The Software Development Approach in Mathematics

Albert-Ludwigs-Universität Freiburg

Konrad Voelkel

“Computer Tools in Pure Math”
Freiburg, November 2014

Goal of this talk

Change your approach to research ...slightly

You could try out some computer tools, think about possible new ones, and help developing them!

Overview

- Why Homotopy Type Theory is interesting
- Non-Computational Computer Tools
- Developing new tools

Most text and images in these slides are hyperlinked.
Please help saving trees by not printing this.

Proofs are Programs

Historical context:

- 1934, 1958 Curry and 1969 Howard: correspondence between certain proof theories and typed lambda calculus. “Curry-Howard correspondence”
- 1960s - de Bruijn: Automath, first proof checker, independently rediscovered “proofs are programs”
- Here, a “proof” is a formal deduction.
- Problems: most people don’t want to think about formal deductions at all, and it’s tedious.

Background:

- 1963 Lawvere: categorical semantics for algebraic theories.
- 1980 Lambek: cartesian closed categorical semantics for simply typed lambda calculus.

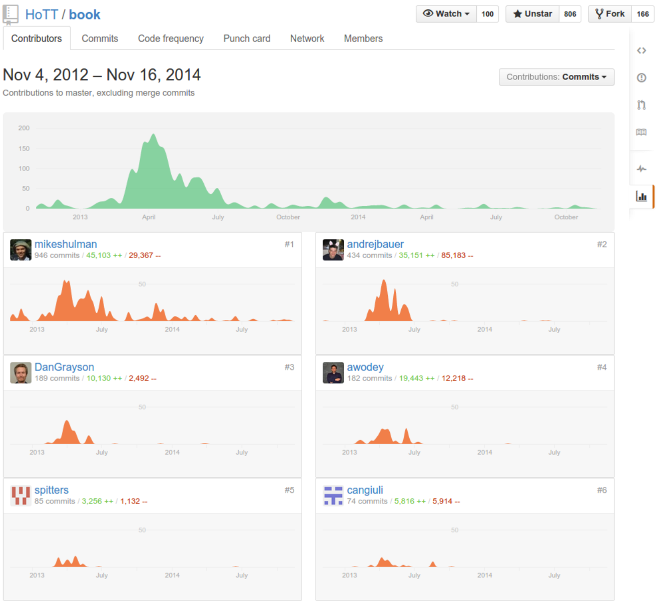
- 1983 Grothendieck: homotopy type = ∞-groupoid.
- 1995 Hofmann and Streicher: model of dependent type theory with non-trivial identity types, turning types into groupoids.
- 2006 Awodey and Voevodsky: identity types as path spaces, turning types into ∞-groupoids.

Homotopy Type Theory

It's interesting for several reasons:

- Conjecturally, internal logic in an $(\infty, 1)$-topos
- Synthetic homotopy theory easy to formalise
- Might entice a new generation of researchers studying formal methods *along* their other research
- IAS wrote a 600-page book with many authors in short time, using GitHub

Example: GitHub, a tool from software development, used for a math book


HoTT / book

Watch 100 | Unstar 806 | Fork 166

Contributors | Commits | Code frequency | Punch card | Network | Members

Nov 4, 2012 – Nov 16, 2014

Contributions to master, excluding merge commits

Contributions: Commits

mikeshulman #1
946 commits / 45,103 ++ / 29,367 --

andrejbauer #2
434 commits / 35,151 ++ / 85,183 --

DanGrayson #3
189 commits / 10,130 ++ / 2,492 --

awodey #4
182 commits / 19,443 ++ / 12,218 --

spitters #5
85 commits / 3,256 ++ / 1,132 --

cangiuli #6
74 commits / 5,816 ++ / 5,914 --

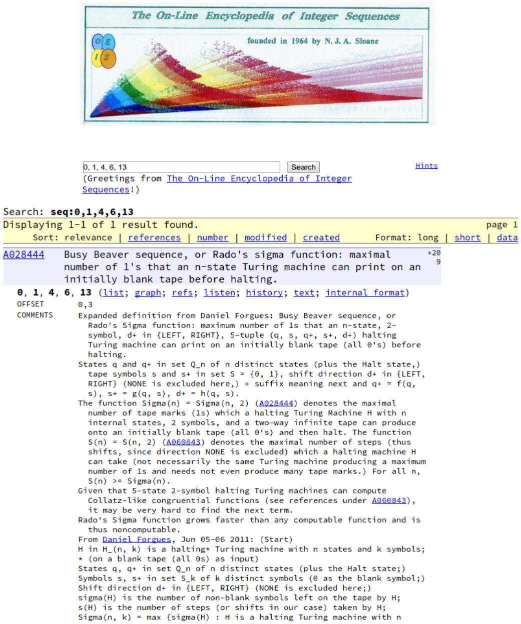
There is more than “proofs are programs”!

Organising content and collaboration:

- Hypertext and Wikis
  No pressure to linearize,
  easy large-scale micro-collaboration.
- (Distributed) Version Control Systems (like Git).
- Databases (with inference engines).

There is also the article by Thomas Hales about recent developments in computer-checked proofs, titled “Mathematics in the Age of the Turing Machine”

Example of a mathematical database: OEIS


The On-Line Encyclopedia of Integer Sequences
founded in 1964 by N. J. A. Sloane
O E I S

0, 1, 4, 6, 13 Search Hints

(Greetings from The On-Line Encyclopedia of Integer Sequences!)

Search: **seq:0,1,4,6,13**

Displaying 1-1 of 1 result found. page 1

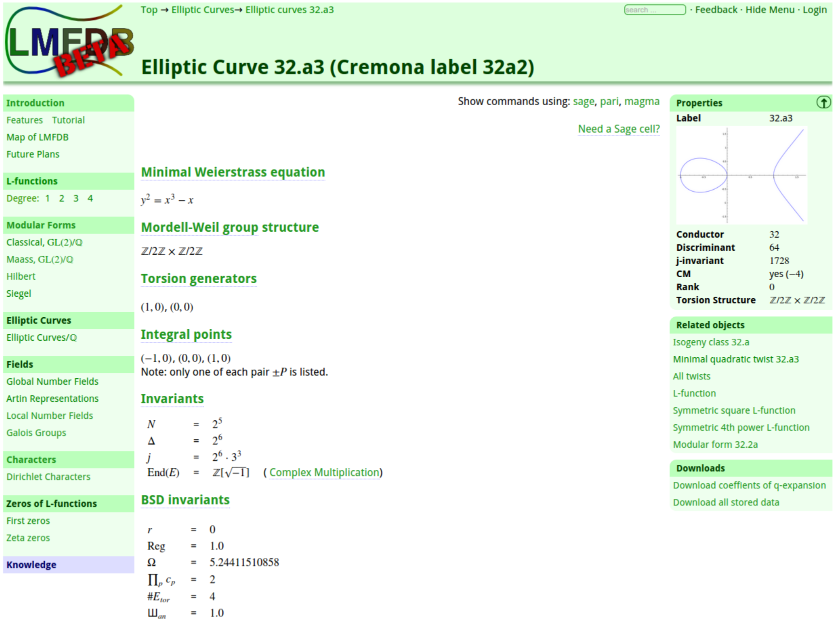
Sort: relevance | references | number | modified | created Format: long | short | data

A028444 Busy Beaver sequence, or Rado's sigma function: maximal number of 1's that an n-state Turing machine can print on an initially blank tape before halting. +20 9

0, 1, 4, 6, 13 (list; graph; refs; listen; history; text; internal format)

```
OFFSET      0,3
COMMENTS    Expanded definition from Daniel Forgues: Busy Beaver sequence, or
              Rado's Sigma function: maximum number of 1s that an n-state, 2-
              symbol, d+ in {LEFT, RIGHT}, 5-tuple (q, s, q+, s+, d+) halting
              Turing machine can print on an initially blank tape (all 0's) before
              halting.
            States q and q+ in set Q_n of n distinct states (plus the Halt state,)
              tape symbols s and s+ in set S = {0, 1}, shift direction d+ in {LEFT,
              RIGHT} (NONE is excluded here,) + suffix meaning next and q+ = f(q,
              s), s+ = g(q, s), d+ = h(q, s).
            The function Sigma(n) = Sigma(n, 2) (A028444) denotes the maximal
              number of tape marks (1s) which a halting Turing Machine H with n
              internal states, 2 symbols, and a two-way infinite tape can produce
              onto an initially blank tape (all 0's) and then halt. The function
              S(n) = S(n, 2) (A060843) denotes the maximal number of steps (thus
              shifts, since direction NONE is excluded) which a halting machine H
              can take (not necessarily the same Turing machine producing a maximum
              number of 1s and needs not even produce many tape marks.) For all n,
              S(n) >= Sigma(n).
            Given that 5-state 2-symbol halting Turing machines can compute
              Collatz-like congruential functions (see references under A060843),
              it may be very hard to find the next term.
            Rado's Sigma function grows faster than any computable function and is
              thus noncomputable.
            From Daniel Forgues, Jun 05-06 2011: (Start)
            H in H_(n, k) is a halting* Turing machine with n states and k symbols;
            * (on a blank tape (all 0s) as input)
            States q, q+ in set Q_n of n distinct states (plus the Halt state;)
            Symbols s, s+ in set S_k of k distinct symbols (0 as the blank symbol;)
            Shift direction d+ in {LEFT, RIGHT} (NONE is excluded here;)
            sigma(H) is the number of non-blank symbols left on the tape by H;
            s(H) is the number of steps (or shifts in our case) taken by H;
            Sigma(n, k) = max {sigma(H) : H is a halting Turing machine with n
```

Example of a mathematical database: LFMDB



Top → Elliptic Curves→ Elliptic curves 32.a3

search ... · Feedback · Hide Menu · Login

Elliptic Curve 32.a3 (Cremona label 32a2)

Introduction
Features Tutorial
Map of LMFDB
Future Plans

L-functions
Degree: 1 2 3 4

Modular Forms
Classical, GL(2)/Q
Maass, GL(2)/Q
Hilbert
Siegel

Elliptic Curves
Elliptic Curves/Q

Fields
Global Number Fields
Artin Representations
Local Number Fields
Galois Groups

Characters
Dirichlet Characters

Zeros of L-functions
First zeros
Zeta zeros

Knowledge

Show commands using: sage, pari, magma

Need a Sage cell?

Minimal Weierstrass equation

$y^2 = x^3 - x$

Mordell-Weil group structure

$\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$

Torsion generators

$(1, 0), (0, 0)$

Integral points

$(-1, 0), (0, 0), (1, 0)$
Note: only one of each pair $\pm P$ is listed.

Invariants

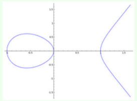
N	=	2^5	
Δ	=	2^6	
j	=	$2^6 \cdot 3^3$	
$\mathrm{End}(E)$	=	$\mathbb{Z}[\sqrt{-1}]$	(Complex Multiplication)

BSD invariants

r	=	0
Reg	=	1.0
Ω	=	5.24411510858
$\prod_p c_p$	=	2
$\#E_{tor}$	=	4
$Ш_{an}$	=	1.0

Properties

Label	32.a3



Conductor	32
Discriminant	64
j-invariant	1728
CM	yes (−4)
Rank	0
Torsion Structure	$\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$

Related objects
Isogeny class 32.a
Minimal quadratic twist 32.a3
All twists
L-function
Symmetric square L-function
Symmetric 4th power L-function
Modular form 32.2a

Downloads
Download coeffients of q-expansion
Download all stored data

From Documents to Knowledge Models

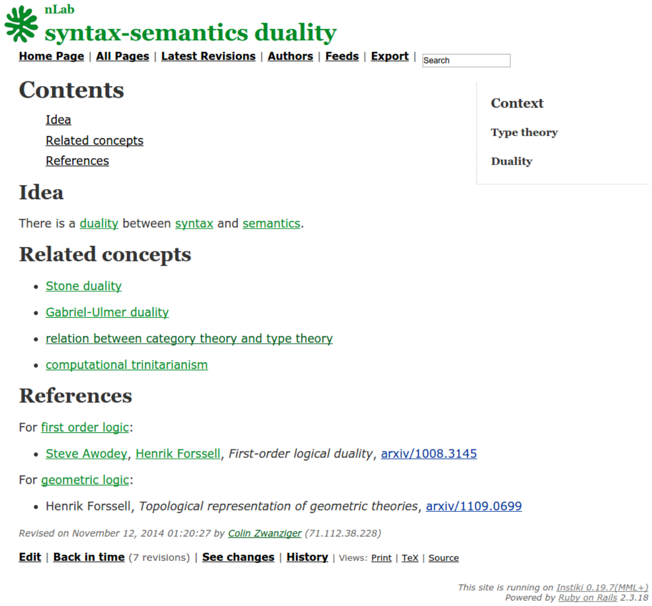
Computers make feasible the maintenance and sharing of the underlying *mental models behind documents*.

Documents

- are static, can be cited.
- have logical structure; meaningful parts can be cited.
- are linear, can be printed.
- are packaged, can be exchanged.

Mental models...? $\Longrightarrow$ Knowledge models!
A comparison of the features of a document with those of knowledge models can be found in an article by Max Voelkel, titled “From Documents to Knowledge Models”

Example of a Knowledge Model: the nLab wiki



nLab

syntax-semantics duality

Home Page | All Pages | Latest Revisions | Authors | Feeds | Export | Search

Contents

Idea

Related concepts

References

Context

Type theory

Duality

Idea

There is a duality between syntax and semantics.

Related concepts

- Stone duality
- Gabriel-Ulmer duality
- relation between category theory and type theory
- computational trinitarianism

References

For first order logic:

- Steve Awodey, Henrik Forssell, *First-order logical duality*, arxiv/1008.3145

For geometric logic:

- Henrik Forssell, *Topological representation of geometric theories*, arxiv/1109.0699

Revised on November 12, 2014 01:20:27 by Colin Zwanziger (71.112.38.228)

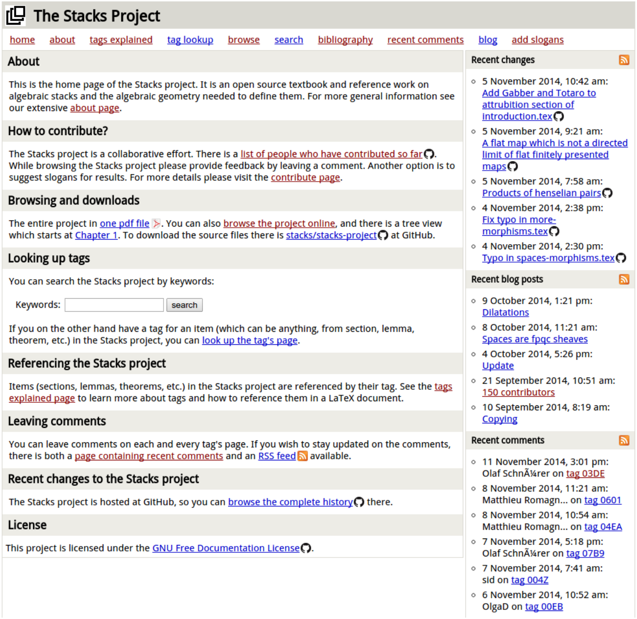
Edit | **Back in time** (7 revisions) | **See changes** | **History** | Views: Print | TeX | Source

This site is running on Instiki 0.19.7(MML+)
Powered by Ruby on Rails 2.3.18

Main problems with knowledge models:

- How to cite a model and parts of it?
- How to linearize (e.g. for printing)?
- Which text format, if not TeX?
- How to handle versions?
- How to package a model or parts?
- There is no standard approach yet.

Example of a Knowledge Model: the Stacks Project



The Stacks Project

home about tags explained tag lookup browse search bibliography recent comments blog add slogans

About

This is the home page of the Stacks project. It is an open source textbook and reference work on algebraic stacks and the algebraic geometry needed to define them. For more general information see our extensive about page.

How to contribute?

The Stacks project is a collaborative effort. There is a list of people who have contributed so far. While browsing the Stacks project please provide feedback by leaving a comment. Another option is to suggest slogans for results. For more details please visit the contribute page.

Browsing and downloads

The entire project in one pdf file. You can also browse the project online, and there is a tree view which starts at Chapter 1. To download the source files there is stacks/stacks-project at GitHub.

Looking up tags

You can search the Stacks project by keywords:

Keywords: [ ] search

If you on the other hand have a tag for an item (which can be anything, from section, lemma, theorem, etc.) in the Stacks project, you can look up the tag's page.

Referencing the Stacks project

Items (sections, lemmas, theorems, etc.) in the Stacks project are referenced by their tag. See the tags explained page to learn more about tags and how to reference them in a LaTeX document.

Leaving comments

You can leave comments on each and every tag's page. If you wish to stay updated on the comments, there is both a page containing recent comments and an RSS feed available.

Recent changes to the Stacks project

The Stacks project is hosted at GitHub, so you can browse the complete history there.

License

This project is licensed under the GNU Free Documentation License.

Recent changes

- 5 November 2014, 10:42 am: Add Gabber and Totaro to attrubition section of introduction.tex
- 5 November 2014, 9:21 am: A flat map which is not a directed limit of flat finitely presented maps
- 5 November 2014, 7:58 am: Products of henselian pairs
- 4 November 2014, 2:38 pm: Fix typo in more-morphisms.tex
- 4 November 2014, 2:30 pm: Typo in spaces-morphisms.tex

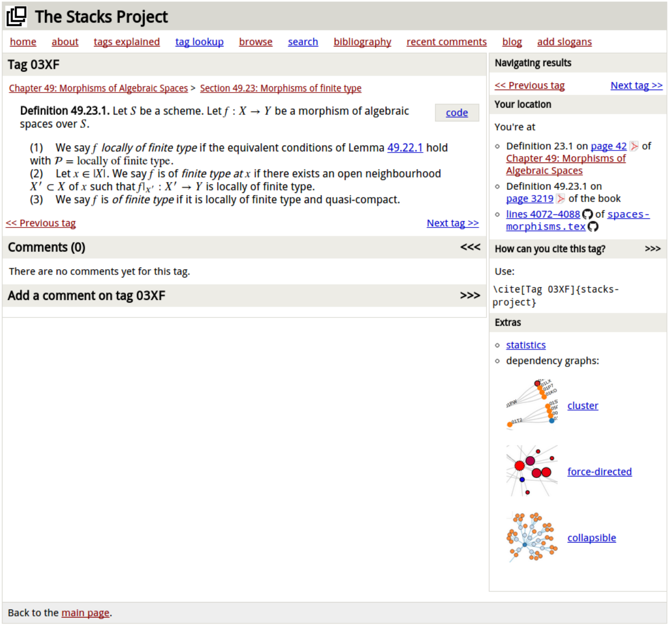
Recent blog posts

- 9 October 2014, 1:21 pm: Dilatations
- 8 October 2014, 11:21 am: Spaces are fpqc sheaves
- 4 October 2014, 5:26 pm: Update
- 21 September 2014, 10:51 am: 150 contributors
- 10 September 2014, 8:19 am: Copying

Recent comments

- 11 November 2014, 3:01 pm: Olaf SchnÃ¼rer on tag 03DE
- 8 November 2014, 11:21 am: Matthieu Romagn... on tag 0601
- 8 November 2014, 10:54 am: Matthieu Romagn... on tag 04EA
- 7 November 2014, 5:18 pm: Olaf SchnÃ¼rer on tag 07B9
- 7 November 2014, 7:41 am: sid on tag 004Z
- 6 November 2014, 10:52 am: OlgaD on tag 00EB

Example of a Knowledge Model: the Stacks Project, Tags



The Stacks Project

home about tags explained tag lookup browse search bibliography recent comments blog add slogans

Tag 03XF

Chapter 49: Morphisms of Algebraic Spaces > Section 49.23: Morphisms of finite type

code

Definition 49.23.1. Let S be a scheme. Let $f : X \to Y$ be a morphism of algebraic spaces over S.

(1) We say f *locally of finite type* if the equivalent conditions of Lemma 49.22.1 hold with $\mathcal{P} = \text{locally of finite type}$.

(2) Let $x \in |X|$. We say f is of *finite type at* x if there exists an open neighbourhood $X' \subset X$ of x such that $f|_{X'} : X' \to Y$ is locally of finite type.

(3) We say f is *of finite type* if it is locally of finite type and quasi-compact.

<< Previous tag Next tag >>

Comments (0) <<<

There are no comments yet for this tag.

Add a comment on tag 03XF >>>

Navigating results

<< Previous tag Next tag >>

Your location

You're at

- Definition 23.1 on page 42 of Chapter 49: Morphisms of Algebraic Spaces
- Definition 49.23.1 on page 3219 of the book
- lines 4072–4088 of spaces-morphisms.tex

How can you cite this tag? >>>

Use:

```
\cite[Tag 03XF]{stacks-project}
```

Extras

- statistics
- dependency graphs:

cluster

force-directed

collapsible

Back to the main page.

There was another big problem, not long ago:

- Math was mostly done with documents because of (the restrictions of) TeX.
- MathJax changed that.
- Now we have a growing number of websites with readable mathematical content.

In the far future

We will employ a computer whereever we can, so humans can focus on the mathematics itself (discovery and communication).

- A large core of “standard” material will be formalised.
  It will be less work to formalise new research than it is now.
- There will be IDEs for doing mathematics.
  With autocompletion, style checking, type checking, ...
- Textbooks will be in the form of hypermedia.
- Research will be communicated as knowledge models.

Change everything?

We won’t change everything over night.

Two directions of change:

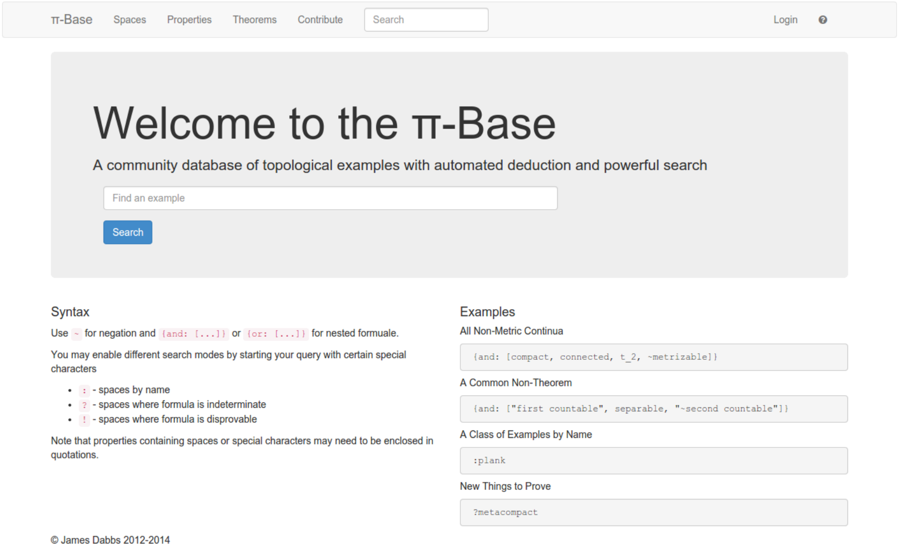
- Bottom-up:
  formalise mathematics from the first definitions on.
- Top-down:
  add semi-formal metadata to a subdomain of mathematics.

Examples around Topological Spaces

- Bottom-up: formalise what a space is in some system, formalise theorems and proofs. Now you can verify proofs.
- Top-down 1: formalise which theorems use which other theorems (just search for \ref commands in your tex file). Now you have a dependency graph.
- Top-down 2: compile a list of space properties and implications and counterexamples. Hook up a simple inference engine, get “new” theorems.
- Top-down 3: ...

In some areas of mathematics, the top-down approach might have a better ROI at the moment.

Example of a semiformal Knowledge Model: π-Base



Welcome to the π-Base

A community database of topological examples with automated deduction and powerful search

Syntax

Use `~` for negation and `{and: [...]}` or `{or: [...]}` for nested formuale.

You may enable different search modes by starting your query with certain special characters

- `:` - spaces by name
- `?` - spaces where formula is indeterminate
- `!` - spaces where formula is disprovable

Note that properties containing spaces or special characters may need to be enclosed in quotations.

Examples

All Non-Metric Continua

```
{and: [compact, connected, t_2, ~metrizable]}
```

A Common Non-Theorem

```
{and: ["first countable", separable, "~second countable"]}
```

A Class of Examples by Name

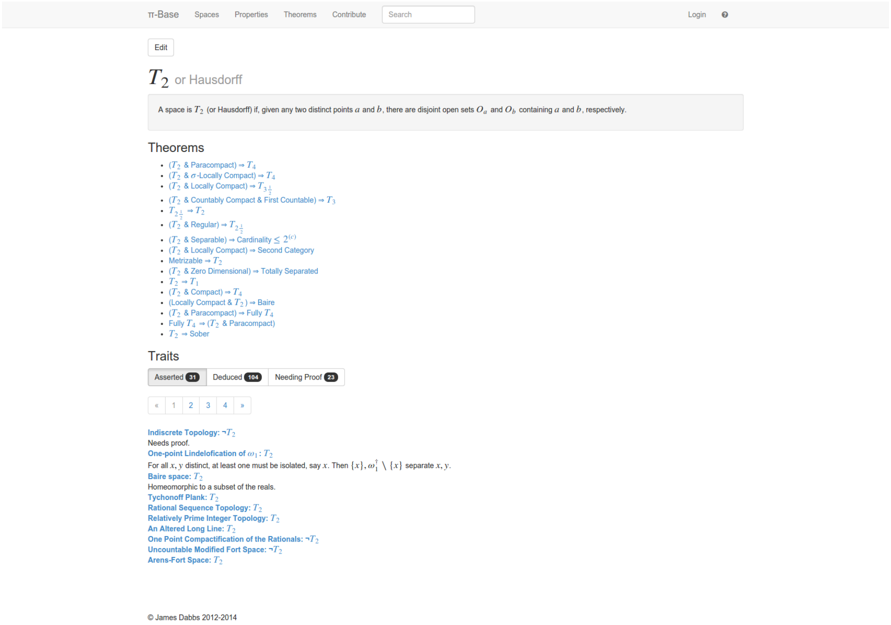
```
:plank
```

New Things to Prove

```
?metacompact
```

© James Dabbs 2012-2014

Example of a semiformal Knowledge Model: π-Base, an item



π-Base Spaces Properties Theorems Contribute Search Login

Edit

T_2 or Hausdorff

A space is T_2 (or Hausdorff) if, given any two distinct points a and b, there are disjoint open sets O_a and O_b containing a and b, respectively.

Theorems

- (T_2 & Paracompact) ⇒ T_4
- (T_2 & σ-Locally Compact) ⇒ T_4
- (T_2 & Locally Compact) ⇒ $T_{3\frac{1}{2}}$
- (T_2 & Countably Compact & First Countable) ⇒ T_3
- $T_{2\frac{1}{2}}$ ⇒ T_2
- (T_2 & Regular) ⇒ $T_{2\frac{1}{2}}$
- (T_2 & Separable) ⇒ Cardinality $\leq 2^{(c)}$
- (T_2 & Locally Compact) ⇒ Second Category
- Metrizable ⇒ T_2
- (T_2 & Zero Dimensional) ⇒ Totally Separated
- T_2 ⇒ T_1
- (T_2 & Compact) ⇒ T_4
- (Locally Compact & T_2) ⇒ Baire
- (T_2 & Paracompact) ⇒ Fully T_4
- Fully T_4 ⇒ (T_2 & Paracompact)
- T_2 ⇒ Sober

Traits

Asserted 31 | Deduced 104 | Needing Proof 23

« 1 2 3 4 »

Indiscrete Topology: $\neg T_2$
Needs proof.
One-point Lindelofication of ω_1 : T_2
For all x, y distinct, at least one must be isolated, say x. Then $\{x\}, \omega_1^\dagger \setminus \{x\}$ separate x, y.
Baire space: T_2
Homeomorphic to a subset of the reals.
Tychonoff Plank: T_2
Rational Sequence Topology: T_2
Relatively Prime Integer Topology: T_2
An Altered Long Line: T_2
One Point Compactification of the Rationals: $\neg T_2$
Uncountable Modified Fort Space: $\neg T_2$
Arens-Fort Space: T_2

Examples around Morphisms of Varieties

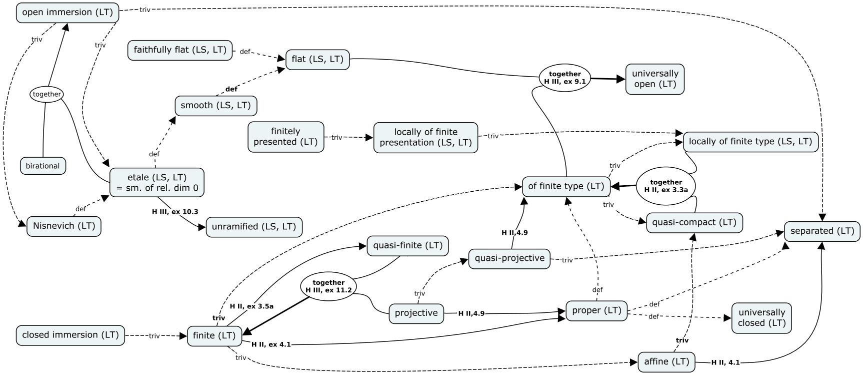
- In algebraic geometry, one considers (after Grothendieck) the principal player to be the morphism, not the object.
- There are many properties a morphism of algebraic varieties (or schemes, spaces, sheaves, stacks) may enjoy.
- On the set of such properties, logical implication gives a partial order. For every term $P \implies Q$ one may either give a counterexample or a reference to a proof in the literature.
- This yields an annotated graph.

Example: A Diagram of Properties of Morphisms of Varieties

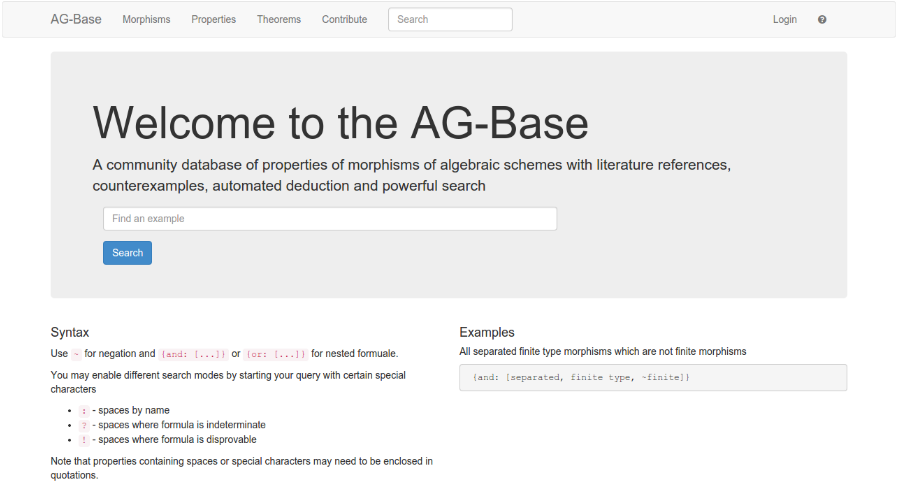
Properties of morphisms in the category of Noetherian schemes, all stable under base change and composition.

Warning: All definitions are used according to Hartshorne, and differ from the definitions of EGA. Therefore, finite doesn't imply projective; smooth is not necessarily locally of finite presentation; quasi-finite is just finite fibers, not necessarily of finite type.

(LT) = local on the target,
(LS) = local on the source.
triv = implies trivially,
def = implies by definition,
H II,3.4 = Hartshorne Chapter II, 3.4.



Mockup for an “AG-Base” tool



Welcome to the AG-Base

A community database of properties of morphisms of algebraic schemes with literature references, counterexamples, automated deduction and powerful search

Syntax

Use `~` for negation and `{and: [...]}` or `{or: [...]}` for nested formuale.

You may enable different search modes by starting your query with certain special characters

- `:` - spaces by name
- `?` - spaces where formula is indeterminate
- `!` - spaces where formula is disprovable

Note that properties containing spaces or special characters may need to be enclosed in quotations.

Examples

All separated finite type morphisms which are not finite morphisms

```
{and: [separated, finite type, ~finite]}
```

Open Questions

- What else could be done with the π-base software? Spitters suggests an instance for algebraic structures.
- How could we ideally encode more complex domains for properties? e.g. spaces *and* maps between spaces.
- How could one encode the mathematical foundations (with or without AC...)?
- How should one use Isabelle or Agda together with π-base?
- How can one make the data from a π-base instance more accessible for others by using established MKM standards?

Conclusion

- Since formal proofs are programs, we can borrow tools from software engineering.
- Flexible knowledge models may often replace documents.
- Some low-hanging fruit is waiting in top-down formalisation.
- You should try out some new tools and help making them better, with feedback, content or code.
- Interested in π-base? Get in touch with James Dabbs!
- Interested in AG-Base?
  Get in touch with Daniel Harrer or me!

Thank you for your attention.