Title

SymEngine, a fast symbolic manipulation library

Authors

Ondřej Čertík, Isuru Fernando

The Brief Description

The goal of [SymEngine](https://github.com/symengine/symengine) is to be the fastest C++ symbolic manipulation library (opensource or commercial), compatible with SymPy, that can be used from many languages (Python, Ruby, Julia, ...). We will present the current status of development, how things are implemented internally, why we chose C++, benchmarks, and examples of usage from Python (SymPy and Sage), Ruby and Julia.

The Detailed Abstract

Motivation

SymPy is a widely used symbolic manipulation library in Python. While SymPy turns out to be very useful for many applications, one of the long term problems with SymPy is that the speed might be insufficient when handling of very large expressions is required. Another problem with SymPy is that due to being written in Python, it can be cumbersome to use from other languages like Julia, Ruby, JavaScript or C++, because it requires, say, a Python to Julia bridge, which might not always be robust and which inflicts additional overhead.

Methods

For these reasons, we implemented SymEngine, an open source C++ symbolic manipulation library, with the goal of being the fastest library for symbolic manipulation (opensource or commercial), and allowing natural wrappers to other languages like Python (the most mature wrapper), Ruby, Julia and others.

Results

We will show benchmarks against other computer algebra systems (opensource and commercial) as well as examples of usage from Python (SymPy and Sage), Ruby and Julia. We will present a roadmap how to port SymPy on top of SymEngine.

We will talk about why we chose C++ and what rules to follow so that the code cannot have an undefined behavior in Debug mode (thus providing similar ease of development as one is used to from Python).

The SymEngine C++ library already has 35 contributors, 101 forks and 126 stars on GitHub, as of March 21, 2016.

Conclusion

SymEngine should fix the slowness of SymPy, while providing a familiar interface, and at the same allowing many languages to naturaly use it, thus creating a common platform/tool that many projects (like SymPy, Sage, ...) can use as their main symbolic engine and all contribute back to it.

Resources

* SymEngine C++ library: https://github.com/symengine/symengine
* SymEngine Python wrappers (with SymPy and Sage integration): https://github.com/symengine/symengine.py
* SymEngine Ruby wrappers: https://github.com/symengine/symengine.rb
* SymEngine Julia wrappers: https://github.com/symengine/symengine.jl
* SymPy: http://sympy.org/