Advanced Languages

Presented by Hormoz Zarnani and Eric Malloy

The Design and Implementation of a Certifying Compiler

George C. Necula and Peter Lee

Compiler Correctness

- Compiler produces correct output
- That is, generated assembly code is functionally equivalent to the high-level code

Traditional Approaches

- Formal compiler verification
  - Hand proofs
  - Mechanical proofs
- Testing
  - Automatically generating test patterns and checking validity of corresponding outputs

Why Traditional Approaches Do Not Work

- Formal compiler verification
  - Verify algorithm rather than implementation
  - Not automatic
    - Requires human intervention and expertise
  - Must redo proofs if compiler changed
- Testing
  - Generated patterns are usually inadequate

Why Traditional Approaches Do Not Work (cont.)

- Cannot handle optimizing compilers
  - Place many restrictions on optimizations

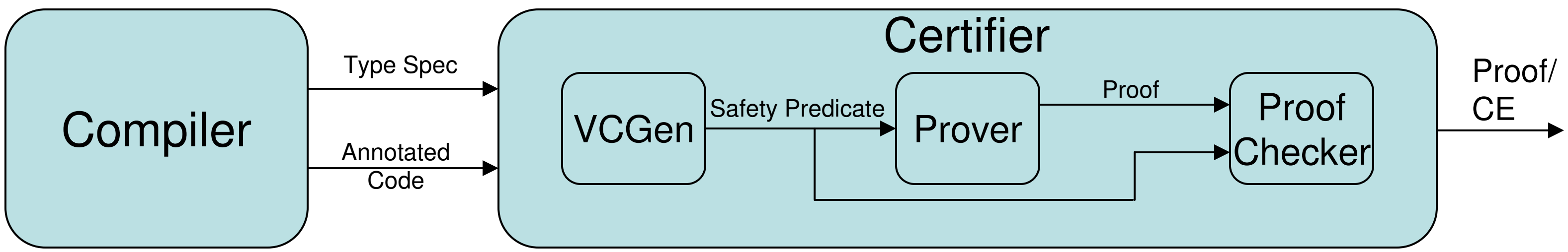
Better Approach

- Proving full correctness is too expensive
- Instead, employ a method that is less expensive and yet gives satisfactory confidence
  - Check individual compilations

Touchstone: A Certifying Compiler

- Compiles a strongly typed subset of C into optimized DEC Alpha assembly language

Structure of Touchstone



Touchstone – Advantages

- Easy to employ
- Also can transform conventional compilers to certifying ones
- Can be applied to any type safe language
- Places no restriction on optimizations allowed
- Only VCGen and Proof Checker have to be correct

Touchstone – Disadvantages

- Applies to only type-safe languages
- But C is not type-safe

Touchstone – Conclusion

- Does not fully address problem
- But is novel work and excellent starting point

Checking System Rules Using System-Specific Programmer-Written Compiler Extensions

Dawson Engler, Benjamin Chelf, Andy Chou and Seth Hallem
Computer Systems Laboratory
Stanford University
Standford, CA 94305

System Rules Violations

- What are the some of the ways we might find violations of system rules in a program?
  - Model Checkers
  - Theorem Provers
  - Testing
  - Code Inspections
  - Compilers

Metal-Level Compilation

- System specific “meta” semantics
  - Essentially these are rules for a systems API’s
  - Implemented as runtime extensions to the compiler
  - Capable of discovering errors in complex code as well as optimization opportunities

Extensions to the xg++ Compiler

- xg++ is and extensible compiler based on g++
- Extensions are written in a high level state machine language called “metal”

How Metal Extensions Work

- Metal is compiled using the mcc compiler and dynamically linked into xg++ at compile time
- Pattern comparisons are done based on xg++'s internal representation

Metal Example

```
{ #include "linux-includes.h" }
sm check_interrupts {
  // Variables
  // used in patterns
  decl { unsigned } flags;

  // Patterns
  // to specify enable/disable functions.
  pat enable = { sti(); }
             | { restore_flags(flags); } ;
  pat disable = { cli(); };

  // States
  // The first state is the initial state.
  is_enabled: disable ==> is_disabled
     | enable ==> { err("double enable"); }
     ;
  is_disabled: enable ==> is_enabled
     | disable ==> { err("double disable"); }
     // Special pattern that matches when the SM
     // hits the end of any path in this state.
     | $end_of_path$ ==>
        { err("exiting w/intr disabled!"); }
     ;
}
```

What Can Be Checked

- Assertion side-effects
- Checking assertions of constant scalar variables
- Temporal orderings of system calls
- Memory management
- Global checking of blocking routines and reference counts

Memory Management Error Counts

Violation	Bug (Linux)	False (Linux)	Bug (OpenBSD)	False (OpenBSD)
No Check	79	9	49	2
Error Leak	44	49	3	1
Use After Free	7	3	0	0
Underflow	2	0	0	0
Total	**132**	**61**	**52**	**3**

Blocking with an Interrupt Disabled

Condition	**Applied**	**Bug**	**False Positive**
Holding Lock	~5400	29	113
Double Lock	-	1	3
Double Unlock	-	1	20
Interrupt Restore	~5800	44	63
Bottom Halves of Interrupt	~180	4	12
Interrupt Flag Restore	~3200	4	49
Total	**-**	**83**	**260**

Advantages\Disadvantages

- What advantages and disadvantages does this approach have?