p-SymPy

Speeding up SymPy's new assumptions

Project Manager: Tilo Reneau-Cardoso, Senior, Math & Computer Science Major
Contact: tiloreneau@gmail.com

Project Overview

Have you ever wanted to contribute to open source software? Join my project and gain the opportunity to do just that! My goal is to enhance the performance of the assumptions module in SymPy, a widely-used Python-based computer algebra system (think WolframAlpha, but in Python). As an expert in SymPy's new assumption system, I've guided newcomers in contributing to it, drawing from my experience as a contributor during Google Summer of Code 2023.

SymPy's assumption system allows users to query properties (or "assumptions") about symbols and expressions, such as whether a symbol represents a positive, real, or integer value. Its primary purpose is to enable more simplifications and manipulations of mathematical expressions. See here for more details about SymPy's assumptions.

SymPy actually has two different assumption systems—the "old" and "new" assumptions. For more than a decade, there has been an effort to transition the new assumption system which allows for more types of queries (e.g. reasoning about inequalities). However, the new assumptions are much slower, so it's the goal of my project to do what we can to improve their speed. See here for a blog post I'm writing which contains additional details.

Tech Stack

- Python

- GitHub/Git

Ethical Considerations

SymPy is liscenced under the New BSD License. This is a permissive liscence which means people can use SymPy freely for pretty much whatever purpose. SymPy is sometimes used by scientists and people doing physics. It's likely that SymPy has at some point been used to help make weapons in some capacity. So by contributing to SymPy you're improving software that can be used by anyone, including potentially nefarious people.

Team Structure

- Target Team Size: ~5 students (flexible)
    - There's a chance I decide to accept a very large number of people.
- Subteam Structure: flexible
- Organizational & Collaboration Frameworks: GitHub

Timeline

Week 1-2

- Everyone will open a pull request attempting to fix some bug or issue in SymPy. This should help everyone become familiar with the process of contributing to SymPy.
- See this link for a list of bugs with the assumptions. I believe one of them is related to the old assumptions, so ignore that one. A lot of pull requests have opened up fixing some of these, so I may have to look for more bugs.
- The pull request doesn't have to be a bug fix. For example, the following PR could be revived: Improved docstring for EncodedCNF.

Week 2 and beyond

- People will be assigned to working on subprojects. As there's only a limited amount of time in the semester, I can't guarantee that any of these subprojects will get completed. However, if they are completed, I recently got merge access to SymPy, so I'll be able to merge your changes if I think they are beneficial to SymPy and meet all of SymPy's standards for contributions.
- I expect some degree of collaboration between everyone. Pull requests are required to be reviewed before they can be merged, so I expect project members to be reviewing the pull requests of other project members.

Project Member Requirements

- Self-initiative
    - You should be proactive in problem-solving, learning, and collaborating with others.
- Time commitment
    - At least 4+ hours per week.
    - Participation in 1-hour weekly meeting is required.
- CS62/CS70 or equivalent
    - If you haven't taken data structures yet, evidence of coding proficiency such as personal projects can substitute it.
- Previous experience with GitHub/Git
    - If lots of people are struggling with Git, and they're not able to help each other, it's not feasible for me to help everyone. So I expect everyone to have some knowledge of Git and be able to help themselves and others if they are experiencing problems. Even having used Git quite a lot, I still struggle with Git from time to time, so it's normal to have difficulties with it, and I don't expect anyone to be an expert or even good at it.
- Python proficiency
- Basic familiarity with logic
    - (e.g. truth tables, boolean variables)
- An interest in math

Additional Skills I'm Looking For

If you've taken any classes related to logic it would be a big plus. For example:

- PHIL060 PO - Logic
- Applied Logic and Automated Reasoning (CS 181U)

Subprojects

The following is an incomplete list of possible subprojects. Depending on the number of people accepted and their interest, some of them may not be assigned anyone. You may be involved in multiple subprojects as I expect project members to review each others pull requests before I review them.

It's important that everyone gets credit for work they did and no one gets credit for work they didn't do. If you're working together on the same subproject, you should either pair program and coauthor commits or break up the work in a way such that each of you makes separate commits. It's okay to do a mix of both, but don't coauthor commits if you weren't pair programing and make sure you do coauthor commits if you were pair programing.

Implement the changes I describe in my blogpost. (3-4 people)

This is by far the largest and most ambitious of the subprojects. Parts 1 and 2 can be worked on concurrently by different people. Part 3 can be most easily completed by the part 2 group, but the part 1 group could also work on it.

1. Implement bad but easy to implement prototype (1-2 people)
    - This might be the most difficult part. Actually implementing the prototype won't be that bad. However, integrating the prototype into the SAT solver may be challenging and other difficulties may arise. The idea is that this group will connect things properly so the part 2 group doesn't have to worry about that.
2. Implement good enough version (1-2 people)
3. Implement efficient minimal conflict clause version
4. In the unlikely event that there's time, we can work on replacing the sathandlers with a similar SMT system.

Make SAT solver incremental (1-2 people)

- See this issue.

Implement Tseytin transformation (1-2 people)

- See here for a wikipedia article discussing the Tseytin transformation.
- See this issue for context about why the Tseytin transformation would be useful to SymPy's assumption system.

Implement handling for non-rational and infinite values in Linear Real Arthmetic Theory Solver (2-3 people)

- I haven't finished writing an issue for this. This is the file that would need to be improved.
- Could be split up into two different sub-subprojects.

Consolidate `satask` and `lra_satask` into one function (1-2 people)

- This can't really happen until lots of other changes are implemented such as speeding up the assumptions significantly. But work can be started on this and whatever roadblocks are present can be documented.

Document the New Assumptions (1-2 people)

- Write a how to guide like this one but for the new assumptions. Or maybe expand that one.