The Concrete Evonne: Visualization Meets Concrete Domain Reasoning

Christian Alrabbaa[1], Franz Baader[1], Raimund Dachselt[2], Alisa Kovtunova[1], and Julián Méndez[2]

[1] Institute of Theoretical Computer Science, TU Dresden, Germany
[2] Interactive Media Lab Dresden, TU Dresden, Germany
{first name.last name}@tu-dresden.de, julian.mendez2@tu-dresden.de

Abstract Evonne is a web application primarily designed to explain Description Logic (DL) entailments using an interactive visualization approach for proofs. This paper introduces an extension of Evonne to DLs with concrete domains, which are needed for formalizing concepts whose definitions involve quantitative information. Specifically, we focus on two extensions of the DL $\mathcal{EL}_\bot$: one with constraints formulated as linear equations and the other with difference constraints. First, we have extended Evonne to enable the generation and presentation of proofs involving these concrete domains. Then, leveraging the unique properties of each domain, we have designed and incorporated alternative visual explanations for the numerical parts of the proofs. Finally, we have assessed the effectiveness of these visual explanations through qualitative user studies and a performance benchmark. While opinions on one of these explanations varied, the other was widely recognized for its clarity and ease of understanding.

Keywords: Explainable AI · Description Logic · Concrete Domains · Visualization · Linear Equations · Difference Constraints.

1 Introduction

Due to the opacity of many machine learning approaches such as deep neural networks [27], explainability (xAI) has become a major research field in Artificial Intelligence [18,19]. Symbolic AI approaches based on logic have the advantage over subsymbolic approaches that they are explainable by design: a consequence computed by an automated reasoner can in principle be explained using a proof, which demonstrates how the consequence can be derived from given axioms by applying simple inference rules, and the non-derivability of a statement can (for some logics) be explained by showing a finite counter-interpretation, which is a model of all axioms, but not of the non-derivable statement. However, to leverage this advantage of logic-based approaches in practice, one must be able to produce proofs (counter-interpretations) that are appropriate for explanation purposes and present them in a comprehensible and cogent way.

In ongoing work, we address these issues in the context of Description Logics (DLs) [12], which are a prominent family of logic-based knowledge representation languages frequently used to formalize ontologies for various application domains. The computation of appropriate proofs and counter-interpretations has been tackled in [3,4] and [9,8], respectively. Both means of explanation can be presented in our interactive visualization tool Evonne[3] [2,23], but the proof-presentation facilities are considerably more mature. Since the publication of the Evonne system description [2] and a journal paper emphasizing its visualization components [23], this tool has been extended by new features and the look and feel of the system has been improved considerably.

Here we concentrate on the extension of the proof visualization facilities of Evonne to DLs with so-called concrete domains [11,22]. In particular, we consider extensions of tractable DLs of the $\mathcal{EL}$ family [10] with two p-admissible concrete domains based on rational numbers, one ($\mathcal{D}_{\mathbb{Q},lin}$) that can use linear equations to formulate constraints [13] and another ($\mathcal{D}_{\mathbb{Q},diff}$) based on difference constraints [10]. Such numerical constraints turn out to be very useful for describing concepts whose definition involves quantitative information, such as the battery capacity of a drone, its flight time, and weather conditions including temperature, which may influence the battery discharge rate. The exact definition of p-admissibility is not relevant for this paper (see [10,13] for details), but note that it is needed to preserve tractability. Both mentioned concrete domains are p-admissible, but their combination is not, though they can both be integrated into the same DL as long as they do not interact. In [5], we have addressed the problem of generating proofs for consequences derived from knowledge bases formulated in such DLs. Basically, the proof system for the extended DL as introduced in [10] uses entailment between and unsatisfiability of sets of concrete domain constraints as applicability conditions. The idea is then to explain the satisfaction of such side conditions by a proof of the entailment (unsatisfiability) if this is requested by the user. The first important new feature of Evonne described in this paper is the extension of its proof presentation facilities by such concrete domain proofs and their interaction with the abstract DL proofs.

However, the most original contribution of this work lies in its introduction of novel visual explanations for unsatisfiability and entailment in the considered numerical concrete domains. These visualizations are designed to reflect the unique properties of the domains and offer more intuitive insight into the underlying numerical reasoning. In the case of $\mathcal{D}_{\mathbb{Q},lin}$, unsatisfiability of a set of constraints using only two variables can be visualized in the 2D Euclidean Space by showing that the lines corresponding to the constraints do not intersect in a single point. Here, we address the challenge of extending this idea to higher dimensions. For $\mathcal{D}_{\mathbb{Q},diff}$, unsatisfiability of a constraint set corresponds to the existence of a negative cycle in the difference graph induced by these constraints. Thus, we developed a visual explanation based on such cycles. A priori, it is not clear how these visual explanations compare to numerical proofs. For this

[3] Evonne's source code, documentation, and evaluation material (user studies, benchmark) are available at: https://imld.de/evonne.

reason, we conducted qualitative user studies to investigate the user reception of these visualization techniques.

In summary, this paper presents the latest extension of Evonne, enabling interactive visualization of proofs for DLs with concrete domains—crucial for modeling concepts involving quantitative constraints. We contribute: (1) the *first* proof visualization tool supporting DLs with linear equations and difference constraints, (2) novel domain-specific visual explanations tailored to enhance comprehension of numerical reasoning, and (3) empirical validation through user studies and benchmarks, demonstrating the effectiveness of our approach. Our assessments showed that the proposed visualizations supported users in understanding conclusions more effectively than when no explanation was provided.

2 Description Logics and Concrete Domains

In this section we recall the Description Logic $\mathcal{EL}_\bot$ [12], and its extension $\mathcal{EL}_\bot[\mathcal{D}]$ with a concrete domain $\mathcal{D}$ [10]. We focus on two particular concrete domains $\mathcal{D}_{\mathbb{Q},diff}$ and $\mathcal{D}_{\mathbb{Q},lin}$ [10,13], both defined over the rational numbers $\mathbb{Q}$.

2.1 Concrete Domains

Concrete domains integrate reasoning about quantitative attributes of objects into DLs [11,22,13]. Let N_n be a set of *concrete predicates*, where every $\Pi \in \mathsf{N}_\mathsf{n}$ has arity $n_\Pi \in \mathbb{N}$. A *concrete domain (CD)* $\mathcal{D} = (\Delta^\mathcal{D}, \cdot^\mathcal{D})$ over N_n consists of a set $\Delta^\mathcal{D}$ and relations $\Pi^\mathcal{D} \subseteq (\Delta^\mathcal{D})^{n_\Pi}$ for all $\Pi \in \mathsf{N}_\mathsf{n}$. We assume that N_n always contains predicates $\bot$ and $\top$, interpreted as $\bot^\mathcal{D} := \emptyset$, and $\top^\mathcal{D} := \Delta^\mathcal{D}$. Let N_V be a set of *variables*. A *constraint* $\Pi(x_1, \ldots, x_{n_\Pi})$, with $\Pi \in \mathsf{N}_\mathsf{n}$ and $x_1, \ldots, x_{n_\Pi} \in \mathsf{N}_\mathsf{V}$, is a predicate with variables as arguments. A constraint $\alpha = \Pi(x_1, \ldots, x_{n_\Pi})$ is *satisfied* by an assignment $s\colon \mathsf{N}_\mathsf{V} \to \Delta^\mathcal{D}$ if $(s(x_1), \ldots, s(x_{n_\Pi})) \in \Pi^\mathcal{D}$. An *implication* is of the form $\mathfrak{C} \to \alpha$, where $\mathfrak{C}$ is a conjunction (set) of constraints. The implication is *valid* if all assignments satisfying all constraints in $\mathfrak{C}$ also satisfy α. A conjunction $\mathfrak{C}$ of constraints is *unsatisfiable* iff $\mathfrak{C} \to \bot$ is valid.

The CD $\mathcal{D}_{\mathbb{Q},diff}$ contains predicates $\top$, $\bot$, $x = q$, $x > q$, and $x + q = y$, for constants $q \in \mathbb{Q}$, with their natural semantics [10]. For instance, the constraint $x + q = y$ is interpreted as $(x + q = y)^{\mathcal{D}_{\mathbb{Q},diff}} = \{(p, r) \in \mathbb{Q} \times \mathbb{Q} \mid p + q = r\}$.

Example 1. Assume a delivery drone with bp representing its current *battery percentage*. The percentage is measured at multiple checkpoints, denoted as bp_0, bp_1, bp_2, with constraints: $bp_0 - 0.25 = bp_1$, $bp_1 - 0.2 = bp_2$, $bp_1 > 0.3$ and $bp_2 > 0.25$. If the initial percentage (bp_0) equals 0.65, then not all the constraints hold, and the drone is not permitted to fly.

For $\mathcal{D}_{\mathbb{Q},lin}$, besides $\top$ and $\bot$, the predicates are given by linear equations $\sum_{i=1}^{n} a_i x_i = b$, for $a_i, b \in \mathbb{Q}$, with their natural semantics [13]. For instance, $x + y - z = 0$ is interpreted as $(x + y - z = 0)^{\mathcal{D}_{\mathbb{Q},lin}} = \{(p, q, s) \in \mathbb{Q}^3 \mid p + q = s\}$.

Example 2. Assume nr and hr represent the average *normal* and *high* battery *discharge rates*, respectively. Under normal conditions, the delivery drone can fly for 8 hours on a single charge with a 30Ah battery, i.e., $8nr = 30$. In cold conditions, one hour of flight increases the battery consumption such that $4nr + hr = 30$. Therefore, if a delivery requires at most 2 hours in cold temperatures, the drone can complete it on a single charge.

2.2 Description Logics

DLs are decidable fragments of first-order logic (FOL) with a special, variable-free syntax and use only unary and binary predicates, called *concept names* and *role names*, respectively. These are used to build *complex concepts*, which correspond to first-order formulas with one free variable, and *axioms*, which correspond to first-order sentences. In this paper we consider the lightweight DL $\mathcal{EL}_\bot$. We use the usual notion of *entailment*, denoted $\mathcal{O} \models \mathsf{A} \sqsubseteq \mathsf{B}$, where A and B are concept names, and $\mathcal{O}$ is a finite set of axioms, called an *ontology*. For more details about the syntax and semantics of DLs, see [12].

The extension of $\mathcal{EL}_\bot$ with a concrete domain $\mathcal{D}$, i.e., $\mathcal{EL}_\bot[\mathcal{D}]$, is obtained by allowing constraints α in $\mathcal{D}$ to be used as $\mathcal{EL}_\bot$ concepts. As in [5], we use the notation $[\alpha]$ to distinguish between constraints and classical concepts. For instance, the statement that a delivery drone has a battery with a battery percentage greater than 0.25 can be expressed in an $\mathcal{EL}_\bot[\mathcal{D}_{\mathbb{Q},diff}]$ axiom as $\mathsf{DD} \sqsubseteq \exists \mathsf{has}.(\mathsf{Battery} \sqcap [bp > 0.25])$.

3 Combined Proofs

Evonne is a web application designed to explain DL entailments. It supports multiple proof types [2] and enhances them with interactive visualizations, helping users understand entailments and debug ontologies [23]. Proofs in Evonne are generated using the Evee library [7], and follow the notion introduced in [3]: A proof $\mathcal{P}$ of $\mathcal{O} \models \mathsf{A} \sqsubseteq \mathsf{B}$ is a finite, acyclic, directed *hypergraph*, where each vertex v is labeled with an axiom $\ell(v)$. Hyperedges are of the form (S, d), where S is a set of vertices and d is a vertex such that $\{\ell(v) \mid v \in S\} \models \ell(d)$. The leaf vertices of a proof are labeled with axioms from $\mathcal{O}$, and the root with $\mathsf{A} \sqsubseteq \mathsf{B}$. Evonne visualizes the proof hypergraphs using *tree* structures. In this paper, we extend Evonne's proofs to cover *combined proofs* for $\mathcal{EL}_\bot[\mathcal{D}_{\mathbb{Q},diff}]$ and $\mathcal{EL}_\bot[\mathcal{D}_{\mathbb{Q},lin}]$ entailments. The following example demonstrates the approach, as introduced in [5].

Let $\mathcal{O} = \{\mathsf{A} \sqsubseteq [\alpha_1], \mathsf{A} \sqsubseteq [\alpha_2], \mathsf{A} \sqsubseteq [\alpha_3], [\beta] \sqsubseteq \mathsf{B}\}$ be an ontology such that $\mathcal{O} \models \mathsf{A} \sqsubseteq \mathsf{B}$. First, we identify the relevant constraints in $\mathcal{O}$ and establish implications between them in order to build the subsumption hierarchy of abstract concepts. By using an appropriate CD reasoner, we test relevant implications such as $\{\alpha_1, \alpha_2\} \rightarrow \beta$. The ontology is then extended with axioms that encode all the valid relevant implications, i.e., $\mathcal{O}' = \mathcal{O} \cup \{[\alpha_1] \sqcap [\alpha_2] \sqsubseteq [\beta], \ldots\}$. By classifying $\mathcal{O}'$ we obtain $\mathcal{O}' \models \mathsf{A} \sqsubseteq \mathsf{B}$. Lastly, by integrating the proof of

$\{\alpha_1, \alpha_2\} \to \beta$ in the proof of $\mathcal{O}' \models \mathsf{A} \sqsubseteq \mathsf{B}$ as a proof for $[\alpha_1] \sqcap [\alpha_2] \sqsubseteq [\beta]$, we obtain the combined proof for $\mathcal{O} \models \mathsf{A} \sqsubseteq \mathsf{B}$.

Example 3. Consider the delivery drone (DD) from Example 2, along with the concept name *large battery drone* (LBD). Additionally, assume the following information is given. First, if operating at a high discharge rate for two hours draws 30Ah, this implies that a drone has a large battery, i.e., $[2hr = 30] \sqsubseteq \mathsf{LBD}$. Second, the delivery drone satisfies the constraints shown in Example 2, i.e., $\mathsf{DD} \sqsubseteq [8nr = 30] \sqcap [4nr + hr = 30]$. From these two axioms, it follows that the delivery drone is a large battery drone, i.e., $\mathsf{DD} \sqsubseteq \mathsf{LBD}$. An explanation of this conclusion is provided by the combined proof shown in Figure 1a, where the proof in Figure 1b shows the inferences at the level of equations.

To complete the picture of how concrete domain-dependent entailments are proven, we now describe the procedures that handle the CD reasoning steps.

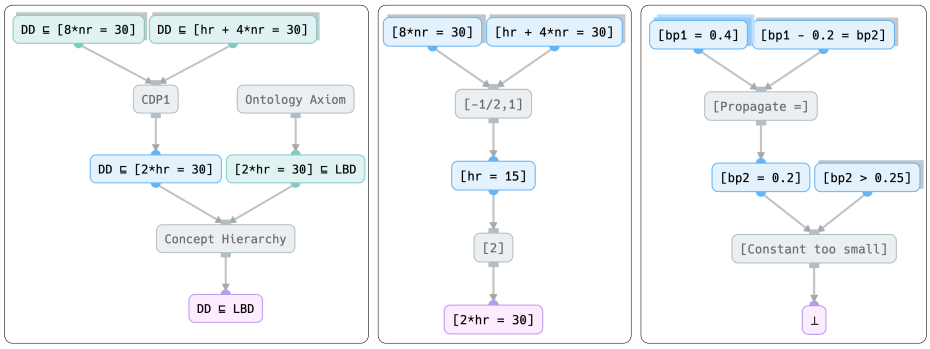
Reasoning in $\mathcal{D}_{\mathbb{Q},lin}$. Deciding the validity of $\mathfrak{C} \to \beta$ is achieved by identifying *linear combinations* that allow β to be derived from the equations in $\mathfrak{C}$. For instance, the implication $\{8nr = 30, 4nr + hr = 30\} \to hr = 15$ can be shown by multiplying $8nr = 30$ by $-\frac{1}{2}$ and adding it to $4nr + hr = 30$. Similarly, the unsatisfiability of a system of equations can be shown by providing a linear combination that results in the derivation of $0 = c$, where $c \neq 0$. In [5], determining the coefficients for the linear combinations is achieved using *Gaussian Elimination*, and these coefficients are used to build the inferences that constitute the proof. An example of a $\mathcal{D}_{\mathbb{Q},lin}$ proof in Evonne is shown in Figure 1b.

Reasoning in $\mathcal{D}_{\mathbb{Q},diff}$. Unlike the reasoning process in $\mathcal{D}_{\mathbb{Q},lin}$, deciding the validity of $\mathfrak{C} \to \beta$ is based on the *saturation* of $\mathcal{D}_{\mathbb{Q},diff}$ constraints, using the rules shown in [5, Fig.1]. Thus, checking whether $\mathfrak{C} \to \beta$ is valid is done by checking if the implication is present in the result of the saturation. In addition, if β is of the form $x > q$, then the implication is valid if either (i) $x = q'$ where $q' > q$ or (ii) $x > q'$ where $q' \geq q$ are derived with the saturation rules. A proof of $\mathcal{D}_{\mathbb{Q},diff}$ entailment is thus built using the instantiations of saturation rules directly. An example of such a proof in Evonne is shown in Figure 1c.

Combined proofs do not depend on the nature of the domains, making them versatile explanations that can be applied to various logics and concrete domains. However, their level of detail can result in large structures with potentially complicated inferences. For instance, the more variables and constraints involved in an entailment, the larger the resulting proof. To address this, we introduce alternative visual explanations, which we discuss in Section 4.

4 Domain-Specific Visualizations

In this section, we introduce two alternative visual explanations that leverage the specific characteristics of each concrete domain to present numerical entailments differently. These explanations do not rely on the DL part of the ontology, making them applicable to other logics or formalisms that incorporate these domains.



(a) Combined proof from Ex. 3 (b) CD proof from Ex. 2 (c) CD proof from Ex. 1

Figure 1: Examples of proofs in EVONNE.

4.1 Explanations of $\mathcal{D}_{\mathbb{Q},lin}$ Entailments

Let β be a linear equation, and $\mathfrak{C}$ be a set of linear equations such that $\mathfrak{C} \to \beta$. The implication means that every *assignment* satisfying all the constraints in $\mathfrak{C}$, i.e., every *solution*, also satisfies β. A proof explains the implication by showing how the conclusion β is derived from equations in $\mathfrak{C}$, i.e., showing that all solutions to $\mathfrak{C}$ are also solutions to β. Hence, a compact representation showing every solution to $\mathfrak{C}$ is a solution to β offers an alternative way to explain the implication.

If we consider systems of linear equations with at most two variables, which are shared across all equations (e.g., $\{\alpha_1, \ldots, \alpha_n, \beta\}$), then we can use *lines* in the 2D *Euclidean Space* to achieve a compact representation of all solutions to the equations, as each solution corresponds to a point in the 2D space. For example, assume $\{\alpha_1, \ldots, \alpha_n\} \to \beta$. If $\beta = \bot$, this means that the system $\{\alpha_1, \ldots, \alpha_n\}$ has *no solutions*. In other words, the lines corresponding to $\alpha_1, \ldots, \alpha_n$ neither intersect at a single point nor overlap. On the other hand, if $\beta \neq \bot$ and $\{\alpha_1, \ldots, \alpha_n\}$ is satisfiable, then the system $\{\alpha_1, \ldots, \alpha_n, \beta\}$ either has a *unique solution*, i.e., exactly one point where all the lines intersect, or *infinitely many solutions*, i.e., all lines overlap. Therefore, by demonstrating the existence or absence of such intersections, we can explain entailments effectively.

Using *dimensionality reduction*, we can apply the same idea to equations with more than two variables. While there are multiple ways to achieve dimensionality reduction [15], we utilize the solution space of a system of equations to project the n-dimensional equations to 2D. Let $\mathfrak{C}$ be a set of equations where $\mathfrak{C} \to \beta$, let s be a solution to $\mathfrak{C}$, and let x and y be any two variables appearing in β. By replacing all the variables, except x and y, in all equations with their corresponding values in s, we obtain equations involving at most two variables. The resulting equations can be viewed as a snapshot of $\mathfrak{C}$ with respect to s, focusing on x and y. Therefore, in a plane defined by x and y, and since s is a solution, all the lines must intersect. Hence, we can explain $\mathfrak{C} \to \beta$ by showing that for every solution s and every plane

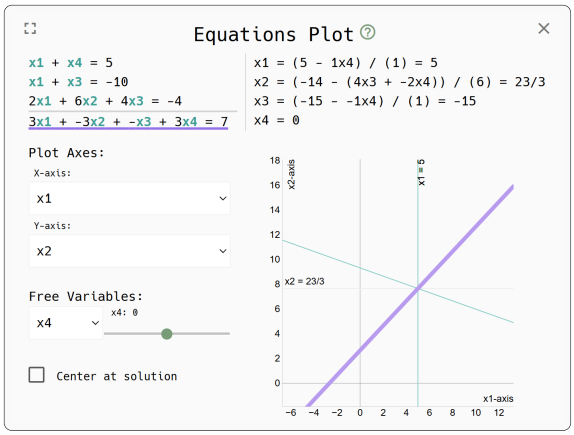


Figure 2: $\mathcal{D}_{\mathbb{Q},lin}$ implication of $3x_1 - 3x_2 - x_3 + 3x_4 = 7$ in Evonne

defined by x and y, the intersection of all the lines of $\mathfrak{C}$ is the same intersection of the line corresponding to β and the lines of $\mathfrak{C}$.

To explain unsatisfiability, we can use a similar approach. If $\mathfrak{C} \to \bot$, then there must be a contradiction in $\mathfrak{C}$. More specifically, at least two contradictory equations must be derivable from $\mathfrak{C}$, i.e., equations of the form $X = q$ and $X = p$, where X is a sum of terms and $p, q \in \mathbb{Q}$, $q \neq p$. Thus, in any 2D plane defined by variables appearing in $\mathfrak{C}$, with at least one variable appearing in X, the lines corresponding to $X = q$ and $X = p$ do not intersect. Consequently, in a plane where all the equations in $\mathfrak{C}$ needed to derive $X = q$ and $X = p$ can be plotted, these lines must also not intersect. Therefore, we can explain $\mathfrak{C} \to \bot$ by highlighting these contradictory equations in $\mathfrak{C}$ and showing that their lines never intersect, regardless of variable assignments.

Figures 2 and 3 show examples of $\mathcal{D}_{\mathbb{Q},lin}$ explanations in Evonne. The system of equations is shown in the top left. Hovering over an equation highlights its corresponding line in the 2D plot, and vice versa, helping users connect the algebraic and geometric views of the constraints. In Figure 2 the top right displays an instantiation of the system's solution based on the chosen value for the free variable. Since this system has one degree of freedom, setting $x_4 = 0$ uniquely determines the remaining variables. Hovering over the question mark icon reveals the solution without the variable assignment. In contrast, in Figure 3, the top right shows the system of equations with respect to the currently chosen variable assignment. Additionally, users can manipulate the visualization directly by switching between different planes and assigning values to (free) variables using the controls beside the plot. In the case of unsatisfiability, the visualization makes contradictions immediately apparent: if a user assigns a value that causes an equation to evaluate to $p = q$ with $p \neq q$, then this inconsistency is highlighted in red, as illustrated by $3 = 0$ in Figure 3.

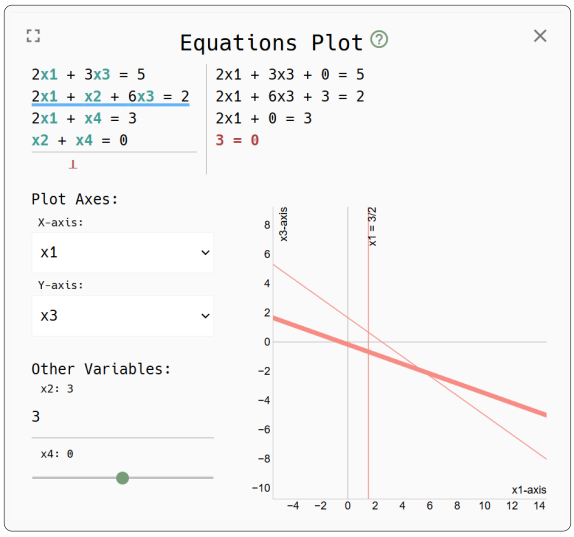


Figure 3: $\mathcal{D}_{\mathbb{Q},lin}$ implication of $\bot$ in Evonne

4.2 Explanations of $\mathcal{D}_{\mathbb{Q},diff}$ Entailments

It is well established that *difference constraints* (i.e., $x-y \leq q$) can be represented as graphs such that every variable corresponds to a vertex and every constraint to a weighted edge [14]. Given a set of difference constraints, deciding whether it is *unsatisfiable* can be reduced to finding a *simple cycle* in the corresponding graph with a *negative weight* [14]. Therefore, we use graphs and negative cycles to explain implications in $\mathcal{D}_{\mathbb{Q},diff}$. However, since predicates in $\mathcal{D}_{\mathbb{Q},diff}$ express more types of constraints, we first need to introduce some necessary transformations.

Let $\mathfrak{C}$ be a set of difference constraints. A *difference graph* is a weighted directed graph $G = (V, E)$, where each variable x_i in $\mathfrak{C}$ corresponds to a vertex $v_i \in V$, and each constraint $x_j - x_i \leq q_{ij}$ corresponds to an edge $(v_i, v_j) \in E$ with weight $q_{ij} \in \mathbb{Q}$. A *path* $\pi(v, u)$ from v to u is a sequence of edges:

$$v \xrightarrow{q_0} v_1 \xrightarrow{q_1} v_2 \ \ldots \ \ v_n \xrightarrow{q_n} u,$$

with weight $\pi^w(v, u) = q_0 + q_1 + \ldots + q_n$. A path is *simple* if all vertices, with the possible exception of v and u, are distinct. A *negative cycle* is a simple path $\pi(v, v)$ where $\pi^w(v, v) < 0$. A set of difference constraints is unsatisfiable iff the corresponding difference graph contains a negative cycle ([14, Theorem 24.9]).

Constraints in $\mathcal{D}_{\mathbb{Q},diff}$ of the form $y-x = q$ can be rewritten as two difference constraints: $y-x \leq q$ and $x-y \leq -q$. Additionally, the *unary* $\mathcal{D}_{\mathbb{Q},diff}$ constraints, i.e., constraints of the form $x \bowtie q$, where $\bowtie \in \{=, >\}$, can be represented as $x - z_0 \bowtie q$ where z_0 is a fresh variable. This formulation is sound because $x \bowtie q$ is satisfiable iff $\{x - z_0 \bowtie q, z_0 = 0\}$ is satisfiable ([14, Lemma 24.8]).

Lastly, we rewrite $x - z_0 > q$ as $z_0 - x \leq -q - \epsilon$, where ϵ is a placeholder for some small positive value. This rewriting is sound for the following reason:

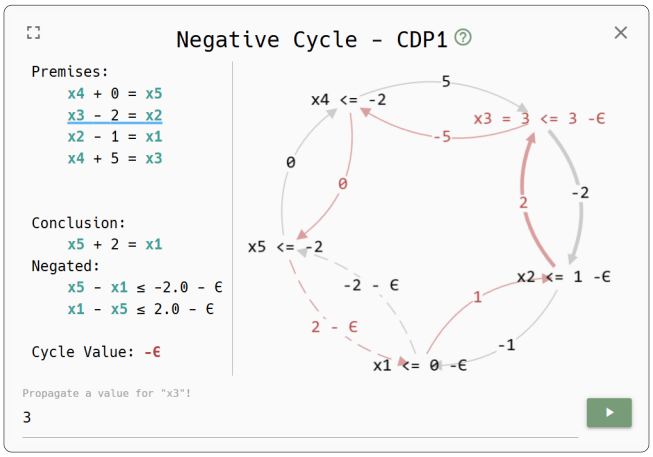


Figure 4: Example of an explanation for $\mathcal{D}_{\mathbb{Q},diff}$ implications in Evonne.

If $\mathfrak{C}$ is satisfiable and $\mathfrak{C} \cup \{x - z_0 > q\}$ is unsatisfiable, then for any positive $e \in \mathbb{Q}$, the constraints $\mathfrak{C} \cup \{x - z_0 \geq q + e\}$ are also unsatisfiable, which is equivalent to $\mathfrak{C} \cup \{z_0 - x \leq -q - e\}$. Meanwhile, for any assignment satisfying $\mathfrak{C} \cup \{x - z_0 > q\}$, there exists a positive $e \in \mathbb{Q}$ such that the same assignment also satisfies $x - z_0 \geq q + e > q$. Thus, $\mathfrak{C} \cup \{z_0 - x \leq -q - e\}$ is also satisfiable. Since such a small positive rational number e can always be found, we use ϵ symbolically as a reference to e whenever we need to transform a constraint with a $>$-predicate, rather than computing the specific value e for each set of constraints.

Consequently, we can represent any set $\mathfrak{C}$ of $\mathcal{D}_{\mathbb{Q},diff}$ constraints as a difference graph, and if $\mathfrak{C} \to \bot$, we can explain the contradiction in $\mathfrak{C}$ by identifying the negative cycle in the graph. However, if $\mathfrak{C}$ is satisfiable and $\mathfrak{C} \to \beta$, an additional step is required. In particular, we can explain that $\mathfrak{C}$ implies β by showing that $\mathfrak{C} \cup \{\neg\beta\} \to \bot$, which allows us to effectively use the notion of negative cycles to explain the implication. If β is of the form $x > q$, then the negative cycle in the difference graph corresponding to $\mathfrak{C} \cup \{x - z_0 \leq q\}$ shows that whenever an assignment satisfies all the constraints in $\mathfrak{C}$, the value of x must be greater than q. If β is of the form $x + y = q$, then $x + y \neq q$ is transformed into $x - y \leq q - \epsilon \vee y - x \leq -q - \epsilon$, leading to two negative cycles in the graph corresponding to $\mathfrak{C} \cup \{x - y \neq q\}$. These cycles explain $\mathfrak{C} \to x - y = q$ by demonstrating that no satisfying assignment for $\mathfrak{C}$ exists unless $x - y = q$ holds.

Figure 4 shows an example of a negative cycle in Evonne. Hovering over a constraint highlights its corresponding edge(s) in the graph, and vice versa. The dotted edges represent the negated conclusion. Furthermore, the negative cycle is animated, allowing users to visually follow its progression. This animation can be triggered by clicking on a vertex or a constraint; the clicked element then determines the starting point of the animation in the graph. Additionally,

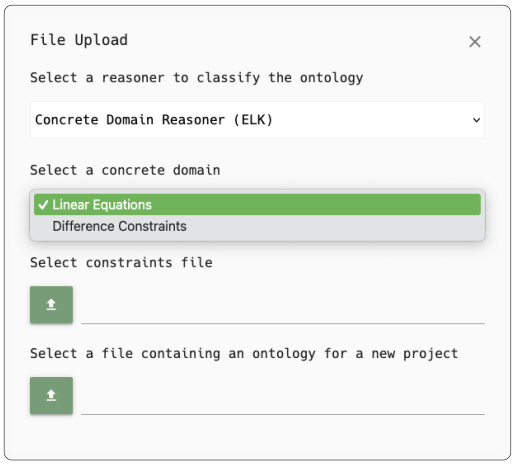


Figure 5: Popover for specifying new project information in Evonne.

users have access to a feature that lets them assign concrete values to variables by double-clicking on them. These values are then automatically propagated along the negative cycle, allowing users to observe how the set of constraints behaves under such assignments. In particular, this makes it possible to see exactly how the cycle leads to logical inconsistencies, which are highlighted in red. An example of such a contradiction is $x_3 = 3 \leq 3 - \epsilon$, as shown in Figure 4.

5 Numerical Explanations in Evonne

The first step to using Evonne to generate CD explanations is to create a new project. As shown in Figure 5, users must first specify which reasoner they would like to use—which, in our case, is the CD reasoner. This selection prompts the user to choose one of the two supported domains. Since the OWL 2 standard [28] does not support all predicate types used in our concrete domains, Evonne requires the user to provide two files for an ontology: one text file that contains the CD constraints of the ontology, and one OWL file that contains the DL part. Once the files are loaded, the user is asked to provide an axiom (concept inclusion) that they would like to have proven. If proving this axiom depends on the concrete domain part of the ontology, then a combined proof is generated and displayed.

By default, all numerical subproofs within a combined proof are collapsed into single inferences. Each of these inferences is labeled with a unique rule identifier—for example, the label *CDP1* used in the proof shown in Figure 1a—which allows for a more compact presentation of the proof. However, users can expand these subproofs to reveal all intermediate inferences. The visual explanations are accessible by clicking the labels of the numerical subproofs, which appear as popovers, as shown in Figures 2, 3, and 4). It is worth noting that

visual explanations rely only on the constraints within the corresponding subproofs and do not use additional constraints from the ontology.

The CD explanations in Evonne use function-plot and cytoscape.js to render the 2D plots and difference graphs, respectively. To handle numerical precision in JavaScript, we represent rational numbers as fractions, using Fraction.js. We implemented a Gaussian elimination solver for evaluating $\mathcal{D}_{\mathbb{Q},lin}$ equations, and a Hamiltonian cycle detector for animating the negative cycles. In our difference graphs, a negative-weight Hamiltonian cycle is guaranteed. This is because our graphs are based on minimal proofs, ensuring that (i) all necessary constraints for constructing the negative cycle are present, and (ii) no constraint is superfluous, as the proof's leaf constraints correspond to a justification, i.e., a minimal set of constraints entailing the proof's root constraint [3].

The current version of Evonne is accessible through https://imld.de/evonne, where all the examples used in the user studies (Section 6.1) and the benchmark (Section 6.2) are listed under the *Play Around* tab of Evonne.

6 Evaluation

We conducted two experiments to evaluate different aspects of our approach. The first experiment consists of user studies that assess the effectiveness of the numerical explanations in terms of user understanding and perceived helpfulness. The second experiment is a benchmark that evaluates the efficiency of rendering these explanations.

6.1 User Studies

To assess the CD explanations we conducted two qualitative studies—one for $\mathcal{D}_{\mathbb{Q},diff}$ and the other for $\mathcal{D}_{\mathbb{Q},lin}$—using online structured interviews. Both studies compared the classical proofs (e.g., Figures 1b, and 1c) with their respective alternative CD explanation (e.g., Figures 2, 3, 4). The goal was to compare the effectiveness of the explanations and collect feedback to improve them.

To ensure adherence to best scientific practices, we preregistered the studies. The preregistration, an extended report, and detailed user feedback are available online [20].

Study Design. We employed a 2x2 factorial design with two independent variables: **representation** (i.e., *plot/cycle* or *proof*), and **task type** (i.e., *unsatifiability* or *entailment*). The task type condition was necessary due to slight differences in plot and cycle representations between cases. Within each domain, four visual explanations of comparable difficulty were created with Evonne. Each task in $\mathcal{D}_{\mathbb{Q},lin}$ involved 3–4 linear equations and, for $\mathcal{D}_{\mathbb{Q},diff}$, 4–6 difference constraints. We used a within-subjects design with a randomized order: all participants experienced all conditions and examples. The dependent variables included ease of use (measured using Single Ease Question, SEQ [24]), user experience (assessed with User Experience Questionnaire, UEQ-S [21,26]),

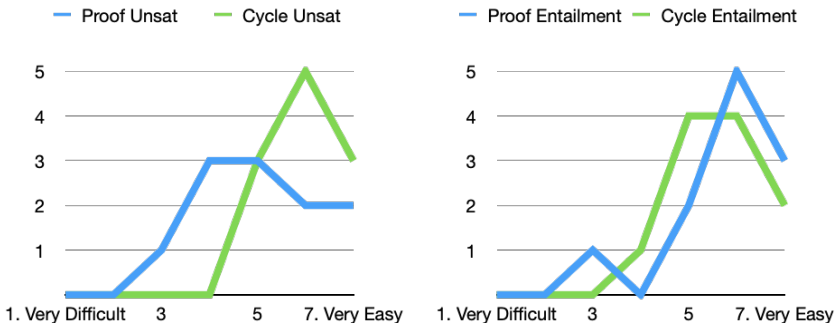


Figure 6: Distribution of SEQ responses in $\mathcal{D}_{\mathbb{Q},diff}$: X-axis represents how difficult it was to understand an explanation, and Y-axis indicates the number of participants who selected each class.

and subjective preference. These dependent variables are interrelated. As demonstrated further, subjective preferences consistently aligned with the results from SEQ and UEQ-S, reinforcing the validity of our findings. The online surveys were hosted on LimeSurvey [1] and included an introductory video for each domain. The $\mathcal{D}_{\mathbb{Q},diff}$ study averaged just 35 minutes, whereas the $\mathcal{D}_{\mathbb{Q},lin}$ study required an average of 50 minutes.

We recruited eleven participants (2 female, 9 male), aged 18–44, from among colleagues familiar with logic and linear algebra, of whom seven were Ph.D.s, two M.Sc.s, and one B.Sc. Participants were informed about the objective of the studies and consented to the use of anonymous data for scientific purposes.

Results for $\mathcal{D}_{\mathbb{Q},diff}$. First, SEQ shows how difficult the explanation is to understand on a Likert scale from 1 "very difficult" to 7 "very easy". Figure 6 demonstrates the following findings: For an *entailment* task, *cycles* and *proofs* received similar evaluations (*cycles*: average 5.6, sd. = 0.9; *proofs*: average 5.8, sd. = 1.2, respectively). However, for *unsatisfiability*, *cycles* received an average score of 6, sd. = 0.8, and were unanimously perceived as more intuitive than *proofs* (average 5, sd. = 1.3).

In addition to SEQ, participants briefly explained why they found a task difficult. They generally found *cycles* and their animations clear and understandable, particularly because the visualization automated calculations, highlighted negative cycles, and helped relate edges to constraints. However, a few participants noted that understanding the interface and interpreting the information required some initial effort, explanations, and practice, as in *"It took me a moment to understand the tool. Once I was given the explanation, I could understand how it works"*. On the other hand, *proofs* were perceived slightly less positively. While participants appreciated the ability to inspect inference steps, clearly identify which equations to combine, and the simplicity of step verification, they also noted some challenges. These included uninformative or redundant information

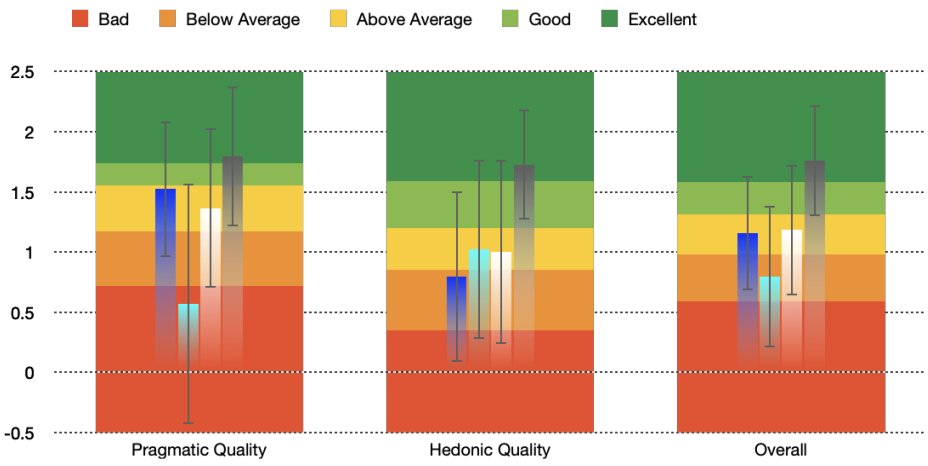


Figure 7: $\mathcal{D}_{\mathbb{Q},lin}$ proof (in blue), plot (in cyan), $\mathcal{D}_{\mathbb{Q},diff}$ proof (in white), and cycle (in gray) UEQ-S scores, their interpretation, and confidence intervals.

(e.g., repetitive left-hand sides of axioms and overly lengthy node labels), the need for manual calculations, and difficulties with the semantics of edge operations. For instance, a participant remarked: *"Proofs take a lot of space, and the edge labelings are unfamiliar"*.

Overall, both methods were well received, with participants appreciating their different strengths. As one participant summarized: *"A positive implication is actually very well represented in a proof. Cycle highlighting, hovering and animation are super cool"*.

Next, we assessed the user experience using the UEQ-S [25], by calculating *pragmatic* and *hedonic* quality scores. Pragmatic usability focuses on the task-oriented nature of an experience, whereas hedonic usability reflects non-utilitarian aspects such as appeal, originality, and joy of use. The white and gray bars in Figure 7 correspond to the following results: *Proofs* received an "above average" rating across all qualities, with scores of 1.364 (pragmatic quality, e.g., usability and functionality), 1 (hedonic quality, e.g., enjoyment and stimulation), and 1.182 (overall quality). In contrast, *cycles* were rated as "excellent" in all three categories, with scores of 1.795 (pragmatic), 1.727 (hedonic), and 1.761 (overall).

After each task, participants were asked whether the explanation service was useful for understanding. All but one participant responded "yes" across both task types and both representations. The outlier cited confusion related to the naming of *proof* edges.

Finally, in the post-test assessment in $\mathcal{D}_{\mathbb{Q},diff}$, most participants preferred *cycles* (8 vs. 2 for *proofs*, 1 for both). This preference grew stronger with more constraints (9 vs. 2 for *proofs*).

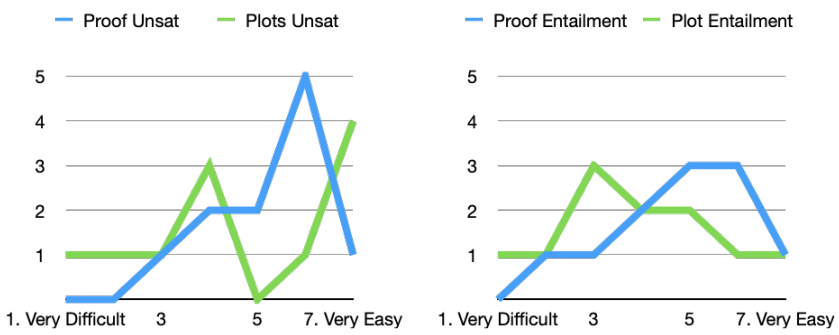


Figure 8: Distribution of SEQ responses in $\mathcal{D}_{\mathbb{Q},lin}$: The X-axis represents how difficult it was to understand an explanation, and the Y-axis indicates the number of participants who selected each class.

Results for $\mathcal{D}_{\mathbb{Q},lin}$. Regarding SEQ, Figure 8 demonstrates that, across both task types—*unsatisfiability* and *entailment*—*plots* (with average scores of 4.7, sd. = 2.2, and 3.9, sd. = 1.8, respectively) were perceived as less intuitive than *proofs* (with average scores of 5.3, sd. = 1.2, and 4.8, sd. = 1.5, respectively). In the case of *plots*, participants found them useful and enjoyable but also cognitively demanding to combine and interpret multiple variables displayed simultaneously, in particular when dealing with a higher number of dimensions. A respondent highlighted this point: *"The projection made me think that the solution was a point at first, before I realized that the solution is higher dimensional"*. *Unsatisfiability* tasks are perceived slightly easier than the *entailment* ones. Over time, familiarity with the visualization also improved, aided by provided instructions. For *proofs*, some participants found the step-by-step approach helpful, as it made the task easier to follow and verify compared to *plots*. They appreciated the ability to trace each step and felt confident in the overall result due to its clear mathematical foundation. However, the mental calculations required for verifying were quite challenging, as one participant noted: *"For proofs, I like that they show each individual step, but I did not figure out the solution myself from proofs, instead I trusted the system"*. Many participants found the notation less accessible, in particular: *"Too many symbols and too long node labels so it was hard to parse"*.

With respect to user experience, the blue and cyan bars in Figure 7 illustrate the following findings: *Proofs* received an overall rating of "above average" (1.159). Specifically, they scored "above average" (1.523) for pragmatic qualities (e.g., usability and functionality) but "below average" (0.795) for hedonic qualities (e.g., enjoyment and stimulation). In contrast, *plots* were rated overall as "below average" (0.795). They scored "bad" (0.568) for pragmatic qualities but achieved an "above average" (1.023) rating for hedonic qualities.

After each task, we asked if the explanation was helpful. For *proofs*, the response was unanimously positive across both task types. Feedback on *plots*

varied: for *unsatisfiability*, 10 said "yes" and 1 was "not sure"; for *entailment*, 6 said "yes", 4 were "not sure", and 1 said "no".

Subjectively, most participants preferred *proofs* (8 vs. 2 for *plots*, 1 for both). However, when linear equations involved only 2–3 variables, preference shifted to *plots* (8 vs. 3 for *proofs*).

Discussion. The user studies demonstrate that concrete domain-specific visualizations can significantly enhance the understanding of numerical reasoning, especially in the case of $\mathcal{D}_{\mathbb{Q},diff}$, where animated cycles were preferred for their clarity and intuitiveness. While proof trees were appreciated for their structure and detail, the choice between concrete domain visualizations and proofs appears task-dependent and influenced by numerical complexity. For simpler $\mathcal{D}_{\mathbb{Q},lin}$ entailments involving few variables, users preferred plots; however, for more complex cases and even with the added cognitive load, proofs were still preferred, as they were perceived as more trustworthy. These insights suggest that offering multiple, complementary explanation forms—proofs and CD visualizations—can accommodate diverse user preferences and experience levels. Future iterations of Evonne could build on this by incorporating adaptive explanation strategies based on characteristics of entailments or user feedback. Additionally, enabling users to adjust the existing visual explanations—or even extend them with user-defined visualizations, as supported by systems like Sterling [17]—could further accommodate individual preferences.

6.2 Performance Benchmark

We measured the response times for rendering the concrete domain explanations using combined proofs from the dataset provided in [5]. To automate the process, we used cypress.io, a testing framework, to load these proofs and open the corresponding CD explanations. Since the goal of the benchmark is to assess the rendering overhead introduced by the visual explanations, we do not measure the time required to compute the combined proofs, as this has already been reported in [6].

On average, response times were 18.32 ms for $\mathcal{D}_{\mathbb{Q},lin}$ and 8.7 ms for $\mathcal{D}_{\mathbb{Q},diff}$. The slowest times were 180.4 ms for $\mathcal{D}_{\mathbb{Q},lin}$ and 124.8 ms for $\mathcal{D}_{\mathbb{Q},diff}$, which correspond to large proofs that push the browser's resources considerably. These higher initial times occur when a CD explanation is opened for the first time, as the required HTML DOM elements are created and the relevant libraries are loaded. However, subsequent renders within the same proof are significantly faster. In the slowest cases, follow-up renders took 44.4 ms in $\mathcal{D}_{\mathbb{Q},lin}$ and 9.3 ms in $\mathcal{D}_{\mathbb{Q},diff}$. Note that all response times are within the 200ms threshold for perceived responsiveness [16].

7 Conclusion

We have extended Evonne to generate and present proofs for $\mathcal{EL}_\bot[\mathcal{D}_{\mathbb{Q},diff}]$ and $\mathcal{EL}_\bot[\mathcal{D}_{\mathbb{Q},lin}]$. Additionally, we have designed and developed alternative visual ex-

planations for numerical entailments, taking into account the characteristics of each concrete domain. Specifically, we illustrate $\mathcal{D}_{\mathbb{Q},diff}$ entailments using negative cycles and $\mathcal{D}_{\mathbb{Q},lin}$ entailments with 2D plots. These visual explanations are specific to the concrete domain yet independent of the underlying logic, making them adaptable and applicable to other formalisms. Furthermore, the visual display of the CD explanations scales better than that of the combined proofs. For instance, in the case of $\mathcal{D}_{\mathbb{Q},lin}$, adding more constraints simply results in additional lines in a 2D plot, without significantly affecting the overall layout. In contrast, increasing the number of equations in a proof expands the entire proof structure, which can make navigation more difficult due to increased scrolling and zooming.

We conducted qualitative user studies to assess the effectiveness of numerical explanations in Evonne. For $\mathcal{D}_{\mathbb{Q},lin}$, participants' opinions varied, though most alluded to a trust factor favoring proofs over visual explanations, suggesting that, in this case, such visual explanations might not be necessary. In contrast, for $\mathcal{D}_{\mathbb{Q},diff}$, participants highly valued the clarity and ease of understanding provided by the animated cycles, making them a more preferred form of explanation.

As future work, we plan to address issues raised by the participants' feedback on both proofs and the visual CD explanations. Additionally, we will refine Evonne's capabilities for explaining non-consequences through counter-interpretations and extend them to support DLs with concrete domains.

Acknowledgments. This work is funded by Deutsche Forschungsgemeinschaft (DFG) under Germany's Excellence Strategy: EXC 2050/1, 390696704 – "Centre for Tactile Interne" (CeTI); by DFG grant 389792660 as part of TRR 248 – CPEC, see `https://perspicuous-computing.science`; and by Bundesministerium für Bildung und Forschung (BMBF) and Saxon State Ministry for Science, Culture and Tourism (SMWK) in Center for Scalable Data Analytics and Artificial Intelligence (ScaDS.AI, SCADS22B). The authors extend gratitude to Dr. Ida Sri Rejeki Siahaan and Nicolás Rojas for their support in the early design and implementation of CD visualizations.

References

1. Der Umfragedienst für sächsische Hochschulen und Berufsakademien, `https://bildungsportal.sachsen.de/umfragen/`
2. Alrabbaa, C., Baader, F., Borgwardt, S., Dachselt, R., Koopmann, P., Méndez, J.: Evonne: Interactive proof visualization for description logics (system description). In: Automated Reasoning - 11th International Joint Conference, IJCAR 2022, Proceedings. Lecture Notes in Computer Science, vol. 13385, pp. 271–280. Springer (2022). `https://doi.org/10.1007/978-3-031-10769-6_16`
3. Alrabbaa, C., Baader, F., Borgwardt, S., Koopmann, P., Kovtunova, A.: Finding small proofs for description logic entailments: Theory and practice. In: LPAR 2020: 23rd International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Alicante, Spain, May 22-27, 2020. EPiC Series in Computing, vol. 73, pp. 32–67. EasyChair (2020). `https://doi.org/10.29007/NHPP`

4. Alrabbaa, C., Baader, F., Borgwardt, S., Koopmann, P., Kovtunova, A.: Finding good proofs for description logic entailments using recursive quality measures. In: Automated Deduction - CADE 28 - 28th International Conference on Automated Deduction 2021, Proceedings. Lecture Notes in Computer Science, vol. 12699, pp. 291–308. Springer (2021). https://doi.org/10.1007/978-3-030-79876-5_17
5. Alrabbaa, C., Baader, F., Borgwardt, S., Koopmann, P., Kovtunova, A.: Combining proofs for description logic and concrete domain reasoning. In: Rules and Reasoning - 7th International Joint Conference, RuleML+RR 2023, Proceedings. Lecture Notes in Computer Science, vol. 14244, pp. 54–69. Springer (2023). https://doi.org/10.1007/978-3-031-45072-3_4
6. Alrabbaa, C., Baader, F., Borgwardt, S., Koopmann, P., Kovtunova, A.: Combining proofs for description logic and concrete domain reasoning (technical report) (2023), https://arxiv.org/abs/2308.03705
7. Alrabbaa, C., Borgwardt, S., Friese, T., Hirsch, A., Knieriemen, N., Koopmann, P., Kovtunova, A., Krüger, A., Popovic, A., Siahaan, I.S.R.: Explaining reasoning results for OWL ontologies with Evee. In: Proceedings of the 21st International Conference on Principles of Knowledge Representation and Reasoning, KR 2024, Hanoi, Vietnam. November 2-8, 2024 (2024). https://doi.org/10.24963/KR.2024/67
8. Alrabbaa, C., Hieke, W.: Explaining non-entailment by model transformation for the description logic $\mathcal{EL}$. In: Proceedings of the 11th International Joint Conference on Knowledge Graphs, IJCKG 2022. pp. 1–9. ACM (2022). https://doi.org/10.1145/3579051.3579060
9. Alrabbaa, C., Hieke, W., Turhan, A.: Counter model transformation for explaining non-subsumption in $\mathcal{EL}$. In: Proceedings of the 7th Workshop on Formal and Cognitive Reasoning co-located with the 44th German Conference on Artificial Intelligence (KI 2021). CEUR Workshop Proceedings, vol. 2961, pp. 9–22. CEUR-WS.org (2021), https://ceur-ws.org/Vol-2961/paper_2.pdf
10. Baader, F., Brandt, S., Lutz, C.: Pushing the EL envelope. In: IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 - August 5, 2005. pp. 364–369. Professional Book Center (2005), http://ijcai.org/Proceedings/05/Papers/0372.pdf
11. Baader, F., Hanschke, P.: A scheme for integrating concrete domains into concept languages. In: Proceedings of the 12th International Joint Conference on Artificial Intelligence. Sydney, Australia, August 24-30, 1991. pp. 452–457. Morgan Kaufmann (1991), http://ijcai.org/Proceedings/91-1/Papers/070.pdf
12. Baader, F., Horrocks, I., Lutz, C., Sattler, U.: An Introduction to Description Logic. Cambridge Univ. Press (2017). https://doi.org/10.1017/9781139025355
13. Baader, F., Rydval, J.: Using model theory to find decidable and tractable description logics with concrete domains. J. Autom. Reason. **66**(3), 357–407 (2022). https://doi.org/10.1007/s10817-022-09626-2
14. Cormen, T., Leiserson, C., Rivest, R., Stein, C.: Introduction to Algorithms, third edition. Computer science, MIT Press (2009), https://books.google.de/books?id=i-bUBQAAQBAJ
15. Cunningham, J.P., Ghahramani, Z.: Linear dimensionality reduction: survey, insights, and generalizations. J. Mach. Learn. Res. **16**, 2859–2900 (2015). https://doi.org/10.5555/2789272.2912091
16. Dabrowski, J.R., Munson, E.V.: Is 100 milliseconds too fast? In: CHI 2001 Extended Abstracts on Human Factors in Computing Systems, CHI Extended Abstracts 2001, Seattle, Washington, USA, March 31 - April 5, 2001. pp. 317–318. ACM (2001). https://doi.org/10.1145/634067.634255

17. Dyer, T., Jr., J.W.B.: Sterling: A web-based visualizer for relational modeling languages. In: Rigorous State-Based Methods - 8th International Conference, ABZ 2021, Ulm, Germany, June 9-11, 2021, Proceedings. Lecture Notes in Computer Science, vol. 12709, pp. 99–104. Springer (2021). https://doi.org/10.1007/978-3-030-77543-8_7
18. Gunning, D., Stefik, M., Choi, J., Miller, T., Stumpf, S., Yang, G.: XAI – explainable artificial intelligence. Sci. Robotics **4**(37) (2019). https://doi.org/10.1126/SCIROBOTICS.AAY7120
19. Holzinger, A., Saranti, A., Molnar, C., Biecek, P., Samek, W.: Explainable AI methods - A brief overview. In: xxAI – Beyond Explainable AI - International Workshop, Held in Conjunction with ICML 2020, Revised and Extended Papers. Lecture Notes in Computer Science, vol. 13200, pp. 13–38. Springer (2020). https://doi.org/10.1007/978-3-031-04083-2_2
20. Kovtunova, A., Alrabbaa, C., Baader, F., Dachselt, R., Méndez, J.: The Concrete Evonne (Feb 2025). https://doi.org/10.17605/OSF.IO/Y4X5T
21. Laugwitz, B., Held, T., Schrepp, M.: Construction and evaluation of a user experience questionnaire. In: HCI and Usability for Education and Work, 4th Symposium of the Workgroup Human-Computer Interaction and Usability Engineering of the Austrian Computer Society, USAB 2008, Graz, Austria, November 20-21, 2008. Proceedings. Lecture Notes in Computer Science, vol. 5298, pp. 63–76. Springer (2008). https://doi.org/10.1007/978-3-540-89350-9_6
22. Lutz, C.: Description logics with concrete domains-a survey. In: Advances in Modal Logic 4, papers from the fourth conference on "Advances in Modal logic," held in Toulouse, France, 30 September - 2 October 2002. pp. 265–296. King's College Publications (2002), http://www.aiml.net/volumes/volume4/Lutz.ps
23. Méndez, J., Alrabbaa, C., Koopmann, P., Langner, R., Baader, F., Dachselt, R.: Evonne: A visual tool for explaining reasoning with OWL ontologies and supporting interactive debugging. Comput. Graph. Forum **42**(6) (2023). https://doi.org/10.1111/CGF.14730
24. Sauro, J., Dumas, J.S.: Comparison of three one-question, post-task usability questionnaires. In: Proceedings of the 27th International Conference on Human Factors in Computing Systems, CHI 2009, Boston, MA, USA, April 4-9, 2009. pp. 1599–1608. ACM (2009). https://doi.org/10.1145/1518701.1518946
25. Schrepp, M.: User Experience Questionnaire Handbook (2015). https://doi.org/10.13140/RG.2.1.2815.0245
26. Schrepp, M., Hinderks, A., Thomaschewski, J.: Design and evaluation of a short version of the user experience questionnaire (UEQ-S). Int. J. Interact. Multim. Artif. Intell. **4**(6), 103–108 (2017). https://doi.org/10.9781/IJIMAI.2017.09.001
27. Søgaard, A.: On the opacity of deep neural networks. Canadian Journal of Philosophy **53**(3), 224–239 (2023). https://doi.org/10.1017/can.2024.1
28. W3C OWL Working Group: OWL 2 Web Ontology Language Document Overview (Second Edition) - W3C Recommendation 11 December 2012 (Dec 2012), http://www.w3.org/TR/owl2-overview/