Data-Driven Debloating of an SMT Solver

Abstract

SMT (Satisfiability Modulo Theories) solvers are essential tools in software verification and industrial applications. However, their increasing complexity raises concerns about code bloat, maintenance challenges, and potential soundness issues. This proposal presents a data-driven approach to debloat SMT solvers by analyzing benchmark usage to identify and eliminate unnecessary code. The goal is to significantly reduce the codebase while retaining a high level of functionality, thereby enhancing maintainability and reliability across these critical tools.

Introduction

SMT solvers play a critical role in verifying software correctness, ensuring security, and facilitating various industrial applications. cvc5, one of the leading SMT solvers, has evolved into a complex system with approximately 325,000 lines of C++ code. While this growth reflects its rich feature set, it also introduces challenges related to code bloat, increased bug surface, and maintenance and feature extension difficulties. In contrast, SAT solvers often maintain a small, sound core, emphasizing simplicity and reliability.

This project proposes a data-driven approach to debloating CVC5 by leveraging standard benchmarks to identify essential code paths. By reducing unnecessary code without compromising functionality, we aim to enhance the solver's performance, maintainability, and soundness.

Problem Statement

The complexity of cvc5's extensive codebase may lead to several issues:

- **Code Bloat**: A significant portion of the code remains unused or is rarely invoked, contributing to unnecessary complexity.
- **Increased Bug Potential**: More code can lead to more bugs, including soundness bugs critical in SMT solvers.
- **Maintenance Challenges**: A large codebase makes adding new features and fixing existing issues harder.
- **Soundness Concerns**: Unused or poorly tested code may harbor undiscovered bugs that could compromise the solver's correctness.

These challenges necessitate a systematic approach to simplify cvc5 without sacrificing its core capabilities.

Objectives

- **Debloat cvc5**: Reduce the codebase by identifying and removing unused or rarely used code, aiming for a significant reduction while maintaining essential functionality.
- **Maintain High Functionality**: Ensure the debloated solver can still solve at least p% of all test cases from standard benchmarks, where p is configurable (e.g. 99%).
- **Enhance Reliability**: Improve the solver's soundness by minimizing potential sources of bugs through code reduction.
- **Improve Maintainability**: Simplify the codebase to facilitate easier maintenance, bug fixing, and feature addition.

Methodology

1. **Benchmark Analysis**:
    - Utilize standard SMT-LIB benchmarks to execute cvc5 and collect data on code usage.
    - Instrument the solver to record which code paths are exercised during benchmark runs.
2. **Data Collection and Analysis**:
    - Analyze collected data to map code coverage, identifying unused or rarely used code segments.
    - Determine thresholds for code usage frequency to decide what code can be considered for removal.
3. **Code Reduction**:
    - Carefully remove or refactor identified code segments, preserving critical functionalities.
    - Modularize the codebase where possible to isolate essential components.
4. **Performance Assessment**:
    - Measure the impact of debloating on solver performance, including execution time and resource consumption.
    - Measure the impact of debloating on development (compile time savings, etc.)
    - Ensure the debloated solver meets or exceeds the original performance in handling the targeted test cases.

Deliverables

Mandatory

- **Problem & Approach formulation (on paper)**
- **Extensive coverage-based testing framework**
- **Realisation/Implementation of the Optimization solving approach (in code)**
- **Realisation of one new solver where at least some life code is debloated (in code)**
- **Simple evaluation of solver completeness on SMT-LIB benchmarks**
- **Documentation of the debloating process and methodologies used**

Optional

- **Automated Debloating Tool, which removes life-code given a certain target completeness**
- **Posthock Completeness Optimization, with regard to higher target completeness**
- **Other Optimizations, based on analysis results**

Conclusion

This project aims to address the challenges posed by CVC5's bloated codebase by implementing a data-driven debloating strategy. We expect to enhance the solver's reliability, maintainability, and performance by retaining essential functionalities and removing unnecessary code. Simplifying CVC5 reduces the potential for bugs and makes it more accessible for future development and feature integration. The anticipated outcome is a leaner, more robust SMT solver that continues to meet the demands of software verification and industrial applications.