GSoC 2024 SageMath Proposal

Atticus Kuhn

March 2024

Contents

1 Personal

- Name: Atticus Kuhn
- email: atticusmkuhn@gmail.com
- Location: California (UTC-8:00)
- University: Trinity College, University of Cambridge

2 Background

What are your technical skills, education, experience, etc. Especially make sure to explain with what level of mathematics you are comfortable with and on what level you would like to program. Who are you? What makes you the best person to work on this particular project? Your personal motivation? What platform and operating-system are you using on your computer? (SageMath development is done best on Linux, !MacOS, or Windows through WSL)

Are you or have you been engaged in other open-source projects? Do you code on your own pet projects? Are you a SageMath user, how long have you known about/used SageMath?

3 Project

- Title: Better Lean Integration with Sage
- Length: long (350 hours)

3.1 Project Synopsis

The aim of this project is better integration between the Lean Theorem Prover and Sage Math. Look at the following tactic from the Lean Mathlib 3: `https://leanprover-community.github.io/mathlib_docs/tactics.html#polyrith`. This tactic calls out to SageMath, and uses the result to construct a proof inside Lean. But the way it does this is janky. See the actual code here: `https://github.com/leanprover-community/mathlib/blob/master/scripts/polyrith_sage.py`. What's more, the newest version of Lean is now Lean4, so the polyrith tactic no longer works in Lean4. My goal in this project is to add features to SageMath to make it easier for Lean (and presumable other proof assistants such as Coq or Isabelle) to call out to SageMath.

SageMath actually does have a lot of theorem-proving powers within it already, such as the

```
solve()
```

function, but it is difficult for proof assistants (Lean, Coq, Isabelle) to call out to these abilities, and so we end up with jank like the polyrith tactic from MathLib3.

As an added bonus, this proposal would solve `https://ask.sagemath.org/question/34991/can-sage-show-step-by-step-solutions/`

3.2 Details

The way I will accomplish this is by having theorem-proving functions have the option to return not only a result, but also a proof of that result's correctness. This could be done with an extra argument, for example. The proof could be in a language-agnostic format, or it could just return a Lean4 expression for convenience.

Here is pseudo-code to describe the behavior after the project is done:

```
sage: expr = x/(x^2 + x)
sage: expr.simplify_full()
1/(x + 1)
sage: expr.simplify_full(show_lean = True)
"""
by
    rw [mul_add x]
    simp
"""
```

3.3 Schedule

TODO

3.4 Risk Management

TODO