Towards Designing a Tool For Understanding Proofs in Ontologies through Combined Node-Link Diagrams

Tamara Flemisch[1], Ricardo Langner[1], Christian Alrabbaa[2], and Raimund Dachselt[1]

[1] Interactive Media Lab Dresden, Technische Universität Dresden
[2] Theoretical Computer Science, Technische Universität Dresden
Nöthnitzer Str. 46, 01187 Dresden, Germany
firstname.lastname@tu-dresden.de

Abstract. As the creation of large ontologies is a difficult and error-prone process, we think it is particularly relevant to develop new visual and interactive tools that support exploring ontologies as well as finding and resolving defects, such as undesired logical entailments. Exhibiting formal proofs for the undesired entailments can help in understanding how a defect happens, and linking the steps in the proof to the ontology can help in determining how the defect can be fixed. We present an interface that visualizes proofs and the corresponding ontology in form of side-by-side node-link diagrams. Building on linked brushing, users benefit from a strong interplay between these views, which allows for discovering and understanding defects. Besides traditional desktop workplaces, our interaction design also considers the use of touch input enabled by interactive displays. As part of an iterative design process, we developed an initial web-based prototype implementation and gathered feedback from an interview with domain experts. With this ongoing research and development, we aim to further investigate the potential and general utility of interactive visualizations for ontology engineering.[3]

1 Introduction and Related Work

Creating and maintaining large ontologies is an error-prone process. The large size and the intricate relation of the knowledge represented in ontologies makes the task of ontology debugging difficult [12]. Examples for errors that appear in ontologies are unsatisfiable concepts or undesired entailments, which we refer to as *defects* in the following. However, tools exist to aid users in finding incorrect or undesired entailments. Exploring formal proofs for the undesired entailments can help in understanding the origin of a defect. Furthermore, connecting the proof of the undesired statement to the ontology can aid in determining how a defect can be fixed. This can be achieved by computing diagnoses for the defect and showing their impact on the whole ontology. We think, it is highly relevant to support this process of exploring ontologies as well as finding and resolving defects by developing new visual and interactive tools.

Visualization and interactive methods have been applied to a multitude of different application domains. Especially linking and combining multiple visualizations, e.g., multiple coordinated views (MCVs), is a common approach to solve complex problems [27]. It has been used to address challenges inherent to visualizations, such as filtering and querying [5], collaborative exploration through meta-visualizations [32], and visualizing change in dynamic networks [18]. Some of these techniques were then applied to domain-specific problems, such as exploring multivariate data [17], observing the sharing of pictures in social networks [18], and visualizing genealogy [26] and biology data [21]. Multiple visualizations have also been investigated in non-traditional setups, such as large, interactive displays [17, 32] and mobile devices [16, 28]. Especially in the domain of visualizing ontologies, plenty of tools exist and were collected and compared in a comprehensive survey by Dudás et al. [6]. An example is Matentzoglu et al.'s [23] Inference Inspector which helps in understanding ontologies by visualizing entailment set changes. Among the variety of different techniques, such as node-link visualizations, radial layouts, and treemaps, are specific examples, such as indented lists used in Protégé [15], Jambalaya [31], OWLViz[4], and KC-Viz [24]. They use a vast selection of interaction techniques, such as filtering, keeping a history, highlighting, and edge bundling. However, most of the existing tools have issues [6]:

- They often have a UI, visualization and interaction design that leaves room for improvement regarding current technologies
- They only focus on class hierarchies of ontologies and are not tailored towards understanding proofs
- They focus on a text based representation instead of a visualization of the content

Interaction plays a huge role when it comes to exploring and understanding data [19, 8] in general. More specifically, previous work has looked at interaction techniques for exploring tree visualizations [20] and node-link diagrams [5, 3, 22]. Furthermore, natural interaction, e.g., through pen and touch input, has also been successfully applied to network visualizations to aid with exploring [7, 18, 14] as well as editing [9].

We present an interface that visualizes proofs and the corresponding ontology in form of side-by-side node-link diagrams for ontology debugging of unwanted consequences. Whereas the view of the ontology, which displays its modular structure, has its origin at the top of the view, the proof tree shows the root node, i.e., the final conclusion of the proof, at the bottom. We build on common interaction techniques, such as linked brushing [2, 27] and overview+detail [4, 25], to create a strong interplay between these components to support the users in discovering and understanding defects. In addition to traditional desktop workplaces, we also consider the use of touch input enabled by interactive displays and propose adaptations to leverage the benefits of natural interaction.

In this work, we describe our tool based on the process we went through. First, we describe the use cases on which we based our design. After elaborating more on the detailed visualization and interaction concepts, we explain the technical details of the implementation. Finally, we discuss adaptations based on what we learned from

[4] `https://github.com/protegeproject/owlviz`

our prototype and the feedback we gathered from the domain experts for our initial prototype and.

2 Application Interface

Our application is a prototypical web application for ontology debugging of unwanted consequences. It consists of two main components: the *Proof Component* which offers an interactive visualization for explaining defects by using proofs and the *Ontology Component* which visualizes the ontology in its modular structure and shows diagnoses of defects and their impact on the ontology.

In this section, we start by describing the use cases we derived from several analytical interviews with expert users. We then explain how these use cases inspired our visualization and interaction design for traditional desktop setups.

2.1 Use Cases

We conducted several analysis sessions experts for Description Logic ontologies [1]. During the first sessions, we made use of paper-based and whiteboard sketches to understand the domain, the data, and the problem with existing tools. In later sessions, we also incorporated their data, i.e., a graph representing the ontology in its modular structure and formal proofs as trees, into exiting graph visualization tools. Our goal was to discover issues with current visualizations, to identify challenges for visualizing both types of data at the same time, and to explore possible scenarios. For all sessions independent of the technologies used, we interviewed the domain experts and derived the following use cases for visualizing proofs along with their respective ontologies with regard to ontology debugging.

Comparing and Contrasting Ontologies are often subject to change and a collaborative effort of a whole community. This results in bugs potentially being introduced by accident from one version to the next. Therefore, it can be beneficial to compare the proofs for undesired entailments in addition to the ontology among different versions. This visual comparison can help to spot differences, identify changes, and potentially even help to understand the origin of these changes [10].

Understanding and Exploring Since ontologies are large constructs that are often hard to grasp for novice users, one use case is the explaining of ontologies. We believe that exploring a proof, i.e., a designated part of an ontology, can help novices in understanding the inner workings and the structure of ontologies. Furthermore, breaking down a graph into chunks that are easier to understand at once, might help in exploring the overall network.

Showing and Explaining Analogous to the previous use case, presenting and exploring a proof might also help when explaining an ontology's structure to other people. The proof's layered and stepwise structure helps in breaking down a larger problem into little chunks. It can further aid in explaining how logical reasoning works by example which can be leveraged for educational purposes.

Repairing and Fixing The most important use case for our application scenario is repairing and fixing parts of an ontology. After having found the undesired entailment, the user aims to fix it by iteratively navigating a proof tree and identifying the origin of the bug. Additionally, the connection between ontology and proof might help the user to decide how to fix a particular bug by computing diagnoses.

After carefully evaluating the possible use cases for visualizing proofs and their corresponding ontology, we decided to mostly focus on the primary use case of *Repairing and Fixing* in addition to a secondary use case of *Understanding and Exploring*. However, our proposed interface does not exclusively support one use case, it still supports other use cases in parts. In summary, our main aim was to assist users in achieving the following tasks:

- Understanding entailments through proof exploration
- Understanding the interaction of axioms in the ontology
- In case of unwanted entailments, understand how to fix the ontology and the effect of axioms removal on the modular structure of the ontology

2.2 Visualization, Interaction Design, and Features

Before designing our application, we decided on some requirements that shaped our design process:

- The application is predominantly used in a traditional desktop environment.
- It should be possible to use the application across distributed devices, i.e., having the two visualizations on different devices.
- Our main goal is to seamlessly integrate both components into one interactive tool.

To make it easier for novices to understand, we are using a commonly used example ontology, the Pizza Ontology in a modified way[5] in our examples and figures. The modification introduces the defect $\mathsf{SpicyIceCream} \sqsubseteq \mathsf{Pizza}$.

The Proof Component The proof component consists mostly of the proof tree of a chosen entailment with the root node being the proven entailment, i.e., the *conclusion* (see Figure 1). In our example, the proof shows the unsatisfiability of SpicyIceCream. The leaf nodes represent a so called *justification* for this entailment, i.e., statements that are part of the ontology and justify the entailment as being true. The levels in-between show the logical reasoning from the justifications to the entailment. By exploring the proof, the user may find other defects that lie within the proof, e.g., that $\mathsf{SpicyIceCream} \sqsubseteq \mathsf{Pizza}$. The *Proof Component* is the center of our application and serves as a starting point for exploring the ontology and finding the origin of a defect. For showing the proofs, we use a node link diagram that puts the leaf nodes on top and the root node at the bottom of the view. This node-link encoding emphasizes the connection among nodes, their depth within the tree, and its topological structure [30], i.e., the various paths that lead to the final conclusion. We decided to create an axis-oriented tree layout since it is very common and most users will be familiar with it [29].

[5] Available at `https://lat.inf.tu-dresden.de/Evonne/PizzaOntology/`

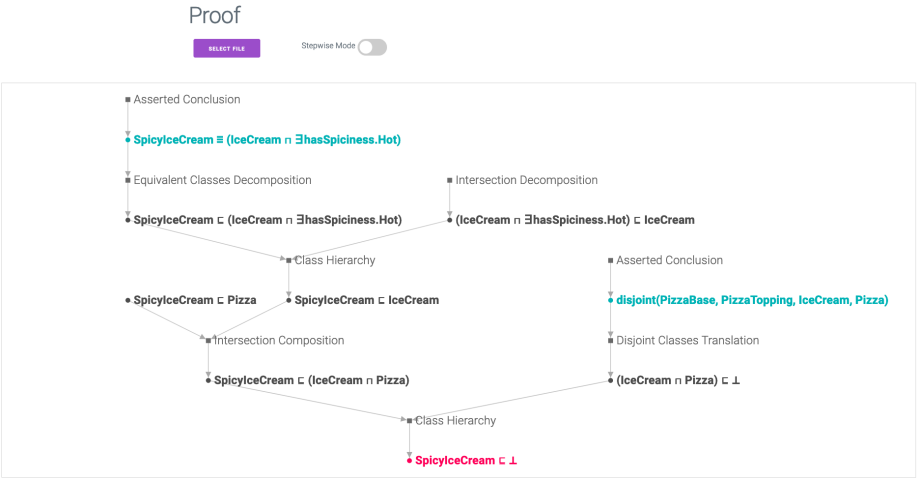

Fig. 1. Screenshot of our tool showing *Proof Component*

Because a proof can be reasonably large and hard to understand at a glance, we provide interactive techniques for navigating the tree and making a large proof easier to understand. The button at the top (cf. Figure 1) loads a chosen proof from a GraphML file and displays it within the component. Clicking on one of the nodes, i.e., axioms of the proof, reveals iconic buttons that allows the user to access navigation and communication functionalities (cf. Figure 2).

The navigation buttons and the switch for the *Stepwise Mode* allow for a stepwise exploration of the proof in both directions, top-down and bottom-up. With top-down and bottom-up, we consider the tree's structure and not the position of the nodes, i.e., bottom is the leaf node level whereas top is the root node.

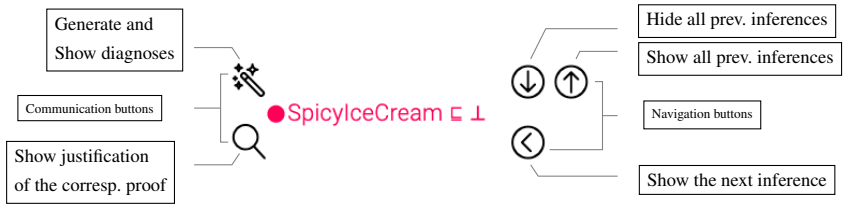

Fig. 2. Buttons associated with axioms in proofs

In the top-down approach, the exploration starts with showing only the final conclusion. Previous inferences can be revealed step-by-step. This allows the user to be fully in control of the exploration and focus on the paths they are interested in. This way, the next node, i.e., inference, is only revealed once the user decides that the visible part has been understood. In the bottom-up approach, i.e., starting from the leaf nodes, the justification, users can collapse certain parts of the proof as understood which thereby

decreases the size of the proof and allows the user to increase the focus by omitting information. By providing both approaches, we allow the users to traverse the proof according to their preferences.

In case of very hard to comprehend parts of the proof, the user can take a *sub-proof* and isolate it from the remaining tree by using the ”delink” button on the edge to the following node (cf. Figure 3). This provides a detail view of all inferences that lead to the chosen link. By clicking on the red node on the bottom, the user is taken back to the original view of the proof.

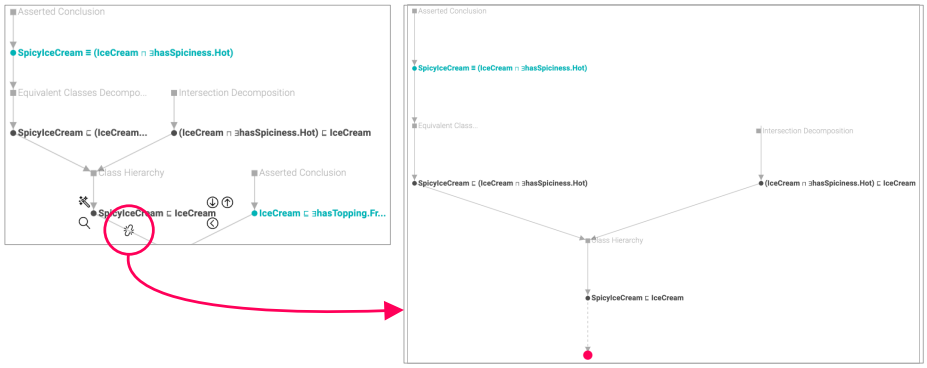

Fig. 3. Clicking on the *delink* icon next to the link in the left image isolates the sub-proof starting from this connection, which can be seen in the right image.

Besides their large size, proofs can also be hard to understand because of the rules that are applied in each step. The rule might be puzzling because the user might not be familiar with it or because the axiom itself is too long to easily spot the application of the rule. To aid the user in comprehending these rules and inference steps, we designed a tooltip that can be invoked by clicking on a particular rule (cf. Figure 4). The tooltip provides the following contents: The abstract rule using substitute variables for the concepts, the current instance of the rule below the abstract rule, and a color encoding that explains how the abstract rule was applied.

The Ontology Component The ontology component computes the diagnoses for the defects and visualizes their impact through the ontology’s modular structure. Its role is to put the proof in context and give an overview of the whole ontology. As shown in Figure 5, the component consists of two parts, the modular structure as a node-link diagram and the part that shows possible diagnoses for a defect. Regarding the layout of the node-link diagram, users can either use a pre-computed force-directed layout [13], arrange the nodes by dragging to create a custom layout which can then be saved, or load a layout. We provide two types of labels for the nodes. The default label lists all axioms that are contained in the node whereas the other option labels the node with the signature of axioms in this node, such as *Toppings*, while respecting the dependency between the nodes. A collapsed side menu shows the computed diagnoses for a certain axiom.

Fig. 4. Explanation of an instance of the `Intersection Composition` rule.

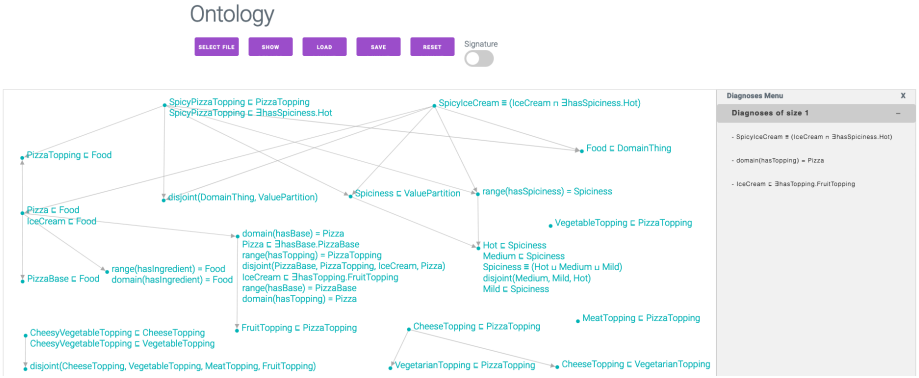

Fig. 5. Screenshot of our tool showing the *Ontology Component*

These diagnoses are grouped into collapsed panels based on their size to add structure to the otherwise convoluted list. Hovering over a diagnosis highlights the corresponding axioms in the node-link diagram by changing their color. Using linked brushing [2] allows us to explore the relations between the different types of data [27]. It also changes the color of all nodes that contain these axioms and their predecessors which shows the impact of the diagnosis on the ontology.

Interplay Between the Components We provide two techniques that connect the two main components of our tool, which are both triggered in the *Proof Component*. However, the effects are shown in the *Ontology Component*. The first technique is highlighting diagnoses. A user can select any node, i.e., axiom, in the *Proof Component* by clicking on it and choosing the button displaying a magic wand (cf. Figure 2) to generate all subset-minimal diagnoses for the selected axiom. This computes a list of diagnoses which is then shown in the *Ontology Component* in the side menu. The second technique is highlighting a justification of an axiom. By clicking the button displaying the magnifying glass, the corresponding node and axiom in the *Ontology Component* can be highlighted through color change. This is an especially important feature for repairing ontologies as it helps users to see the axioms in the ontology that lead to the selected one in the currently considered proof.

3 Prototype Implementation

Our prototype was implemented as a web application using Express[6] and Node.js[7] on the server side. The Node Express server supports multiple clients, which can either display the *Proof Component* or the *Ontology Component*. Thereby, we allow the user to distribute the views according to their preferences on different displays or even on different devices in the same network. We made this choice early on to ensure that we have the possibility to extend the usage of our tool to multi-device environments (MDEs). On the client side, we use D3.js[8] for the visualizations and the CSS framework Milligram[9]. Furthermore, the tool supports GraphML files to load the data and to load and store the layouts. Diagnoses and highlighting is calculated in separate Java applications. To enable the clients to properly communicate among themselves and with the server, we use Socket.IO[10].

4 Adaptations and Expert Interview

In this section, we elaborate on the insights we gained from implementing the prototype and from an interview with domain experts.

4.1 Adaptions for Natural Interaction on Interactive Surfaces

From the beginning of the design, we had the idea of extending our application to also work with natural interaction, such as pen and touch input. Touch devices have become ubiquitous and are frequently used for analysis tasks [19, 11]. Therefore, we made specific design choices to ensure that the tool is extendable in the future, such as having a client-server architecture that allows for a multi-device setup. Regarding our interface, we created interactive mockups using Adobe XD for a future design iteration of two main parts: (1) the design of the nodes and (2) the navigation of the proof.

As mentioned in the previous sections, we want to provide the user with a variable top-down and bottom-up approach. To improve our current design, we were inspired by natural and fluid interaction [8] for visualizations. In the fully collapsed state of a proof (cf. Figure 6), only the leaf and the root node are visible. All other nodes are hidden behind a *Magic Rule*. By pulling a node away from the *Magic Rule*, the next step that was hidden behind the *Magic Rule* is revealed. Pushing a node back towards the *Magic Rule* hides the last step of the proof. This approach can be used until there is no *Magic Rule* left as well as to create several *Magic Rules* within one, possibly large, proof.

As shown in Figure 7, the new node design is less minimal than the old one and includes all functionality within the node instead of using the space around the node. It has two states: the collapsed, i.e., default, and the expanded state. The collapsed state only provides the most important features, i.e., expanding and collapsing the child or

[6] http://expressjs.com/
[7] https://nodejs.org/en/
[8] https://d3js.org/
[9] https://milligram.io/
[10] https://socket.io/

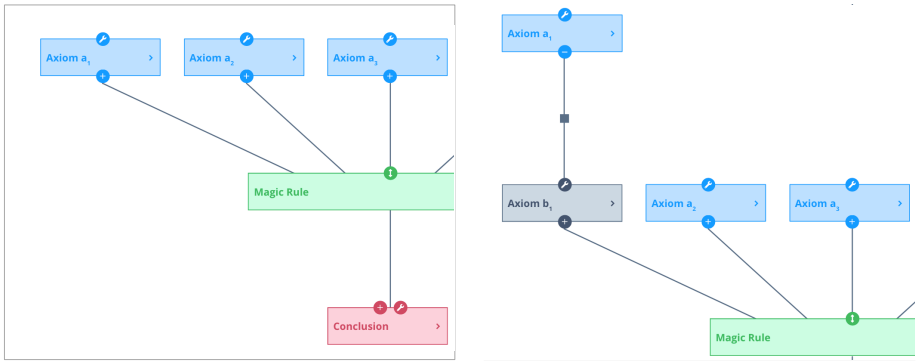

Fig. 6. Left: A proof with only leaf and root nodes being visible, Right: After pulling on axiom a_1 to reveal another step of the proof.

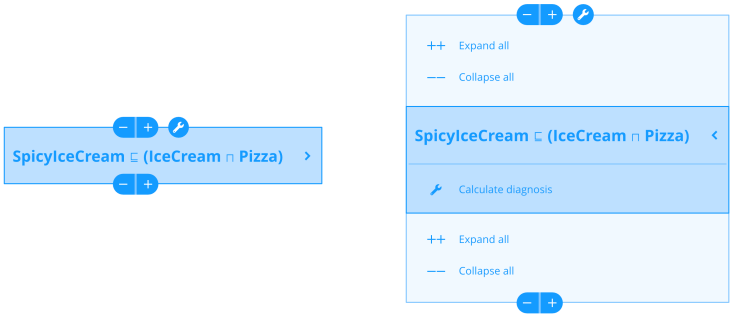

Fig. 7. Left: A node of an axiom in its collapsed state, Right: The node in its expanded state showing additional functionalities

parent node. The expanded state, however, allows access to all features and additionally explains features that are otherwise only accessible through an icon, e.g., expanding and collapsing.

Even though the designs are not yet implemented in the prototype, we aim to provide the users with a more natural and thus intuitive interaction for working with a proof and an ontology by improving the overall node design and the navigation techniques.

4.2 Expert Interview

As part of our iterative design process, we conducted an interview with three domain experts working daily with Description Logics ontologies. These were not the same experts that took part in our initial analysis sessions (cf. subsection 2.1). We showed them our prototype and explained how we imagined the workflow with the application. Additionally, after receiving their feedback on the current prototype, we also presented the design adaptations (cf. subsection 4.1). The comments were mostly about the theoretical background of the modular structure we use for the *Ontology Component* since

it is a rather uncommon way of representing an ontology. This structure was not immediately clear to the users and could only be fully understood after an explanation and discussion with one of the authors. This, however, is conflicting with our goal of being easy to use for domain experts. Therefore, we consider switching to a more traditional concept-based approach for the *Ontology Component*. The experts mentioned a lot of minor design and implementation related issues, like improving icons or increasing the font size. Most of the minor comments are already addressed by our design revisions. Especially, our new version of navigating the proof and the introduction of the *Magic Rule* received very favorable comments. In general, the experts were convinced by the concept itself but expressed concerns about the modular structure of the ontology and whether it would be reasonably easy to understand and thereby helpful.

5 Conclusion

We presented an interface that visualizes proofs and the corresponding ontology as side-by-side node-link diagrams to facilitate ontology debugging of unwanted entailments. We provided an overview of our visualization and interaction design including the *Proof Component* and the *Ontology Component* and some technical details. Furthermore, we elaborated on further adaptations for natural interaction and interactive surfaces and the feedback we gained from an expert interview. In the future, we aim to incorporate our adaptions into our prototype and conduct a larger user study to further investigate the potential and utility of interactive visualizations for ontology engineering.

References

[1] Franz Baader et al., eds. *The Description Logic Handbook: Theory, Implementation and Applications*. 2nd ed. Cambridge University Press, 2007. DOI: 10.1017/CBO9780511711787.

[2] Richard A. Becker and William S. Cleveland. "Brushing Scatterplots". In: *Technometrics* 29.2 (May 1987), pp. 127–142. ISSN: 0040-1706. DOI: 10.1080/00401706.1987.10488204.

[3] W. Chen et al. "Structure-Based Suggestive Exploration: A New Approach for Effective Exploration of Large Networks". In: *IEEE Transactions on Visualization and Computer Graphics* 25.1 (Jan. 2019), pp. 555–565. ISSN: 1077-2626. DOI: 10.1109/TVCG.2018.2865139.

[4] Andy Cockburn, Amy Karlson, and Benjamin B. Bederson. "A Review of Overview+detail, Zooming, and Focus+context Interfaces". In: *ACM Comput. Surv.* 41.1 (Jan. 2009). ISSN: 0360-0300. DOI: 10.1145/1456650.1456652.

[5] C. Collins and S. Carpendale. "VisLink: Revealing Relationships Amongst Visualizations". In: *IEEE Transactions on Visualization and Computer Graphics* 13.6 (Nov. 2007), pp. 1192–1199. ISSN: 1077-2626. DOI: 10.1109/TVCG.2007.70521.

[6] Marek Dudáš et al. "Ontology visualization methods and tools: a survey of the state of the art". In: *The Knowledge Engineering Review* 33 (2018). DOI: 10.1017/S0269888918000073.

[7] Philipp Eichmann et al. "Orchard: Exploring Multivariate Heterogeneous Networks on Mobile Phones". In: *Computer Graphics Forum* (2020). ISSN: 1467-8659. DOI: 10.1111/cgf.13967.

[8] Niklas Elmqvist et al. "Fluid interaction for information visualization". en. In: *Information Visualization* 10.4 (Oct. 2011), pp. 327–340. ISSN: 1473-8716. DOI: 10.1177/1473871611413180.

[9] Mathias Frisch, Jens Heydekorn, and Raimund Dachselt. "Investigating Multi-touch and Pen Gestures for Diagram Editing on Interactive Surfaces". In: *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*. ITS '09. New York, NY, USA: ACM, 2009, pp. 149–156. ISBN: 978-1-60558-733-2. DOI: 10.1145/1731903.1731933.

[10] Michael Gleicher et al. "Visual comparison for information visualization". In: *Information Visualization* 10.4 (2011), pp. 289–309. DOI: 10.1177/1473871611416549.

[11] P. Isenberg et al. "Data Visualization on Interactive Surfaces: A Research Agenda". In: *IEEE Computer Graphics and Applications* 33.2 (Mar. 2013), pp. 16–24. ISSN: 0272-1716. DOI: 10.1109/MCG.2013.24.

[12] Aditya Kalyanpur et al. "Repairing Unsatisfiable Concepts in OWL Ontologies". In: *The Semantic Web: Research and Applications, 3rd European Semantic Web Conference, ESWC 2006, Budva, Montenegro, June 11-14, 2006, Proceedings*. Vol. 4011. Lecture Notes in Computer Science. Springer, 2006, pp. 170–184. DOI: 10.1007/11762256_15.

[13] Tomihisa Kamada and Satoru Kawai. "An algorithm for drawing general undirected graphs". In: *Information Processing Letters* 31.1 (1989), pp. 7–15. DOI: https://doi.org/10.1016/0020-0190(89)90102-6.

[14] Nam Wook Kim et al. "DataToon: Drawing Dynamic Network Comics With Pen + Touch Interaction". In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. CHI '19. event-place: Glasgow, Scotland Uk. New York, NY, USA: ACM, 2019, 105:1–105:12. ISBN: 978-1-4503-5970-2. DOI: 10.1145/3290605.3300335.

[15] Holger Knublauch et al. "The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications". In: *The Semantic Web – ISWC 2004*. Ed. by Sheila A. McIlraith, Dimitris Plexousakis, and Frank van Harmelen. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 229–243. ISBN: 978-3-540-30475-3.

[16] R. Langner, T. Horak, and R. Dachselt. "VISTILES: Coordinating and Combining Co-located Mobile Devices for Visual Data Exploration". In: *IEEE Transactions on Visualization and Computer Graphics* PP.99 (2017), pp. 1–1. ISSN: 1077-2626. DOI: 10.1109/TVCG.2017.2744019.

[17] R. Langner, U. Kister, and R. Dachselt. "Multiple Coordinated Views at Large Displays for Multiple Users: Empirical Findings on User Behavior, Movements, and Distances". In: *IEEE Transactions on Visualization and Computer Graphics* 25.1 (Jan. 2019), pp. 608–618. ISSN: 1077-2626. DOI: 10.1109/TVCG.2018.2865235.

[18] Alexandra Lee, Daniel Archambault, and Miguel Nacenta. "Dynamic Network Plaid: A Tool for the Analysis of Dynamic Networks". In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. CHI '19. event-place: Glasgow, Scotland Uk. New York, NY, USA: ACM, 2019, 130:1–130:14. ISBN: 978-1-4503-5970-2. DOI: 10.1145/3290605.3300360.

[19] Bongshin Lee et al. "Beyond Mouse and Keyboard: Expanding Design Considerations for Information Visualization Interactions". In: *IEEE Transactions on Visualization and Computer Graphics* 18.12 (Dec. 2012), pp. 2689–2698. ISSN: 1077-2626. DOI: 10.1109/TVCG.2012.204.

[20] Bongshin Lee et al. "TreePlus: Interactive Exploration of Networks with Enhanced Tree Layouts". In: *IEEE Transactions on Visualization and Computer Graphics* 12.6 (Nov. 2006), pp. 1414–1426. ISSN: 1077-2626. DOI: 10.1109/TVCG.2006.106.

[21] A. Lex et al. "Caleydo: Design and evaluation of a visual analysis framework for gene expression data in its biological context". In: *2010 IEEE Pacific Visualization Symposium (PacificVis)*. Mar. 2010, pp. 57–64. DOI: 10.1109/PACIFICVIS.2010.5429609.

[22] T. Major and R. C. Basole. "Graphicle: Exploring Units, Networks, and Context in a Blended Visualization Approach". In: *IEEE Transactions on Visualization and Computer Graphics* 25.1 (Jan. 2019), pp. 576–585. ISSN: 1077-2626. DOI: 10.1109/TVCG.2018.2865151.

[23] Nicolas Matentzoglu et al. "Inference Inspector: Improving the verification of ontology authoring actions". In: *Journal of Web Semantics* 49 (2018), pp. 1–15. ISSN: 1570-8268. DOI: https://doi.org/10.1016/j.websem.2017.09.004.

[24] Enrico Motta et al. "A Novel Approach to Visualizing and Navigating Ontologies". In: *The Semantic Web – ISWC 2011*. Ed. by Lora Aroyo et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 470–486. ISBN: 978-3-642-25073-6.

[25] Tamara Munzner. *Visualization Analysis and Design*. A K Peters Visualization Series. CRC Press, Dec. 2014. ISBN: 9781466508910.

[26] C. Nobre et al. "Lineage: Visualizing Multivariate Clinical Data in Genealogy Graphs". In: *IEEE Transactions on Visualization and Computer Graphics* 25.3 (Mar. 2019), pp. 1543–1558. ISSN: 1077-2626. DOI: 10.1109/TVCG.2018.2811488.

[27] J. C. Roberts. "State of the Art: Coordinated Multiple Views in Exploratory Visualization". In: *Fifth International Conference on Coordinated and Multiple Views in Exploratory Visualization (CMV 2007)*. July 2007, pp. 61–71. DOI: 10.1109/CMV.2007.20.

[28] Ramik Sadana and John Stasko. "Designing Multiple Coordinated Visualizations for Tablets". en. In: *Computer Graphics Forum* 35.3 (June 2016), pp. 261–270. ISSN: 1467-8659. DOI: 10.1111/cgf.12902.

[29] H. Schulz. "Treevis.net: A Tree Visualization Reference". In: *IEEE Computer Graphics and Applications* 31.6 (Nov. 2011), pp. 11–15. ISSN: 1558-1756. DOI: 10.1109/MCG.2011.103.

[30] H. Schulz, S. Hadlak, and H. Schumann. "The Design Space of Implicit Hierarchy Visualization: A Survey". In: *IEEE Transactions on Visualization and Computer Graphics* 17.4 (Apr. 2011), pp. 393–411. ISSN: 1941-0506. DOI: 10.1109/TVCG.2010.79.

[31] Margaret-Anne Storey et al. "Jambalaya: Interactive visualization to enhance ontology authoring and knowledge acquisition in Protégé". In: *Workshop on Interactive Tools for Knowledge Capture (K-CAP-2001)*. 2001.

[32] Matthew Tobiasz, Petra Isenberg, and Sheelagh Carpendale. "Lark: Coordinating Co-located Collaboration with Information Visualization". In: *IEEE Transactions on Visualization and Computer Graphics* 15.6 (Nov. 2009), pp. 1065–1072. ISSN: 1077-2626. DOI: 10.1109/TVCG.2009.162.