David Van Horn

Curriculum Vitae

Dept. of Computer Science & UMIACS
University of Maryland
A.V. Williams Building
College Park, MD 20742

www.cs.umd.edu/~dvanhorn
dvanhorn@cs.umd.edu
(202) 460-4104

Education

Ph.D., Brandeis University, Comp. Sci., *The Complexity of Flow Analysis in Higher-Order Languages*, 2009
M.S., University of Vermont, Comp. Sci., *Algorithmic Trace Effect Analysis*, 2006
B.S., University of Vermont, Comp. Sci. & Info. Sys., 2003

Employment

University of Maryland	Assistant Professor, Computer Science	2014–*present*
	Assistant Professor, UMIACS	2014–*present*
Northeastern University	Research Assistant Professor	2012–2013
	Visiting Assistant Professor	2011–2012
	CRA Computing Innovation Fellow	2009–2011

Awards

NSF CAREER Award	2019
OOPSLA Distinguished Paper	2018
ACM Computing Reviews Notable Book	2013
Northeastern University Excellence in Teaching Award Nominee	2013
Communications of the ACM, Research Highlight	2011
CRA Computing Innovation Fellow	2009

Grants

CAREER: Gradual Verification: From Scripting to Proving — 1/2019-1/2024
National Science Foundation — $573,376
Lead investigator

SHF: Small: Collaborative Research: Online Verification-Validation — 9/2016–8/2019
National Science Foundation — $140,009
Lead investigator

REU for SHF: Small: Collaborative Research: Online Verification-Validation — 9/2016–8/2019
National Science Foundation — $16,000
Lead investigator

TWC: Large: Collaborative: The Science and Applications of Crypto-Currency — 7/2015–6/2017
National Science Foundation — $593,941
Co-investigator

Sound Over- & Under-Approximations of Complexity & Information Security — 4/2015–4/2018
Defense Advanced Research Projects Agency — $830,100
Co-investigator

Trustworthy and Composable Software Systems with Contracts 2/2014–7/2017
National Security Agency $301,910
Co-investigator

Scalable and Precise Abstractions of Programs for Trustworthy Software 12/2013–6/2016
Defense Advanced Research Projects Agency $541,560
Lead investigator

SHF: Small: Behavioral Software Contract Verification 9/2012–5/2015
National Science Foundation $400,000
Co-investigator

CRA Computing Innovation Fellow 9/2009–5/2011
National Science Foundation $267,500
Lead investigator

Publications

Names of student co-authors are annotated with superscripts to indicate being from UMD (m), NEU (n), or other (o) institutions; as being undergraduate (u), graduate (g), or post-doctoral (p) students at the time of publication. Advisees (at the time of publication) are indicated in **bold**.

Publications: Books

Realm of Racket.
Matthias Felleisen, David Van Horn, Conrad Barski, and Northeastern undergraduates: Forrest Bice[n,u], Rose DeMaio[n,u], Spencer Florence[n,u], Feng-Yun Mimi Lin[n,u], Scott Lindeman[n,u], Nicole Nussbaum[n,u], Eric Peterson[n,u], Ryan Plessner[n,u].
No Starch Press 2013.

Publications: Articles in Refereed Journals and Conference Proceedings

Size-Change Termination as a Contract.
Phúc C. Nguyen[m,g], **Thomas Gilray**[m,p], Sam Tobin-Hochstadt, and David Van Horn.
PLDI 2019 (To appear).

Type-Level Computations for Ruby Libraries.
Milod Kazerounian[o,g], **Sankha Guria**[m,g], **Niki Vazou**[m,p], Jeffrey S. Foster, and David Van Horn.
PLDI 2019 (To appear).

Constructive Galois Connections.
David Darais[m,g] and David Van Horn.
JFP 2019 (To appear). (**ICFP 2016 Special Issue.**)

Gradual Liquid Type Inference.
Niki Vazou[m,p], Éric Tanter, and David Van Horn.
OOPSLA 2018. (**Distinguished Paper Award.**)

Theorem Proving for All: Equational Reasoning in Liquid Haskell.
Niki Vazou[m,p], Joachim Breitner[o,p], **Rose Kunkel**[m,u], David Van Horn, and Graham Hutton.
Haskell Symposium 2018.

Soft Contract Verification for Higher-order Stateful Programs.
Phúc C. Nguyen[m,g], **Thomas Gilray**[m,p], Sam Tobin-Hochstadt, and David Van Horn.
POPL 2018.

Abstracting Definitional Interpreters.
David Darais[m,g], **Phúc C. Nguyen**[m,g], **Nicholas Labich**[m,g], and David Van Horn.
ICFP 2017.

Higher-Order Symbolic Execution for Contract Verification and Refutation.
Phúc C. Nguyen[m,g], Sam Tobin-Hochstadt, and David Van Horn.
JFP 2017. (**ICFP 2014 Special Issue.**)

A Vision for Online Verification-Validation.
Matthew A. Hammer, Bor-Yuh Evan Chang, and David Van Horn.
GPCE 2016.

Constructive Galois Connections: Taming the Galois Connection Framework for Mechanized Metatheory.
David Darais[m,g] and David Van Horn.
ICFP 2016.

Pushdown Control-Flow Analysis for Free.
Thomas Gilray[o,g], Steven Lyde[o,g], Michael D. Adams[o,p], and Matthew Might.
POPL 2016.

Incremental Computation with Names.
Matthew A. Hammer[m,p], Joshua Dunfield, Kyle Headley[m,u], **Nicholas Labich**[m,g], Jeff Foster, Michael Hicks.
OOPSLA 2015.

Galois Transformers and Modular Abstract Interpreters.
David Darais[m,g], Matthew Might, and David Van Horn.
OOPSLA 2015.

Running Probabilistic Programs Backwards.
Neil Toronto[m,p], Jay McCarthy, and David Van Horn.
ESOP 2015.

Abstracting Abstract Control.
Dionna A. Glaze[n,g] and David Van Horn.
DLS 2014.

Pruning, Pushdown Exception-Flow Analysis.
Shuying Liang[o,g], Weibin Sun[o,g], Matthew Might, Andrew W. Keep[o,p], and David Van Horn.
SCAM 2014.

Soft Contract Verification.
Phúc C. Nguyen[m,g], Sam Tobin-Hochstadt, and David Van Horn.
ICFP 2014.

Pushdown flow analysis with abstract garbage collection.
Dionna A. Glaze[n,g], Ilya Sergey[o,p], Christopher Earl[o,g], Matthew Might, and David Van Horn.
JFP 2014. (**ICFP 2012 Special Issue.**)

Optimizing Abstract Abstract Machines.
Dionna A. Glaze[n,g], **Nicholas Labich**[n,u], Matthew Might, and David Van Horn.
ICFP 2013.

Higher-Order Symbolic Execution via Contracts.
Sam Tobin-Hochstadt and David Van Horn.
OOPSLA 2012.

Systematic Abstraction of Abstract Machines.
David Van Horn and Matthew Might.
JFP 2012. (**ICFP 2010 Special Issue.**)

Introspective Pushdown Analysis of Higher-order Programs.
Christopher Earl[o,g], Ilya Sergey[o,g], Matthew Might, and David Van Horn.
ICFP 2012.

A Family of Abstract Interpretations for Static Analysis of Concurrent Higher-Order Programs.
Matthew Might and David Van Horn.
SAS 2011.

Abstracting Abstract Machines: A Systematic Approach to Higher-Order Program Analysis.
David Van Horn and Matthew Might.
CACM 2011. (**Research Highlight.**)

Abstracting Abstract Machines.
David Van Horn and Matthew Might.
ICFP 2010.

Evaluating Call-By-Need on the Control Stack.
Stephen Chang[n,g], David Van Horn, and Matthias Felleisen.
TFP 2010. (**Best Student Paper Award.**)

Resolving and Exploiting the k-CFA Paradox: Illuminating Functional vs. Object-Oriented Program Analysis.
Matthew Might, Yannis Smaragdakis, and David Van Horn.
PLDI 2010.

Deciding kCFA is complete for EXPTIME.
David Van Horn and Harry G. Mairson.
ICFP 2008.

Flow Analysis, Linearity, and PTIME.
David Van Horn and Harry G. Mairson.
SAS 2008.

Types and Trace Effects of Higher Order Programs.
Christian Skalka, Scott Smith, and David Van Horn.
JFP 2008.

Relating Complexity and Precision in Control Flow Analysis.
David Van Horn and Harry G. Mairson.
ICFP 2007.

Publications: Articles in Workshop Proceedings

Sound and Precise Malware Analysis for Android via Pushdown Reachability and Entry-Point Saturation.
Shuying Liang[o,p], Andrew W. Keep[o,p], Matthew Might, Steven Lyde[o,g], Thomas Gilray[o,g], Petey Aldous[o,g], and David Van Horn.
ACM Workshop on Security and Privacy in Smartphones & Mobile Devices 2013.

AnaDroid: Malware Analysis of Android with User-supplied Predicates.
Shuying Liang[o,p], Matthew Might, and David Van Horn.
Workshop on Tools for Automatic Program Analysis 2013.

Concrete Semantics for Pushdown Analysis: The Essence of Summarization.
Dionna A. Glaze[n,g] and David Van Horn.
Workshop on Higher-Order Program Analysis 2013.

From Principles to Practice with Class in the First Year.
Sam Tobin-Hochstadt and David Van Horn.
Workshop on Trends in Functional Programming in Education 2013.

Semantic Solutions to Program Analysis Problems..
Sam Tobin-Hochstadt and David Van Horn.
PLDI 2011, FIT Session.

Pushdown Control-Flow Analysis of Higher-Order Programs.
Christopher Earl[o,g], Matthew Might, and David Van Horn.
Workshop on Scheme and Functional Programming 2010.

A Type and Effect System for Flexible Abstract Interpretation of Java.
Christian Skalka, Scott F. Smith, and David Van Horn.
ACM Workshop on Abstract Interpretations of Object-Oriented Programs 2005.

Publications: Reports, Drafts, and Non-Refereed Monographs

Sound Over- & Under-Approximations of Complexity & Information security (SOUCIS).
Michael Hicks, David Van Horn, Jeff Foster, Eric Koskinen, Dawn Song, Timos Antopoulos.
AFRL Technical Report AFRL-RI-RS-TR-2018-229, 2018.

An Introduction to Redex with Abstracting Abstract Machines.
David Van Horn.
Draft 2018.

Automated Techniques for Higher-Order Program Verification.
Naoki Kobayashi, Luke Ong, and David Van Horn.
Progress in Informatics, No. 10, 2013.

Pushdown Abstractions of Javascript.
David Van Horn and Matthew Might.
CoRR, abs/1109.4467, 2011.

Talks

Size-Change Termination as a Contract.
IBM Programming Languages Day, December 2018.

From Scripting to Proving: Gradual Verification with a Scheme.
Keynote, The ACM SIGPLAN Scheme Workshop, September 2018.

From Scripting to Proving: Gradual Verification for Expressive Programming Languages.
Yale University, CS Colloquium, December 2018.
University of Maryland, CS Colloquium, September 2018.

Symbolic Execution for Higher-Order Program Verification.
Vrije Universiteit Brussel, May 2018.
Princeton University, May 2018.
UCLA, Compilers Colloquium, April 2018.
University of Washington, PLSE Colloquium, April 2018.

Redex, Abstract Machines, and Abstract Interpretation.
Oregon Programming Languages Summer School (OPLSS), July 2017. Three days of lectures.

Verification and Refutation of Behavioral Contracts with Higher-Order Symbolic Execution.
University of Chile, PLEAID Seminar, January 2016.
Johns Hopkins University, PL Seminar, October 2015.
Indiana University, PL Wonks Seminar, January 2015.

Tutorial: Introduction to Redex with Abstracting Abstract Machines.
University of Chile, PLEAID Seminar, January 2016.

Young Researcher Panel.
ACM SIGPLAN Programming Languages Mentoring Workshop (PLMW) at ICFP, August 2015.

Abstracting Abstract Machines.
University of Utah, PLT Redex Summer School, July 2015.

Soft Contract Verification.
Dagstuhl Seminar on Scripting Languages and Frameworks: Analysis and Verification, July 2014.
NII Workshop on Software Contracts for Communication, Monitoring, and Security, May 2014.

Analysis for Trustworthy Software.
Third Annual Maryland Cybersecurity Center Symposium, June 2014.

Synthesis from Contracts.
Defense Advanced Research Projects Agency, March 2014.

Program Analysis for Trustworthy Software.
Laboratory for Telecommunication Sciences, March 2014.

Abstracting Definitional Interpreters.
Mid-Atlantic Programming Languages Seminar, April 2013.

Analysis for Trustworthy Software.
University of Maryland, CS Colloquium, March 2013.

Analyzing Software Contracts.
DARPA Clean-slate design of Resilient Adaptive Secure Hosts, December 2012.

Raising the Level of Discourse with GnoSys.
DARPA Clean-slate design of Resilient Adaptive Secure Hosts, November 2012.

Towards the Verification of Behavioral Software Contracts.
Microsoft Research, RiSE Group invited lecture, Redmond, Washington, November 2012.

Program Verification via Abstract Reduction Semantics.
Harvard University, Advanced Functional Language Compilation (invited lecture), November 2012.

Optimized Machines for Program Analysis.
Harvard University, Advanced Functional Language Compilation (invited lecture), November 2012.

Abstract Machines for Program Analysis.
Harvard University, Advanced Functional Language Compilation (invited lecture), November 2012.

Scalable Abstractions for Trustworthy Software.
DARPA Automated Program Analysis for Cybersecurity, October 2012.

Low-level Analysis for High-level Assurance.
DARPA Clean-slate design of Resilient Adaptive Secure Hosts, October 2011.

Verification via Abstract Reduction.
NII Workshop: Automated Techniques for Automated Higher-order Program Verification, September 2011.

The Complexity of kCFA.
NII Workshop: Automated Techniques for Automated Higher-order Program Verification, September 2011.

What Program Analysis Can and Cannot Do for You.
Rice University CS Colloquium, March 2011.

What Program Analysis Can and Cannot Do for You.
University of Utah, CS Colloquium, February 2011.

The Paradox of Flow Analysis, Or: What We Talk About When We Talk About Higher-Order Flow Analysis.
MIT Programming Languages Working Group, February 2011.

Modular Analysis via Abstract Reduction Semantics.
New Jersey Programming Languages and Systems Symposium, Rutgers University, December 2010.

Pushdown Control-Flow Analysis of Higher-Order Programs.
IBM Programming Languages Day, July 2010.

Abstracting Abstract Machines: Storing and Stacking Continuations.
Harvard University, Programming Languages Seminar, July 2010.

Abstracting Abstract Machines.
New England Programming Languages and Systems Symposium, April 2010.

Resolving and Exploiting the k-CFA Paradox.
University of Oregon, CIS Colloquium, April 2010.
New England Programming Languages and Systems Symposium, MIT, December 2009.

Subcubic Control-Flow Analysis Algorithms.
ACM Symposium in Honor of Mitchell Wand, Northeastern University, August 2009.

The Complexity of Flow Analysis.
New England Programming Languages and Systems Symposium, November 2008.
Northeastern University, Graduate Programming Languages Seminar, October 2008.

Relating Complexity and Precision in Control Flow Analysis.
Northeastern University, Programming Languages Seminar, May, 2007.
IBM Programming Languages Day, May 2007.

Linearity and Program Analysis.
Northeastern University, Graduate Programming Languages Seminar, October 2007.

Advising: Undergraduate Research Advisor

Deena Postol		2018–*current*
Rachael Zehrung		2018–*current*
Cameron Moy		2018–*current*
Rose Kunkel	Now: PhD student at University of California, San Diego	2016–2018
Kyle Headley	Now: PhD student at University of Colorado, Boulder	2014–2015
Rebecca MacKenzie	Now: Academic IT Coordinator, Northeastern University	2014

Advising: Masters Advisor

Nicholas Labich	Now: United Income	2018
Javran Cheng	Now: Google	2018

Advising: Doctoral Advisor

Sankha Guria	(Co-advised with Jeff Foster.)	
Phuc C. Nguyen	*Higher-order Symbolic Execution*	expected 2019
David Darais	*Mechanizing Abstract Interpretation* Now: Asst. Professor, University of Vermont	2017
Dionna A. Glaze	*Automating Abstract Interpretation of Abstract Machines*, Northeastern University Now: Google	2014

Advising: Doctoral Thesis Committee Member

Quentin Stiévenart	*Scalable Designs for Abstract Interpretation of Concurrent Programs: Application to Actors and Shared-Memory Multi-Threading*, Vrije Universiteit Brussel	2018
Kristopher Micinski	*Interaction-Based Privacy Policies for Mobile Apps* Now: Visiting Asst. Professor, Haverford College	2017
Piotr Mardziel	*Modeling, Quantifying, and Limiting Adversary Knowledge* Now: Systems Scientist, Carnegie Mellon University	2015
Vincent St-Amour	*How to Generate Actionable Advice about Performance Problems* Now: Assistant Professor of Instruction, Northwestern University	2015
Stephen Chang	*On the Relation Between Laziness and Strictness*, Northeastern University Now: Associate Research Scientist, Northeastern University	2014
Shuying Liang	*Static Analysis of Android Applications*, University of Utah Now: HP Fortify Labs.	2014
Letterio Galletta	*Adaptivity: Linguistic Mechanisms and Static, Analysis Techniques*, University of Pisa Now: Asst. Professor, IMT School for Advanced Studies, Lucca.	2014

Advising: Postdoctoral Advisor

Niki Vazou	Now: Asst. Professor, IMDEA	2016–2018
Thomas Gilray	Now: Asst. Professor, University of Alabama, Birmingham	2016–2018
Shiyi Wei	Now: Asst. Professor, University of Texas at Dallas	2015–2017
Matthew Hammer	Now: Asst. Professor, University of Colorado, Boulder	2014
Neil Toronto	Now: Microsoft Research, Cambridge	2014

Service: Activities for Funding Agencies

NSF Directorate for CISE, Panelist, 2011.
NSF Directorate for CISE, Panelist, 2010.

Service: Activities for Journals, Conferences, and Workshops

Steering Committee: ICFP 2013–2015, HOPA 2014.

Chair: Programming Langauges Mentoring Workshop (PLMW) at ICFP 2019, PLMW at ICFP 2018, POPL Workshops 2013–2015, TFP 2016, HOPA 2014, NII Workshop on Automated Techniques for Higher-Order Program Verification 2011, New England Programming Languages and Systems Symposium 2011.

Program Committee: OOPSLA 2019, PADL 2018, TFP 2017, SAS 2017, POPL 2017, ECOOP 2016, ICFP 2015, OBT 2015, ESOP 2014, PADL 2014, LOLA 2014, TFP 2014, TFPIE 2014, Scala 2013, HOPA 2013, TFP 2013, ICFP 2011, Scheme 2011, Scheme 2009.

External Review Committee: ICFP 2018, POPL 2016, POPL 2013.

Reviewer: POPL 2019, VMCAI 2016, POPL 2015, POPL 2014, VMCAI 2014, ICFP 2014, DLS 2014, OOPSLA 2012, DLS 2012, ESOP 2011, ICFP 2010, POPL 2008, LICS 2007, CSL 2007.

Journal Reviewer: ACM Computing Surveys, ACM Transactions on Computational Logic, ACM Transactions on Programming Languages and Systems, Higher-Order and Symbolic Computation, Journal of Functional Programming, Science of Computer Programming.

Service: Departmental Service

Faculty Search Committee	2018–*current*
Graduate Student Review Committee, Chair	2016–*current*
Middle States Committee	2016–2018
Graduate Admissions Committee	2014–2016
Space Planning Committee for Iribe Center	2014–2018
Education Committee	2013–*current*
PL/SE/HCI Field Committee	2013–*current*

Teaching: Curriculum Development

CMSC 131A: Systematic Program Design I: This course (along with 132A) is a complete redesign of the first year introductory programming curriculum at UMD. The course is an introduction to computing and programming. Its major goal is to introduce students to the principles of systematic problem solving through programming and the basic rules of computation. This course exposes students to the fundamental techniques of program design: an approach to the creation of software that relies on systematic thought, planning, and understanding from the very beginning, at every stage and for every step. I wrote extensive notes, a complete set of labs, assignments, midterms, practice problems, practice exams, and final exams. Video recordings of all lectures, including screen captures of in-class coding, are available on Panopto.

CMSC 132A: Systematic Program Design II: This is the follow-up course to 131A. It studies the class-based program design and the design of abstractions that support the design of reusable software and libraries. It covers the principles of object oriented program design, the basic rules of program evaluation, and examines the relationship between algorithms and data structures, as well as basic techniques for analyzing algorithm complexity. I wrote extensive notes, a complete set of labs, assignments, midterms, practice problems, practice exams, and final exams. Video recordings of all lectures, including screen captures of in-class coding, are available on Panopto.

CMSC 430: Introduction to Compilers: Each semester I have taught this course, I have revised existing materials and written new midterms and exams.

CMSC 631: Program Analysis and Understanding: Upon joining the faculty at UMD, I did a from-scratch redesign of the PhD-level PL and program analysis course. The course objectives include to introducing students to the complementary research areas of programming languages and program analysis and exposing students to the basic principles of research processes in computer science: how to ask/articulate questions and how to recognize elements of solutions. It covers basic theoretical ideas and practical techniques for modeling and analyzing programming languages; and leveraging those techniques to mechanically reason about programs. As part of the course development, I wrote an extensive set of course notes, designed research projects, developed lectures, exams, and programming assignments. I have signficantly revised and refined the course over three iterations.

Honors Intro. to Programming and Computing II (Northeastern University): With Sam Tobin-Hochstadt, I designed this course from scratch to offer a second semester programming course for Honors College and advanced students to take as an alternative to the standard CS II course. We developed an extensive set of course notes, a pedagogicaly oriented programming language for exploring class- and object-oriented designs, and a complete set of exams, labs, and problem sets.

Honors Intro. to Programming and Computing I (Northeastern University): With Olin Shivers, I adapted the existing intro course for onors Collegeand advanced students to take as analternative to the standard CS I course. We developed several new units on advanced topics such as interpreters and control operators as well as accompanying labs, exams, and problem sets.

Teaching: Courses Taught

Make Your Own Video Games: An Introduction to Programming and Computing (Terps Young Scholars program), CMSC 198Q, Summer 2019 (future).
Systematic Program Design II, CMSC 132A (40), Spring 2019 (future).
Systematic Program Design I, CMSC 131A (40), Fall 2018
Systematic Program Design II, CMSC 132A (75), Spring 2018
Systematic Program Design I, CMSC 131A (140), Fall 2017
Introduction to Compilers, CMSC 430 (40), Spring 2017
Introduction to Compilers, CMSC 430 (40), Spring 2016
Program Analysis and Understanding, CMSC 631 (20), Fall 2015
Introduction to Compilers, CMSC 430 (40), Spring 2015
Program Analysis and Understanding, CMSC 631 (15), Fall 2014
Program Analysis and Understanding, CMSC 631 (20), Spring 2014
(Northeastern University) Intro. to Programming and Computing I (247), 2007–2010.
(Northeastern University) Intro. to Programming and Computing I, Honors (134), 2009–2011.
(Northeastern University) Intro. to Programming and Computing II (312), 2008–2009, 2012–2013.
(Northeastern University) Intro. to Programming and Computing II, Honors (110), 2011-2013.