TGView3D: a System for 3-Dimensional Visualization of Theory Graphs*

Richard Marcus[0000−0002−6601−6457], Michael Kohlhase[0000−0002−9859−6337], and Florian Rabe[0000−0003−3040−3655]

Computer Science, FAU Erlangen-Nürnberg

Abstract. We describe the TGView3D system, an interactive graph viewer optimized for exploring mathematical knowledge as 3D graphs. To exploit all three spatial dimensions, it extends the commonly-used force-directed layout algorithms with hierarchical components that are more suitable for the typical structure of mathematical knowledge. TGView3D can also communicate with OMDoc-based knowledge management tools to offer semantic, mathematics-specific interaction with the graphs.

1 Introduction

Digital libraries of both informal and formal mathematics have reached enormous sizes. For instance, at least half a dozen theorem prover libraries exceed 10^5 statements. Thus, it is getting more and more difficult to organize this knowledge in a way that humans can understand and access it. While library sources, generated presentations, and IDEs such as PIDE [Wen19] give good access to local knowledge structures, global properties of the induced knowledge spaces are very difficult to assess.

Theory graphs provide a good representation for these global properties: the nodes are **theories** and their edges **theory morphisms** that define interrelations between theories. Concretely, we use OMDoc/MMT [Koh06; RK13], which distinguishes multiple kinds of morphisms for theory graphs: Most importantly, **inclusions** represent the inheritance relation, and **views** represent translations and interpretations.

However, standard graph visualization techniques are not ideal for theory graphs. Inclusions are highly prevalent and induce a directed acyclic subgraph, which captures the primary structure of the graph, in particular the inheritance hierarchy; therefore, they must be prioritized in the layout. Views may introduce cycles or connect very distant theories; therefore, they must be layouted with care to avoid intersecting edges, which can lead to a messy layout, especially in the 2-dimensional case. For example, we have never been satisfied with the visualization and user interaction features that state-of-the-art tools could provide

* The authors were supported by DFG grant RA-1872/3-1, KO 2428/13-1 OAF and EU grant Horizon 2020 ERI 676541 OpenDreamKit. They are also grateful for hardware support from and very helpful discussions about layout algorithms with Roberto Grosso and Marc Stamminger as well as Jonas Müller.

for our own graph of logic formalizations (LATIN; see [Cod+11]), containing (only) a few hundred nodes and many includes representing the modular design of logics and views representing logic translations. We will use this LATIN theory graph as a running example.

Superseding our previous two-dimensional theory graph viewer [RKM17], TGView3D is a three-dimensional theory graph visualization tool that adapts traditional force-directed layout algorithms to make use of hierarchies and clusters in theory graphs, using an approach similar to [DKM06] except extended to three dimensions.

TGView3D is based on the **Unity** game engine [UGE]. While there are dedicated tools for interactive 3D graph visualization such as **Gephi** [BHJ09] or web applications and frameworks (e.g., based on WebGL), we opted for Unity as it allows fast implementation of typical 3D interactions and flexible platform support as well as efficient rendering of large graphs. Unity also allows building two versions of TGView3D: a WebGL version that we can embed into browser-based interfaces for casual users, and executables for VR hardware that offer better performance for power users.

While Unity is proprietary, all of our code is licensed under GPLv3 and is available at `https://github.com/UniFormal/TGView3D`. The web application runs at `https://tgview3d.mathhub.info` and a demo video for the VR executable is available at `https://youtube.com/watch?v=Mx7HSWD5dwg`.

2 Layouting and Interaction

3D Layouting To compute the layout for large graphs, force-directed graph drawing is the typical choice. It introduces forces so that nodes repel each other in general but that connected ones attract each other, aiming at a layout of connected groups of nodes and short edges. However, this approach does not offer special treatment for directed edges, and in theory graphs the directed acyclic inheritance hierarchy is a central cognitive aspect. To better visualize this, layered graph drawing can be used instead. This 2D approach first places the nodes

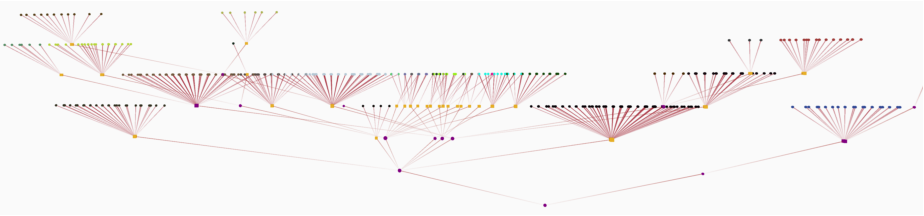

Fig. 1: LATIN Graph with Static Layers

on a minimal number of layers with all edges pointing in the same direction and then minimizes edge crossings by reordering the nodes within their layers. While successful for the inclusion hierarchy, the restriction to layers makes it difficult to incorporate arbitrary additional edges. The latter is needed all the time for

theory graphs, where edges relating distant nodes are among the most interesting. A key benefit of the 3D approach is that we can utilize the third spatial dimensions to devise layout algorithms that cater to the structure of mathematical knowledge. Concretely, we can map the hierarchy to the graph vertically and still get the advantages of force-directed layout algorithms.

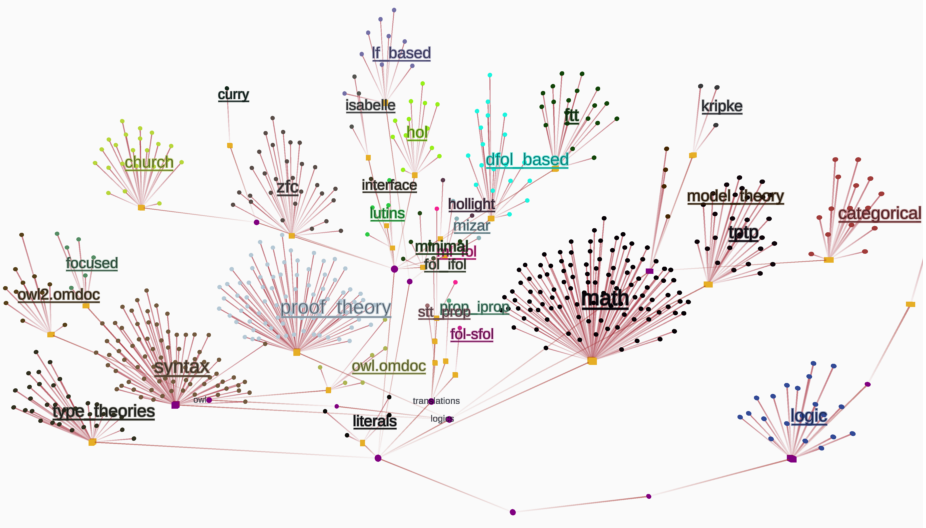

Fig. 2: LATIN Graph with Hierarchic Forces

However, static layers in the style of layered graph drawing (cf. Figure 1) are still problematic in this combination: it prevents nodes from forming groups vertically, which can lead to inefficient use of space, e.g. all nodes could end up on a single layer. Therefore, we use a less restrictive **relative hierarchy** instead, i.e., we aim at a consistent edge direction going from bottom to top. We accomplish this by adding a force that pushes nodes connected by inclusions either downwards or upwards without statically fixing a set of layers. In many cases, this hierarchic force already influences the layout sufficiently to yield good visualizations. But we also added a way to force every node that includes N to appear above N: we first position the nodes in any way that conforms to the hierarchy (e.g., placing them all on the same layer) and then restrict the force-directed node movement so that nodes may only "overtake" each other in the correct direction. This achieves our goals to preserve the relative hierarchy while allowing the force-directed algorithm to work relatively freely (cf. Figure 2).

Now, the layout algorithm can organize the theory graph efficiently: hierarchic relations create a vertical ordering, and minimizing the length of other edges creates node groups. Adding the view edges to the layout in Figure 2 would then reorganize the positions of node clusters but keep the relative hierarchy intact.

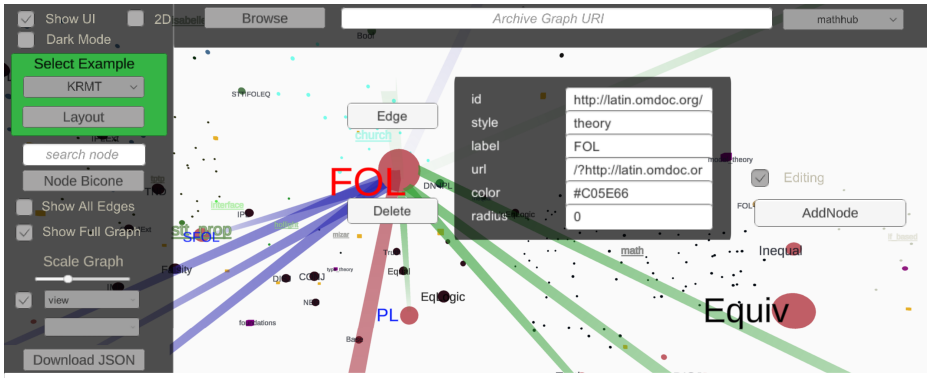

Fig. 3: TGView3D User Interface: Theory FOL within LATIN

Interaction Figure 3 gives an example of the TGView3D user interface. It shows the node for the FOL theory in the LATIN graphs with its attributes. Nodes and edges may be typed, and colors are used to differentiate the types visually. Users explore the theory graph by moving through the 3-dimensional visualization and using interaction features that can be accessed within the UI. Additionally, we provide graph editing features for advanced users like developers or library maintainers, e.g., adding and removing nodes and edges.

Compared to 2D, the nodes have more space to form recognizable clusters, but a problem of the 3D-visualization is the visual overlap induced by the placement of nodes along the third dimensions. To cope with this, TGView3D provides the option to hide parts in the distance, thus presenting the user a vertical slice of the graph. Even so, showing all types of edges at once can still result in cluttered layouts, but, since users often want to focus on certain aspects of the theory graph, the main interaction concepts in TGView3D revolve around giving users control over the layout composition. Accordingly, TGView3D allows the user to hide currently not required edge types and, optionally, recalculate the layout based on this selection. The latter, in particular, can be used to analyze how the types of theory morphisms affect the graph layout and thus to get insights about different dependencies in the theory graph.

Another core feature is following the inheritance hierarchy of inclusions. In practice, this means that we need to support the transition between inspecting the graph globally and exploring local structures. Both are important for mathematical knowledge: looking at the whole graph at once reveals groups of theories and dependencies between these groups, whereas the relation between individual nodes give insights about the respective theories and theory morphisms. Given the limitations of space, separating groups visually by packing nodes closely together will eventually result in too much local overlap, while a more even spread makes it harder to recognize clusters. Therefore, TGView3D gives the user direct control over the node spacing in addition to the possibility of moving through the graph. To allow crawling through the graph and focusing on the local neighbor-

hood of nodes, we give users the option to hide all edges except those of selected nodes. Last, to bridge the gap between local and global exploration, TGView3D can also compute **node bicones**, which show the transitive inclusions of a node, i.e., the two trees of nodes that can be reached by following the inclusion relation forwards and backwards. This gives the user information about the role of an individual node in relation to the full graph.

Hierarchical Clustering In MMT theory graphs, all nodes and edges are labeled in two orthogonal ways: with their logical URI, which follows the namespace structure chosen by the user, and their physical source URL, which follows the project and folder structure of the source files. TGView3D uses this information to define clusters, which are visualized by using the same color for the respective nodes and adding a cluster label. Beyond that, TGView3D permits collapsing these clusters into a single bigger node to reduce graph complexity and enable step-wise graph exploration. In that case, all edges of the original nodes are propagated to the cluster node. This also allows for nested clusters, which is important to efficiently reduce the graph to a size where humans can recognize clear structures and computers can handle the computational load better. With this method, we can compress the graph shown in Figure 2 drastically (cf. Figure 4) and still show all edge types at the same time.

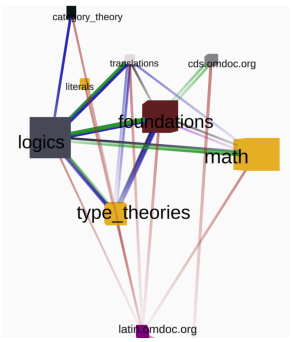

Fig. 4: LATIN Graph: Hierarchic Clustering

Indeed, mathematical libraries often yield large theory graphs with a single connected component, and theory graphs visualizations should not always be self-contained. As an complementary approach to clustering, TGView3D can also be opened with a subgraph built for a particular theory, containing some neighborhood of that theory. The key difference is that instead of collapsing nodes into clusters, the user preselects a certain cluster to reduce the size of the loaded graph. In that case, TGView3D reveals the origin of external nodes and gives users the option to load the respective subgraphs to add them to the current one, thus gradually increasing the size of the visible subgraph.

Integration with Other Systems While TGView3D is a standalone system, one of its key motivations is to serve as a component of our larger MathHub system (hosted at `https://MathHub.info`), a web portal for formal mathematical libraries in OMDoc/MMT format. In particular, the access of subgraphs via namespaces is enabled by the MMT system. For integration with other systems in general, the TGView3D web application is called by URL parameters that govern which graph to load. It can call other systems by opening URLs attached to the nodes and edges, e.g., in response to user interaction. Thus, every library, namespace, and theory viewed

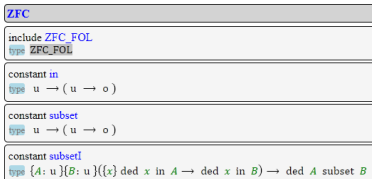

Fig. 5: Source View in MathHub

in MathHub allows opening a corresponding subgraph in TGView3D in a new page. Vice versa, the MMT URI of every node or edge in TGView3D can be used to view the sources of the respective object in MathHub (cf. Figure 5). It is also straightforward to add the functionality of opening nodes and edges in a locally running version of MMT's source editor instead.

3 Conclusion and Future Work

TGView3D is an interactive 3D graph viewer that can handle hierarchical relations and clusters efficiently. While it can handle arbitrary graphs, it is designed to particularly support theory graphs as they occur in mathematical libraries. Therefore, it allows for hierarchical clustering and filtering methods and our layout algorithm makes use of the third spatial dimension to visualize hierarchies and optimize the node organization in a force-directed manner.

In addition to continuous improvements to the graph viewer itself, future work will be to create an ecosystem that simplifies the process of importing different kinds of graphs into TGView3D. Extending this, we want to allow more customizability and offer preconfigured builds that are tailored towards domain-specific use cases.

References

[BHJ09] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. "Gephi: an open source software for exploring and manipulating networks". In: *Third international AAAI conference on weblogs and social media*. 2009.

[Cod+11] Mihai Codescu et al. "Project Abstract: Logic Atlas and Integrator (LATIN)". In: *Intelligent Computer Mathematics*. Ed. by James Davenport et al. LNAI 6824. Springer Verlag, 2011, pp. 289–291. URL: https://kwarc.info/people/frabe/Research/CHKMR_latinabs_11.pdf.

[DKM06] Tim Dwyer, Yehuda Koren, and Kim Marriott. "Drawing directed graphs using quadratic programming". In: *IEEE Transactions on Visualization and Computer Graphics* 12.4 (2006), pp. 536–548.

[Koh06] Michael Kohlhase. *OMDoc – An open markup format for mathematical documents [Version 1.2]*. LNAI 4180. Springer Verlag, Aug. 2006. URL: http://omdoc.org/pubs/omdoc1.2.pdf.

[RK13] Florian Rabe and Michael Kohlhase. "A Scalable Module System". In: *Information & Computation* 0.230 (2013), pp. 1–54. URL: http://kwarc.info/frabe/Research/mmt.pdf.

[RKM17] Marcel Rupprecht, Michael Kohlhase, and Dennis Müller. "A Flexible, Interactive Theory-Graph Viewer". In: *MathUI 2017: The 12th Workshop on Mathematical User Interfaces*. Ed. by Andrea Kohlhase and Marco Pollanen. 2017. URL: http://kwarc.info/kohlhase/papers/mathui17-tgview.pdf.

[UGE] *Unity Game Engine*. URL: https://unity3d.com (visited on 03/07/2019).

[Wen19] Makarius Wenzel. "Interaction with Formal Mathematical Documents in Isabelle/PIDE". In: *Intelligent Computer Mathematics (CICM) 2019*. Ed. by Cezary Kaliszyck et al. LNAI 11617. Springer, 2019, pp. 1–15. DOI: 10.1007/978-3-030-23250-4.