Sid's Plan for 2026-2027

September 2025

Invariants

- Location: Cambridge (London is dispreferred, but can-do with a hybrid schedule).
- Work: Well-Paid.
- Visa: Global Talent (preferably), or Skilled Worker Sponsorship.
- Long-Term Location: As Luisa prefers, UK / EU.

I'll be staying where Luisa lives. Cam/London for the next couple of years; EU/UK long-term.

Uncertainties

Global Talent Visa Timelines: Do I get it in time for 2026 summer?

Post-Phd Option: Post-doc

This is the most straightforward option. Since I'm applying for a grant with Tobias with me as the co-grant-writer, this path also offers a global talent visa. I'd like to switch to a postdoc position sooner rather than later, for both salary, as well as starting the clock on my stay in the UK as a skilled worked. I must talk to tobias about this.

My current feeling is that I should start a postdoc early next year (say, April), and in parallel, wait for the global talent visa to come through either route (via the grant application, or via the endorsement route).

Post-Phd Option: Work At Interesting Places

Jobs in descending order of preference. I'm looking for roles that are well-paid, and can sponsor a Skilled Worker Visa if needed. Apple *(Cam)* / AWS *(Cam)* / ARM *(Cam)* / AMD *(Cam)* / Intel *(Cam? London Flexible?)* / DeepMind *(London Flexible)* / Antithesis *(London Flexible)* / OpenAI *(?)* / Jane Street *(London Flexible)* / FAANG *(London Flexible)*

There seem to be sufficiently many people in the compilers space who would hire me, but I want to reality-check this assumption.

Post-Phd Option: Starting a Verification Company

The way I frame the problem for starting a company that deploys verification is to ask:

> Who is deathly afraid of their code being incorrect?

In theory, having correct code is a force multiplier when working on complicated software such as compilers. In practice, fuzz testing and weaker forms of static analysis is extremely effective for weeding out bugs.

Most obviously, hardware companies care deeply about getting their hardware right, as tapeouts cost millions. However, this space is quite saturated.

Cryptography, fully homomorphic encryption, and blockchains are domains where security is critical. While important, it's not clear to me how much of their need for verification is for **appearances**, and how much is legitimate. It may well be that we provide a luxury service of **appearing trustworthy**, ala Jepsen

Furthermore, databases are another domain where people are paranoid about getting their code right, and formal verification is routinely deployed in this space to ensure correctness.

As a case study, Google seem to guard then nonlinear ILP solver in OR-TOOLS, so this seems to be an interesting case where having good tech actually makes a difference. Similarly, hardware companies carefully guard their secret sauce of model checkers, place and route algorithms, and so on.

A lot of the market is about appearances, so our technological choices must be guided by such considerations. using Python makes one appear unserious due to its weak typing and semantics. However, having the core solver be written in Rust/Lean, and being able to **interface** with code written in Python would be fantastic, due to the large user base available.

FV seems useful for e.g. the TOSA folk, who seem to care about getting their semantics right.

I could imagine that formal verification is useful for **debugging** crazy hardware, such as Cerebras.