KAT: Enabling the Semantification of STEM Documents

Felix Schmoll
f.schmoll@jacobs-university.de

Tom Wiesing
t.wiesing@jacobs-university.de

Jacobs University Bremen

Abstract

Considering a constantly growing body of mathematical knowledge it becomes more and more difficult for individuals to take full advantage of all information available. To semantify documents, we need to be able to create annotations efficiently and conveniently – marking definitions or declarations as well as usages of concepts – on a large corpus of documents. Eventually this can be achieved automatically, but as a first step a gold standard has to be created by humans. KAT – the KWARC Annotation Tool – has the goal of allowing users to create, view and update annotations on arbitrary (X)HTML documents. We have presented our approach before and in this paper we want to give an update on our progress.

1 Introduction

STEM – Science, Technology, Engineering and Mathematics – documents often not only introduce the user to new areas but build heavily on previous knowledge. We want to make STEM documents, in particular mathematical documents, more accessible and lower the burden of gaining access to a new topic. Users should be able to intuitively navigate through existing knowledge as to make the process of understanding more efficient.

In the context of mathematical documents this requires annotating documents and marking up definitions, declarations and other linguistic phenomena. Furthermore we want to mark up all usages of these concepts within the document to allow readers to, for example, click on a concept and navigate to its definition. In order to bring us closer to this goal it is necessary to do this on a large scale.

As it is non-trivial to annotate a huge corpus of documents one wants eventually to do this automatically, with only little human interaction. During the development of tools to achieve this it is however common to annotate a small subset of documents manually, creating a "gold standard", that can then be used as a basis for further the development – either using automated machine learning approaches or smart rule-based software. Annotation tools to create, edit and view annotations are here necessary and can further be helpful during later phases of development – in order to evaluate performance[1]. Additionally they can they can be used by mathematical readers to interactively navigate through annotated content.

STEM documents usually consist of a multitude of content, ranging from pure text over mathematical/chemical formulae to tables and diagrams. Most common annotation tools, such as brat [BR], Yawas [YW], and Annotatie [AN], only work properly with textual content – they store the position of annotations as offsets in a character string. While some of them can work with more advanced content, they do so by treating it as a blackbox and replacing it with a placeholder. Such a treatment however prevents for example the annotation of sub-formulae as any embedded formulae are considered as a single object.

Copyright © by the paper's authors. Copying permitted for private and academic purposes.

[1] Indeed our tool has been used during the development of a declaration spotter in order to evaluate performance, we refer interested readers to [Sch16].

We have developed KAT – the KWARC Annotation Tool – to address and solve these problems. KAT is a browser-based tool that will allow users to annotate arbitrary (X)HTML documents. In this paper we want to present the newest iteration of it, building on the previous versions presented in [DGKMMW14] and [GLKW15]. We proceed in Section 2 by giving a short review of the existing KAT system architecture. In Section 3 we then proceed by giving a more detailed look into how annotations can be created in the browser and in particular focus on the progress we have achieved since the last iteration. We then conclude in Section 4 with an outlook on how we plan to develop the system further.

2 Recap of KAT system architecture

KAT is implemented as a JavaScript library working with (X)HTML5 documents and can be integrated into arbitrary websites with little effort. The basic system architecture can be seen in Figure 1. This architecture consists of four main components: *i*) the KAT Annotator, the KAT tool running in the browser; *ii*) the KAnnSpec, which serves as a description of *iii*) the Annotation ontology and *iv*) a document management system (here called CorTeX).

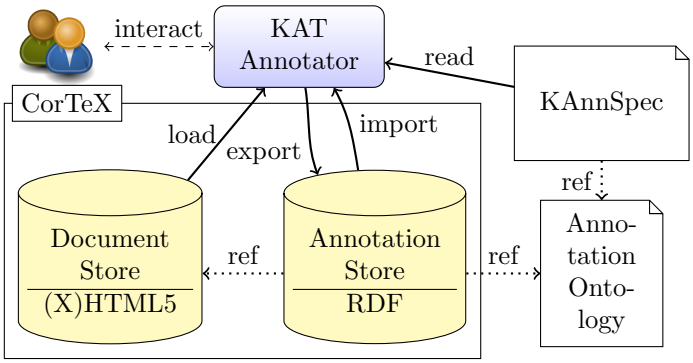

Figure 1: The KAT System Architecture

Instead of describing this architecture in detail, we refer interested readers to [DGKMMW14] and [GLKW15]. KAT is capable of working as a standalone tool, however it is best used in the context of a corpus management system such as CorTeX system [CT]. This scenario is intended for creating and improving a "gold standard". Users receive a document from the document store in CorTex and create annotations for it using KAT. These annotations are then sent back to the system in RDF form. Alternatively the users receive an already annotated document and check existing annotations. These two scenarios drive the KAT development.

KAT annotations are represented as RDF subject / predicate / object triples. The subjects are the text fragments that we are annotating whereas the objects are either concepts from the annotation ontology (in the case of classificational annotations) or other text fragments (in the case of relational annotations). KAT is not tied to a particular annotation ontology. On the contrary, at startup time it loads a set of annotation descriptions referred to as KAT Annotation Specification, or KAnnSpec for short. These are a set of custom XML documents that describe the annotation ontology, its concepts and additional constraints for the KAT user interface.

As KAT is XHTML based we reference to text fragments using URIs. For this we make use of the XPointer framework [GMMW03] and developed a custom XPointer scheme. Each text fragment is a contiguous range of elements in the DOM tree and can be identified by giving the first and last elements contained in it. This obviously imposes the limitation that text fragments must consist of entire elements, making it very difficult to annotate linguistically meanigful concepts. In previous iterations of the system we worked around this by TEI-tokenizing document, that is adding elements around each word in the document, however we are in progress of implementing an updated XPointer scheme that allows us to reference text ranges within elements and thereby eliminating the limitation entirely.

3 Editing Annotations In The Browser

KAT offers three different operating modes that can be navigated via a sidebar. Each of them facilitates a different kind of working with annotations:

1. *Annotation Mode.* This mode provides a user interaction to create and edit annotations on documents. This is semantically seperate from the viewing and evaluating of annotations by a different mode, allowing specialized operations e.g. on right-clicking.

2. *Reading Mode.* Once annotations have been created they can be used to obtain a better understanding of the structure of a mathematical document as a whole. This mode attempts to provide as much information as possible about all given annotations in an intuitive way.

3. *Review Mode.* When automating the creation of annotations, one wants to go over the created annotations and evaluate them. This mode allows an evaluation of annotations by providing an interface for judging them as good (thumbs up) or bad (thumbs down).

3.1 Creating Annotations

As KAT is a browser-based tool, the process of creating annotations is a heavily form-oriented process. A range of text is selected and then an annotation category is chosen from a right-click context menu. Subsequently a form appears in the sidebar where more specific information can be entered. An example of this can be seen in Figure 2.

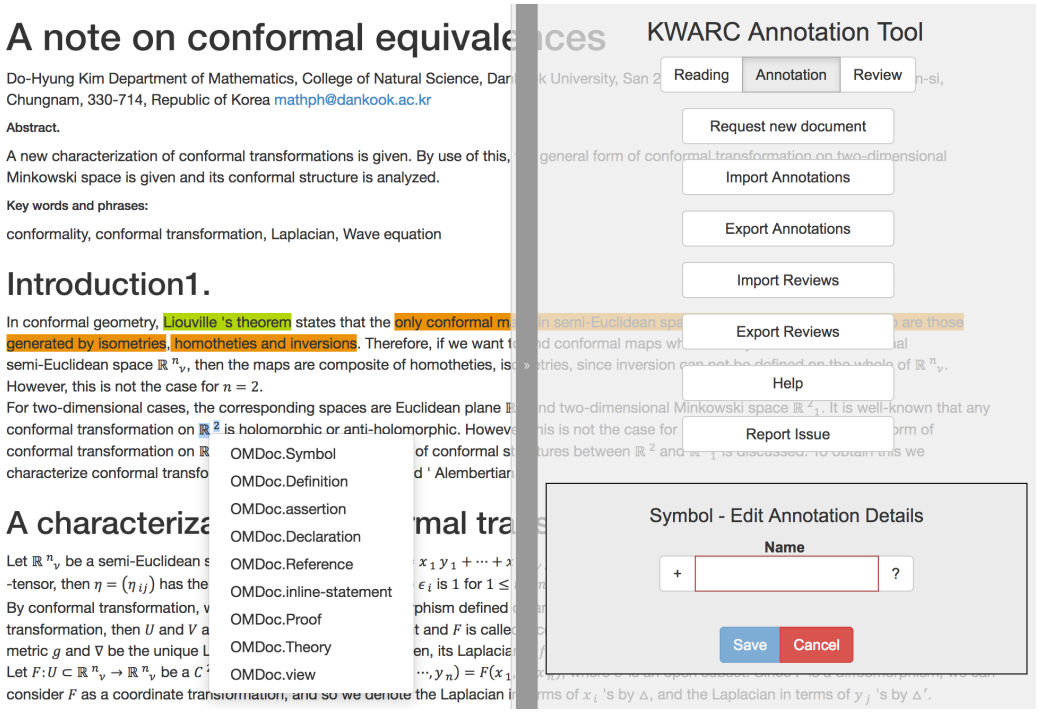

Figure 2: Creating annotations in KAT.

It is possible to make restrictions on input in the KAnnSpec, such as the format of a name by providing a Regular Expression to be fulfilled. To ensure that entered content is valid it is only possible to save an annotation once all conditions are fulfilled. To visually indicate this the color of a text field changes from red to green, and only once the validation constraints are fullfilled the save button becomes active. Additionally the question mark right next to the input-field can be used to obtain more information about said constraints (in this particular case the Regular Expression restricting the input format would be shown).

3.2 Visualizing Annotations And References

In Reading mode all annotations are displayed to the user. The respective information are conveyed via multiple means:

- *Highlighting.* Each concept from the KAnnSpec is assigned its own color and each annotation is highlighted in the appropriate color.
- *Tooltip.* When hovering an annotation, information about its fields is displayed. The tooltip is generated using the `<template/>` tag in the KAnnSpec.
- *Relations.* Some annotations may have referential fields. These are visualised on demand.

Upon creation of an annotation it is assigned a UUID (universal unique identifier). This is later on the only way to refer to a specific annotation. As this is not a very intuitive way to reference an annotation there is a need for visual cues in displaying relations.

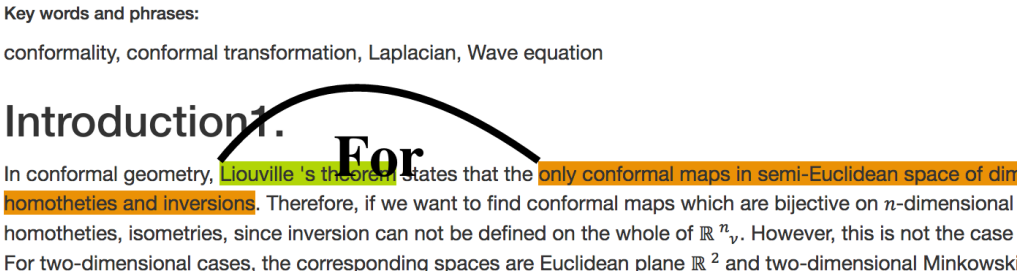

Figure 3: Paths can be used to visualize references.

To achieve this, KAT provides the ability to show the relations using paths between annotations that appear by clicking on them. The paths are labeled with the type of reference between two given annotations. The direction of a path becomes clear due to the user interaction that causes them to be displayed.

Internally this is implemented using an SVG-overlay over the document. Paths are then drawn from the selected annotation to all other annotations that are referenced in respective fields of the KAnnSpec and decorated with captions describing the kind of relation. As start and end points of the paths the upper left corners of their selections are used.

3.3 Import And Export Of Annotations

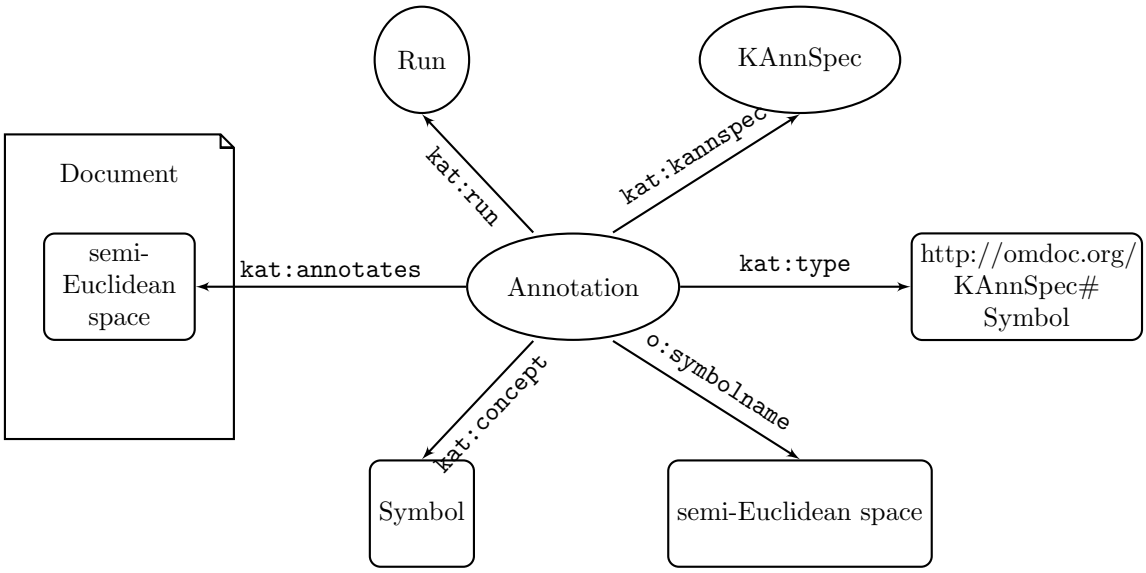

Figure 4: An Annotation Graph

By itself KAT only saves the state of annotations for the length of a given session. To make annotations persistent, it is possible to export and later re-import annotations via buttons in the side pane, which is implemented based on the Resource Description Framework [SR14]. This approach has been successfully tested in the creation of a gold standard of declarations in math [Sch16].

The current implementation also features a prototype for retrieving new documents from a document storage and submitting annotations to it. If one would fully integrate the system with CorTeX it would then no longer be necessary to store annotations manually but could store it centralised together with the document.

In Listing 1 we show a sample of how an annotation is exported to RDF. Each annotation consists of a single node (lines 6-12). Additionally, each annotation has meta-data, such as the used KAnnSpec, which is omitted here. To generate this node, we use the properties of the annotation graph shown above in Figure 4.

Listing 1: Exported RDF generated for a single annotation of an OMDOC Symbol

```
1 <rdf:RDF xmlns:o="http://omdoc.org/KAnnSpec#" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
    syntax-ns#" xmlns:kat="https://github.com/KWARC/KAT/">
```

```
2   <!-- omitted a lot of meta-information here -->
3
4   <rdf:Description rdf:nodeID="KAT_1433087821332_4477">
5     <kat:annotates rdf:resource="https://kwarc.github.io/KAT/content/sample1.html#cse(%2F%2F
          *%5B%40id%3D'sentence.11'%5D%2C%2F%2F*%5B%40id%3D'word.202'%5D%2C%2F%2F*%5B%40id%3D'
          word.203'%5D)" />
6     <kat:run rdf:nodeID="kat_run"/>
7     <kat:kannspec rdf:nodeID="KAT_1433087757661_OMDoc"/>
8     <kat:concept>Symbol</kat:concept>
9     <kat:type rdf:resource="http://omdoc.org/KAnnSpec#Symbol" />
10    <o:symbolname>semi-Euclidean space</o:symbolname>
11  </rdf:Description>
12
13  </rdf:RDF>
```

The export starts by declaring the text fragment it annotates using the *kat:annotates* relation (line 5). For this we use the KAT XPointer scheme – in this case given by the URI `https://kwarc.github.io/KAT/content/sample1.html#cse(//*[@id='sentence.11'],//*[@id='word.202'],//*[@id='word.203'])`. In this URI `cse` stands for Container, Start and End. The URI itself consists of

- the document URI `https://kwarc.github.io/KAT/content/sample1.html`, the document in which the annotated text fragment is located;
- an XPath to the deepest element that fully contains the annotated text fragment, here `//*[@id='sentence.11']`;
- an XPath that points to the start of the annotated text fragment – the first element that is contained in it – here `//*[@id='word.202']` and
- an XPath that points to the end of the annotated fragment – the last element that is contained in it – here `//*[@id='word.203']`.

Next, a *kat:run* is passed (line 6). This is intended to provide meta-information such as when and how this annotation was generated, which has been omitted from the listing. Continuing, it references a KAnnSpec (line 7) and then the actual concept it annotates (line 8). We further specify the type of the annotation with the *kat:type* relation (line 9) as given in the KAnnSpec. Finally, we provide all the fields and their values. In this case, we just give the concept the name *semi-Euclidean space* (line 10).

3.4 Evaluating Annotations In The Review Mode

After annotations have been created it is possible to evaluate them using a special review mode. Here automatically created annotations can be assessed by a human operator. An example of the review mode can be found in Figure 5.

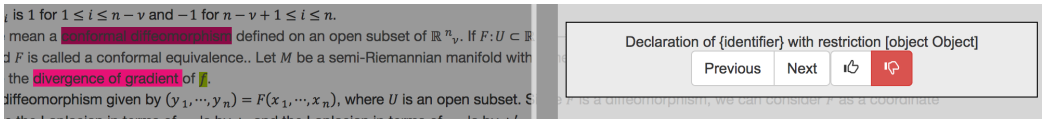

Figure 5: Evaluating annotations using the review mode.

At any given time exactly one annotation is in focus – the one that is supposed to be assessed. This annotation is visually contrasted by a dark overlay over the rest of the document. An information field in the side pane displays all available information for the current annotation (which would appear in a tooltip when hovering over an annotation in reading mode) and facilitates endorsing or flagging of a given annotation, such that the automatic annotation generation can be improved based on the given feedback. The user can then iterate through the annotations in the order of their appearance in the document.

Using this field it is possible to evaluate the accuracy of automated annotation-systems or to use ensure the quality of existing annotations. One possible use case is as a stage in the development cycle in order to obtain relevant feedback, as was done is the development of the declaration spotter in [Sch16].

The reviews of annotations made in review mode can likewise be exported, whereas the format is essentially a list of (UUID, Boolean)-pairs indicating which annotation was reviewed and what the binary review choice was.

4 Conclusion And Future Work

We have presented the KAT system, an open and browser-based annotation system for STEM documents encoded in XHTML5. In particular we have presented the new and improved visualisation of referential annotations and the new review mode. The code base is released freely under the terms of the GNU Public License and available at [KG]. A running demo is further available at [KD].

While the basic utility of annotating documents is functional, there are some aspects that we want to improve upon in order to allow a smoother user experience. The most important ones are

1. *Allowing annotated text fragments within elements.* At the moment it is only possible to annotate text fragments on a DOM Element basis. Thus KAT works best with documents are TEI-tokenized into words in order to process them. We are working on an XPointer-implementation that would allow any XHTML5 document to be immediately marked up. This would furthermore allow a better integration with CorTeX [CG].

2. *Distinction between overlapping annotations.* Currently different annotation categories are visually set apart by color. This is however not sufficient once there are overlapping annotations. In future versions the ranges of overlapping annotations should be discernable by giving each of them a different height.

3. *Adding feedback and other improvements to the review mode.* It should be possible to provide more specific feedback in review mode, as it is currently only possible to make a binary choice. One way would be to provide additionally an input field in order to give an explanation, another one to pick a specific reason why an annotation is inappropriate from a drop-down menu. Furthermore it is an open question how to properly export the feedback given in review mode.

4. *Using arrows for visualising paths.* One might want to extend the visualisation of relations as to use actual arrows indicating a direction. The current paths are suitable while using the system, however insufficient when looking at a static image of the representation.

5. *Allowing changing document content.* Published papers are rarely changed, but one might nevertheless want to accommodate the ability to handle a changing document structure for example when annotating just sections of a document and merging them later on. The current implementation does not allow this.

We will also try to get more immediate user feedback by providing a version of the system to a larger audience.

4.1 Acknowledgements

We thank Frederik Schaefer for his feedback on further improvements of KAT especially in terms of usability and Deyan Ginev for his input on how to avoid the need to tokenize documents. Additionally we would like to thank Michael Kohlhase for his supervision.

References

[AN] Annotation tool. URL: `Http://www.annotatiesysteem.nl` (visited on 02/15/2014).

[BR] Brat rapid annotation tool. URL: `http://brat.nlplab.org` (visited on 02/15/2014).

[CG] GitHub repository. URL: `https://github.com/dginev/CorTeX/`.

[CT] CorTeX framework. URL: `http://cortex.mathweb.org` (visited on 02/14/2014).

[DGKMMW14] Mircea Alex Dumitru, Deyan Ginev, Michael Kohlhase, Vlad Merticariu, Stefan Mirea, and Tom Wiesing. System description: KAT an annotation tool for STEM documents. 2014. URL: `http://kwarc.info/kohlhase/submit/cicm14-kat.pdf`.

[GLKW15] Deyan Ginev, Sourabh Lal, Michael Kohlhase, and Tom Wiesing. KAT: an annotation tool for STEM documents. In *Mathematical user interfaces workshop at CICM*. Andrea Kohlhase and Paul Libbrecht, editors, July 2015. URL: `http://www.cermat.org/events/MathUI/15/proceedings/Lal-Kohlhase-Ginev_KAT_annotations_MathUI_15.pdf`.

[GMMW03] Paul Grosso, Eve Maler, Jonathan Marsh, and Norman Walsh. W3c xpointer framework. W3C Recommendation. World Wide Web Consortium (W3C), March 25, 2003. URL: http://www.w3.org/TR/2003/REC-xptr-framework-20030325/.

[KD] URL: http://kwarc.github.io/KAT/ (visited on 07/18/2016).

[KG] GitHub repository. URL: https://github.com/KWARC/KAT/.

[Sch16] Jan Frederik Schaefer. Declaration spotting in mathematical documents. B. Sc. Thesis. Jacobs University Bremen, 2016.

[SR14] Guus Schreiber and Yves Raimond. RDF 1.1 primer. W3C Working Group Note. World Wide Web Consortium (W3C), 2014. URL: http://www.w3.org/TR/rdf-primer.

[YW] Yawas - the original web highlighter. URL: http://www.keeness.net/yawas/ (visited on 02/15/2014).