Computational Logic Center: Research, Education, Outreach

Aaron Stump   Cesare Tinelli

Dept. of Computer Science
The University of Iowa
Iowa City, Iowa, USA

The U. Iowa Computational Logic Center (CLC)

Apply Computational Logic (CL) to
Solve problems in fields like Verification, and
Train the next generation of CL researchers.

The U. Iowa Computational Logic Center (CLC)

Apply Computational Logic (CL) to
Solve problems in fields like Verification, and
Train the next generation of CL researchers.

In this talk:

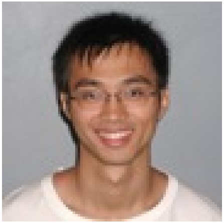
- About the group.
- Current and upcoming research projects.
- Teaching and outreach activities.

About the CLC

- Started in 2008.
- Merged AS/CT groups from U. Iowa, Washington U. in St. Louis.
- Current personnel (13 total):
  - 2 faculty.
  - 2 postdocs.
    - Garrin Kimmell. PhD, Kansas U., 2008. Working with AS.
    - Temesghen Kahsai. PhD, U. Swansea (UK), 2010. Working with CT.
  - 4 doctoral students.
    - Frank Fu. Second year, advised by AS.
    - Tianyi Liang. Third year, advised by CT.
    - Duckki Oe. Third year, advised by AS.
    - Andrew Reynolds. Third year, advised by CT.
  - 3 Master's students.
    - Harley Eades III (AS), Cuong Thai (AS), Jed McClurg (AS/CT).
  - 2 undergraduates.
    - JJ Meyer (AS), Austin Laugesen (AS).

Some Pictures

Harley Eades

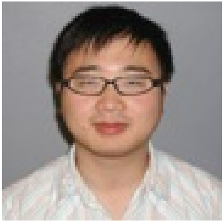


Frank Fu

Andrew Reynolds

Teme Kahsai

Garrin Kimmell



Tianyi Liang

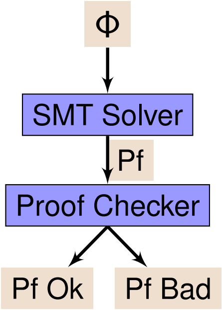
Selected Research Projects (Currently Funded)

- Parallel Solvers (NSF).
  - CT with C. Barrett (NYU). $113,936/$250,000, 2010-2012.
- StarExec (NSF).
  - Planning grant for cross-community solver execution web service.
  - AS/CT with G. Sutcliffe (U. Miami). $84,197/$100,000, 2010-2011.
  - Pending proposal: $1,889,817/$2,060,144.
- Fast Proof Checking (NSF ARRA).
  - Fast proof checking for SMT solvers.
  - AS/CT with C. Barrett (NYU). $299,986/$449,986, 2009-2011.
- TRELLYS (NSF).
  - New programming lang. for verification (dependent types).
  - AS with S. Weirich (U. Penn.), T. Sheard (Portland State).
  - $691,207/$2,090,953, 2009-2013.
- SMT-based Model Checking (AFOSR).
  - CT with C. Barrett (NYU), $457,844 / $1,058,366, 2009-2013.

Fast Proof-Checking with LFSC

Proofs and SMT Solvers

- SMT solvers large (50-100kloc), complex.
- To increase trust, have solvers emit proofs.
- Check proofs with much simpler checker (2-4kloc).

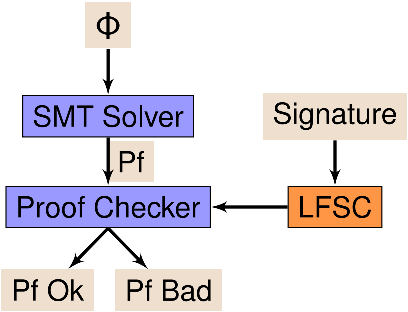


- Large, complex formulas => large proofs.
- Proofs easily 100s MBs or GBs.
- Proof-checking speed important!

The LFSC Proof Format

- “Logical Framework with Side Conditions”.
- Goal: a standard proof format for SMT.
- Developed over past 4 years (5 papers in SMT, LFMTP).
- LFSC is a meta-language.
  - Describe abstract syntax, proof rules in a *signature*.
  - LFSC then compiles that signature.
  - Supports many logics (not just SMT).
  - Result: fast custom proof checker.
  - Benefits: speed and flexibility.

LFSC Proofs and SMT Solvers



Benefits of LFSC

- Trustworthiness :
  - Declarative specification of proof checker.
  - Trusted: signature + generic LFSC compiler.
  - More trustworthy than hand-implemented checker.
  - More human-understandable (cf. CVC3's C++ rules).
- Flexibility :
  - SMT solvers have hundreds of rules.
  - No consensus on single "right" proof system.
  - Easily change signature.
  - Auto-generate C++ code for proof production (in progress).
- Performance :
  - Compilation removes overhead of using meta-language.
  - New optimizations implemented once in LFSC.
  - All proof systems can take advantage.

Logical Framework with Side Conditions

- Based on Edinburgh Logical Framework (LF) [Harper et al., '93]
- View proof-checking as type-checking.
- Adds support for computational side conditions [Stump, Oe '09].
- For example, resolution:

$$\frac{\vdash C_1 \quad \vdash C_2}{\vdash C_3} \; \text{resolve}(C_1, C_2, v) = C_3$$

- LFSC supports continuum of proof systems.

Purely Computational — Practical — Purely Declarative

LFSC Proof-Checking Optimizations

1. Compile declarative part of signature [Zeller,Stump,Deters '07].
   - Basic checker: `bool check(sig *s, pf *p)`
   - Partially evaluate this w.r.t. `sig *s`.
   - Custom checker: `bool check-s(pf *p)`
2. Compile side-condition code [Oe,Reynolds,Stump '09].
3. Incremental checking [Stump '08].
   - Traditionally: parse to AST, then check proof.
   - Optimized: parse and check together.
   - Avoid building AST for proof in memory.

5x speedup for SMT benchmarks with each of these.

Next Steps

- Experiment with trade-off between declarative, computational.
  - Comparing Proof Systems for Linear Real Arithmetic with LFSC.
    Reynolds, Haderean, Tinelli, Ge, Stump, Barrett. SMT '10
- New implementation of LFSC compiler (for fall '10).
- New input syntax.
  - BNF for abstract syntax, textual versions of rules:

```
formula f ::= true | false | and f1 f2 .

holds f1,  holds f2
------------------------- and_intro
holds (and f1 f2)
```

- Public release, tool paper.

Teaching and Outreach

Classroom Teaching

- Programming Language Foundations (185).
  - grad-level course, denotational/operational/axiomatic semantics.
  - concurrency, lambda calculus, types,
  - AS has book under contract: *Programming Language Foundations*.
- Logic in Computer Science (188).
  - grad-level applied logic.
  - propositional, predicate, temporal, modal logics.
  - applications in verification, AI, databases, etc.
- Formal Methods in Software Engineering (186).
  - grad-level formal-methods course.
  - tool-based (e.g., Alloy), emphasis on formal specification.
- Programming Language Concepts (111).
  - undergraduate programming-languages course.
  - emphasis on functional programming (OCaml).
- CLC Grad Seminar: currently, term rewriting.

Major Outreach Activities

- SMT-LIB Initiative.
  - Developed series of standards for SMT formulas.
  - Enabled major increase in productivity.
  - Co-ran competition (SMT-COMP) 2005-2010, SMT-EXEC service.
  - Haifa Verification Conference 2010 research award (with 3 others).
- Midwest Verification Day (MVD).
  - Organized 2009 and 2010 at U. Iowa.
  - 2009: 40 registered attendees, 8 institutions.
  - 2010: 55 registered attendees, 13 institutions.
  - 2011: being planned for elsewhere...

Other Outreach Activities

- Collaboration with Intel Strategic CAD Labs.
  - Interpolant generation (CT).
- Collaboration with Rockwell Collins.
  - Proposals for proof-producing model checker (AS/CT).
  - Met at RC August, 2010.
  - They co-sponsored MVD '10.
- Academic collaborations:
  - NICTA, Chalmers, INRIA, T.U. Vienna, Stanford, T.U. Barcelona, ...
- Visiting grad students:
  - T.U. Barcelona, U. Kansas, U. Missouri, U. Penn, UIUC, Stanford, ...
- Introductory teaching:
  - Intro. to Computer Science (005).
  - First-year seminars (002).
- Academic blog: QA9 (AS).

Conclusion and Future Directions

- Dynamic, growing group.
- Expanding research agenda in CL, Verification, PL.
- Future directions:
  - proof-producing model checker (AS/CT, Rockwell Collins).
  - compile-time analysis of memory management (AS).
    - use linear types to track memory.
    - support controlled aliasing.
    - memory-safe programming with no GC.
    - exploring applications to real-time systems with Jan Vitek (Purdue).
  - adding induction capabalities for CVC4 (CT).
    - allow inductive types, primitive recursive functions.
    - apply techniques for automated induction to answer queries.

LFSC Signatures by Example

Mathematical version:

$$\text{formula } f \ ::= \ \text{true} \mid \text{false} \mid p \mid (\text{and } f_1 \ f_2) \mid \ldots$$

$$\frac{\vdash f_1 \quad \vdash f_2}{\vdash (\text{and } f_1 \ f_2)} \text{ and-intro}$$

LFSC version:

```
(declare formula type)
(declare true formula)
(declare false formula)
(declare and (! f1 formula (! f2 formula formula)))

(declare holds (! x formula type))
(declare andi (! f1 formula
               (! f2 formula
               (! u1 (holds f1)
               (! u2 (holds f2)
               (holds (and f1 f2))))))))
```

A Sample Proof

Mathematical version:

$$\dfrac{\vdash p \qquad \dfrac{\vdash q \quad \vdash q}{\vdash (\text{and } q\ q)}}{\vdash (\text{and } p\ (\text{and } q\ q))}$$

LFSC version:

```
(% p formula
(% q formula
(% u1 (holds p)
(% u2 (holds q)
   (andi _ _ u1 (andi _ _ u2 u2))))))
```

- LFSC assumptions introduce with %.
- _ for the formulas proved by subproofs.