Autoformalizing Intersection Type Systems

Gabriele Vanoni

Université Paris Cité

Abstract

In this talk I will report on my efforts of using LLMs to (auto)formalize intersection type systems. Starting from a solid `Agda` codebase, I have been able to formalize soundness and completeness of several variants of non-idempotent intersection types, including the characterization of space consumption of a sophisticated abstract machine. I did this with almost no previous experience with proof assistants.

Large language models (LLM) are something we cannot just dismiss saying "*it is just a trend*". They are here to stay, and they will likely change the way in which we are doing mathematics [5]. This will happen in several aspects, for example:

- in how we *develop* mathematical ideas: interacting with an LLM has been compared to interacting with a graduate student. New ideas can be tested and refined at a speed never seen before;
- in how we *write* mathematics: LLMs are great at writing, whatever the language. This way, they can write hundreds or thousands of code in minutes. Since LLM output cannot be trusted, it becomes fundamental to *certify* what they produce. This is where proof assistants come into play. We do not just want LLMs to generate mathematics, but to generate certified one.

In this talk, I will report on my experience about the latter point. I want to immediately highlight this was the first time I used a proof assistant seriously, and my knowledge about them does not go beyond understanding definitions and statements.

The Original Codebase. I started from the non-idempotent intersection type `Agda` formalization of Bob Atkey [2]. This is about the standard de Carvalho's result about the quantitative correspondence with Krivine's abstract machine (KAM) [4]. He uses an intrinsically well-scoped de Bruijn representation that makes the implementation very elegant.

The First Easy Extensions. I wanted to play with his code and then I have first implemented an optimization of the KAM, called unchaining. Of course, also the type system had to be modified accordingly. I did the definitions by hand, and then used `Gemini 3.1 Pro` to handle the proofs. This was done in an *interactive* way, through the chat. Typically, the `Agda` code did not type check at the first time, and a few iterations are needed to fix it. I repeated the same process introducing free variables, using a *locally nameless, globally named* approach. I want to stress that not being familiar with the formalization of λ-calculi and type systems, I discussed the different possible approaches with the LLM itself. All of this did not take me more than two days of work.

Formalizing an ICFP Paper. The next step I took was to formalize completely my ICFP paper "Multi types and Reasonable Space" [1]. Here, there are more difficulties, as both the abstract machine is more complex and the quantitative reasoning is about space instead of time, requiring more care. Probably because of my inexperience in dependent type theory, it took me approximately one week to formalize the full paper, using the same interactive approach.

Future Directions. I think this is just the beginning, as I just scratched the surface of what is possible to do using LLMs. For example I did not use any *agent*, sticking to an "artisanal" chat-based approach. I guess that using autoformalization methods such as those developed in the past few months [3], it would be possible to convert full papers into certified code just from the PDF file without any human intervention.

References

[1] Beniamino Accattoli, Ugo Dal Lago, and Gabriele Vanoni. Multi types and reasonable space. *Proc. ACM Program. Lang.*, 6(ICFP):799–825, 2022. doi:10.1145/3547650.

[2] Bob Atkey. *Quantitative Typing with Non-idempotent Intersection Types*, 2020. URL: https://bentnib.org/posts/2020-08-13-non-idempotent-intersection-types.html.

[3] Dustin Bryant, Jonathan Julián Huerta y Munive, Cezary Kaliszyk, and Josef Urban. Munkres' general topology autoformalized in isabelle/hol. *CoRR*, abs/2604.07455, 2026. doi:10.48550/ARXIV.2604.07455.

[4] Daniel de Carvalho. Execution time of λ-terms via denotational semantics and intersection types. *Math. Str. in Comput. Sci.*, 28(7):1169–1203, 2018. doi:10.1017/S0960129516000396.

[5] Konstantin Kakaes. The ai revolution in math has arrived. *Quanta Magazine*, 2026. URL: https://www.quantamagazine.org/the-ai-revolution-in-math-has-arrived-20260413/.