debugging erroneous entailments. In this section, we describe our design rationale and specific concepts for our tool. Our development process consisted of interdisciplinary ideation sessions and discussions between experts in the fields of logics and visualization. In an iterative manner, we (1) identified needs of the DLs community, (2) collected and presented the techniques that could be used, (3) developed concepts for tailoring the general techniques to the existing needs and (4) evaluated the realization of these concepts with respect to existing workflows. As a result of this, we identified a set of design goals, which characterize our proposed visualization and interaction concepts.

3.1. Design goals

Our set of design goals (referred throughout the paper as DG-T1 to DG-G2) is distilled from our interdisciplinary conversations. Our target users are logicians and ontology engineers at various levels of expertise. Therefore, our goals address the needs of the DL community, and foster novelty in their workflows beyond the required basic visualization support. The list of goals is followed by our reasoning to define them. The goals are related to the major *Tasks* we support (DG–T), characteristics of our *Users* (DG–U) and *General* usability (DG–G) in dynamic scenarios.

DG-T1: Enable Interactive Exploration
Proofs lack visualization support and using ADs for visualization is atypical, as mentioned in Section 2. We aim to enable self-contained but also linked interactive visualizations for these.

DG-T2: Support Decision-Making
Towards debugging of ontologies through the selection of diagnoses, our tool should organize the options to help users compare, filter and select adequately.

DG-U1: Accommodate User Preferences
We want to provide configuration options, layout alternatives and different view settings to allow tasks to be completed based on individual preferences.

DG-U2: Support Different Levels of Expertise
We aim at an intuitive tool for users with less experience, that remains non-intrusive for the more experienced users. This refers to both tool and domain expertise.

DG-U3: Build on Familiar Representations
Our tool should build upon visual representations familiar to logicians (*i.e.* node-link diagrams). However, we learned that readability and the semantic structure represent the most important needs, and they must be balanced in our solution.

DG-G1: Improving Existing and Enabling Novel Workflows
Traditionally our target users have desktop environments with command-line or simple visual tools. We want to improve their work setting while also opening the possibility for new workflows which might involve collaboration, multiple devices, *etc*.

DG-G2: Minimize Setup Difficulties
We aim to provide a ready-to-use tool that is on par with the setup complexity of other state of the art tools such as WebProtégé [HGN*19], which can be executed locally or remotely due to the potentially heavy computational load.

To define the goals, we selected existing workflows for explanation of logical entailments, and translated them conceptually using modern visualization approaches. Take, for example, the process of *reading a proof*. Generally, it involves traversing the structure step by step to understand the different inferences. Depending on the length of the proof, the complexity of the axioms and inferences, personal preferences (DG-U1) and expertise (DG-U2), one may approach the reading from different directions: *(a) Top-down*: Given a conclusion, what are the premises that lead to it? *(b) Bottom-up*: Given a set of premises, what is the inferred conclusion? Or *(c)* a combination of both. The support for this workflow is reflected in DG-T1. Another example of a major workflow is the process of *debugging unwanted entailments*. Roughly speaking, given an unwanted entailment, the user needs to find out why this entailment exists, and select a diagnosis that eliminates it while affecting the ontology minimally. There are different measures to determine the impact of a diagnosis, and the number of diagnoses can be exponential in the size of the ontology. This is mainly supported by DG-T2, but DG-T1 also plays an important role here since the influence of the ontology on the entailment and vice-versa should be clear to the users. Several other minor workflows are supported by our tool, such as *learning logical rules by examples*, *comparing proofs computed by different reasoners*, *optimizing proofs w.r.t. different measures*, *etc*.

DG-U3 is applicable to all workflows, as the usage of screen space must never obscure the readability of the axioms (details) nor the semantic structure (local or global overview). In order to better support their mental process, users should be allowed to dynamically manipulate this balance, which results in sub-optimal screen space usage. While we aim to fulfil the requirements of these workflows without overwhelming the users with big changes in their workspaces, we are also interested in how modern visualization support can improve the work of the DL community. With this in mind, DG-G1 disallows design choices that are too far away from the current work environments of logicians (*e.g.* immersive technologies). Nevertheless, we want to provide features that make use of multiple screens and devices, and can be used by multiple concurrent users. Lastly, with DG-G2, we make sure that even the modern features do not complicate the setup of our solution.

3.2. Overview of the user interface and general features

Addressing these design goals, we developed an interactive visualization tool we call Evonne (see Figure 1). Its user interface consists of two major views: the *Proof View* and the *Ontology View*. Both use node-link diagrams, but they behave differently, according to their semantic structure. The look-and-feel of the tool is tailored towards desktop environments as this is the main environment of our target community (DG-U1, U2, G1). Users may want to use either only one view at a time, or both simultaneously, which can depend on preferences, technical setups or the tasks at hand. For instance, understanding an entailment is possible using only the Proof View, without considering the related ontology (DG-T1). However, using both views simultaneously (1) aims at enhancing the understanding of a proof in the context of the ontology and (2) is better suited for more complicated tasks such as repairing the ontology. Furthermore, the AD is constructed w.r.t. the entailment in the Proof View, and therefore, the linked interactions that we provide are triggered from the Proof View to inspect the context of the proof within the Ontology View. When both views are shown together, we refer to it as a *split view*, as depicted in Figure 1. Showing them separately, *e.g.* on different screens or devices, is suitable for large structures that need a lot of screen space.

Both views support independent zoom and pan navigation and include a *minimap* feature (DG-T1), helping users to navigate the details while being aware of the overall structure (see Figure 1(d)). Looking at the various spatial configurations that proofs and ontologies may need, we designed settings to adjust the visual data distribution more efficiently with the given space. These settings affect the visualizations at three levels: *textual*, *structural* and *behavioural*. For example, to deal with readability issues on both views (DG-U1, U3), we provide two methods for shortening textual elements (*e.g.* axioms and rule names), called *fixed-length* and *camel case*. The former limits the amount of characters in concepts. The latter removes lower case letters from concept names (*e.g.* "SpicyIceCream" becomes "SIC"), while role names keep the first lower case letter and otherwise are treated the same (*e.g.* "∃hasSpiciness.Hot" becomes "∃hS.H".). While fixed-length gives more control w.r.t. size, camel case might produce better recognizable labels (DG-U3).

All of the settings are located in a sidebar on the right side of the screen (Figure 1(c)). In addition, each view has more detailed configurations and specific interaction modalities that are listed in this sidebar (DG-U1). In the next two sub-sections, we will detail the *Proof View* and *Ontology View* with all their functionalities. The coloured underlines in these sub-sections refer to the colours that are used in the tool, as shown in Figures 1–5.

3.3. Proof view

The first major view is the Proof View, and it can be self-sufficient for understanding entailments (DG-T1). We indicate the used colours with underlines. As introduced in Section 2.3, we present proofs using hypertrees, where we distinguish the nodes as follows:

- *Final conclusion*: the entailment under inspection (root).
- *Asserted conclusions*: axioms from the ontology that constitute a justification of the entailment.
- *Inferred axioms*: the intermediate conclusions computed for each inference step, excluding the asserted conclusions.
- *Fixed knowledge*: (1) the DL inference rules (rule nodes) and (2) *known* content (*i.e.* nodes that hide either axioms or entire sub-proofs).

When we mention *axiom nodes*, we refer to both *asserted* and *inferred* statements (including the *root entailment*). On the other hand, when discussing *traversal*, we refer to the structure of the tree: *Top/up* refers to the root, which in our tool is located at the bottom of the view, while *bottom/down* refers to the leaves towards the top of the view. Figure 1(a) shows the *default layout* of the Proof View. Among the settings specific to the proof, we allow the users to increase or decrease the space between nodes vertically and horizontally using sliders, which is helpful to fine-tune space usage and address readability issues (DG-U3). We learned in our design conversations (and confirmed in our study) that optimal space usage was not always desired because our users often look for a balance between readability and visualizing the whole structure—or as much of it as possible. For users that immediately prefer the full structure without any type of overlap, we also provide an option to automatically distribute the nodes to avoid overlap entirely, instead of the default setting that fits the tree to the available screen space.

Furthermore, the various *DL inference rules* we show may not be trivial to understand, considering the expertise of the users and their familiarity with the naming convention we use (DG-U2). Thus, we provide *Rule Explanation* tooltips (Figure 1(e)) that show an abstract representation of a rule, and its instantiation using the axioms in the proof. Some users will have sufficient knowledge about some parts of the ontology, so that inferences from those parts do not need to be explained to them. To reflect this, we allow users to mark concept and role names as *known* by uploading a *signature file*. The axioms and sub-branches that only use these names are hidden under *"known" nodes* (DG-U2).

Hovering over axiom nodes reveals options to collapse branches and to reveal inferences step by step. An example of these buttons can be seen at the root node of the proof in Figure 1(a)). If a user wants to explore the proof starting from the *root*, the *Show Previous Step* button () can be used to gradually reveal the tree towards the *leaves*. Likewise, the user may use the *Hide All Previous* button () to mark sub-trees as checked and move from the *asserted conclusions* towards the *final conclusion*. This action can

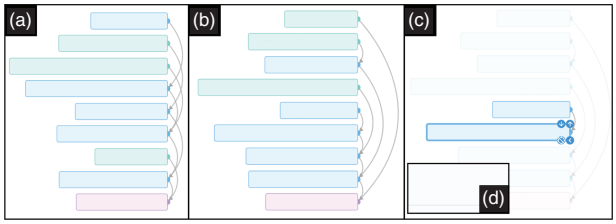

Figure 2: *Linear Proof options: (a) Shows the optimized premise distance, where edges intersect. (b) Avoids edge intersections at the cost of premise proximity. (c) Shows the effect of the Highlight inference button ( ) the premise and conclusion are highlighted, and (d) a Rule Explanation tooltip appears.*

be undone with the *Show All Previous* button ( ). In addition, users can click on an edge to *cut* a particular branch and inspect it separately. Only outgoing links from axioms may be cut to ensure that we always visualize a coherent inference tree. The resulting subtree is presented with an *indicator at its root* which can be clicked to restore the full proof. Double clicking any axiom node gives access to actions coupled to the ontology view: The justification button ( ) reveals a *justification* of the corresponding axiom, and the repair button ( ) triggers the computation of the *diagnoses* for the selected axiom. For both of these linked interactions, further details are discussed in Section 3.4, where Figure 5 depicts their effect. Besides the default tree representation of the proof which we have described so far, we designed two alternative behaviours for the layout of the tree structure: *Linear* and *Magic Mode*. We describe these layout modes in the following sections.

3.3.1. *Linear mode: vertical layout*

When writing a proof on a sheet of paper, a logician often uses a so called "linear", vertical organization of the axioms that are used to infer new entailments until the final conclusion is reached (*i.e.* one uses a line for each axiom and the inferences follow in new lines). This way of organizing information is familiar to logicians (both novice and experienced), as we learned in our interdisciplinary conversations. Inspired by that (DG-U2, U3), we provide a linear proof layout as an alternative that can be activated from the settings sidebar. The resulting layout can be seen in Figure 2. On paper, proofs are easier to follow when axioms are used soon after they have been introduced. Thus, we minimize the distance between premises by default. However, this creates many intersections between the links. We call this "optimized premise distance" (Figure 2(a)), but this can be disabled to instead optimize for planarity, which shows the links more clearly, but moves the premises further apart for some inferences (Figure 2(b)). The nodes in this mode retain the same actions as in the default tree mode (*i.e.* , and as well as the colour coding. The main difference content-wise is that there are no nodes for rule names. Instead, inferred axioms (nodes) have an additional *Highlight inference* ( ) button. This button shows the explanation of the used rule and highlights the involved axioms in the proof. This is showcased in Figure 2(c).

The linear proof can also present a partial solution to readability issues of the default proof layout at the cost of losing the visually

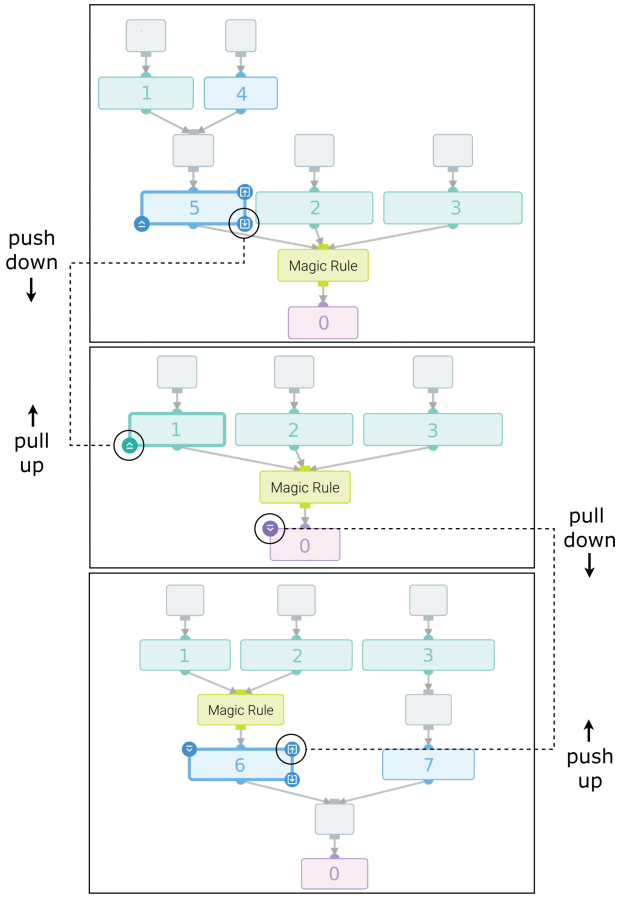

Figure 3: *Diagram of the Magic Mode actions, showing three states of the same proof. The dashed lines indicate the interaction flow of the reversible pull and push actions, triggered by clicking the circled buttons from the nodes.*

branching tree structure, which a portion of users finds preferable. Another indirect advantage of this mode is that organizing the axioms vertically leaves a lot of space for the ontology in the split view. The arrangement of the views is much simpler to achieve in this mode than with the default tree that grows horizontally, as the linear layout scales almost only vertically.

3.3.2. *Magic mode: bidirectional layout*

The reading order while traversing a proof (*i.e.* bottom-up, top-down) comes down to personal preference and the understanding process of each user. Therefore, we designed a mode that permits users to explore proofs in both directions simultaneously (DG-T1, U3). To achieve this, we allow users to expand and collapse parts of the proof—in either direction—on demand. This mode is illustrated in Figure 3. We use special nodes that represent hidden parts of (or entire) inference steps. We call these *Magic Rules*, or just *magic nodes* for simplicity. The behaviour of the rest of the nodes changes for this mode compared to the default and linear modes. From a particular axiom, users can (1) request more details about a hidden inference step by *pulling nodes out of* a magic node or (2)

hide nodes from the inferences by *pushing them into* a magic node. Our current solution uses a set of up to four buttons per node to realize these movements (pull: , , push: ,). These buttons appear on hovering, but only if the actions can be performed. This can be seen in Figure 3. The pull and push actions can also be accessed from a context menu on the node, to familiarize users with their meaning (DG-U2). They stand for (): *pull conclusion of this premise* (up), (): *pull premise(s) leading to this* (down), (): *push premise(s)* (upwards) and (): *push conclusion* (downwards). To help distinguish between these actions we (1) used distinct icons, (2) grouped the pull and push buttons at the sides of the node (pull on the left, push on the right) and (3) placed the respective buttons closest to the magic nodes that will be interacted with (up or down).

In order to always display a *semantically correct proof*, the rule nodes (both *magic* and *normal*) must always connect a set of premises to a conclusion, and thus cannot be connected to other rule nodes. As a result, the tree can change structure in potentially unexpected ways. For example, when pulling for a conclusion, another sub-tree can be consequentially revealed—affecting in total more than just one inference. Furthermore, (1) magic nodes that represent only one inference rule are automatically unravelled, (2) pushing a node when no magic nodes surround it will create a magic node and (3) pushing a node surrounded by magic nodes will merge the surrounding magic nodes into one. When a combination of these effects are triggered in chain, the resulting tree structure may not be easy to predict. This can be addressed by providing a preview of the result as an animation, and by allowing users to easily undo their inputs. For future versions, we want to investigate the feasibility of using different interactions in this mode (*e.g.* dragging, gestures). Doing so is not trivial because of the changing availability of the actions at every state of the proof and the potentially unexpected structural changes that the tree can go through.

As a last remark, the reason we chose the word "magic" to identify this mode is the quote "Magic is just science that we don't understand yet" (Arthur C. Clarke), which in our case refers more specifically to *science that we don't show yet*.

3.4. Ontology view

Commonly, ontologies are presented based on their subsumption hierarchies [DLSP18], visualized using indented lists. In our *Ontology View*, the second major view provided by EVONNE, we instead show an AD [VHP*20] using a node-link diagram (see Figure 1(b)), where the nodes correspond to the atoms and the links represent the dependency relation. As discussed by Flemisch *et al.* [FLAD20], one could argue that the more traditional approach to visualize the ontology (using subsumption hierarchies) better suits the familiarity we strive for. However, the advantages of using the AD discussed by Alrabbaa *et al.* [ABD*20] make it more precise for localizing justifications and the impact of diagnoses, which in our case is preferable (DGs T1, T2). To minimize complexity, instead of showing the entire ontology, we visualize a module for the signature of the entailment explained in the proof view. This substantially smaller subset of the ontology contains all axioms that could be used for inferring the entailment. Moreover, it is self-contained in the sense that any information expressed using the terms used in the module can be derived from the module alone. Thus, the module contains all axioms

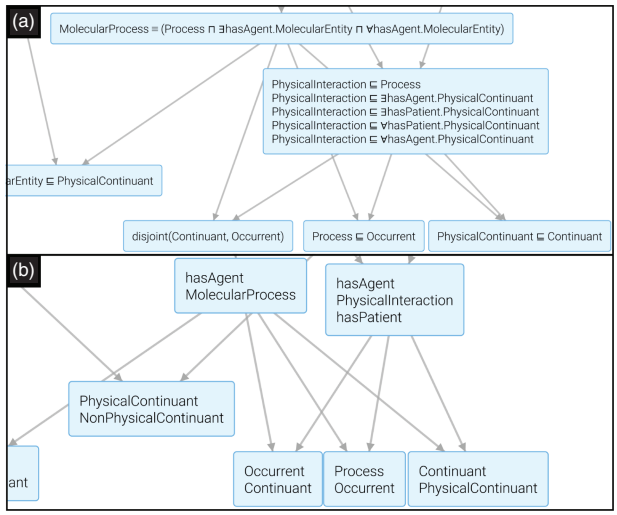

Figure 4: *Effect of the Signature Mode (Ontology View): (a) shows a set of atom nodes with full-length axioms, (b) same atom nodes with Signature Mode active.*

relevant to the entailment and possible diagnoses of it. To accommodate for the semantic and structural complexity of the ADs, we use a force-directed layout [GFV13] that makes effective use of the viewport. Additionally, we allow manipulation of the flow of the layout to create a sense of hierarchy. This is a compromise between the full hierarchical structure that could be presented in a subsumption hierarchy (DG-U3) and the benefits from our AD (DGs T1, T2).

In order to save space, the ontology view is first loaded in what we call the *Signature mode*. The effect of this mode can be seen in Figure 4(b). Instead of showing axioms in the atoms (nodes), we show a list of names, *i.e.* the signature, relevant to these atoms (see also end of Section 2.2). This mode gives an idea about what these atoms talk about without actually showing the explicit logical statements. The user can reveal the full axioms by turning off the signature mode from the settings sidebar (Figure 1(c)). Likewise, it is important to compute an initial state of the graph with minimal or no overlap, while also showing the links in reasonable proximity. From the aforementioned sidebar, the shortening options described in Section 3.2 (*fixed-length* and *camel case*) can be applied to the ontology view as well. Additionally, we provide line-wrap to limit the length of each line within a node and miscellaneous configurations for the simulation of the layout. For instance, adjusting the directional force horizontally or vertically allows the user to arrange the AD as a hierarchy and to control the space given for each depth level.

The colours in this view complement those of the Proof View:

- *Atoms* represent knowledge which was computed, similar to the *inferred axioms* from the proof.
- *Justification axioms* match the *asserted conclusions* from the proof, when highlighted using the justification button ().
- For *diagnoses*, we use contrasting colours to indicate the *impact* of replacing axioms to repair the ontology ().

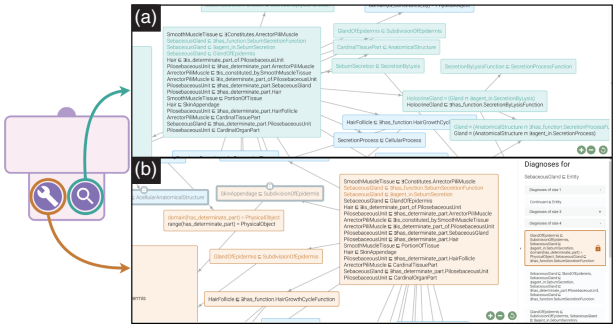

Figure 5: *Effect of the linked interactions: (a) Justification of the axiom from the Proof View on the Ontology View. (b) Preview of the impact of the selected diagnosis from the sidebar on the right.*

> Atoms and all their axioms can be marked as *trusted to be correct*. This filters out all the diagnoses that include any such axioms, extending the *fixed knowledge* concept from the Proof View.

3.5. Justifications and diagnoses

As introduced in Section 3.3, the linked interactions (*i.e.* to highlight justifications and compute diagnoses) are triggered from the axiom nodes on the Proof View.

Justifications: The (🔍) button on axiom nodes from the proof triggers the display of the justification for that axiom on the Ontology View. This justification corresponds to the *asserted conclusions* occurring in the sub-tree of the proof under the axiom. For the *root*, this would be all the *asserted axioms* that appear in the Proof View. In the Ontology View, the atom nodes are highlighted if they contain any of the axioms of the justification. In addition, these axioms are also highlighted within the nodes (see Figure 5(a)). The goal here is to help users in assembling a mental picture to understand the proof (DG-T1) and to potentially identify faulty axioms in the ontology (DG-T2), especially for large and complex ontology and proof pairs. To avoid confusion, only one justification can be highlighted at a time.

Diagnoses: The (🔧) button on each axiom node triggers the computation of diagnoses for the selected axiom, that is, sets of axioms such that removing them, or modifying them appropriately, prevents the axiom from being deducible. This computation can be costly. In fact, the expressiveness of the logic and the size of the ontology influence the computation time, which can range between milliseconds and several minutes. Once they are computed, we show the diagnoses grouped by *size* in the sidebar. The size of a diagnoses is one criterion that can give an initial insight to how much of a change the diagnosis proposes. However, it is not sufficient, because changing a small number of axioms can have a comparably larger impact on the ontology. Which diagnosis is the "correct" one is in the end something only the user can decide (DG-T2).

There are two challenges related to this process: (1) The impact of a small diagnosis can be large enough to render the ontology useless, while a larger diagnosis can have a more contained impact (2) using a diagnosis to repair the ontology will fix the ontology w.r.t.the erroneous entailment at hand, but might leave other problems, if the diagnosis is not properly analysed. To understand the impact of the axioms of a diagnosis, we follow the idea introduced by Alrabbaa *et al.* [ABD*20] to highlight not only *the diagnosis* itself in the ontology view, but also *the dependent atoms*, and thus the part of the ontology that would be potentially affected by changing the axioms that appear in the diagnosis. To show such impact on the ontology, the user can hover over the diagnoses on the sidebar to preview the impact, or click on them to select them and explore the view while they are highlighted. The sidebar and highlighting are shown in Figure 5(b).

To help users in comparing different diagnoses, and under the discussed challenges (DG-T2), we designed a filtering mechanism where users can *lock nodes* of the AD to indicate that their axioms must not be modified (*i.e.* marking these as *known* and *trusted*). In doing so, the length of the list of diagnoses can be reduced significantly. Diagnoses with axioms from the locked nodes are hence filtered out, but not the diagnoses that impact the locked nodes indirectly. In future versions, we may consider more advanced ways of filtering diagnoses using the ontology view.

4. Evonne: Technologies and Tools

Having the concepts of our tool explained in the previous section, we address in this section the specifics of our implementation. In particular, we describe the technologies we use, how we generate the data we visualize, and the improvements of the tool since its conception [ABD*20, FLAD20]. An online demo, local installation instructions, test data, pre-computed examples and a video walkthrough can be found at https://imld.de/evonne. Furthermore, the source code and documentation of Evonne are also publicly available at https://github.com/imldresden/evonne.

Evonne is a web-based application implemented with Node.js and Express.js for the server. For the client, we use d3 (both views) and WebCola for the force simulation and its settings in the ontology view. The look-and-feel is adapted from the Materialize styling. Since the initial prototype presented by Alrabbaa *et al.* [ABD*20], the majority of the visual characteristics of the interface were substantially reworked and a thorough refactoring of the original implementation took place to enable maintainability and extensibility of the tool. An example of this is the adjustment of the front-end to include reusable templates using the library Sprightly. The interface was adjusted to have a desktop application feel, which was overwhelmingly preferred by the experts we consulted (DG-U1, G1, G2). The visual rework includes an update to the styling of the nodes and links, smoother interactions and animations, a navigation bar with project identifier and several menus for both specific and common features of the views, as described in Section 3.2.

The back-end uses a Java application which computes the following: (1) the AD of a given OWL ontology, using the algorithm from Horridge *et al.* [VHP*20]; (2) various types of proofs for a given entailment using the approaches described by Alrabbaa *et al.* [ABB*22]; (3) all diagnoses for a given axiom, which are computed based on INCA, a tool for navigating answer sets of ASP programs [ARS18]. In order to serve the data to multiple independent

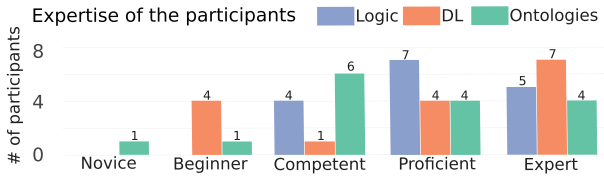

Figure 6: *Self-assessment of expertise from the participants from Novice (lowest) to Expert (highest) in the topics of general Logic, Description Logics (DL) and real-world Ontologies.*

views, we establish socket connections using SOCKET.IO. These connections are identified by a project ID and additional metadata that is required for the interactions, such as the origin of the interaction and the intended goal (*e.g.* the axiom which must be justified and the action name "highlight justification"). Combining this with the restful server, we communicate the views across devices as long as they are using the same project ID. This allows multiple sessions of the same major view to be active and coordinated at the same time, which can be used, for example, in a multi-window, a multi-device and even in a collaborative setting (DG-G1).

5. Evaluation: Expert Feedback Sessions

In order to evaluate our concepts and tool, EVONNE was tested in a formal qualitative user study that we describe in this section. The real-life ontologies that were used in the study were obtained from a 2017 snapshot of BioPortal [MP17], a repository containing ontologies from the bio-medical domain [RMKM08]. We chose to use BioPortal because the ontologies of this repository are often used for evaluation purposes in the DLs community, and because biology and medicine are central application domains of ontologies. Please note that the purpose of the study is not to evaluate how EVONNE can be used to obtain new knowledge about the domain of an ontology, but rather how it can be used to explain the phenomena entailed by the modelling of the domain in the ontology.

5.1. Study design

Participants: We recruited 16 logicians (3 female, 13 male) from the local university, with a mean age of $M = 32.28$ ($SD = 5.24$) and different levels of expertise in logics as detailed in Figure 6. Out of the 16 participants, 11 teach or have taught logics and related topics. We used a scale of low/mid/high to ask about their familiarity with the topics. All participants had medium or high familiarity with the notion of formal proofs (mid = 9, high = 7). The majority had at least medium familiarity with the notions of justifications (low = 3, mid = 7, high = 6) and diagnoses (low = 4, mid = 10, high = 2).

Methodology and goals: The sessions were conducted remotely with a video conference and shared screen. Each participant was interviewed individually for up to 90 min., with an average time of approximately 68 min. The web application was hosted and the participants were given links to access it. We recorded the video conferences for further inspection, after getting the consent of the participants. Two interviewers guided each session, one responsible for introducing the tool and guiding the participants through the tasks, the other taking notes about the reactions and answers of the participants, but also asking complementary questions as the participants followed a think-aloud protocol. Our goals were to assess the value of our concepts and tool for understanding proofs of entailments and for repairing ontologies, to collect qualitative feedback on the implemented features, and to distil new requirements and ideas for future work from the expectations of the participants.

Procedure and tasks: The sessions had three parts: First, a *tutorial* of the tool using toy data (*i.e.* the Spicy Ice Cream example). Then, a *guided walk-through* where the participants used various real-life data (ontologies and proofs) that presented difficulties (*e.g.* a very wide proof with long axioms, examples with complex concept and role names without much meaning, large ontologies with varying diagnosis impacts, etc.). The task of the participants would be to walk through these examples in such a way that they felt comfortable exploring both views w.r.t. the available features (*e.g.* finding an adequate layout for the wide proof, using a shortening mechanism to hide "unnecessary" domain knowledge in the axioms, filtering the diagnoses using our locking mechanism). Besides thinking aloud, we asked participants to state opinions regarding the features, to check against our estimations.

Lastly, the users were given a concluding offline questionnaire with the *User Experience Questionnaire* (UEQ, [HST18, SHT]) and a few additional questions about the usage of EVONNE as part of the current tool sets for editing ontologies, whether the tool could be used for teaching purposes, and open comments.

5.2. Results and reflection of design goals

The results of the UEQ assessment are shown in Figure 7. In the normalized $[-2, 2]$ scale used by the UEQ data analysis tool, EVONNE remained between 1 and 2 for all evaluated aspects. From lowest to highest: perspicuity (1.27), efficiency (1.47), dependability (1.59), attractiveness (1.66), stimulation (1.73) and novelty (1.72). To avoid confusion, we decided to remove the security aspect from the UEQ evaluation, since our scope did not include security features (*e.g.* restricted or partial access to ontologies). We found that despite their preferred method for traversing a proof, the users would switch reading directions when the possibility was presented to them. *"It depends on what I want—if I want to understand the whole proof, I would use the linear mode, but if I want to focus on a specific part, I would use the Magic mode"*. For example, users who preferred reaching the conclusion last while using the linear mode (*i.e.* bottom-up), would then find the step-wise explanations useful (*i.e.* top-bottom). Likewise, those who preferred starting from the root of the default tree (*i.e.* top-bottom) would be pleased to know they could collapse branches after all the nodes had been analysed (*i.e.* bottom-up). Furthermore, the participants that explicitly approached the proof from both directions simultaneously were pleased with the Magic mode, though most of the participants agreed that the interactions in this mode involved a learning curve. *"This "push" and "pull" (metaphor)—like with a door sign, I need to try both ways"*. Even though some participants grasped the interactions quickly, we identify this as an area of improvement for our tool.

Our participants also answered positively on the value of our tool for teaching purposes. They pointed out that even if some

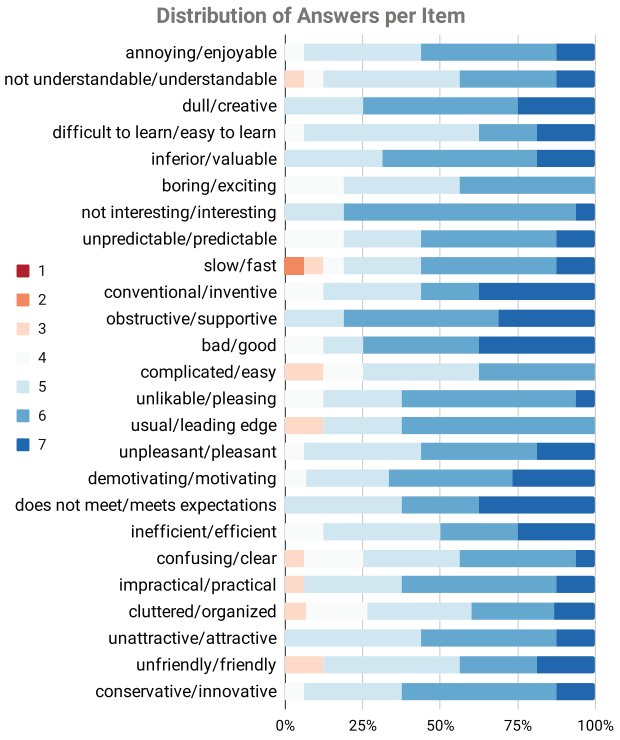

Figure 7: *Distribution of answers for the UEQ questionnaire; generated with the data analysis tool from the UEQ website [HST18].*

interactions are complicated, our colour coding, features for supporting explanations and the alternative layout options were very helpful. For example, when asked about the usefulness of the Rule Explanation tooltip: *"It is definitely useful (…) for training purposes. It is the first thing I clicked. It answers the question: what's going on here?"* Because of the involvement of most of our participants in teaching, this result is meaningful to us.

We identified other areas of improvement and potentially new feature concepts. As limitations, despite our multiple features to deal with over-plotting and readability—which were well received by our participants with varying preferences—no single configuration suffices for all scenarios, and thus we wish to investigate smart automatic configurations based on the proof and AD sizes. Regarding new features requested, some examples are (1) new levels of filtering diagnoses, (2) more ways to manipulate the shown text and (3) minor usability improvements. For instance, being able to hide all the rule names in a proof is one of the suggested improvements for the default tree layout. Most of the minor feedback was already addressed in our development revisions.

With respect to our design goals (Section 3.1), we consider that the feedback from the participants, both during the interviews and in the UEQ assessment, reflects our intentions positively. For instance, the *innovative, inventive, supportive, valuable* and *leading edge* points from UEQ confirm our goal to not just provide support for existing workflows, but also encourage new ways to fulfil the tasks (DG-T1, T2, G1). The results for *predictable, easy to learn, understandable* and *clear* are positive indicators of the fulfilment of our DGs U1, U2, U3. These results also hint at areas where Evonne can be improved, such as further optimization and more efforts to communicate insights about how to use the tool and how to interpret the visualized data. Furthermore, we are pleased to confirm that for our general usability goals (DG-G1, G2), participants of our study see a value in our tool for scenarios like teaching, where the flexibility of our web implementation enables complex multi-device and potentially multi-user setups.

6. Discussion and Future Directions

Evonne supports the processes of understanding proofs and debugging ontologies at an axiom level. The integrated combination of useful visualization and interaction concepts, with a focus on the aforementioned needs of the DLs community, is what makes our tool a valuable contribution, as confirmed by our feedback sessions and user study.

Knowledge representation is an area of AI that is, at least in theory, *explainable by design*, because decisions made by the AI system are based on symbolic reasoning from explicitly stated information. To make it also *explainable in practice*, there is a great need for more visualization work in knowledge representation and symbolic AI, because the (admittedly sound) existing explanations can be difficult to digest. Similar to our case with Evonne, the systematic combination, application and adaptation of visualization techniques to enhance the accessibility of this knowledge enables further research to support the understanding of complex reasoning processes. In the following, we discuss limitations of our tool and highlight future research directions.

Integration into the ontology development environment. Explaining entailments is only a part of the ontology development process, and as such, it should be integrated into the development environment used by ontology engineers, which would include aspects such as editing, versioning and documentation. Our web service architecture allows for an integration into Protégé or WebProtégé, which would provide an additional front-end to the editing capabilities of those tools. The integration should be bidirectional: (1) visualizing the effect of modifications to the ontology in Evonne and (2) using Evonne to directly lead the user to the axioms to modify in the other tools.

Complexity of the diagnoses selection. The exponential number of different diagnoses makes filter functionalities essential, especially when users need to decide which one to adopt. By leveraging the dependencies between the atoms in the AD, we provide users with a visual representation of the potential impact of diagnoses, which then supports them in selecting an appropriate diagnosis. The classical approach for repairing an erroneous entailment is to remove all axioms in the diagnosis. A more gentle approach is to change these axioms in a more fine-grained manner. In theory, there can be infinitely many ways to perform these "gentle repairs" [BKNP18], and we will look into supporting users in choosing the appropriate one. With *non-monotonic logics*, such as Defeasible DLs [PT18, BPS20], eliminating unwanted entailments can be achieved by adding

knowledge instead of removing it. However, investigating this is out of the current scope of Evonne.

Different logical formalisms. OWL ontologies are of course only one area in which automated reasoning is used—at least regarding proof visualization, our techniques can be used in other such areas where proofs are already available or easy to generate: This includes logic-based programming as with Prolog [Coh88] or Answer Set Programming [BET11], database reasoning with datalog [CGT89] or existential rules [CGK13, BLMS09], and theorem proving [BG01]. The rule set used in Evonne can be changed to support different reasoners, though this requires adapting the format of the proofs so that they can be read in our system. In fact, one of the participants in our study asked whether Evonne is sufficiently modular to integrate other reasoning systems to provide debugging support for other types of logics. This shows that the interest is not only on the OWL-specific computations used in our tool, but especially on the visual features that it provides.

Collaboration and multiple devices. Even though our prototype is currently focused on single-user setups, real applications of ontologies almost always happen in multi-user environments [SLR14]. Ontologies are usually developed in heterogeneous teams with differing expertise (domain experts, logic experts), in which proof visualizations could be used to explain the mechanics of an ontology to others, or to explore in a team how to fix or extend an ontology. We also envision collaborative settings in teaching. Proof visualizations could be used to explain logics to students, and students could explore proofs together to better understand reasoning in OWL ontologies. Our prototype and its client–server architecture can serve as a basis for enabling such use cases, for both co-located and remote collaboration. As mentioned in Section 3.2, Evonne allows the major views to be shown separately. While this does not make our tool tailored towards co-located collaboration, it enables, for example, a proof view to be displayed on a larger shared screen and ontology views to be presented on the mobile devices (*e.g*. tablets, laptops) of multiple team members.

7. Conclusion

We introduced Evonne, a web-based tool for explaining entailments of OWL ontologies. Entailments are explained through an interactive visualization of formal proofs, a main innovation being that those proofs are not shown in isolation, but seamlessly linked to a second view showing their context in the ontology. Both views come with interactive components that allow the exploration of the proof and the role of the involved axioms in the ontology. In addition to just explaining entailments, Evonne supports ontology debugging by showing users different ways to fix erroneous entailments discovered during the exploration of the proof. Exploring those fixes is supported by a visualization of their impact on other components of the ontology. The proof view allows for different ways of displaying and navigating a proof. In addition, the novel idea of the magic mode gives the user control over the structure of the proof, by enabling a bidirectional exploration. Evonne was very positively perceived by logic experts in our qualitative study, motivating further research in this direction. With this work, we hope to have paved the ground for further work in the area of formal proof and ontology visualization and, more broadly, of visual explanations in AI.

Acknowledgements

We thank Vincent Thiele and Tamara Flemisch for their contributions towards Evonne. This work was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy: EXC-2068, 390729961—Cluster of Excellence "Physics of Life" and EXC 2050/1, 390696704—Cluster of Excellence "Centre for Tactile Internet" (CeTI) of TU Dresden, by DFG Grant 389792660 as part of TRR 248—CPEC (see https://perspicuous-computing.science), and by the DFG Research Training Group QuantLA, GRK 1763 (https://lat.inf.tu-dresden.de/quantla).

Open access funding enabled and organized by Projekt DEAL.

References

[ABB*20] Alrabbaa C., Baader F., Borgwardt S., Koopmann P., Kovtunova A.: Finding small proofs for description logic entailments: Theory and practice. In *Proceedings of the 23rd International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR), EPiC Series in Computing* (2020), vol. *73*, EasyChair, pp. 32–67. http://doi.org/10.29007/nhpp

[ABB*21] Alrabbaa C., Baader F., Borgwardt S., Koopmann P., Kovtunova A.: Finding good proofs for description logic entailments using recursive quality measures. In *CADE, Lecture Notes in Computer Science* (2021), vol. *12699*, Springer, pp. 291–308. http://doi.org/10.1007/978-3-030-79876-5_17

[ABB*22] Alrabbaa C., Baader F., Borgwardt S., Dachselt R., Koopmann P., Méndez J.: Evonne: Interactive proof visualization for description logics (system description). In *Automated Reasoning* (2022), Springer International Publishing, pp. 271–280. http://doi.org/10.1007/978-3-031-10769-6_16

[ABD*20] Alrabbaa C., Baader F., Dachselt R., Flemisch T., Koopmann P.: Visualising proofs and the modular structure of ontologies to support ontology repair. In *Proceedings of the 33rd International Workshop on Description Logics (DL)* (2020), vol. *2663*, CEUR-WS.org. http://ceur-ws.org/Vol-2663/paper-2.pdf

[ARS18] Alrabbaa C., Rudolph S., Schweizer L.: Faceted answer-set navigation. In *Rules and Reasoning. Lecture Notes in Computer Science* (2018), vol. *11092*, Springer, pp. 211–225. http://doi.org/10.1007/978-3-319-99906-7_14

[BC87] Becker R. A., Cleveland W. S.: Brushing scatterplots. *Technometrics 29*, *2* (1987), 127–142. http://doi.org/10.1080/00401706.1987.10488204

[BET11] Brewka G., Eiter T., Truszczynski M.: Answer set programming at a glance. *Communications of the ACM 54*, 12 (2011), 92–103. http://doi.org/10.1145/2043174.2043195

[BG01] Bachmair L., Ganzinger H.: Resolution theorem proving. In *Handbook of Automated Reasoning*. Elsevier and MIT Press, 2001, pp. 19–99. http://doi.org/10.1016/b978-044450813-3/50004-7

[BGCF20] Braun G., Gimenez C., Cecchi L., Fillottrani P.: crowd: A visual tool for involving stakeholders into ontology engineering tasks. *KI - Künstliche Intelligenz 34* (2020). http://doi.org/10.1007/s13218-020-00657-8

[BHLS17] Baader F., Horrocks I., Lutz C., Sattler U.: *An Introduction to Description Logic*. Cambridge University Press, Cambridge, 2017. http://doi.org/10.1017/9781139025355

[BKNP18] Baader F., Kriegel F., Nuradiansyah A., Peñaloza R.: Making repairs in description logics more gentle. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixteenth International Conference, KR 2018, 30 October–2 November 2018* (Tempe, Arizona, 2018), M. Thielscher, F. Toni and F. Wolter (Eds.), AAAI Press, pp. 319–328. https://aaai.org/ocs/index.php/KR/KR18/paper/view/18056

[BLMS09] Baget J., Leclère M., Mugnier M., Salvat E.: Extending decidable cases for rules with existential variables. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence, IJCAI 2009* (2009), pp. 677–682. http://ijcai.org/Proceedings/09/Papers/118.pdf

[BPS20] Bonatti P. A., Petrova I. M., Sauro L.: Defeasible reasoning in description logics: An overview on DLN. In *Applications and Practices in Ontology Design, Extraction, and Reasoning*. IOS Press, Amsterdam, 2020. http://doi.org/10.3233/SSW200043

[CC07] Collins C., Carpendale S.: VisLink: Revealing relationships amongst visualizations. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1192–1199. http://doi.org/10.1109/TVCG.2007.70521

[CGH*19] Chen W., Guo F., Han D., Pan J., Nie X., Xia J., Zhang X.: Structure-based suggestive exploration: A new approach for effective exploration of large networks. *IEEE Transactions on Visualization and Computer Graphics 25*, 1 (2019), 555–565. http://doi.org/10.1109/TVCG.2018.2865139

[CGK13] Calì A., Gottlob G., Kifer M.: Taming the infinite chase: Query answering under expressive relational constraints. *Journal of Artificial Intelligence Research 48* (2013), 115–174. http://doi.org/10.1613/jair.3873

[CGT89] Ceri S., Gottlob G., Tanca L.: What you always wanted to know about datalog (and never dared to ask). *IEEE Transactions on Knowledge and Data Engineering 1*, 1 (1989), 146–166. http://doi.org/10.1109/69.43410

[CKB09] Cockburn A., Karlson A., Bederson B. B.: A review of overview+detail, zooming, and focus+context interfaces. *ACM Computing Surveys 41*, 1 (2009). http://doi.org/10.1145/1456650.1456652

[Coh88] Cohen J.: A view of the origins and development of prolog. *Communications of the ACM 31*, 1 (1988), 26–36. http://doi.org/10.1145/35043.35045

[CSM09] Catenazzi N., Sommaruga L., Mazza R.: User-friendly ontology editing and visualization tools: The owleasyviz approach. In *Proceedings of the 13th International Conference on Information Visualisation, IV* (2009), IEEE Computer Society. https://doi.org/10.1109/IV.2009.34

[DLSP18] Dudás M., Lohmann S., Svátek V., Pavlov D.: Ontology visualization methods and tools: A survey of the state of the art. *The Knowledge Engineering Review 33* (2018), e10. http://doi.org/10.1017/S0269888918000073

[DVKP*11] Del Vescovo C., Klinov P., Parsia B., Sattler U., Schneider T.: DeMoST: A tool for exploring the decomposition and the modular structure of owl ontologies. In *Proceedings of the 10th International Semantic Web Conference (ISWC-11)* (2011), pp. 191.

[EMJ*11] Elmqvist N., Moere A. V., Jetter H.-C., Cernea D., Reiterer H., Jankun-Kelly T.: Fluid interaction for information visualization. *Information Visualization 10*, 4 (2011), 327–340. http://doi.org/10.1177/1473871611413180

[Fal] Falconer S.: Ontograf protégé plugin (2010). https://protegewiki.stanford.edu/wiki/OntoGraf, Accessed: 2022-12-10.

[FLAD20] Flemisch T., Langner R., Alrabbaa C., Dachselt R.: Towards designing a tool for understanding proofs in ontologies through combined node-link diagrams. In *Proceedings of the 5th International Workshop on Visualization and Interaction for Ontologies and Linked Data (VOILA)* (2020), vol. *2778*, CEUR-WS.org. http://ceur-ws.org/Vol-2778/paper3.pdf

[FNS13] Fu B., Noy N. F., Storey M.-A.: Indented tree or graph? A usability study of ontology visualization techniques in the context of class mapping evaluation. In *The Semantic Web – ISWC* (2013), Springer, Berlin Heidelberg, pp. 117–134. http://doi.org/10.1007/978-3-642-41335-3_8

[Gen04] Gene Ontology Consortium: The Gene Ontology (GO) database and informatics resource. *Nucleic Acids Research 32* (2004), 258–261. http://doi.org/10.1093/nar/gkh036

[GFV13] Gibson H., Faith J., Vickers P.: A survey of two-dimensional graph layout techniques for information visualisation. *Information Visualization 12*, 3-4 (2013), 324–357. http://doi.org/10.1177/1473871612455749

[GHKS07] Grau B. C., Horrocks I., Kazakov Y., Sattler U.: Just the right amount: Extracting modules from ontologies. In

Proceedings of the 16th International Conference on World Wide Web, (WWW) (2007), ACM, pp. 717–726. http://doi.org/10.1145/1242572.1242669

[GHM*14] Glimm B., Horrocks I., Motik B., Stoilos G., Wang Z.: Hermit: An OWL 2 reasoner. *Journal of Automated Reasoning 53*, 3 (2014), 245–269. http://doi.org/10.1007/s10817-014-9305-1

[GPNT12] García-Peñalvo F. J., Palacios R. C., Navarro J. F. G., Therón R.: Towards an ontology modeling tool. A validation in software engineering scenarios. *Expert Systems with Applications 39*, 13 (2012), 11468–11478. http://doi.org/10.1016/j.eswa.2012.04.009

[HB11] Horridge M., Bechhofer S.: The OWL API: A java API for OWL ontologies. *Semantic Web 2*, 1 (2011), 11–21. http://doi.org/10.3233/SW-2011-0025

[HGN*19] Horridge M., Gonçalves R. S., Nyulas C. I., Tudorache T., Musen M. A.: Webprotégé: A cloud-based ontology editor. In *Companion Proceedings of the WWW'19* (2019), Association for Computing Machinery, pp. 686–689. http://doi.org/10.1145/3308560.3317707

[HKP*09] Hitzler P., Krötzsch M., Parsia B., Patel-Schneider P. F., Rudolph S.: Owl 2 web ontology language primer. *W3C Recommendation 27*, 1 (2009), 123.

[HMP*14] Horridge M., Mortensen J. M., Parsia B., Sattler U., Musen M. A.: A study on the atomic decomposition of ontologies. In *The Semantic Web – ISWC* (2014), Springer International Publishing, pp. 65–80. http://doi.org/10.1007/978-3-319-11915-1_5

[Hor] Horridge M.: Owlviz protégé plugin (2010). https://protegewiki.stanford.edu/wiki/OWLViz, Accessed: 2022-12-10.

[Hor08] Horrocks I.: Ontologies and the semantic web. *Communications of the ACM 51*, 12 (2008), 58–67. http://doi.org/10.1145/1409360.1409377

[Hor11] Horridge M.: Justification Based Explanation in Ontologies. PhD thesis. University of Manchester; 2011. https://www.research.manchester.ac.uk/portal/files/54511395/FULL_TEXT.PDF

[HSG15] Hoehndorf R., Schofield P. N., Gkoutos G. V.: The role of ontologies in biological and biomedical research: a functional perspective. *Briefings in Bioinformatics 16*, 6 (2015), 1069–1080. http://doi.org/10.1093/bib/bbv011

[HST18] Hinderks A., Schrepp M., Thomaschewski J.: Ueq: User experience questionnaire. https://www.ueq-online.org/ (2018), Accessed: 2022-12-10.

[IB14] Ivanovic M., Budimac Z.: An overview of ontologies and data resources in medical domains. *Expert Systems with Applications 41*, 11 (2014), 5158–5166. http://doi.org/10.1016/j.eswa.2014.02.045

[KC20] Koopmann P., Chen J.: Deductive module extraction for expressive description logics. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence, IJCAI 2020* (2020), C. Bessiere (Ed.), ijcai.org, pp. 1636–1643. http://doi.org/10.24963/ijcai.2020/227

[KK15] Kazakov Y., Klinov P.: Advancing ELK: Not only performance matters. In *Proceedings of the 28th International Workshop on Description Logics (DL)* (2015), CEUR-WS.org. http://ceur-ws.org/Vol-1350/paper-27.pdf

[KKS14] Kazakov Y., Krötzsch M., Simancik F.: The incredible ELK—from polynomial procedures to efficient reasoning with $\mathcal{EL}$ ontologies. *Journal of Automated Reasoning 53*, 1 (2014), 1–61. http://doi.org/10.1007/s10817-013-9296-3

[KKS17] Kazakov Y., Klinov P., Stupnikov A.: Towards reusable explanation services in Protégé. In *Proceedings of the 30th International Workshop on Description Logics (DL), CEUR Workshop Proceedings* (2017), vol. *1879*. http://www.ceur-ws.org/Vol-1879/paper31.pdf

[KLWW08] Konev B., Lutz C., Walther D., Wolter F.: Semantic modularity and module extraction in description logics. In *Proceedings of the 18th European Conference on Artificial Intelligence, ECAI 2008. Frontiers in Artificial Intelligence and Applications* (2008), M. Ghallab, C. D. Spyropoulos, N. Fakotakis and N. M. Avouris (Eds.), vol. *178*, IOS Press, pp. 55–59. http://doi.org/10.3233/978-1-58603-891-5-55

[KPSG06] Kalyanpur A., Parsia B., Sirin E., Grau B. C.: Repairing unsatisfiable concepts in OWL ontologies. In *The Semantic Web – ESWC 2006. Lecture Notes in Computer Science* (2006), vol. *4011*, Springer, pp. 170–184. http://doi.org/10.1007/11762256_15

[LdKLC13] Lee D., de Keizer N., Lau F., Cornet R.: Literature review of SNOMED CT use. *Journal of the American Medical Informatics Association 21*, e1 (2013), e11–e19. http://doi.org/10.1136/amiajnl-2013-001636

[LDMH*21] Lieber S., De Meester B., Heyvaert P., Brückmann F., Wambacq R., Mannens E., Verborgh R., Dimou A.: Visual notations for viewing RDF constraints with unshacled. *Semantic Web* (2021), 1–36. http://doi.org/10.3233/SW-210450

[LIRC12] Lee B., Isenberg P., Riche N. H., Carpendale S.: Beyond mouse and keyboard: Expanding design considerations for information visualization interactions. *IEEE Transactions on Visualization and Computer Graphics 18*, 12 (2012), 2689–2698. http://doi.org/10.1109/TVCG.2012.204

[LPP*06] Lee B., Parr C., Plaisant C., Bederson B., Veksler V., Gray W., Kotfila C.: Treeplus: Interactive exploration of networks with enhanced tree layouts. *IEEE Transactions on Visualization and Computer Graphics 12*, 6 (2006), 1414–1426. http://doi.org/10.1109/TVCG.2006.106

[MB19] Major T., Basole R. C.: Graphicle: Exploring units, networks, and context in a blended visualization approach. *IEEE*

Transactions on Visualization and Computer Graphics 25, 1 (2019), 576–585. http://doi.org/10.1109/TVCG.2018.2865151

[MMV11] Moodley K., Meyer T., Varzinczak I. J.: Root justifications for ontology repair. In *Web Reasoning and Rule Systems. Lecture Notes in Computer Science* (2011), vol. *6902*, Springer, pp. 275–280. http://doi.org/10.1007/978-3-642-23580-1_24

[MP17] Matentzoglu N., Parsia B.: BioPortal snapshot 30.03.2017. 2017. https://doi.org/10.5281/zenodo.439510, Accessed: 2022-12-10.

[Mun14] Munzner T.: *Visualization Analysis and Design. A K Peters Visualization Series*. CRC Press, Florida, 2014. http://doi.org/10.1201/b17511

[Mus15] Musen M. A.: The Protégé project: A look back and a look forward. *AI Matters 1*, 4 (2015), 4–12. http://doi.org/10.1145/2757001.2757003

[MVJS18] Matentzoglu N., Vigo M., Jay C., Stevens R.: Inference inspector: Improving the verification of ontology authoring actions. *Journal of Web Semantics 49* (2018), 1–15. http://doi.org/10.1016/j.websem.2017.09.004

[PPH18] Paulovics P., Pukancová J., Homola M.: SIVA: an educational tool for the tableau reasoning algorithm. In *Proceedings of the 31st International Workshop on Description Logics (DL)* (2018), vol. *2211*, CEUR-WS.org. http://ceur-ws.org/Vol-2211/paper-29.pdf

[PSDB20] Pernischová R., Serbak M., Dell'Aglio D., Bernstein A.: Chimp: Visualizing ontology changes and their impact in protégé. In *Proceedings of the 5th International Workshop on Visualization and Interaction for Ontologies and Linked Data* (2020), vol. *2778*, CEUR-WS.org, pp. 47–60. http://ceur-ws.org/Vol-2778/paper5.pdf

[PSK05] Parsia B., Sirin E., Kalyanpur A.: Debugging OWL ontologies. In *Proceedings of the 14th International Conference on World Wide Web (WWW)* (2005), ACM, pp. 633–640. http://doi.org/10.1145/1060745.1060837

[PT18] Pensel M., Turhan A.: Reasoning in the defeasible description logic $\mathcal{EL}^+$ - computing standard inferences under rational and relevant semantics. *International Journal of Approximate Reasoning* (2018). http://doi.org/10.1016/j.ijar.2018.08.005

[RBB02] Rost U., Bornberg-Bauer E.: TreeWiz: interactive exploration of huge trees. *Bioinformatics 18*, 1 (2002), 109–114. http://doi.org/10.1093/bioinformatics/18.1.109

[RDH*] Rector A. L., Drummond N., Horridge M., Rogers J., Knublauch H., Stevens R., Wang H., Wroe C.: OWL pizzas: Practical experience of teaching OWL-DL: Common errors & common patterns. In *Proceedings of the EKAW 2004. Lecture Notes in Computer Science*, vol. *3257*, Springer, pp. 63–81. http://doi.org/10.1007/978-3-540-30202-5_5

[RMKM08] Rubin D. L., Moreira D. A., Kanjamala P. P., Musen M. A.: Bioportal: A web portal to biomedical ontologies. In *AAAI Spring Symposium Series, Symbiotic Relationships between Semantic Web and Knowledge Engineering* (2008), Stanford University Press.

[Rob07] Roberts J. C.: State of the art: Coordinated multiple views in exploratory visualization. In *5th International Conference on Coordinated and Multiple Views in Exploratory Visualization (CMV)* (2007), pp. 61–71. http://doi.org/10.1109/CMV.2007.20

[SC03] Schlobach S., Cornet R.: Non-standard reasoning services for the debugging of description logic terminologies. In *IJCAI'03: Proceedings of the 18th International Joint Conference on Artificial Intelligence* (2003), Morgan Kaufmann, pp. 355–362. https://www.ijcai.org/Proceedings/03/Papers/053.pdf

[SHT] Schrepp M., Hinderks A., Thomaschewski J.: Applying the user experience questionnaire (UEQ) in different evaluation scenarios. In *Proceedings of the DUXU 2014*, Springer International Publishing, pp. 383–392. http://doi.org/10.1007/978-3-319-07668-3_37

[SLG14] Steigmiller A., Liebig T., Glimm B.: Konclude.: System description. *Journal of Web Semantics 27-28* (2014), 78–85. https://doi.org/10.1016/j.websem.2014.06.003

[SLR14] Simperl E., Luczak-Rösch M.: Collaborative ontology engineering: A survey. *The Knowledge Engineering Review 29*, 1 (2014), 101–131. http://doi.org/10.1017/S0269888913000192

[SMS*01] Storey M., Musen M., Silva J., Best C., Ernst N., Fergerson R., Noy N.: Jambalaya: Interactive visualization to enhance ontology authoring and knowledge acquisition in Protégé. In *Proceedings of the Workshop on Interactive Tools for Knowledge Capture (K-CAP)* (2001). https://www.isi.edu/~blythe/kcap-interaction/papers/storey.pdf

[TM03] Teoh S. T., Ma K.-L.: Paintingclass: Interactive construction, visualization and exploration of decision trees. In *KDD'03: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2003), Association for Computing Machinery, pp. 667–672. http://doi.org/10.1145/956750.956837

[TS20] Tominski C., Schumann H.: *Interactive Visual Data Analysis*. CRC Press, Boca Raton, FL, 2020. http://doi.org/10.1201/9781315152707

[TSP*08] Thomas E., Sleeman D. H., Pan J. Z., Reul Q., Lam J. S. C.: The Aberdeen University ontology reuse stack. In *Proceedings of the Symbiotic Relationships between Semantic Web and Knowledge Engineering* (2008), AAAI, pp. 83. http://www.aaai.org/Library/Symposia/Spring/2008/ss08-07-013.php

[Ves13] Vescovo C. D.: The Modular Structure of an Ontology: Atomic Decomposition and its Applications. PhD thesis, University of Manchester, UK, 2013.

[VHP*20] Vescovo C. D., Horridge M., Parsia B., Sattler U., Schneider T., Zhao H.: Modular structures and atomic decomposition in ontologies. *Journal of Artificial Intelligence Research 69* (2020), 963–1021. http://doi.org/10.1613/jair.1.12151

[WLA18] Wiens V., Lohmann S., Auer S.: Webvowl editor: Device-independent visual ontology modeling. In *Proceedings of the ISWC-P&D-Industry-BlueSky* (2018), vol. *2180*, CEUR-WS.org. http://ceur-ws.org/Vol-2180/paper-75.pdf

Supporting Information

Additional supporting information may be found online in the Supporting Information section at the end of the article.

Data S1