Verified Software Construction

Aaron Stump

Computational Logic Center
Computer Science Department
The University of Iowa

UI Students: Andrew Reynolds, Duckki Oe.

Funding from NSF.

Engineering Perfection

- Engineering is judged by its artifacts.
- Criteria: cost, reliability, aesthetics, durability, etc.
- Most basic criterion: correct function.
- Beautiful cheap airplanes must fly, ugly chainsaws ok if they work.

A Great Engineering Example

The St. Louis Arch

- Construction took 2.5 years, finished in 1965.
- Tallest national monument in U.S.
- Two legs constructed simultaneously, then joined.
- Margin of error for this was 1/64 of an inch.
- Truly an incredible example of very precise engineering.

From Very Precise to Flawless

- Physical objects can never be absolutely perfect.
- Virtual objects are different.
- Software can be tested, debugged, to low margins of error.
- But we can go beyond this.

Verified Software

- *Verification* applies formal reasoning to software.
- Prove that code is correct.
  - No low-level bugs: null pointer access, array bounds violation.
  - Richer specifications: sorting returns sorted list for any input.
- Many different approaches developed over 40+ year history.
- *Algorithmic* verification attacks existing code.
  - Goal: completely automatic verification.
  - Targets existing languages like C/C#/Java.
  - Great success with finite-state systems (*model checking*).
  - Obstacle: verification requiring ingenuity beyond automation.
- Alternative: *language-based* verification.

Languages of the Future

- Design new programming languages for verification.
- The time is ripe.
  - Pressure for correctness high.
  - Design space wide open.
- Surpass fully automatic approaches.
  - Greater expressiveness.
  - Can design away from problematic language features (e.g., C).
- Verification empowers programmers!
  - Write flawless code!
  - Attempt more complex, riskier techniques!

The GURU Programming Language

- A *verified programming* language.
- Combines a functional programming language and a logic.
- Can write code, prove properties about it.
- Type/proof checker, compiler to efficient C.
- Growing standard library, case studies (20kloc GURU).
- *Internal* and *external* verification:

```
sort : Fun(A:type)(cmp:Fun(a b:A).bool)(l:<list A>).
         <list A>.
sorted : Fun(A:type)(cmp:Fun(a b:A).bool)(l:<list A>).bool
sort_sorts : Forall(A:type)(cmp:Fun(a b:A).bool)(l:<list A>).
               { (sorted cmp (sort cmp l)) = tt }.
```

vs.

```
sort : Fun(A:type)(cmp:Fun(a b:A).bool)(l:<list A>).
        <sorted_list cmp A>.
```

Computational Logic Center

- New collaboration beginning this fall.
- Faculty: AS, Cesare Tinelli, Hantao Zhang.
- Goal: foster research and student, faculty development in CL.
- Main topics: verification, automated theorem proving.
- Activities: reading group this fall.
  - Meeting 10-11:30am Thursdays in MacLean B13.
  - Topic: categorical semantics for type theory.
  - Only prereq. is some mathematical maturity.
  - Talk to me if interested.

2008-2009 Teaching

- This fall: CS 185, “Programming Language Foundations”.
  - Semantics of imperative programs.
  - Nondeterminism and concurrency.
  - Untyped lambda calculus.
  - Functional programming.
  - Type systems.
- Spring: “Verified Software Construction”.
  - Goal: collaborate to build a non-trivial piece of verified software.
  - Will use the GURU verified programming language.
  - Course will be divided between lecture and studio time.
  - Software will be released as open-source at end of class.