Auto-hyperlinking the Stacks Project

Johan Commelin[1] and Josef Urban[1]

Radboud University Nijmegen

Abstract. This paper describes an effort to automatically hyperlink mathematical terms to their definitions in the Stacks Project.
The Stacks Project is an open source, collaborative textbook on Algebraic Geometry. It covers the definition of an algebraic stack (unrelated to the notion of stack in Computer Science), and beyond; providing a technical reference of the state of the art for researchers in the field. It provides a web interface and a stand-alone PDF (currently over 4600 pages).
Throughout the project there is a thorough linking to earlier results used in proofs. However, as is customary in mathematics, nouns and symbols are not hyperlinked to their definitions. In this paper we outline our initial approach to auto-hyperlinking terminology to their definitions, and our future plans.

1 Introduction: Stacks Project

The Stacks Project[1] was initiated in 2005 by Aise Johan de Jong with the aim of collaboratively writing an online introductory text on algebraic stacks (a rather advanced topic in Algebraic Geometry). Over time it has evolved into a textbook covering the foundations of Algebraic Geometry, and serves as an online reference for algebraic geometers. The text is written in a very restricted subset of LaTeX, with a minimal amount of packages and custom commands. This allows custom translation of the text to HTML implemented by the project's PHP code, and relying on MathJaX for math-mode rendering. At the moment the compiled PDF consists of more than 4600 pages.

A crucial element of the Stacks Project is its web interface, and the concept of *tags*. Every result (theorem, lemma, equation, etc, but also chapters and sections) has a LaTeX-label, used for internal references. Besides that, each such label is assigned a *tag*, a 4-character alphanumerical string. The tag is stable, and the web interface provides an easy method to look up the mathematical statement associated with a tag. This also solves the problem of referencing results in the Stacks Project, since the tag provides a permanent URI for the mathematical statement. It should be understood that the content of a tag will not change (up to corrections of minor mathematical mistakes, typographical errors, and clarifications/expansions of proofs).

Part of the Stacks Project now also consists of several pieces of software (mostly written by Aise Johan de Jong and Pieter Belmans) covering amongst more:

[1] `http://stacks.math.columbia.edu/`

– tools to assign new tags to new results;
– scripts that parse the LaTeX source files, generating chunks of source for each tag, and converting the LaTeX to HTML (+MathJaX);
– an API that can be used to request the LaTeX source for a tag;
– dependency graphs for each tag (recursively showing which results are used in the proof).

The authors of the Stacks Project have initially decided (as is customary in pen-and-paper mathematics) that symbols in the text are not hyperlinked to their definitions.

While expert mathematicians usually understand texts written by other expert mathematicians, nonintrusive (e.g., Wikipedia-style) hyperlinking of terminology can be useful for non-experts and students. It is also a prerequisite and one of the first steps needed for making such texts – at least partially, and possibly with human assistance – computer-understandable and verifiable by formal proof assistants. Below we outline our initial approach to auto-hyperlinking symbols to their definitions. It consists of (i) heuristically collecting defined terms from the texts (Section 2) and (ii) heuristically linking symbols in an arbitrary Stacks text to their estimated definitions (Section 3).

2 Collecting definition data

The Stacks Project website employs an SQLite database to store all the information about tags that it needs. We queried this database for a list of all tags that correspond to a LaTeX `definition` environment. We then parsed the source of these tags for strings of the form `{\it foobar}`, to generate a list of all defined terminology, together with the tag where they were defined. The list consists of 2238 items, and we can already make some observations about it:

1. Certain tags define multiple terms. This is not a problem for our purposes.
2. Certain terms have definitions in multiple tags. For example, the term *flat* is defined in the following tags:

   `00HB 01U3 0251 0253 02N3 03ER 03ML 04JB 05ND 06PW.`

   Tag `00HB` defines what *flat* module over a ring is, and when a morphism of rings is *flat*. Next, `01U3` defines when a sheaf of modules is flat at some point, and when a morphism of schemes is *flat*. This list continues, and finally `06PW` defines when a morphism of algebraic stacks is *flat*.
   For (expert) mathematicians it is usually clear from the context which definition is meant. Of course for our purpose this poses a challenge, because we somehow have to take the context into account.
3. Certain terms occur as substrings of other terms. In most cases this can be solved by greedily choosing the longest matching term. Sometimes we may also need to take context into account, as in the previous point.

3 Auto-linking

While ultimately we are interested in trying as sophisticated context-based algorithms as possible,[2] initially we have tried to make the whole website work with two simple methods. A particular problem with using e.g. machine-learning approaches is that we do not have annotated training data, as for example in the Wikifier [8] project,[3] where disambiguation can be learned on the large amount of manually hyperlinked concepts, or in our related work on parsing informalized large formal corpora [7,6] with the help of strong large-theory automated reasoning "hammers" [5,1].

The first method goes through all the defined terms (series of words) sequentially from the longest to the shortest, and in the target text it globally rewrites matched strings to special unique markers that cannot be matched by any other defined term. This way, the longest matched concepts are greedily removed, avoiding clashes in the form of possible further matches of their substrings. For example, once *flat module* has been matched at a certain position, neither of its constituent words can be matched. This is a simple longest-first greedy heuristic, which could likely be extended to Knuth-Morris-Pratt-like algorithm ensuring maximal cover by longest possible strings, using e.g. heuristics trading the average length of the matched strings for the total matched ratio of the whole text. While the efficiency of doing sequential scan with all the defined terms in SP seemed far from optimal, in practice the speed of linking turned out not to be an issue on our hardware and happens in real time.

Having the first method running allowed us to see its main deficiencies by randomly browsing dozens of the auto-linked pages. One frequent and easily removable deficiency is linking to future. The Stacks tags have a chronological ordering (as common in textbooks). Only very rarely do mathematicians allow use of concepts that have not been introduced yet, and SP explicitly forbids this. Hence our second method: when rendering a particular tag, we still go through all the defined concepts from the longest, however we only allow replacement of the matched term if it has been defined in a tag that precedes the currently rendered tag. Again, the information about the chronological ordering of the tags can be easily extracted from the SQLite database of tags. A side-by-side comparison of the second version running on our server[4] with the unmodified (slightly later) version from the Stacks website is shown on Figure 1.

4 Evaluation Methods

As mentioned above, we do not have any "ground truth" data for evaluating the quality of the autolinking and comparing different methods. Instead, we use or plan to use the following methods for evaluation:

[2] See, e.g., [2,3,4] for the decade of work on the much older PlanetMath corpus, which is however quite different in terms of the technology used, focus, and coverage of advanced topics.

[3] `http://cogcomp.cs.illinois.edu/page/demo_view/Wikifier`

[4] `http://mws.cs.ru.nl:8008`

– Random browsing through several topics, possibly using side-by-side comparison of the same text rendered with different methods. To make this easier, we use a copy-on-write filesystem (BTRFS[5]) to minimize the overhead of several simultaneous differently modified installations (each over 2GB big) of Stacks, and we optionally use javascript code that immediately previews on mouse-over the linked pages.
– We have written scripts that go through all the tags using a tracing version of the autolinking methods, resulting in a large file with the statistics of how often a particular disambiguation was used in each tag. We then compare the traces for different versions of the autolinking methods. Full tracing for all the 12500 tags takes about 30 minutes. Table 1 compares the most frequent disambiguations for the two autolinking methods explained above.
– We are working on an evaluation interface that will present readers (mathematicians, students, or just us) for each autolinked item on a page with a selection window, allowing to choose the correct disambiguation from all the options. Such choices will be stored on the server, eventually generating the ground-truth data against which we will be comparing and training the algorithms.

No position filtering			Position filtering		
count	term	tag	count	term	tag
14522	morphism	03UM	13184	finite	09G3
8941	scheme	01IJ	8984	scheme	01IJ
7793	finite	09G3	7727	morphism	03UM
7104	open	06U2	7340	algebraic	09GC
6850	algebraic	09GC	5910	category	0014
6794	flat	06PW	5225	functor	003N
6135	surjective	04ZS	4724	isomorphism	0017
5910	category	0014	3910	quasi-compact	090H
5806	affine	03WF	3396	finite type	01T1
5323	functor	003N	3225	field	09FD

Table 1: Initial statistics of the two linking methods.

5 Future Work

Even the simplest auto-linking methods described above seem to be already useful, but there is a wealth of research we can draw on. The obvious extensions include use of machine learning on the collected data, use of the(bag-of-words) context for disambiguation, stemming of the words and their permuting, detecting typing information for supplying more advanced context, etc. Since the proof

[5] `https://btrfs.wiki.kernel.org`

style is quite uniform, a distant dream is to eventually try creation of formally correct formulas and proof sketches by semi-automated methods, and attempting their discharging with strong large-theory automated reasoning tools.

References

1. J. C. Blanchette, C. Kaliszyk, L. C. Paulson, and J. Urban. Hammering towards QED. Accepted to Journal of Formalized Reasoning, preprint at `http://www4.in.tum.de/~blanchet/h4qed.pdf`, 2015.
2. J. J. Gardner, A. Krowne, and L. Xiong. NNexus: Towards an automatic linker for a massively-distributed collaborative corpus. In E. Blanzieri and T. Zhang, editors, *2nd International ICST Conference on Collaborative Computing: Networking, Applications and Worksharing, CollaborateCom 2006, Atlanta, GA, USA, November 17-20, 2006*. IEEE Computer Society / ICST, 2006.
3. J. J. Gardner, A. Krowne, and L. Xiong. NNexus: An automatic linker for collaborative web-based corpora. *IEEE Trans. Knowl. Data Eng.*, 21(6):829–839, 2009.
4. D. Ginev and J. Corneli. NNexus reloaded. In Watt et al. [9], pages 423–426.
5. C. Kaliszyk and J. Urban. HOL(y)Hammer: Online ATP service for HOL Light. *Mathematics in Computer Science*, 9(1):5–22, 2015.
6. C. Kaliszyk, J. Urban, and J. Vyskocil. Learning To Parse on Aligned Corpora (Rough Diamond). Accepted for publication in ITP'15, preprint at `http://mws.cs.ru.nl/~urban/itp15/paper1-final.pdf`, 2015.
7. C. Kaliszyk, J. Urban, J. Vyskocil, and H. Geuvers. Developing corpus-based translation methods between informal and formal mathematics: Project description. In Watt et al. [9], pages 435–439.
8. L. Ratinov, D. Roth, D. Downey, and M. Anderson. Local and global algorithms for disambiguation to Wikipedia. In D. Lin, Y. Matsumoto, and R. Mihalcea, editors, *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*, pages 1375–1384. The Association for Computer Linguistics, 2011.
9. S. M. Watt, J. H. Davenport, A. P. Sexton, P. Sojka, and J. Urban, editors. *Intelligent Computer Mathematics - International Conference, CICM 2014, Coimbra, Portugal, July 7-11, 2014. Proceedings*, volume 8543 of *Lecture Notes in Computer Science*. Springer, 2014.

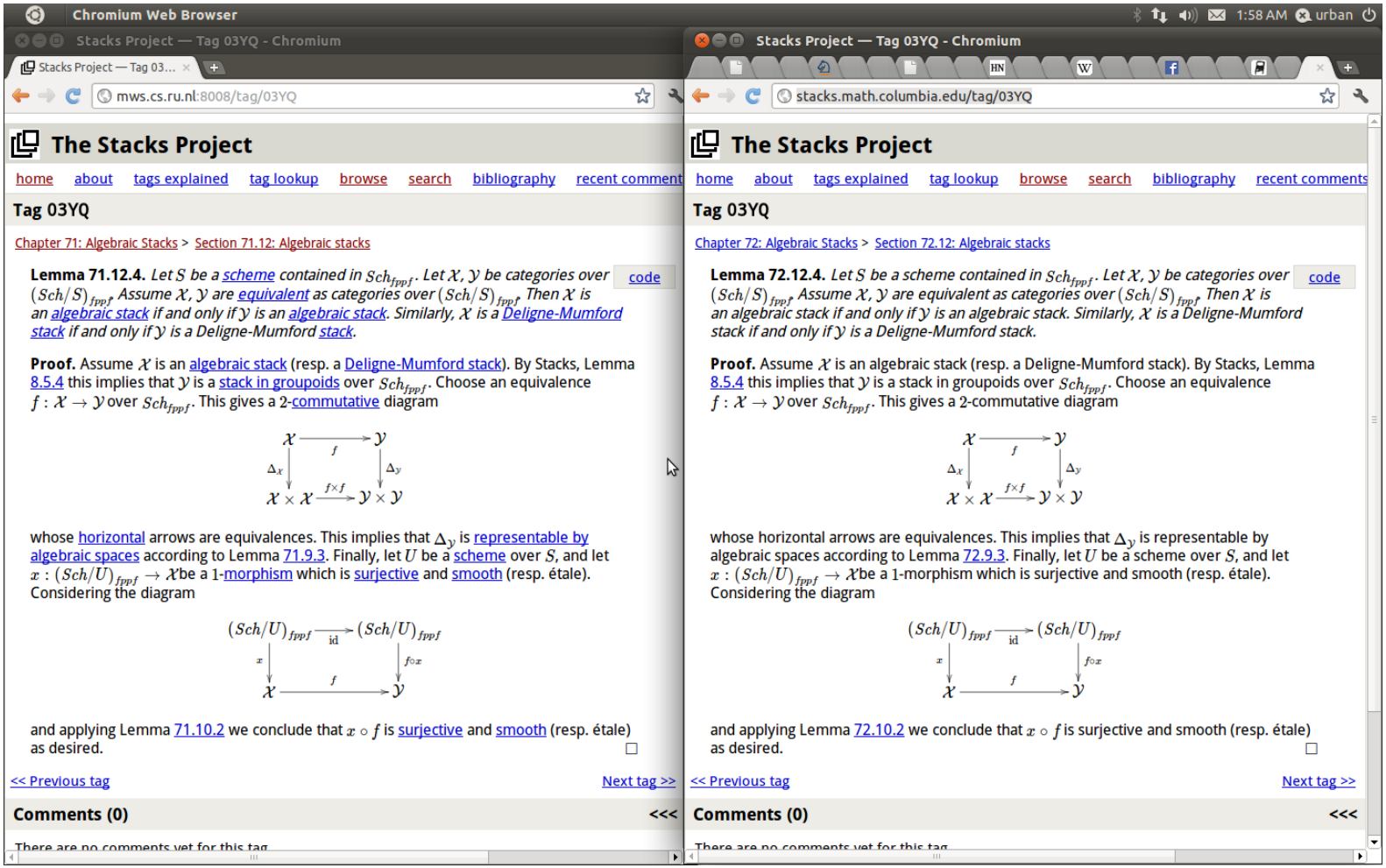

Fig. 1: Auto-linked and original page for Tag 03YQ