130k Lines of Formal Topology in Two Weeks: Simple and Cheap Autoformalization for Everyone?

130k Lines of Formal Topology in Two Weeks: Simple and Cheap Autoformalization for Everyone?

Josef Urban
AI4REASON, Prague, Czech Republic
University of Gothenburg, Sweden
Chalmers University of Technology, Gothenburg, Sweden

Abstract

This is a brief description of a project that has already autoformalized a large portion of the general topology from the Munkres textbook (which has in total 241 pages in 7 chapters and 39 sections). The project has been running since November 21, 2025 and has as of January 4, 2026, produced 160k lines of formalized topology. Most of it (about 130k lines) have been done in two weeks, from December 22 to January 4, for an LLM subscription cost of about $100. This includes a 3k-line proof of Urysohn's lemma, a 2k-line proof of Urysohn's Metrization theorem, over 10k-line proof of the Tietze extension theorem, and many more (in total over 1.5k lemmas/theorems).

The approach is quite simple and cheap: build a long-running feedback loop between an LLM and a reasonably fast proof checker equipped with a core foundational library. The LLM is now instantiated as ChatGPT (mostly 5.2) or Claude Sonnet (4.5) run through the respective Codex or Claude Code command line interfaces. The proof checker is Chad Brown's higher-order set theory system Megalodon, and the core library is Brown's formalization of basic set theory and surreal numbers (including reals, etc). The rest is some prompt engineering and technical choices which we describe here. Based on the fast progress, low cost, virtually unknown ITP/library, and the simple setup available to everyone, we believe that (auto)formalization may become quite easy and ubiquitous in 2026, regardless of which proof assistant is used.

2012 ACM Subject Classification Theory of computation → Automated reasoning; Theory of computation → Higher order logic; Theory of computation → Logic and verification

Keywords and phrases Autoformalization, Automated reasoning, Interactive theorem proving, Formal proof assistants, Machine learning, Language Models

Digital Object Identifier 10.4230/LIPIcs.ITP.2026.31

Category Short Paper

Related Version *Full Version*: `https://arxiv.org/abs/2601.03298` [13]

Supplementary Material
Software: `https://github.com/mgwiki/mgw_test/blob/main/mglib/topology.mg`
archived at `swh:1:dir:5514ac5dd418710e862d80cce366bc74efde42c2`

Funding This work was supported by the ERC project NextReason, Amazon Research Awards, the Czech MEYS under the ERC CZ project *POSTMAN* no. LL1902, Czech Science Foundation grant no. 25-17929X, and the sponsors of the AI4REASON institute.[1]

Acknowledgements Chad Brown has largely developed Megalodon, its hammer, and the initial core set-theoretical library, helped me to set it up for the experiment, and provided encouraging comments on the early system's formalizations. Zar Goertzel has kept me informed about his experiments with LLMs and the related coding agents like Claude Code and Codex. I took his advice and fully switched to ChatGPT 5.2 (and the $200/month Pro subscription) after he reported encouraging results. He has already finished a 13k-line autoformalization of the Ramsey(3,6) theorem. Atle Hahn and Adrian De Lon have been doing interesting experiments with LLM-based autoformalization into Naproche. Atle's impressive (and working) prompts demonstrate the power and potential of "programming" LLMs to do autoformalization. Thanks to John Harrison for supporting our

[1] `https://ai4reason.eu/sponsors.html`

early autoformalization efforts and also to Larry Paulson and Dominic Mulligan for encouragement. Cezary Kaliszyk has helped with the Megalodon/hammer development and related infrastructure, and spearheaded the first learning-based autoformalization ideas, projects and approaches with me since 2014. The autoformalization community has grown a lot since then – thanks to all these (and especially the early) believers.

Declaration about GenAI/LLM use As explained in the paper, the formalization was done exclusively by LLMs and the writing of the article was assisted by LLMs.

1 How it started

I have been working (more or less seriously) with my colleagues on learning-assisted autoformalization since 2014[2] [8, 7]. A bit of autoformalization history is given in my 2024 Bonn talk[3] and its slides.[4] Long story short, since the first MaLARea experiments in 2005/6 [11], I have become quite convinced about the power of feedback loops between reasoning/logic/deduction and learning, and believed that we are ourselves in such a feedback loop when we formalize math (see e.g. the last two sentences in [12]). In 2018, I got further impressed by the neural language models (especially attention-based) showing surprisingly good informal-to-formal translation capability in our first experiments over a large synthetic Mizar-to-LaTeX corpus [15]. I started to believe my own autoformalization propaganda, in the sense that autoformalization would become usable much sooner than my 2014 conservative 25-year horizon estimate.[5] Since 2020, the field has become increasingly popular in industry teams with much more resources [16, 10, 6], and (seeing their heavy buy-in[6]) I have (with some exceptions) mostly focused on other (less resource-heavy) feedback loops in ATP and conjecturing [14, 5, 3, 4, 9]. I have occasionally tried to get ChatGPT (and similar LLMs) to autoformalize smaller papers (such as the Ramsey(3,6) 4-page paper [2]) since it appeared, but it initially didn't seem very good at carrying out deeper and longer work – at least not without building some infrastructure around it.

The immediate impulse for starting the experiment described here were Zar Goertzel's reports of good autoformalization progress with coding agents such as Codex and Claude Code. In particular, in late 2025 (overlapping with this work) Zar managed to finish a full Lean proof of my testing paper on Ramsey(3,6) with Claude Code in about 13k lines of Lean.[7] After some false starts and brief discussion with Chad Brown, I have decided to set up the current experiment in formalization of topology with the Megalodon proof assistant and the LLM-based command-line coding agents.

2 Current Setup

The latest setup has been (almost) stable since December 22 and used for (almost) automated formalization resulting in about 130k lines code and the above mentioned fully proved major theorems. The setup is as follows:

---

[2] I have likely coined the "auto-formalization" term in this context in my CICM'14 talk `https://t.ly/PaeKH` [8].

[3] `https://www.youtube.com/watch?v=4JeezEGc_gQ`

[4] `http://grid01.ciirc.cvut.cz/~mptp/bonn24_af.pdf`

[5] `http://ai4reason.org/aichallenges.html`

[6] Sometimes perhaps too heavy – see the claims about "introducing autoformalization" at `https://archive.ph/8Ib1k`.

[7] `https://github.com/zariuq/ai-agents/tree/main/lean-projects/ramsey36/Ramsey36`

1. A ChatGPT Pro ($200/month) subscription, giving access to the OpenAI's Codex coding agent.
2. I always use the ChatGPT 5.2 model with high (not extra high) reasoning.[8]
3. I isolate the coding agent using the Linux bubblewrap (bwrap[9]) sandboxing tool. I could have likely used any other reasonable sandboxing solution. I want to give the coding agents unlimited powers (calling all sorts of dangerous tools), hence their isolation inside the sandbox. I map several outside directories into the sandbox with various binaries and project data. While no serious harm was done by the agents so far inside the sandbox, there were already several occasions (like the agents getting confused about data location, file names, etc) that made me feel vindicated about having the isolation. I periodically back up the project data directory from the outside.
4. Inside bwrap, I run the command-line coding agent (now Codex, but I can and did run Claude Code too with similar approach and results).
5. Bwrap itself is run in the tmux terminal manager/multiplexer, typically in a long-running (multiday) session that is being logged and interacted with from the outside. I could likely have used other similar tools for this such as GNU Screen.
6. I use the Megalodon (higher-order set theory) proof checker binary for the proof checking. The binary used is the one from the Megalodon Wiki (mgwiki) Github repo, which ensures compatibility with the mgwiki toolchain. This allows to push any compiling formalization into the mgwiki, where it is immediately html-ized. This cross-linked HTML presentation allows us to easily study the developing formalization.
7. As the core library I use Brown's formalization of basic set theory and surreal numbers (including reals, etc). We deleted most of the proofs there, replacing them with "admit.", so that the initial library file is smaller and (hopefully) easier to study for the LLMs. I however leave several of Brown's proofs there (such as Diaconescu proof of excluded middle), so that the LLM can (hopefully) study how to prove things in Megalodon.
8. That's it. The rest is some prompting and some additional tools (optional and added later) described below.

3 Prompts

With only a few exceptions, since December 22, the following short constant prompt has been repeatedly given to Codex each time it reports finishing its work (i.e., each time Codex CLI passes the control back to the user, asking for another prompt):

```
Read the file CLAUDE.md .  Treat it as authoritative work instructions.
Follow those instructions exactly for all subsequent actions and responses.
That means work as long as possible (as specified) without stopping.
```

My estimate is that this prompt has been automatically given to Codex about 1000-2000 times during the two weeks.[10] The current (14th) version of the "rules of work" file CLAUDE.md is available at the full arxiv version of this paper [13] and online.[11]

[8] Users with more resources could try to repeat the experiment with other settings and provide further ablations. I don't use the extra high setting because already the high reasoning depletes its weekly usage limit in less than a week.
[9] `https://github.com/containers/bubblewrap`
[10] This can likely be calculated precisely after parsing the Codex logs.
[11] `https://pastebin.com/raw/bGpnimpL`

3.1 Prompting Automation

Before December 22, I was "babysitting" the command-line interface, i.e., looking every now and then if the agent has finished its work and waits for another prompt. This was manageable, because my early $20/month LLM subscriptions would quickly run out of credits anyway and most of the time would be spent waiting for the weekly credit limit to be reset.

With the $200/month ChatGPT Pro subscription the limits on usage got much higher and automating the prompting became necessary. I have implemented it by writing a script[12] that watches the tmux session and feeds it the prompt followed by "Enter" whenever there has not been any change in the session for 60 seconds.

3.2 Initial Prompts

Initially, I have done quite a lot of experiments with the prompting, and also had some special prompts for special tasks like the initial creation of all definitions and theorems/lemmas corresponding to the textbook. See the full arxiv version for details. A major initial effort was to make the coding agent work as independently as possible and as long as possible, without them coming back, asking additional questions, or being derailed in various ways. Ultimately, I started to get independent runs of over 1 hour, and with the Pro subscription sometimes even 2 hours, before giving it the next prompt (which was very often just the same prompt over and over, until I automated it fully as explained above).

3.3 Focused Prompting

On December 31, the system has already done quite some work on the Urysohn lemma but then started to switch to other things. Since this looked like an interesting end-of-year challenge, I issued the following special prompt:

```
Special rule for today:  Return to the proof of Urysohn lemma and finish it.
Don't switch to other things before it is fully finished without any admits,
axioms, etc.  Finishing it is your singular goal for now.
```

This has backfired a bit because the system then started to try all sorts of normally forbidden tricks. I told it to keep following the CLAUDE.md rules but focus on finishing Urysohn. The 3k-line proof indeed got finished by the end of 2025.

I made a similar focused attempt on January 4, after the system made a significant (but likely redundant – see below) initial effort on the Tietze extension theorem, but then started to switch to other things. Since this was interesting I tried to focus it as follows (this time I was wiser and also told it to follow the rules – which worked well):

```
Read the file CLAUDE.md .  Treat it as authoritative work instructions.
Follow those instructions exactly for all subsequent actions and responses.
That means work as long as possible (as specified) without stopping.  Special
rule for today:  Return to the proof of Tietze_stepII_real_extension_nonempty
and finish it.  Don't switch to other things before it is fully finished.
```

This indeed led to the massive effort that finished the 10k-line proof (and all its prerequisites) after about 20 hours. I believe this now provides quite a usable way of combining the system's automated work on a large (textbook) formalization with the user occasionally impatiently overriding it when he wants a particular proof done first.

[12] Available at the full arxiv version of this paper [13] and online at `https://pastebin.com/raw/Ah0pWyS1`

4 Further Utilities

Hammer (aby) attempts. Megalodon has since 2024 a simple higher-order hammer "aby" [1]. In particular, it can export the problems to the TH0 TPTP format and call e.g. higher-order Vampire. There is so far no proof reconstruction (as e.g. in Naproche). Initially it was tempting to save the LLM the work on detailed Megalodon proofs and instead employ the hammer as much as possible. I have written the scripts that automate that (to some extent) but so far this has not been very useful and likely requires further work on caching and (at least simple) premise selection, possibly also some of the well-known translations to FOF (first-order). A major issue that should be addressed (in the absence of proof reconstruction) is smart re-running of the ATPs when something changes.

Theorem and Dependency Tracking. Relatively late in the development I wanted the LLM to have a better overview of the theorem dependencies and the biggest "choke points". It looked like for example the completeness of reals was a major cause of recursive admits, even though it should be quite feasible based on Brown's surreal numbers development (which turned out to be true – see R_lub_exists below). So I have added a simple dependency tracking script that at every iteration gets called on the current file and produces for each theorem a one-line info with its current status (fully proved, admitted, recursively admitted), length of the (partial) proof, and the current proof dependencies. It seems that this has helped the LLM to orient itself better (see also the advice about this in CLAUDE.md).

5 Enforcing the order of work

This is a relatively uncharted territory. People who formalize math have developed good routines, but I am not aware of some "handbook of good formalization practices".

Some of the problems that appeared were:

1. The LLMs were initially (occasionally) lazy in formalizing the theorem and definition statements. Sometimes they created stubs, ad-hoc or special cases instead of general cases, etc. This is dangerous if the LLM later forgets about it (its context window is finite) and takes such simplified specs seriously. I have countered it by telling it to be very cautious about such things, and once or twice even doing a special prompt to assess/remove the inaccuracies.
2. Initially it had a tendency to go for easy lemmas (often exercises/examples), rather than the hard/long theorems. Also, it could endlessly jump around searching for something easy. Again see CLAUDE.md for my attempts to prevent that (without being too strict).
3. I also wanted it to be continuously aware of all sections, rather than getting stuck in just one section for long time. Because this could later lead to major refactoring. I have tried to convince it to keep a PROGRESS (status) file (see CLAUDE.md) but this attempt has so far only mixed results. The trade-offs between going deep and proving hard theorems and being aware of the full formalization picture are obviously nontrivial also in large-scale human formalization projects.

6 Miscellaneous Comments

In general, I currently do not have the time and energy to do a full-scale analysis of what went well and what went wrong. LLMs and other tools could be used in the future to find automatically interesting cases from the complete logs (which I am keeping) and the publicly available git commits.[13] Perhaps this could lead to a feedback loop augmenting the currently used prompts and tools based on such analysis.

Resources Used. While the Pro subscription is indeed only $200/month, I have tried to use independent tools to measure the USD cost of calling the LLM. See the full arxive version for details. Note that the reported costs are much higher than the $100 spent over the two weeks. One wonders what kind of business policies the LLM companies are implementing (vicious fight for the market/subscribers?, overcharging companies for the API usage?, trying to get as much data as possible?, bait-and-switch?, etc).

Context Compactification. This is a major issue with the LLM-based technology I use. Every now and then, the coding agent needs to get rid of unessential parts in its (limited) context window and keep only the parts essential for further processing. The results of the compactification (which is likely as proprietary as the LLMs I use) can be quite unpredictable.

Even worse, there seems to be a hard limit on how much compactification can be done for a continuous session. I have reached this hard limit already twice: once (December 26) when the ChatGPT session file reached about 252MB, and another time (January 2) when the session file reached 351MB. Once this hard limit is reached, it seems the session can no longer be continued in any way. The standard advice I found seems to be "do not run long sessions". Instead, people should maintain the essential status and progress themselves when doing longer projects like these, always starting a new session with such updated status/progress.

Since this looked like a bigger project and I liked how the long-running session behaved (not wanting to lose all it learned), I decided to ignore this advice. Instead of resetting completely, I reset to an earlier version of the session. In particular, when I reach the hard limit, I just rewrite the (big) ChatGPT session (jsonl) file with a (much) younger (and smaller) version of it. In particular, for both resets I used an early 65MB-big version of the session file from December 18. This is likely an unsupported and dangerous trick, but it has so far worked quite well for me.

Running Megalodon. As the file grows, running Megalodon gets slower and possibly problematic. While it took under 1s on the initial library file, the latest verification on my notebook takes 32s. However the file has now almost 170k lines and 8MB An obvious solution is to start to split the file or to start admitting (the `@proof` pragma in Mizar) the finished theorems. So far, I haven't done that because: (i) I prefer the proofs to be possibly refactored by the LLMs, (ii) running Megalodon doesn't seem to be the bottleneck compared to the LLM usage and available credits. I.e., the number of calls to the LLM is still quite high and even with the Pro subscription I run out of the LLM credits before the usage limit resets.

Major Theorems Proved. See the arxiv version and its appendix for the listing of the fully proved major theorems. This includes `Tietze_stepII_real_extension_nonempty`, `Urysohn_lemma`, `Urysohn_metrization_theorem`, `one_point_compactification_exists`, `unit_interval_connected` and many others.

[13] `https://github.com/mgwiki/mgw_test/commits/main/`

7 The Battle of the Tietze Hill

A large part of the last two days has been spent on the formalization of the Tietze extension theorem, specifically its Step II. The proof[14] takes 10369 lines and has 142 direct dependencies (36 definitions and 106 theorems).

The textbook proof uses facts about infinite sums of real functions and their uniform convergence, which is a major external dependency not present in the textbook. The agent seems to have been stumped by this and instead kept unrolling the inductive (infinite) construction 12 times (12 terms of the series), producing about 6k lines that are not useful for the final proof (even though they may be useful for figuring out the machinery more concretely before the induction is done).

It has however recovered from this and decided to gradually formalize the necessary prerequisites. The ChatGPT analysis of the unfinished 9k-line proof attempt is available online[15]. The arxiv version shows its periodic report, showing how the fact that complete metric spaces are dealt with much later in the textbook confused the LLM.

In more detail, the problem is that complete metric spaces and Cauchy sequences in them are indeed only defined in section 43 of Munkres, while Tietze is in section 35. The LLM has tried hard to follow the formalization rules. As a result, it has done 11 useless iterations and correction terms (from u2 in CHANGES3650 up to u12 in CHANGES3685), blowing up the proof from 865 lines to 5408 lines in these 35 changes. This took almost 4 hours of work and wasted quite some resources. After some futile attempts, the LLM has reasoned itself into doing the hard work on the prerequisites and moving them before Tietze (see the full arxive version forv details). Note that this qualifies as a large gap, external dependency or even an (unsafe) forward reference in the textbook – something that can be a serious hurdle for human formalizers. The ChatGPT analysis of the final 10k-line proof and its difference to the previous unfinished 9k-line proof is available online[16]. Once this hard battle is finished, the standard statement of Tietze from Munkres which is as follows:

> ▶ **Theorem 35.1** (Tietze extension theorem). *Let X be a normal space; let A be a closed subspace of X.*
>
> **(a)** *Any continuous map of A into the closed interval $[a, b]$ of $\mathbb{R}$ may be extended to a continuous map of all of X into $[a, b]$.*
>
> **(b)** *Any continuous map of A into $\mathbb{R}$ may be extended to a continuous map of all of X into $\mathbb{R}$.*

is (relatively easily) proved via these two Megalodon theorems:

```
Theorem Tietze_extension_real_bounded_interval : forall X Tx A a b f:set,
  Rle a b ->
  normal_space X Tx -> closed_in X Tx A ->
  continuous_map A (subspace_topology X Tx A) R R_standard_topology f ->
  (forall x:set, x :e A -> apply_fun f x :e closed_interval a b) ->
  exists g:set, continuous_map X Tx R R_standard_topology g /\
    (forall x:set, x :e A -> apply_fun g x = apply_fun f x).
```

[14] https://mgwiki.github.io/mgw_test/topology.mg.html#Tietze_stepII_real_extension_nonempty, also line 129871 of https://raw.githubusercontent.com/mgwiki/mgw_test/48bca1fc5df4576f1a28416aa4e423ecfa556f6c/mglib/topology.mg

[15] https://chatgpt.com/share/695a7928-9efc-8002-860f-bb2c0f33663b

[16] https://chatgpt.com/share/695a7928-9efc-8002-860f-bb2c0f33663b

```
Theorem Tietze_extension_real : forall X Tx A f:set,
  normal_space X Tx -> closed_in X Tx A ->
  continuous_map A (subspace_topology X Tx A) R R_standard_topology f ->
  exists g:set, continuous_map X Tx R R_standard_topology g /\
    (forall x:set, x :e A -> apply_fun g x = apply_fun f x).
```

8 Conclusion

This is obviously quite surprising and exciting and I wonder where this will all go. The price tag here (and my coding effort) was really low. It is quite possible that in 2026 we will get most of (reasonably written) math textbooks and papers autoformalized. Or perhaps not – there may be limits this will eventually run into. Perhaps a (sorely needed?) reality check for some (insert a popular proof assistant name here) fundamentalists is that the current experiments were done in an almost unknown proof assistant based on higher-order set theory, with very little exposure to the existing LLMs. The future of AI-assisted (auto)formalization seems to be quite open and democratic.

References

1 Chad E. Brown, Cezary Kaliszyk, Martin Suda, and Josef Urban. Hammering higher order set theory. In Valeria de Paiva and Peter Koepke, editors, *Intelligent Computer Mathematics - 18th International Conference, CICM 2025, Brasilia, Brazil, October 6-10, 2025, Proceedings*, volume 16136 of *Lecture Notes in Computer Science*, pages 3–20. Springer, 2025. doi:10.1007/978-3-032-07021-0_1.

2 David Cariolaro. *On the Ramsey number R (3, 6)*. Department of Mathematical Sciences, Aalborg University, 1999.

3 Thibault Gauthier, Miroslav Olsák, and Josef Urban. Alien coding. *Int. J. Approx. Reason.*, 162:109009, 2023. doi:10.1016/J.IJAR.2023.109009.

4 Thibault Gauthier and Josef Urban. Learning conjecturing from scratch. In Clark W. Barrett and Uwe Waldmann, editors, *Automated Deduction - CADE 30 - 30th International Conference on Automated Deduction, Stuttgart, Germany, July 28-31, 2025, Proceedings*, volume 15943 of *Lecture Notes in Computer Science*, pages 423–445. Springer, 2025. doi:10.1007/978-3-031-99984-0_23.

5 Jan Jakubuv, Karel Chvalovský, Zarathustra Amadeus Goertzel, Cezary Kaliszyk, Mirek Olsák, Bartosz Piotrowski, Stephan Schulz, Martin Suda, and Josef Urban. Mizar 60 for mizar 50. In Adam Naumowicz and René Thiemann, editors, *14th International Conference on Interactive Theorem Proving, ITP 2023, Białystok, Poland, July 31 - August 4, 2023*, volume 268 of *LIPIcs*, pages 19:1–19:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ITP.2023.19.

6 Albert Qiaochu Jiang, Sean Welleck, Jin Peng Zhou, Timothée Lacroix, Jiacheng Liu, Wenda Li, Mateja Jamnik, Guillaume Lample, and Yuhuai Wu. Draft, sketch, and prove: Guiding formal theorem provers with informal proofs. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL: https://openreview.net/forum?id=SMa9EAovKMC.

7 Cezary Kaliszyk, Josef Urban, and Jirí Vyskocil. Learning to parse on aligned corpora (rough diamond). In Christian Urban and Xingyuan Zhang, editors, *Interactive Theorem Proving - 6th International Conference, ITP 2015, Nanjing, China, August 24-27, 2015, Proceedings*, volume 9236 of *Lecture Notes in Computer Science*, pages 227–233. Springer, 2015. doi:10.1007/978-3-319-22102-1_15.

8 Cezary Kaliszyk, Josef Urban, Jirí Vyskocil, and Herman Geuvers. Developing corpus-based translation methods between informal and formal mathematics: Project description. In Stephen M. Watt, James H. Davenport, Alan P. Sexton, Petr Sojka, and Josef Urban, editors, *Intelligent Computer Mathematics - International Conference, CICM 2014, Coimbra, Portugal, July 7-11, 2014. Proceedings*, volume 8543 of *Lecture Notes in Computer Science*, pages 435–439. Springer, 2014. doi:10.1007/978-3-319-08434-3_34.

9 Jelle Piepenbrock, Josef Urban, Konstantin Korovin, Miroslav Olsák, Tom Heskes, and Mikolás Janota. Invariant neural architecture for learning term synthesis in instantiation proving. *J. Symb. Comput.*, 128:102375, 2025. doi:10.1016/J.JSC.2024.102375.

10 Christian Szegedy. A promising path towards autoformalization and general artificial intelligence. In Christoph Benzmüller and Bruce R. Miller, editors, *Intelligent Computer Mathematics - 13th International Conference, CICM 2020, Bertinoro, Italy, July 26-31, 2020, Proceedings*, Lecture Notes in Computer Science, pages 3–20. Springer, 2020. doi:10.1007/978-3-030-53518-6_1.

11 Josef Urban. Malarea: a metasystem for automated reasoning in large theories. In Geoff Sutcliffe, Josef Urban, and Stephan Schulz, editors, *Proceedings of the CADE-21 Workshop on Empirically Successful Automated Reasoning in Large Theories, Bremen, Germany, 17th July 2007*, volume 257 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007. URL: https://ceur-ws.org/Vol-257/05_Urban.pdf.

12 Josef Urban. Automated reasoning for mizar: Artificial intelligence through knowledge exchange. In Piotr Rudnicki, Geoff Sutcliffe, Boris Konev, Renate A. Schmidt, and Stephan Schulz, editors, *Proceedings of the LPAR 2008 Workshops, Knowledge Exchange: Automated Provers and Proof Assistants, and the 7th International Workshop on the Implementation of Logics, Doha, Qatar, November 22, 2008*, volume 418 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008. URL: https://ceur-ws.org/Vol-418/paper1.pdf.

13 Josef Urban. 130k lines of formal topology in two weeks: Simple and cheap autoformalization for everyone? *CoRR*, abs/2601.03298, 2026. doi:10.48550/arXiv.2601.03298.

14 Josef Urban and Jan Jakubuv. First neural conjecturing datasets and experiments. In Christoph Benzmüller and Bruce R. Miller, editors, *Intelligent Computer Mathematics - 13th International Conference, CICM 2020, Bertinoro, Italy, July 26-31, 2020, Proceedings*, volume 12236 of *Lecture Notes in Computer Science*, pages 315–323. Springer, 2020. doi:10.1007/978-3-030-53518-6_24.

15 Qingxiang Wang, Cezary Kaliszyk, and Josef Urban. First experiments with neural translation of informal to formal mathematics. In Florian Rabe, William M. Farmer, Grant O. Passmore, and Abdou Youssef, editors, *Intelligent Computer Mathematics - 11th International Conference, CICM 2018, Hagenberg, Austria, August 13-17, 2018, Proceedings*, volume 11006 of *Lecture Notes in Computer Science*, pages 255–270. Springer, 2018. doi:10.1007/978-3-319-96812-4_22.

16 Yuhuai Wu, Albert Qiaochu Jiang, Wenda Li, Markus N. Rabe, Charles Staats, Mateja Jamnik, and Christian Szegedy. Autoformalization with large language models. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL: http://papers.nips.cc/paper_files/paper/2022/hash/d0c6bc641a56bebee9d985b937307367-Abstract-Conference.html.