Manasij Mukherjee | CV

manasij7479@gmail.com

Research and Academic Experience

- **Master's Thesis** — **Chennai Mathematical Institute**
  Advisor: Prof. Mandayam Srivas — *September 2017–April 2018*

  I worked on function-summary based bounded model checking in the 2ls tool in the CBMC/CProver framework and investigating if this interprocedural approach can be competitive against the conventional practice of inlining function bodies at each call site. We found that this approach becomes competitive (for SVCOMP benchmarks), if conditional expressions affecting control flow are simplified by utilizing a SAT solver.

- **DSL for Linux Kernel Modules** — **Accepted for DSLDI 2017 workshop**
  Independent Work — *July 2017–September 2018*

  I developed a Domain Specific Language (DSL) for developing Linux Kernel Modules, particularly drivers. The language compiles to C and the user-definable libraries (named Utilities) are templates for various common kernel-space operations. A module provides one or more Services by using the available utilities to provide implementations for system calls. Links : Extended-Abstract Event-Page Slides

- **Teaching Assistant of Prof. Mandayam Srivas** — **Chennai Mathematical Institute**
  Course: Software Verification using SMT Solvers — *August 2017–November 2017*

  I developed some programming assignments focusing on encoding simple verification problems with SMT and delivered two lectures focusing on the design and use of SAT solvers respectively. I also held a tutorial session explaining the usage of the CBMC and 2ls tools.

- **Project Associate** — **Tata Institute of Fundamental Research**
  Advisor: Prof. Ashutosh Gupta — *May 2017–July 2018*

  I worked on a decision procedure for Partial Orders in the Z3 SMT Solver. The problem boils down to answering positive and negative connectivity queries for a changing directed graph. My primary contribution was developing a hybrid graph search which proceeds in a depth-first manner for previously seen paths. This enabled a 20-40% performance improvement for randomly generated problem instances.

Internships

- **Google - YouTube Data** — **Mountain View, California**
  Software Engineering Intern — *January 2017–April 2017*

  I worked on a new columnar evaluation engine for the database system that powers YouTube Analytics and several other internal and external systems. I came up with a fast multi-key sorting technique for the database columns and implemented a substantial portion of the GoogleSQL standard library for our new engine and in the process benchmarked the new engine and influenced its design.

- **Google - Chrome** — **Mountain View, California**
  Software Engineering Intern — *May 2016–August 2016*

  I worked on compiler optimizations for the Subzero code generator in Portable Native Client in Chrome. I implemented Common Subexpression Elimination, Loop Invariant Code Motion, Live Range Splitting, Aggressive Short Circuit Evaluation and various tweaks to the existing address optimizer functionality. This resulted in a 5% geomean improvement in benchmark (SPEC CPU2000) results for a 15% compile time tradeoff. I also developed a python script to automate bisection debugging to pinpoint tricky mis-compilations which was well received and led to a bonus. Links : Code Slides Bonus

- **Apple - Developer Tools** — **Cupertino, California**
  Software Engineering Intern — *May 2015–July 2015*

  I interned at Apple's LLVM Source Tools team, working on tools based on the Swift and Clang compilers for extracting semantic information about code during the build and finding interesting uses for it in Code Browsers, IDEs and other internal tools. This eliminates redundant invocations of the compiler front-ends for a fixed compile time overhead, which for C++ and ObjC was 4% and was negligible for Swift. This feature was launched as part of Xcode 9 in 2017.

- **CERN - ROOT/Cling Project** — **Remote**
  Google Summer of Code Student — *April 2014–August 2014*

  Cling is a C++ interpreter based on Clang and LLVM. My primary task was to implement interactive hints for missing required headers in user code. We approached this by injecting auto-generated annotated forward declarations into the interpreter environment. Links : Code Slides Proposal

Education

Universities

- **The University of Utah** — **Salt Lake City Utah**
  PhD Computer Science — *2018-Current*
- **Chennai Mathematical Institute** — **Chennai, India**
  M.Sc. Computer Science, CGPA : 8.88/10 — *2015-2018*
- **St. Xavier's College** — **Kolkata, India**
  B.Sc. Computer Science, CGPA Major: 8.69/10, CGPA Overall: 7.59/10 — *2012-2015*

Highlighted Coursework

- **Software Verification Using SMT Solvers** *Fall 2016*

  This class was focused on the design and use of SMT solvers and was responsible for making Mathematical Logic click for me. I am a TA for the current offering of this course.
- **Program Analysis** *Spring 2016*

  I visited a neighboring institution, IIT Madras, for this course. This was my first introduction to LLVM IR level compiler optimizations and was a great help in preparing for my first internship at Google afterwards. I learnt about data flow analysis, with a focus on alias analysis. The course also covered polyhedral models, shape analysis, program slicing and run time profiling. The assignments involved implementing program analysis algorithms as LLVM passes.
- **Implementation of Functional Programming Languages** *Spring 2016*

  We followed a book by Simon Peyton Jones after which the course is named. This course gave me some perspective on how program language design influences compiler optimizations and vice versa, especially in the context of the design and evolution of functional languages.
- **Proofs and Types** *Fall 2016*

  We studied Intuitionistic Logic, some introductory Type Theory, the Curry Howard Isomorphism and how they relate to Programming Languages and Theorem Provers.
- **Model Checking and Systems Verification** *Spring 2016*

  This was my first course on verification and my main takeaway was using SAT solvers in practice for intractable problems. I also learnt verification techniques like bounded model checking and k-induction. The course also covered LTL and CTL in some detail.