Improving Safety in Clad-Generated Mathematical Functions

Ivan Velev
High School of Mathematics “Akademik Kiril Popov” Plovdiv
Mentor: Vassil Vassilev

About Me

School Email: ivanvelev_20b@schoolmath.eu

Personal Email: Vanko.Z.Velev@gmail.com

GitHub: CopyPasteExpert

My name is Ivan Velev and I am a high school student with a strong interest in programming and compiler technologies. Through my work with Compiler Research, I aim to gain practical experience in real-world software development and improve my understanding of compiler infrastructure and numerical computing. This project focuses on improving the reliability and safety of automatically generated mathematical functions.

About Clad

Clad is a compiler-based automatic differentiation tool built on top of LLVM and Clang. It allows developers to automatically generate derivative functions from existing C++ code. This is extremely useful in scientific computing, optimization, machine learning, and other fields that require efficient derivative calculations.

The Problem

When Clad generates derivative functions, some mathematical operations may perform out-of-bounds access or receive inputs that are outside the valid domain of a function. This can lead to undefined behavior, which may cause incorrect results, hidden bugs, or even program crashes.

Undefined behavior makes debugging difficult because the program may fail unpredictably or produce invalid outputs without clear warnings.

The Solution

The goal of this project is to introduce clearly defined behavior for Clad-generated functions when invalid inputs occur.

Instead of producing undefined behavior, the functions will return a well-defined value such as NaN (Not a Number). This makes it much easier for developers to detect errors and trace issues during debugging.

However, adding safety checks for every function call may introduce a performance overhead. To address this, the project introduces an optional safety mode using:

```
#define CLAD_SAFE_MATH
```

and conditional compilation with `#ifdef` statements.

This allows developers to enable safety checks during debugging while keeping maximum performance when they are confident that their inputs are valid.

This will form the first phase of the project. Future work may include defining function-specific fallback behavior or returning approximate values in cases where strict accuracy is not required.

Timeline:

Weeks	Tasks	Deliverables
Week 1	Prepare a presentation explaining the problem of undefined behavior in Clad-generated functions and the idea behind the CLAD_SAFE_MATH safety checks.	Finished presentation describing the project and the proposed approach.
Week 2	Study the Clad codebase and start implementing checks for basic arithmetic operations that may cause problems.	Initial fixes for arithmetic operations and notes about where boundary checks are needed.
Week 3	Add boundary checks for **root and power functions** such as sqrt, cbrt, pow, and hypot, where invalid inputs can easily cause undefined behavior.	Safe handling for root and power functions.
Week 4	Work on **logarithmic functions** including log, log10, log2, and log1p, making sure invalid inputs return a defined value like NaN.	Boundary-safe behavior for logarithmic functions.
Week 5	Add fixes for **inverse trigonometric functions**: asin, acos, atan, and atan2, making sure the domain restrictions are handled correctly.	Implemented checks for inverse trigonometric functions.
Week 6	Implement checks for **regular trigonometric functions** such as sin,	Safer behavior for trigonometric functions

	cos, and tan, especially handling cases where the derivative may become undefined.	in generated derivatives.
Week 7	Work on **hyperbolic functions** like sinh, cosh, tanh, asinh, acosh, and atanh, adding checks for inputs outside their valid ranges.	Implemented fixes for hyperbolic functions.
Week 8	Add checks for **exponential functions** including exp, exp2, and expm1, focusing on handling overflow or extreme inputs.	Safer exponential function handling.
Week 9	Add **debugging support** so that when CLAD_SAFE_MATH is enabled the program can warn when a boundary violation happens.	Debug messages or logging for invalid inputs.
Week 10	**Buffer and polishing week** – fix bugs, clean up the implementation, and write documentation explaining how to enable or disable safe math checks.	Final cleaned-up implementation and documentation.