A Flexible, Interactive Theory-Graph Viewer

Marcel Rupprecht Michael Kohlhase Dennis Müller

Computer Science, FAU Erlangen-Nürnberg

Abstract

Many mathematical software systems are based on the explicit representation of mathematical knowledge and organize that modularly. Surprisingly, most of these systems do not give the user a direct way to interact with the represented knowledge via its structure.

In this paper we present TGView, a flexible, adaptable theory graph viewer for the OMDoc/MMT ecosystem. Graph layout and interaction are browser-based given theory graph data provided by the MMT API. As OMDoc/MMT is a general integration format for mathematical knowledge, TGView can serve as an interface for any system that has an OMDoc/MMT interface.

1 Introduction

Many mathematical software systems are based on the explicit representation of mathematical knowledge and organize that modularly. Computer algebra systems like GAP [GAP] or Sagemath [Sage] organize mathematical objects in a class hierarchy and dispatch computational methods along that. Axiom [OA] even adds axiomatizations to the object classes to encode system knowledge. Theorem prover system often organizes axioms, definitions, theorems, and proofs into theories, which are connected by inclusions and other theory morphisms (theory graphs). IMPS [FGT93] has pioneered the "little theories paradigm", and the dominant provers like Coq [Coq], Isabelle [Pau94], and PVS [ORS92] follow its intuitions; even the Mizar [Miz] system has a system of articles that can be understood as a lightweight theory graph.

But the use of modular theory-graph like knowledge organization principles is not restricted to computational and formal systems: OMDoc-based active documents [Koh+11] are generated from a theory graph of document fragments, and the SMGloM terminology [Koh14] uses theory graphs to model both domain concept dependencies as well as multilinguality. Linguistic relations like hypernymy or antonymy between technical terms can be derived from the theory graph structure.

Even at the very informal end graphs abound, for instance Wikipedia is often seen as a large "citation graph" of knowledge and books sometimes end the preface with a graph-like diagram of chapter dependencies to allow readers multiple paths through the book.

Surprisingly, most of the systems discussed above do not give the user a direct way to interact with the represented knowledge via its graph structure. Many of the systems do allow to generate a static graph via the Graphviz system [GV]. Even though one can add limited interactions by adding links to SVG versions of the graph, any change leads to server-side re-layouting, which cannot take display requirements of the client into account. An exception to this rule is the Protégé system for ontology development, which sports numerous graph display plugins [PV] for interaction with the ontologies.

For theory-graph-based systems, understanding the modular structure of theories and their connections is crucial for accessing the knowledge in the system and understanding its behavior, and visualizing them as graphs is the most natural way of conveying this knowledge. However, most such graphs are much too large for normal screens, if presented in their full glory. Therefore, the user needs to interact pan, tilt, and zoom, the graph, colorize, hide, and cluster nodes, and drill down on the information of the theories and morphisms. It is this important interaction aspect that is not supported in the current state of the art.

In this paper we present TGView, a flexible, adaptable theory graph viewer for the OMDoc/MMT ecosystem. Graph layout and interaction are browser-based given theory graph data provided by the MMT API. As OMDoc/MMT is a general integration format for mathematical knowledge, TGView can serve as an interface for any system that has an OMDoc/MMT interface. In the next section we introduce the OMDoc/MMT ecosystem before we discuss the TGView system in Section 3. Section 4 concludes the paper.

2 The OMDoc/MMT Ecosystem

OMDoc/MMT is a knowledge representation system building on OMDoc, a wide-coverage representation language for mathematical knowledge (formal) and documents (informal/narrative). The format is designed to be foundation-independent and introduces several concepts to maximize modularity and to abstract from and mediate between different foundations, to reuse concepts, tools, and formalizations.

OMDoc/MMT offers very few primitives, which have turned out to be sufficient for most practical settings. These are

1. *constants* with optional types and definitions; these are *objects*, which are syntax trees with binding, using previously defined constants as leaves,
2. *theories*, which are lists of constant declarations and
3. *theory morphisms*, that map declarations in a domain theory to expressions built up from declarations in a target theory.

Using these primitives, logical frameworks, logics and theories *within* some logic are all uniformly represented as OMDoc/MMT theories, rendering all of those equally accessible, reusable and extendable. Constants, functions, symbols, theorems, axioms, proof rules etc. are all represented as constant declarations, and all terms which are built up from those are represented as objects.

Theory morphisms represent truth-preserving maps between theories. Examples include theory **inclusions**, **translations**/**isomorphisms** between (sub)theories and **models**/**instantiations** (by mapping axioms to theorems that hold within a model), as well as

meta-theory relations, that include logical symbols to talk about the domain. All of this naturally give us the notion of a **theory graph**, which relates theories (represented as nodes) via edges representing theory morphisms, being right at the design core of the OMDoc/MMT language.

There is already a large body of mathematical knowledge encoded in OMDoc/MMT, ranging from the LATIN logic atlas, a meta-level theory graph with over 1000 nodes and edges, over the libraries of more than a dozen theory libraries exported to OMDoc/MMT (about 200,000 definitions, theorems, and proofs; theory graph structure varies), to informal, but theory-graph annotated teaching documents (ca. 4000 theories). All of these are organized in **math archives**, a special file system layout that facilitates mathematical knowledge management services on OMDoc/MMT collections.

OMDoc/MMT is implemented in the MMT system [Rab13; MMT], an application that can read OMDoc/MMT archives and perform knowledge management operations on them. For TGView, it is important that MMT gives acces to the specifics of the theory graphs loaded from the respective archives via a RESTful web interface, which can be extended flexibly by specialized MMT plugins.

The math archives are collected, managed, and made accessible to the community on the MathHub portal [Ian+14; MH]. It provides versioned storage, build management & error-reporting facilities, knowledge based services (via the MMT system), and interactive browsing facilities for collections of math archives.

The TGView system, which is the main contribution we report on in this paper evolved as the theory graph interface of MathHub, mostly on request by users who use MathHub for developing modular formalizations of mathematical models – see [Koh+17]. In these developments, the understanding and interacting with the graph structure induced by the formalization is of essential importance.

3 The Graph-Viewer System

The central design decision in the TGView system is to build a browser-based graph viewer with client-side layouting. Existing systems use libraries like GraphViz to layout the graph in backend and send an image or vector-graphic to the frontend. Such a system needs to recalculate the whole graph server-side for most user interactions. Given web-based technologies, simple interactions like node highlighting might be performed client-side, but any more intrusive interaction – e.g. hiding/showing nodes, clustering/unclustering subgraphs, or changing layout styles – will trigger a page refresh from the server, reduce the speed and usability of interactions with graphs dramatically. Moreover, server-side layouting faces scalability issues and does not support responsive UI design very well.

We overcome these limitations by implementing TGView completely client-side in the browser. This allows TGView to scale nearly infinitely and to enable responsive graph interaction. An Ajax-based implementation in JavaScript allows users to interact with theory graphs over a wide range of devices, since nowadays almost every device supplies modern browsers, which natively support the JS platform. A Java-based solution, which would give access to a wide range of graph layout libraries would necessitate browser plugins, which are available only for a limited range of devices.

TGView has a graphical user interface, which allows all actions to be carried out via mouse point, click and drag operations without any server requests except the first one to receive graph

data. As heart of TGView we use the network library from vis.js [VJS], a graph drawing library, which offers plenty of native interactions like node clustering, different layouts (hierarchical vs. force-driven), good user experience and efficient rendering of big graphs.

To give the user access to the math archives on MathHub, we use a tree-structured menu where we can select single or even multiple theories to plot. We chose to use the JavaScript library jsTree with its rich tree API. For the UI we use jQuery-UI, one of the state-of-the-art libraries for UI-Design. A screen shot of the TGView system in action can be seen in Figure 1.

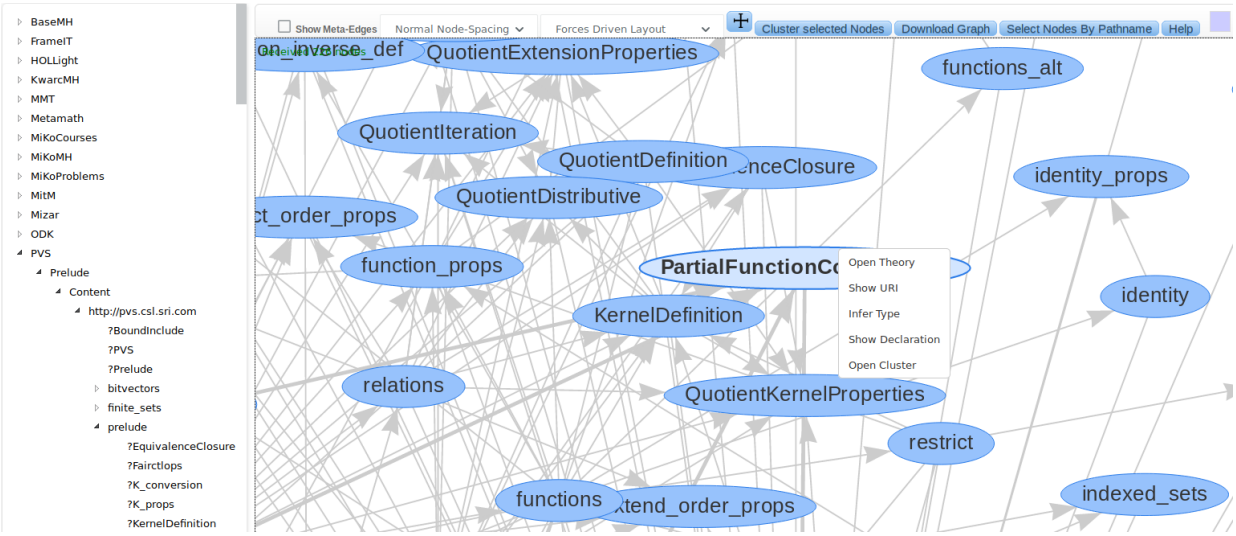

Figure 1: Left: Math Archives; Top: Header bar for editing/selection; Center: Graph area

Figure 2 shows the architecture behind TGView. The server, in our case MathHub– it could be any kind of server – sends the requested graph data to browser. This data is processed by TGView, which then updates the HTML-DOM and canvas via vis.js. From this time on, TGView only needs to communicate with the server, when the user switches or extends graphs. To keep our system modular we divide functionality between client – which is responsible for graph layout and user interaction – and the server, which only needs to supply generic graph data in JSON. This approach is also visible in the invocation pattern of our system. TGView is a JavaScript library that can be included in any web page – in MathHub it is at `http://mathhub.info/mh/mmt/tgview.html`; it adds the `graphdata` query parameter for the web service that supplies the graph data. Thus the invocation pattern for a concrete graph is `http://mathhub.info/mh/mmt/tgview.html?graphdata=mathhub.info/graph/pvs.json`. This is also the invocation used when a user requests a new graph from the math archives sidebar.

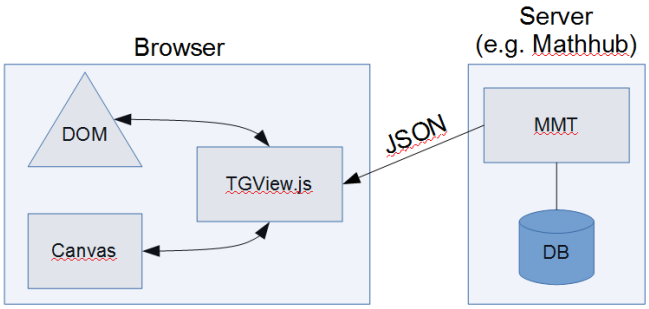

Figure 2: Architectural view

As different graph layout styles give different insights, TGView gives the user the choice of layout formalisms from the header bar. Currently the system supports the three most suitable general layout schemes from vis.js: forces driven layout, semi-hierarchical layout – used to structure weakly hierarchical graphs – and strictly hierarchical layout – see Figure for 3.

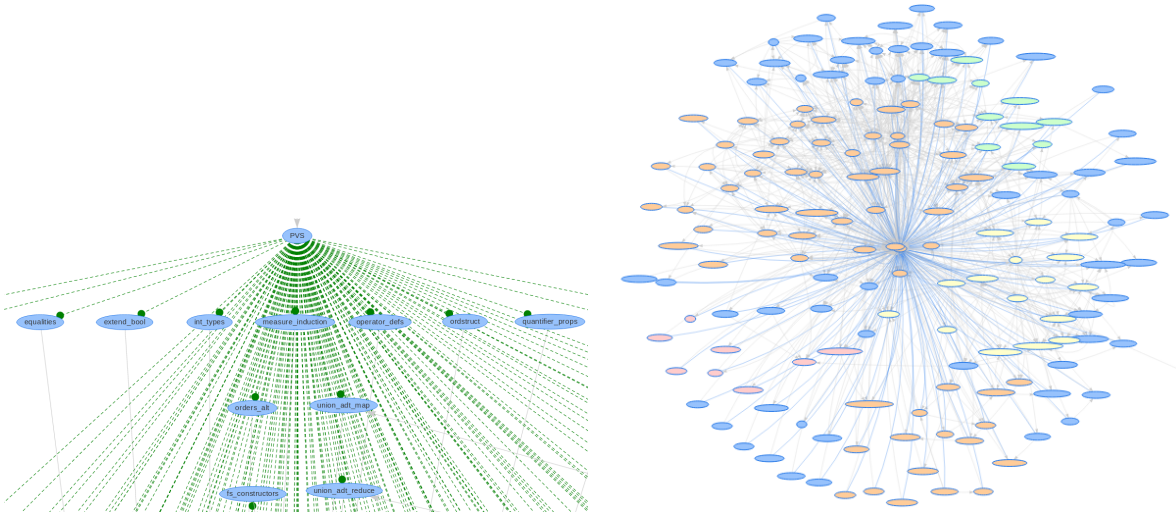

Figure 3: Left: Architectural layout; Right: Forces driven layout

Furthermore, TGView provides a custom layout for **model pathway diagrams** (MPDs) – theory graphs with additional role metadata for theories; see [Koh+17] for details and Figure 4 shows the MPD view of a theory graph about drift-diffusion models for electrons and holes in one-dimensional semiconductors.

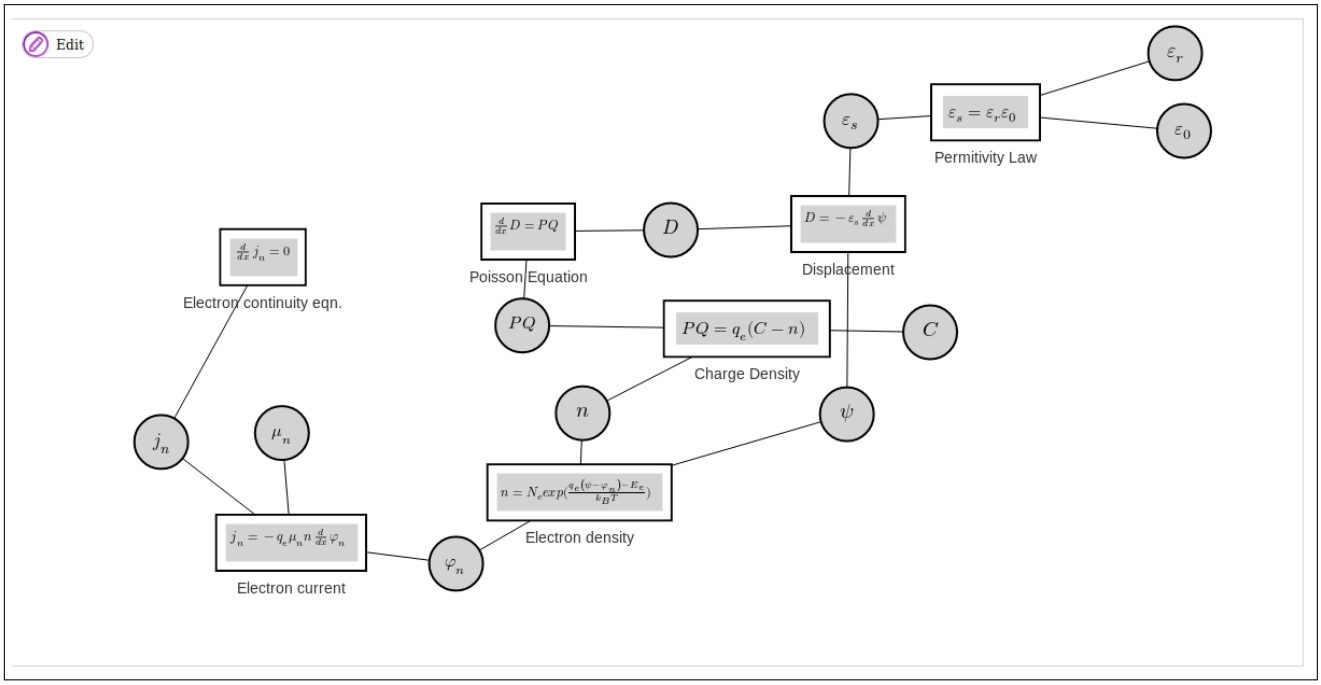

Figure 4: MPD View of the van Roosbroeck Drift/Diffusion Model

As nodes and edges in the displayed graph represent theories and morphisms, which are complex objects of themselves. Fortunately, the MMT web interface already supplies methods for browsing and interacting with them – see [Rab14] –, so we just dispatch the UI to that using a right-click context menu, see Figure 1 for the current context menu.

4 Conclusions, Availability, Evaluation, and Future Work

We have present TGView, a flexible, adaptable theory graph viewer for the OMDoc/MMT ecosystem. Graph layout and interaction are browser-based given theory graph data provided by the MMT API. The system is licensed under the Gnu Public License and is available from GitHub at `https://github.com/uniformal/TGView`.

TGView is able to handle even huge graphs with ten of thousands of edges and nodes efficiently on current hardware, which is way beyond what humans can usefully process visually on normal screens. Unfortunately, browsers limit the canvas to about 3000x3000 pixels. This is enough for showing the image on most modern displays but makes printing big graphs on posters or using extremely big displays like the one at LRI Paris [LRI] impossible.

In spite of its relative youth, users report better comprehension of the theory graphs at a global level. Important features for this are the ability to hand-tweak computed layouts – e.g. pulling nodes apart where they are too dense – and highlighting nodes together with their incoming and outgoing edges. In particular our design decision of using a client-side layout approach has been fully validated.

We have shown the adaptability of the TGView architecture by providing a MPD viewer, which interprets (suitably enriched) theory graphs as model path diagrams, which highlight salient features of model-oriented theory graphs to computational scientists [Koh+17]. Note that this viewer uses MathML for formula display even though this is not (natively) supported by all browsers. We feel that this is not a grave limitation since this viewer ist used mostly in a desktop setting, where FireFox that has native MathML support is available.

Other special viewers that would be desirable would be a TGView mode for the multilingual theory graph structures underlying SMGloM terminology [Koh14] collapsing the signature/language-binding graphs into single nodes.

It is very important to have a nicely structured layout for big graphs, otherwise the amount of manual work is infeasible high. Some automatic clustering beside the possibility to manual clustering would another nice-to-have algorithm (see Figure 5 for a manual cluster example).

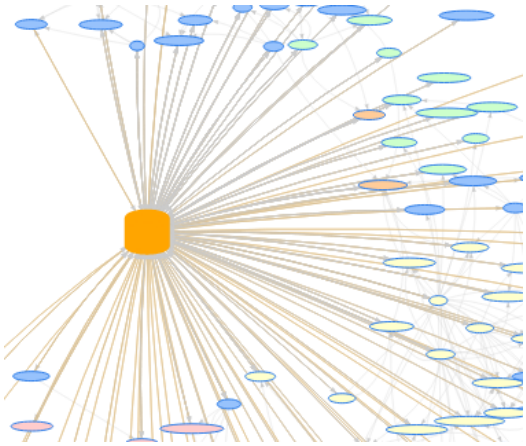

Figure 5: Clustered nodes

As OMDoc/MMT is a general integration format for mathematical knowledge, TGView can serve as an interface for any system that has an OMDoc/MMT interface. Indeed, we are currently integrating TGView into the MathHub portal, which contains OMDoc/MMT exports of many theorem prover libraries, computer algebra systems, and semi-formal collections ranging from flexiformal course materials to mathematical vocabularies.

In the future, we want to extend TGView by a facility for editing theory graphs, combining "drag-and-drop" like editing features for the theory structure with MMT surface syntax editing facilities for the declaration level. Another useful extension would be a MPD editor, where we can select subgraphs into MPDs, together with the TGView editor this would give a graphical modeling tool for shared model libraries. In this vein, we would like to have a facility for saving/sharing layout tweaks to generated theory graphs – possibly for subgraphs – to take

advantage of human improvements.

Acknowledgements: We gratefully acknowledge EU funding for the OpenDreamKit project in the Horizon 2020 framework under grant 676541. Our discussions have particularly profited from contributions by Andrea Kohlhase, Thomas Koprucki, Florian Rabe, and Karsten Tabelow.

References

[Coq] *The Coq Proof Assistant*. URL: http://coq.inria.fr/ (visited on 07/31/2010).

[FGT93] William M. Farmer, Joshua D. Guttman, and F. Javier Thayer. "IMPS: An Interactive Mathematical Proof System". In: *Journal of Automated Reasoning* 11.2 (Oct. 1993), pp. 213–248.

[GAP] The GAP Group. *GAP – Groups, Algorithms, and Programming*. URL: http://www.gap-system.org (visited on 08/30/2016).

[GV] *Graphviz – Graph Visualization Software*. URL: http://www.graphviz.org/ (visited on 05/22/2017).

[Ian+14] Mihnea Iancu, Constantin Jucovschi, Michael Kohlhase, and Tom Wiesing. "System Description: MathHub.info". In: *Intelligent Computer Mathematics 2014*. Conferences on Intelligent Computer Mathematics. (Coimbra, Portugal, July 7–11, 2014). Ed. by Stephan Watt, James Davenport, Alan Sexton, Petr Sojka, and Josef Urban. LNCS 8543. Springer, 2014, pp. 431–434. ISBN: 978-3-319-08433-6. URL: http://kwarc.info/kohlhase/papers/cicm14-mathhub.pdf.

[Koh+11] Michael Kohlhase et al. "The Planetary System: Web 3.0 & Active Documents for STEM". In: *Procedia Computer Science* 4 (2011): *Special issue: Proceedings of the International Conference on Computational Science (ICCS)*. Ed. by Mitsuhisa Sato, Satoshi Matsuoka, Peter M. Sloot, G. Dick van Albada, and Jack Dongarra. Finalist at the Executable Paper Grand Challenge, pp. 598–607. DOI: 10.1016/j.procs.2011.04.063. URL: http://kwarc.info/kohlhase/papers/epc11.pdf.

[Koh+17] Michael Kohlhase, Thomas Koprucki, Dennis Müller, and Karsten Tabelow. "Mathematical models as research data via flexiformal theory graphs". In: *Intelligent Computer Mathematics (CICM) 2017*. Conferences on Intelligent Computer Mathematics. (July 17–21, 2017). LNAI. in press. Springer, 2017. URL: http://kwarc.info/kohlhase/papers/cicm17-models.pdf.

[Koh14] Michael Kohlhase. "A Data Model and Encoding for a Semantic, Multilingual Terminology of Mathematics". In: *Intelligent Computer Mathematics 2014*. Conferences on Intelligent Computer Mathematics. (Coimbra, Portugal, July 7–11, 2014). Ed. by Stephan Watt, James Davenport, Alan Sexton, Petr Sojka, and Josef Urban. LNCS 8543. Springer, 2014, pp. 169–183. ISBN: 978-3-319-08433-6. URL: http://kwarc.info/kohlhase/papers/cicm14-smglom.pdf.

[LRI] *The LATIN Theory Graph on an Extremely Large Touchscreen*. URL: https://kwarc.info/people/frabe/Research/florian_rabe_latin_graph_at_lri.jpg.

[MH] *MathHub.info: Active Mathematics*. URL: http://mathhub.info (visited on 01/28/2014).

[Miz] *Mizar*. URL: http://www.mizar.org (visited on 02/27/2013).

[MMT] *MMT – Language and System for the Uniform Representation of Knowledge*. project web site. URL: https://uniformal.github.io/ (visited on 08/30/2016).

[OA] *OpenAxiom: The Open Scientific Computation Platform*. URL: http://www.open-axiom.org (visited on 05/22/2017).

[ORS92] S. Owre, J. M. Rushby, and N. Shankar. "PVS: A Prototype Verification System". In: *Proceedings of the 11th Conference on Automated Deduction*. Ed. by D. Kapur. LNCS 607. Saratoga Springs, NY, USA: Springer Verlag, 1992, pp. 748–752.

[Pau94] Lawrence C. Paulson. *Isabelle: A Generic Theorem Prover*. LNCS 828. Springer Verlag, 1994.

[PV] *Visualization - Protege Wiki*. URL: hTtps://protegewiki.stanford.edu/wiki/Visualization (visited on 05/22/2017).

[Rab13] Florian Rabe. "The MMT API: A Generic MKM System". In: *Intelligent Computer Mathematics*. Conferences on Intelligent Computer Mathematics. (Bath, UK, July 8–12, 2013). Ed. by Jacques Carette, David Aspinall, Christoph Lange, Petr Sojka, and Wolfgang Windsteiger. Lecture Notes in Computer Science 7961. Springer, 2013, pp. 339–343. ISBN: 978-3-642-39319-8. DOI: 10.1007/978-3-642-39320-4.

[Rab14] Florian Rabe. "A Logic-Independent IDE". In: *Workshop on User Interfaces for Theorem Provers*. Ed. by Cristoph Benzmüller and Bruno Woltzenlogel Paleo. Elsevier, 2014, pp. 48–60. DOI: 10.4204/EPTCS.167.7.

[Sage] The Sage Developers. *SageMath, the Sage Mathematics Software System*. URL: http://www.sagemath.org (visited on 09/30/2016).

[VJS] *vis.js - A dynamic, browser based visualization library*. URL: http://visjs.org (visited on 06/04/2017).

[Wat+14] Stephan Watt, James Davenport, Alan Sexton, Petr Sojka, and Josef Urban, eds. *Intelligent Computer Mathematics*. Conferences on Intelligent Computer Mathematics. (Coimbra, Portugal, July 7–11, 2014). LNCS 8543. Springer, 2014. ISBN: 978-3-319-08433-6.