nev: A Visual Query Tracer and Builder for Declarative Logic Programming on Nemo

J. Méndez[1], L. Gerlach[2], T. Wieland[1], A. Ivliev[2], M. Krötzsch[2], R. Dachselt[1]

[1]Interactive Media Lab Dresden, TU Dresden, Germany
[2]Knowledge-Based Systems Group, TU Dresden, Germany

Abstract

We introduce Nemo Explain Visualizer (nev), an interactive visual query tracer and builder that communicates with the web interface of the Datalog engine Nemo. Scarce visual resources exist to support learning and reasoning with Datalog and similar declarative logic programming languages. Therefore, we designed nev as a web-based visualization tool that not only allows inspection of computation traces (for derived facts), but also shape-based proof tree querying and explanations.

CCS Concepts

• ***Human-centered computing*** → ***Visualization systems and tools;*** *Visual analytics;* • ***Theory of computation*** → *Constraint and logic programming; Programming logic;*

1. Introduction & Related Work

Declarative logic programming languages like *Datalog* have recently received renewed interest for source code analysis (e.g., Semmle/Github), database query (e.g., Google Logica), web data extraction, and many other fields [Krö25]. In practice, Datalog extensions enable features such as datatypes (numbers, strings, etc.), negation, aggregation, or existentially quantified variables. Prominent examples with subsets of these features are *RDFox* [NPM*15], *Soufflé* [JSS16], *VLog* [CDG*19], and our own *Nemo* [IGM*24], which we build upon. Visualization has supported declarative logic programming since early 1990s works [Wie90, ARR*93] with the intuition carried onto modern-day approaches [CGS08, KLS12]. However, modern tools [BOP*13, CGP*18, AR24, BGMS25] rarely exploit the declarative nature of logic programs using visualization. Soufflé [JSS16] and *Nemo* [IGM*24] are capable of computing derivation tree structures, but only a recent prototype we developed for *Nemo* has attempted to visualize them [GIM*24].

Our work constitutes a web-based programming environment consisting of (1) the Datalog reasoner *Nemo*; (2) the programming frontend *Nemo Web*: a dedicated editor (with syntax highlighting, error checking, etc.) with runtime controls; and (3) the *Nemo Explain Visualizer* (*nev*): a visual interface to interactively manipulate query traces and retrieve computed facts on demand. *Nemo* and *Nemo Web* existed before [IGM*24], but can now compute explanation data on demand and in new ways that enable its dynamic exploration. The interactive visualizer *nev* is completely new, inspired by query execution trace visualizers (i.e., [Dal25, Chr26, Pet26, Noë25]) for relational databases, and proof visualizers (e.g., [Wie90, ARR*93, AF07, KKS17, MAK*23]) for ontology reasoning. The traces and proofs supported by these systems are often trees or directed acyclic graphs (DAGs), and they may contain repeating patterns and recursive structures. Naturally, well-established visualization techniques are employed to explore node-link diagrams [CC07], such as linking multiple coordinated views [Rob07] and showing details on demand within nodes and onto juxtaposed views [JE12]. Furthermore, the survey by Dudáš et al. [DLSP18] presents typical components of ontology visualization tools that we incorporated in *nev*, such as undo/redo, graphical zoom and pan, entity focus, and search and highlight.

All of our tools are free and open source, with all resources summarized at `https://imld.de/nev`. Our Datalog environment is fully web-based and thus without installation requirements. It is also scalable to datasets of millions of facts (see reported benchmarks [IGM*24]). However, *nev*'s main contribution is its support for querying explanations based on their *shape*, which can be used to discover computation patterns.

2. Visual Trace Inspection and Editing for *Nemo*

In Datalog, if a *fact* states that $A(c)$ holds, then the rule $B(x) \leftarrow A(x)$ allow us to infer that $B(c)$ is also true. Thus, by the very nature of the semantics of a Datalog program, we can backtrack the rules that contributed to the inference, starting from initial facts and resulting in a traced *proof tree*. We refer to *Nemo*'s queries as *T1* and *T2*. In essence, one can query for a *fact* (*T1*), which returns a proof tree, and for the *shape* of a proof tree (*T2*), which returns the given tree with facts from all traces that include a partial proof tree of the given shape. *T1* explains individual facts, while *T2* was introduced for debugging purposes, to see how different rule applica-

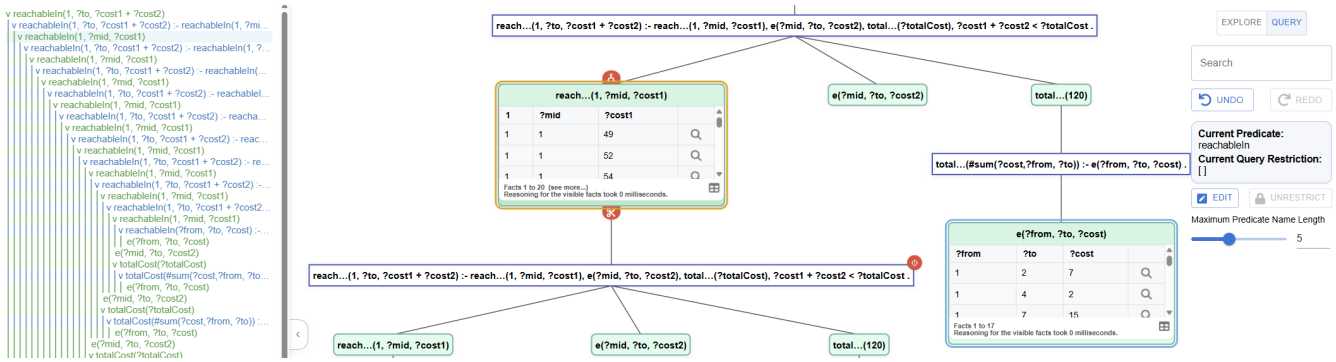

Figure 1: *Screenshot of* nev*, illustrating its* trace explorer *(indented tree) on the left and main* visual editor *on the center. The floating menu on the right shows the search bar, undo/redo, query restriction, and other configuration options.*

tions interact without having to change the program. Additionally, execution times of the individual rule applications are included in the responses, which can assist in analyzing performance. For example, when only querying for a single rule in a *T2*, this will yield that total time that *Nemo* spent on executing this rule in the reasoning process. Textually handling *T1* and *T2* inputs and responses would be very challenging, so we developed *nev* as a web-based visualization tool that handles these queries by communicating with *Nemo Web*. Since *T2* uses the trace as input and output, *nev* simultaneously supports trace editing and inspection.

nev consists of an *explorer* (indented tree), a main *visual editor*, and a *fact-tables* view. The explorer and editor (visible in Figure 1) distinguish between two types of nodes: *predicates* using green highlights (i.e., border and fill color) and *rules* using blue borders. The rule labels of the merged proof tree are decoupled into their own auxiliary nodes for visual clarity. The indented tree can be used to highlight nodes on the main view (on hover) or to bring them to the center of the view (on click). Clicking on a predicate node in the main view reveals the facts associated with it on a small table embedded in the node. These tables can be pinned to the fact-tables view at the bottom for inspection (using pagination and counts) and comparison. Unlike the modal dialog approach previously available to *Nemo Web* [GIM*24], *nev* is loaded onto a separate browser tab. The user can thus configure their screen(s) flexibly (e.g., side-by-side, different monitors) without cluttering the code editor view. However, the most versatile aspect of *nev* is its communication with *Nemo Web* on the basis of *T2*. For *T1*, we refer to a *query restriction* that specifies derived facts to look for. This restriction can be lifted using the settings located to the right. Without it, visually editing the tree shape triggers *T2*s to be sent to *Nemo Web*, which populates *nev*'s views accordingly. Our visualization may be used in two modes. On **query editing** mode, users may *add or remove rules* using the red and green buttons atop and at the bottom of predicate nodes. Users can also *focus on a rule*, which constructs a *T2* using only the rule and its head and body predicates. This is equivalent to removing all nodes above and below the rule on focus, and is useful for inspecting all applications of a specific rule across all possible derivations. Since the sense-making process involves a lot of back and forth between the different traces corresponding to these features, we also incorporate history features (undo and redo) that save the state of the tree (including restrictions) after any modification. To mitigate the risk of accidental clicks, as well as to support incremental inspection, we also enable an **exploration mode** that disables editing, where the users can collapse and expand the tree from any node, and focus on specific rules by blurring the rest of the tree instead of removing it. In both modes, for cases where the predicate names are long, the main view can be configured through a slider to show a maximum predicate length and replace the exceeding text with ellipsis. Lastly, we provide a search feature to look for specific predicate tuples, which searches within all nodes and highlights any matches.

Implementation: We used TypeScript, React, and Vite to implement *nev* as a static web application. The React component library Material UI (MUI) defines styling and the standard d3-flextree layout is used to visualize the traces. The tables employ MUI DataGrid [MUI26], but the pagination is custom, achieved through *T2* properties. This helps reduce memory requirements. The serverless browser tab messaging uses the Broadcast Channel API [MDN26].

3. Conclusion

We presented *nev*, the new addition to our web-based *Nemo* development environment. It merges visual analysis, debugging, and query building tasks on the basis of queries that use the shape of inference trees to debug and understand patterns within the logic programs. Considering the scarcity of modern resources to visually explain and learn Datalog-like languages, *nev* constitutes a valuable contribution for declarative logic programming.

Acknowledgements

This work is funded by Deutsche Forschungsgemeinschaft (DFG) under Germany's Excellence Strategy: EXC 2050/2, 390696704 – "Centre for Tactile Internet" (CeTI); by DFG grant 389792660 as part of TRR 248 – CPEC; 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); and by BMBF and German Academic Exchange Service (DAAD) in project 57616814 (SECAI, School of Embedded and Composite AI).

References

[AF07] ADACHI Y., FURUSAWA Y.: Logichart: A prolog program diagram and its layout. *Electronic Communications of the European Association of Software Science and Technology 7* (2007). doi:10.14279/TUJ.ECEASST.7.95. 1

[AR24] ALVIANO M., REINERS L. A. R.: ASP chef: Draw and expand. In *Proceedings of the 21st International Conference on Principles of Knowledge Representation and Reasoning* (2024), Marquis P., Ortiz M., Pagnucco M., (Eds.). doi:10.24963/KR.2024/68. 1

[ARR*93] ARORA T., RAMAKRISHNAN R., ROTH W. G., SESHADRI P., SRIVASTAVA D.: Explaining program execution in deductive systems. In *Deductive and Object-Oriented Databases, Third International Conference* (1993), Ceri S., Tanaka K., Tsur S., (Eds.), vol. 760 of *LNCS*, Springer, pp. 101–119. doi:10.1007/3-540-57530-8_7. 1

[BGMS25] BELLOMARINI L., GENTILI A., MAGNANIMI D., SALLINGER E.: Vadacode: A logician-friendly IDE for Datalog+/-. *Proceedings of the VLDB Endowment 18*, 12 (2025), 5411–5414. URL: https://www.vldb.org/pvldb/vol18/p5411-gentili.pdf, doi:10.14778/3750601.3750684. 1

[BOP*13] BUSONIU P., OETSCH J., PÜHRER J., SKOCOVSKY P., TOMPITS H.: SeaLion: An eclipse-based IDE for answer-set programming with advanced debugging support. *Theory and Practice of Logic Programming 13*, 4-5 (2013), 657–673. doi:10.1017/S1471068413000410. 1

[CC07] COLLINS C., CARPENDALE S.: VisLink: Revealing Relationships Amongst Visualizations. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1192–1199. doi:10.1109/TVCG.2007.70521. 1

[CDG*19] CARRAL D., DRAGOSTE I., GONZÁLEZ L., JACOBS C., KRÖTZSCH M., URBANI J.: VLog: A rule engine for knowledge graphs. In *The Semantic Web – ISWC* (2019), Ghidini C., Hartig O., Maleshkova M., Svátek V., Cruz I., Hogan A., Song J., Lefrançois M., Gandon F., (Eds.), Springer International Publishing, pp. 19–35. doi:10.1007/978-3-030-30796-7_2. 1

[CGP*18] CALIMERI F., GERMANO S., PALERMITI E., REALE K., RICCA F.: Developing ASP programs with ASPIDE and LoIDE. *KI - Künstliche Intelligenz 32*, 2-3 (2018), 185–186. doi:10.1007/S13218-018-0534-Z. 1

[CGS08] CABALLERO R., GARCÍA-RUIZ Y., SÁENZ-PÉREZ F.: A theoretical framework for the declarative debugging of datalog programs. In *Semantics in Data and Knowledge Bases, 3rd International Workshop* (2008), Schewe K., Thalheim B., (Eds.), vol. 4925 of *LNCS*, Springer, pp. 143–159. doi:10.1007/978-3-540-88594-8_8. 1

[Chr26] CHRISTOFIDES M.: pgMustard: Review Postgres query plans quickly. https://www.pgmustard.com/, 2026. Accessed: Jan 2026. 1

[Dal25] DALIBO: Pev2. https://github.com/dalibo/pev2, 2025. Accessed: 2025-07-01. 1

[DLSP18] DUDÁŠ 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. doi:10.1017/S0269888918000073. 1

[GIM*24] GERLACH L., IVLIEV A., MÉNDEZ J., MEUSEL S., DACHSELT R., KRÖTZSCH M.: EvonNemo - a symbiosis of Datalog tracing and proof tree visualization. In *The 5th Workshop on Explainable Logic-Based Knowledge Representation* (11 2024). URL: https://imld.de/cnt/uploads/2024-XLoKR-EvonNemo.pdf. 1, 2

[IGM*24] IVLIEV A., GERLACH L., MEUSEL S., STEINBERG J., KRÖTZSCH M.: Nemo: Your friendly and versatile rule reasoning toolkit. In *Proceedings of the 21st International Conference on Principles of Knowledge Representation and Reasoning* (2024), Marquis P., Ortiz M., Pagnucco M., (Eds.), IJCAI Organization, pp. 743–754. doi:10.24963/kr.2024/70. 1

[JE12] JAVED W., ELMQVIST N.: Exploring the design space of composite visualization. In *2012 IEEE Pacific Visualization Symposium* (2012), pp. 1–8. doi:10.1109/PacificVis.2012.6183556. 1

[JSS16] JORDAN H., SCHOLZ B., SUBOTIC P.: Soufflé: On synthesis of program analyzers. In *Computer Aided Verification - 28th International Conference* (2016), Chaudhuri S., Farzan A., (Eds.), vol. 9780 of *LNCS*, Springer, pp. 422–430. doi:10.1007/978-3-319-41540-6_23. 1

[KKS17] KAZAKOV Y., KLINOV P., STUPNIKOV A.: Towards reusable explanation services in Protégé. In *Proceedings of the 30th International Workshop on Description Logics* (2017), vol. 1879 of *CEUR Workshop Proceedings*. URL: http://www.ceur-ws.org/Vol-1879/paper31.pdf. 1

[KLS12] KÖHLER S., LUDÄSCHER B., SMARAGDAKIS Y.: Declarative datalog debugging for mere mortals. In *Datalog in Academia and Industry - 2nd International Workshop* (2012), Barceló P., Pichler R., (Eds.), vol. 7494 of *LNCS*, Springer, pp. 111–122. doi:10.1007/978-3-642-32925-8_12. 1

[Krö25] KRÖTZSCH M.: Modern Datalog: Concepts, methods, applications. In *Joint Proceedings of the 20th and 21st Reasoning Web Summer Schools (RW 2024 & RW 2025)* (2025), Artale A., Bienvenu M., García Y. I., Murlak F., (Eds.), vol. 138 of *OASIcs*, Dagstuhl Publishing. doi:10.4230/OASIcs.RW.2024/2025.7. 1

[MAK*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. *Computer Graphics Forum* (3 2023). doi:10.1111/cgf.14730. 1

[MDN26] MDN: Broadcast Channel API. https://developer.mozilla.org/en-US/docs/Web/API/Broadcast_Channel_API, 2026. Accessed: Jan 2026. 2

[MUI26] MUI: React Data Grid Component. https://mui.com/x/react-data-grid/, 2026. Accessed: Jan 2026. 2

[Noë25] NOËL M.: Duckdb execution plan visualizer. https://db.cs.uni-tuebingen.de/explain/, 2025. Accessed: 2025-07-01. 1

[NPM*15] NENOV Y., PIRO R., MOTIK B., HORROCKS I., WU Z., BANERJEE J.: RDFox: A highly-scalable RDF store. In *The Semantic Web – ISWC* (2015), Arenas M., Corcho Ó., Simperl E., Strohmaier M., d'Aquin M., Srinivas K., Groth P., Dumontier M., Heflin J., Thirunarayan K., Staab S., (Eds.), vol. 9367 of *LNCS*, Springer, pp. 3–20. doi:10.1007/978-3-319-25010-6_1. 1

[Pet26] PETRY T.: Visual EXPLAIN for MySQL. https://mysqlexplain.com/, 2026. Accessed: Jan 2026. 1

[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* (2007), pp. 61–71. doi:10.1109/CMV.2007.20. 1

[Wie90] WIELAND C. A.: Two explanation facilities for the deductive database management system DeDEx. In *Proceedings of the 9th International Conference on Entity-Relationship Approach* (1990), Kangassalo H., (Ed.), ER Institute, pp. 189–203. 1