Hi, People at the University of Birmingham!

My name is Siyuan Yan, and I preferred the name Matt to avoid unnecessary tongue-twisting. I have broad interests in math and computer science. Coding and open-source has been my hobby since high school, in the meantime, doing academic research has always been my life goal.

Undergraduate Studies

I began my undergraduate studies in a Computer Science Specialization Program with Math Minor. From the outset, I took Honours equivalent of all the math requirements, where courses are oriented towards proofs and methodology rather than applications. As a payoff, I applied to and was admitted to the Honours program in my second year, with math major and computer science minor.

Starting from 2019, while maintaining a full enrolment, I worked as a research assistant in the linguistic department of the University of Alberta for 1.5 years during my undergraduate studies, under the supervision of Dr. Antti Arppe, a computational linguist. From the work experience, I gained hands-on experience in assorted aspects of software development.

Academically, I had a hiccup in my fourth year, when I tried to start an online service with applied machine learning and diverted too much attention. As a consequence, I had to drop from the Honours program to a general science program.

I eventually fought back on track. Considering my over-all GPA and advanced course selections, rather than the current program listed on the transcript, the school lets me graduate with a B.Sc. in Math Specialization, which has stricter GAP requirements and course requirements. I have requested proof statements from the school of the degree change, yet the school says this won't show up on the transcript until June.

Research Interests

Having a career and in academic research is always my biggest dream. Yet not being able to decide on a specific area, I didn't hurry to apply to a school, but instead spent lots of time surveying different areas of mathematics. Not too late, I found myself deeply intrigued by the foundations of math. I started reading books in set theory, proof theory, and logic. This is when I inevitably came across homotopy type theory (HoTT), an alternative yet impactful foundation system of mathematics and an innovation in type theory. I find many of its concepts familiar thanks to my familiarity with the Rust programming language, a functional programming language with applied ML type theory.

I find type theory immensely beautiful in two ways, both philosophically and pragmatically.

First there is the intuitionist philosophy behind. It always touches my soul when I think of the debates that happened in the early 1900s, where foundations of mathematics were being built and choices in the logic of mathematics were being made, when set theory was chosen to prevail. Yet things have changed a lot since then. As Gödel shows in the incompleteness theorems, theories are not perfect in their proving power. Alternative theories can provide the strength of different proving powers. On the other hand, the nature of intuitionist logic used in type theory always feels appealing to me. The omission of the law of excluded middle in type theory, for example, seems to fit well with the interpretations brought by contemporary physics, which Hilbert and mathematicians back then didn't have access to. Just like how software gets better with refactoring and iterations, with the benefit of hindsight, reconstructions of the foundations of mathematics from a modern perspective will bound to bring benefits.

Secondly, the constructive nature of type theory serves as a bridge between computers and proofs. Type theory are the underlying theory in proof assistants, programs that understand proofs. With proof assistants, math theorems can be completely computerized and checked meticulously by the machine. It's just exciting to imagine every math theorem so neatly formalized and their correctness checked. While I am learning Lean, a proof assistant with increasing popularity, I'm amazed at the community of mathematicians implementing advanced math concepts in code. With recently advances in human-machine interaction brought by type theory, proof assistants is really becoming a feasible research tool as well as an education tool in mathematics. And I believe it's bound to revolutionize how math is researched in the future. Just as what the Dr. Andrej Bauer said in 2021:

> But will formalized mathematics go mainstream? I think this is the completely wrong question to answer, ... of course it will ... the question is, WHEN ...[1]

Another person who inspired my a lot is Dr. Kevin Buzzard at Imperial College London, who is applying proof assistants to both undergraduate math education and in advanced research topics like algebraic geometry. If

[1]video: `https://youtu.be/zp6WleEjHUg?t=2279` timestamp relevant

there's a chance, I wish to become an evangelist like him and integrate proof assistants as an inherent part of my future research. I wouldn't doubt in the future, coding will become as important as LaTeX for mathematicians. And I want to contribute to this future.

"I am excited by the vision of a future where computers can do research by themselves" were the words I wrote back then on my application to undergraduate schools. As crazy as it sounded like, it's partially becoming the reality, not mentioning researchers exploring further on this idea by applying AI to proof assistants.

With all the excitements I have, I crave an opportunity to study more about the relavant fields in the University of Birmingham before I can contribute to the research. During my search for ideal supervisors, I found Dr. Eric Finster [2] in the University of Birmingham in the authors of *the HoTT book*[3], I believe he can be an ideal mentor for my further studies, or at least, a person I can turn to in my path of graduate studies.

I sincerely hope my application be thoroughly considered. Please feel free to ask any further questions via my email syan4@ualberta.ca.

[2] `https://ericfinster.github.io/`

[3] `https://homotopytypetheory.org/book/`