# Amittai F. Aviram

9 Sewall Avenue
Apt. 103
Brookline, MA 02446-5120

E-mail: amittai.aviram@gmail.com
Website: http://www.amittai.com
Telephone (mobile): 646-483-2639

## Experience

**Medtronic, Inc. Boston, MA. Senior Software Engineer. September 2016–present.**
Minimally Invasive Therapies Group (formerly Covidien, Inc.) Robotics Research and Development Lab. Developing software for the Einstein project, a new robotic assistance system for abdominal surgery. (C++, C, MATLAB, Simulink, Bash, CMake,

- Developed custom Simulink blocks to support real-time logging across a distributed system.
- Designed and implemented a multi-phased automatic code generation scheme using C++, object-oriented MATLAB, Bash scripting, CMake, and Google Protocol Buffers.
- Developed formal requirements and design documents for the distributed logging subsystem.
- Maintained and extended the logging subsystem.

**Wentworth Institute of Technology, Boston, MA. Adjunct Instructor. October 2016–present.**

- Developed and taught Programming Languages and Paradigms (CPSC 7050), an online, intensive, project-driven, 7-week, master's-level course, which introduces students to four languages and paradigms in four weeks (imperative [C], functional [Haskell], logic [Prolog], and object-oriented [C++]), as well as parallel, distributed, real-time, and embedded systems.
- Taught *Introduction to Programming* (COMP1000), a traditional, 15-week, undergraduate course, introducing students to programming concepts and practices through Java, using Eclipse and GitHub.

**MathWorks, Inc. Natick, MA. Software Engineer . September 2012–May 2016.**
Code Generation Intermediate Representation team, improving and optimizing the IR for tools to translate from MATLAB and Simulink down to C or HDL. (C++, Perl.)

- Developed new late-stage transforms to translate IR into LLVM IR for a new back end—e.g., hoisting local array and structure constants into global space, eliminating duplicates and enabling functions to share such objects for space efficiency; insertion of error code to catch integer overflow and other causes of undefined behavior at runtime.
- Extended and improved a Sanity Checker library to detect unsound changes to the IR across transforms and to enforce desired semantics on the IR as it evolved, such as strong and correct typing. Included a new framework to mark functions at major stages of compilation and condition sanity checks accordingly;
- Developed a type conversion library to enable transform writers to insert correct high-level type conversions (e.g., from one matrix type to another) and thus avoid common semantic bugs.
- Developed a numerical testing framework to help ensure transform soundness by executing (constant-folding) IR before and after the transform on select numerical inputs and comparing results and invoking the Sanity Checker on "before" and "after" IR states—heavily used in unit tests throughout the CGIR code base.
- Improved generated code performance by as much as 100-fold by adjusting transform heuristics so as to avoid expensive computations within nested loops.
- Refactored legacy code using test-driven development and Clean Code principles.
- Supervised QE projects, including an additional numerical testing framework.
- Fixed and prevented bugs—including potentially catastrophic bugs—caused by misunderstood semantics, hidden numerical transformations, and vulnerabilities to undefined behavior.
- Led research compiler research reading groups.

**Google, Inc. New York, NY. May–September 2011 (Summer Intern).**
Site Reliability Engineering team. Designed and developed a tool to gather and report data on high-latency operations in the distributed storage infrastructure (C++).

**Google, Inc.  New York, NY. May–August 2010 (Summer Intern).**
DoubleClick team.  Designed and developed an integrated testing infrastructure for their new advertisement tracking tag server  (Python).

**Ellington Management Group.  Old Greenwich, CT.  May–Aug. 2007 (Summer Intern).**
Developed a custom Web services infrastructure, including parser, serializer, asynchronous parallel message handler, deserializer, and CLR IR code generator (C#), together with unit and system tests (Python).

Microsoft Research.  Redmond, WA.  May–August 2006 (Summer Intern).
Designed and developed a demonstration project to infer a context-free grammar from the source code of a hand-written parser, using Microsoft's Phoenix compiler toolkit (C++ .Net).

## Education

**Yale University.  PhD, Computer Science.**
Dissertation project: Deterministic Parallel OpenMP.  Advisor: Bryan Ford.
Co-author, USENIX OSDI Best Paper Award, 2010.

**Columbia University. BS, Computer Science.**
Theory track.  Russell C. Mills Award.  Contributed to NLP research project on Arabic morphology.

**Yale University.  PhD, English Language and Literature.**

**Columbia University.  BA, English and Comparative Literature.**

## Certificate

Machine Learning. Coursera.  9 August 2016.

## Previous Career

**Associate Professor, English and Comparative Literature,**
University of South Carolina, Columbia, SC.   August 1984–August 2004.
Research and teaching on poetry and poetics.  Mellon Postdoctoral Fellowship, Cornell University, 1986.
Tenure: 1994.  Fulbright Senior Scholar, Germany, 2001.

## Technical Skills

**Programming Languages**
C++, C, C#, Java, Perl, Python, PHP, Haskell, ML, R

**Markup and Query Languages**
HTML, LaTeX, Dot, SQL

**Operating Systems Used**
Linux, Windows, Mac OS X

**Tools Used**
GCC, GDB, Clang/LLVM, LLDB, Xcode, Visual Studio, Eclipse, VI (Gvim), Qt