Have Your Verified Compiler
And Extend It Too

Zachary Tatlock
Sorin Lerner

UC San Diego
Compiler Correctness

Building robust compilers is difficult.
Complex interactions resist testing.

Compiler bugs are contagious.
Invalidate source level guarantees.

Few users extend their compiler.
Hand optimized, unreadable code.
Verified Compilers

- Implement compiler in proof assistant
- Prove compiler correct interactively
- CompCert [Leroy], Lambda Tamer [Chlipala]

Strong Guarantee

Difficult to Extend
DSL-based Compilers

- Domain Specific Language for optimizations
- DSL opts proven correct automatically
- Rhodium [POPL 05], PEC [PLDI 09]
Contribution

- CompCert
- XCert
- PEC

Addability
- Add L Execution Engine + Extensibility
- Correctness Proof

Harder to extend
- Stronger guarantee
- Weaker guarantee

Easier to extend
- Reduce TCB
Main Theorem Proved in Coq:
XCert Rules Locally Correct

[PLDI 09]
- Formal Correctness Proof in Coq
- Bulk of the development effort

CompCert
C
Asm

Extensible & Correct Compiler

Rewrite
Locally Correct
Extensible & Correct Compiler

1. Rewrite Rule

2. CompCert

3. Challenges and Evaluation

PEC [PLDI 09]
`while (C) {
  I += 2
  I++
  x = 0
  while (x < 10) {
    x++
  }
  return x
}`

**Rewrite Rule**
- Find & Replace
- Match Pattern
  - C → x < 10
  - I → x
- Apply Subst

---

**[PLDI 09]**
PEC Checker

1. Convert to CFG
2. Guess Sync Points
3. Check w/ SMT

SMT Checked

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>A</th>
<th>✓</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>→</td>
<td>A</td>
<td>✓</td>
</tr>
<tr>
<td>A</td>
<td>→</td>
<td>B</td>
<td>✓</td>
</tr>
</tbody>
</table>
XCert Module

1. Rule in Coq
2. SMT Checks

SMT Checked

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A → A</td>
<td>A → B</td>
<td></td>
</tr>
</tbody>
</table>
Extensible & Correct Compiler

1. Rewrite Rule

PEC

[PLDI 09]

2. XCert

3. Challenges and Evaluation
XCert Correctness Proof

Small Step

- Execute instruction
- Step state $S$ to $S'$
XCert Correctness Proof

Equivalent Executions

- Initial Equiv $\rightarrow$ Final Equiv
- Prove Simulation Diagram

CompCert Small Step Library:

$\exists R : R \rightarrow R \quad \xRightarrow{\text{Sim Diagram}} \quad \text{Progs, Equiv}$
Extensible & Correct Compiler

1. Rewrite Rule

PEC [PLDI 09]

2. XCert

Challenges and Evaluation
Challenges (see paper)

XCert Execution Engine
- CFG pattern matching
- CFG splicing

XCert Correctness Proof
- Managing case explosion
- Verified validation [Tristan and Leroy]
- Preserving non-terminating behaviors
Evaluation

Engine : 1,000 lines of Coq functional code
Proof  : 3,000 lines of Coq proof script

Trusted Computing Base (TCB)
- Compcert : Coq + Coq encoding of C sem
- XCert adds : SMT + SMT encoding of C sem
Evaluation

Extensibility: Support PEC Opts [PLDI 09]

- No manual proof effort or TCB increase
- Maintain CompCert end-to-end correctness

Sample of Optimizations Run:

<table>
<thead>
<tr>
<th>Loop Invariant Code Hoist</th>
<th>Loop Peeling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Pipelining</td>
<td>Conditional Speculation</td>
</tr>
<tr>
<td>Loop Unswitching</td>
<td>Partial Redundancy Elim</td>
</tr>
</tbody>
</table>
Extensible & Correct Compiler

1. Rewrite Rule
   [PLDI 09]

2. XCert

Thank You!