

Daniel R. Licata

Personal Information:

E-mail: dr1@cs.cmu.edu
Web: <http://www.cs.cmu.edu/~dr1/>
Home Address: 5742 Northumberland St. #2
Pittsburgh, PA 15217
Mobile Phone: +1 (412) 889-0106
Citizenship: United States

Education:

Carnegie Mellon University 2004 to present

PhD Candidate in Computer Science. Advised by Robert Harper.

Courses include:

- Type Systems
- Extensible Software
- Planning, Execution, and Learning
- Computation and Deduction
- Logic Programming
- Computer Networks

Brown University

2000 to 2004

Bachelor of Science in Mathematics and Computer Science.

Honors Degree. *Magna cum laude*. Courses include:

- Programming Languages
- Machine Learning
- Number Theory
- Abstract Algebra
- Distributed Systems
- Information Theory
- Differential Geometry
- Non-Euclidean Geometry
- Software Verification
- Cryptography
- Topology
- Philosophy of Education

Internships:

Microsoft Research, Cambridge UK

Summer, 2007

Research Intern. Advised by Simon Peyton-Jones.

I designed and implemented a new feature called *view patterns* for GHC, a compiler for the programming language Haskell.

Publications:

Daniel R. Licata, Noam Zeilberger, and Robert Harper. Focusing on Binding and Computation. In the proceedings of *Logic in Computer Science*, 2008.

Robert Harper and Daniel R. Licata. Mechanizing Metatheory in a Logical Framework. *Journal of Functional Programming*, 17(4–5), pages 613–673. July, 2007.

Tom Murphy VII, Daniel Spoonhower, Chris Casinghino, Daniel R. Licata, Karl Crary, and Robert Harper. The Cult of the Bound Variable: The 9th Annual ICFP Programming Contest. Technical Report CMU-CS-06-163, 2006.

Daniel R. Licata and Robert Harper. A Formulation of Dependent ML with Explicit Equality Proofs. Technical Report CMU-CS-05-178, 2005.

Daniel R. Licata and Shriram Krishnamurthi. Verifying Interactive Web Programs. In the proceedings of *Automated Software Engineering*, 2004. IEEE Press.

Daniel R. Licata, Christopher Harris, and Shriram Krishnamurthi. The Feature Signatures of Evolving Programs. In the proceedings of *Automated Software Engineering*, 2003. IEEE Press.

Research Projects:	<p>Dependently Typed Programming 2004 to present With Robert Harper, CMU Current type systems for languages such as ML and Haskell ensure important but limited program properties. My primary research project is the design of a new programming language in which programmers can define logics and use them to verify detailed properties of their code. For example, a programmer might define an access-control logic and use it to statically prevent unauthorized access to controlled resources. This language can also be used to prove theorems about logics, in the style of Twelf. Work on this project has been published in <i>Logic in Computer Science</i>, 2008; further work is described in a draft submitted to <i>POPL '09</i>.</p> <p>A Module System for Twelf 2006 to present With Robert Harper, Frank Pfenning, Rob Simmons, and Daniel Lee, CMU Twelf is a tool used to formalize, implement, and prove theorems about formal deductive systems such as programming languages and logics. Unfortunately, the Twelf language currently lacks a module system. We have designed an ML-style module system for Twelf that alleviates the burdens of namespace management and provides greater opportunities for code extensibility and reuse. The implementation of this design is currently in progress.</p> <p>Verifying Interactive Web Programs 2003 and 2004 With Shriram Krishnamurthi, Brown University Despite the ever-growing commercial importance of Web programs, many remain notoriously buggy and unreliable. We developed a method for proving correctness properties of and finding bugs in interactive Web programs. This work has been published in the proceedings of <i>Automated Software Engineering</i>, 2004.</p> <p>Feature Signatures Summer 2002 With Shriram Krishnamurthi and Chris Harris, Brown University Software maintainers would benefit greatly from knowing why a given piece of code was introduced and how it has previously been modified; unfortunately, this information is rarely documented. We designed an automated technique for generating the <i>feature signatures</i> of a program and applying them to several software engineering tasks, including code rationale generation and aspect mining. This work has been published in the proceedings of <i>Automated Software Engineering</i>, 2003.</p>
Students Advised:	<p>Arbob Ahmad, CMU. Advised jointly with Robert Harper Equality in the Finitary λ-calculus 2007-2008 Arbob completed two summers and three semesters of undergraduate research on a new, proof-theoretic algorithm for deciding program equality in the λ-calculus with finite products and coproducts. Arbob's work resulted in a talk contributed to the <i>ACM SIGPLAN Workshop on Mechanizing Metatheory</i>, 2007, and we expect a submission to a major conference in the next few months. Arbob is proceeding to the PhD program in computer science this fall.</p>
Teaching Experience:	<p>CMU 15-312: Principles of Programming Languages Teaching Assistant, Spring 2006 With Robert Harper This course, taken primarily by sophomores and juniors, is a rigorous introduction to type systems and operational semantics for functional programming, control and state effects, and concurrency. As a TA, I wrote and graded the assignments and exams, held office hours, and lectured at weekly recitation sections.</p>

Brown CS017/018: An Integrated Introduction to Computer Science

Head Teaching Assistant, 2002-2003 and 2003-2004; Teaching Assistant, 2001-2002

With John F. Hughes and Philip Klein

CS17-18 is a year-long introductory sequence that teaches programming in Scheme, ML, and Java as well as design and analysis of algorithms and data structures. As a TA, I held office hours, led weekly lab sections, graded, and developed new homeworks and exams. As a Head TA, I additionally developed new projects and course software, managed a course staff of seven to nine people, and collaborated with the professor to decide on course content and pedagogical techniques.

Service:

The 9th Annual ICFP Programming Contest

The ICFP Programming Contest is an annual three-day competition associated with the International Conference on Functional Programming. The 2006 contest asked participants to uncover the secrets of a (fictional) ancient society of computer scientists by solving a series of puzzles based on programming languages research. Seven hundred participants on 365 teams from all over the world competed, more than in any previous year. As one of the four primary developers of the contest, I designed puzzles, implemented contest software, and answered questions from participants.

Popularizing Twelf

Twelf is a tool used by programming language researchers to mechanically verify their work. I have written several tutorials designed to make Twelf more accessible to more researchers. Robert Harper and I wrote a technical introduction to LF and Twelf, published in the *Journal of Functional Programming*. I have also written a more informal introduction to Twelf, along with many other tutorials on Twelf techniques, for the Twelf Wiki (<http://www.twelf.org>).

CMU CSD PhD Admissions Committee, 2006 and 2007

In both 2006 and 2007, I was a member of a dozen-person committee of faculty and students that evaluated over 700 applicants for admission to the computer science department's PhD program.

ConCert Reading Group Organizer, 2006 to 2007

I chose research papers for a weekly reading group meeting.

Awards & Fellowships:

Carnegie Mellon University Anonymous Fellowship in Computer Science, 2008

Finalist for Computing Research Association Outstanding Undergraduate Award, 2004.

Honorable Mention - NSF Graduate Research Fellowship, 2004 and 2005.

Brown University Senior Prize in Computer Science, 2004.

Karen T. Romer Undergraduate Teaching and Research Assistantship, Summer 2003.

References

Robert Harper. Carnegie Mellon University. rwh@cs.cmu.edu.

Frank Pfenning. Carnegie Mellon University. fp@cs.cmu.edu.

Simon Peyton-Jones. Microsoft Research. simonpj@microsoft.com.