THE TWENTY-FIRST ANNUAL

HIGH CONFIDENCE SOFTWARE AND SYSTEMS CONFERENCE

HCSS

https://cps-hcss.org

The Twenty-First Annual

HIGH CONFIDENCE SOFTWARE AND SYSTEMS CONFERENCE

https://cps-hcss.org

May 3-6, 2021

WELCOME MESSAGE

On behalf of the HCSS steering committee and the NITRD CNPS Interagency Working Group, it is our great pleasure to welcome you to the 21st annual High Confidence Software and Systems (HCSS) Conference.

For a second year in a row, the conference has to be held virtually due to the continued world's health situation, and we truly hope we will be able to be back in Annapolis next year.

This year's program continues a tradition of excellence at HCSS. World-class speakers from academia, industry, and government will draw on a broad range of experiences to deliver four days of technical talks. These presentations will describe new scientific and technological foundations that will enable new generations of engineered systems for computing, communication, and control. The technologies discussed will enable the creation of high-assurance systems essential for life-, safety-, security-, and mission-critical operation.

This year's HCSS talks focus on the themes of Proof Robustness, Exploring Composability, and Continuous Development and Formal Methods.

We hope that you will find the 2021 Conference stimulating and informational. We greatly appreciate your attendance, and look forward to your participation and support at future HCSS conferences.

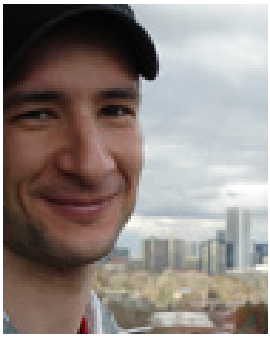
HCSS Co-Chairs

June Andronick, Proofcraft & UNSW Sydney
Lee Pike, Amazon Web Services

CO-CHAIRS

June Andronick
Proofcraft & UNSW Sydney

June Andronick is CEO and co-founder of the newly created Proofcraft company. She previously led the Trustworthy Systems group, world-leading in the area of verified operating systems software, known worldwide for the formal verification of the seL4 microkernel. She is also conjoint Professor at UNSW Sydney, Australia. Her main research interest is in formal verification and certification of software systems, more precisely in formal proofs of correctness and security properties of programs using interactive theorem proving, as well as concurrency reasoning, targeting interruptible and multicore systems. She was recognised in 2011 by MIT's Technology Review as one of the world's top young innovators (TR35). She previously worked in industry for the smart-card manufacturer Gemalto in Formal Methods research, where she did her PhD in collaboration with University of Paris Sud.



Lee Pike
Amazon Web Services

Lee Pike is currently a Principal Applied Scientist at Amazon Web Services. He previously led the compiler team at Groq, Inc. Before that, he directed the Cyber-Physical Systems program at Galois, Inc. He has been the PI on externally-funded research contracts funded by NASA, DARPA, AFRL, DOT, and other agencies and Fortune 100 companies. His research focuses on applying techniques from functional programming, run-time verification, and formal verification to the areas of operating systems, compilers, cryptographic systems, avionics, and control systems. Previously, he was a research scientist in the NASA Langley Formal Methods Group and has a Ph.D in Computer Science from Indiana University.

CONFERENCE ORGANIZATION

PROGRAM CO-CHAIRS

June Andronick, Proofcraft & UNSW Sydney
Lee Pike, Amazon Web Services

STEERING COMMITTEE

Perry Alexander, University of Kansas
Kathleen Fisher, Tufts University
John Hatcliff, Kansas State University
John Launchbury, Galois, Inc.
Stephen Magill, MuseDev / Sonatype
Brad Martin, National Security Agency
Ray Richards, DARPA
William Scherlis, DARPA
Eric W. Smith, Kestrel Institute
Sean Weaver, National Security Agency
Matthew Wilding, Collins Aerospace

MEETING ORGANIZERS

Publicity and Communications:
Katie Dey, Vanderbilt University

Virtual Logistics:
Regan Williams, Vanderbilt University

Onsite Support:
Anne Dyson, CPVI
Alexis Rodriguez, Vanderbilt University

Graphic Design:
Amy Karns, Vanderbilt University

TABLE OF CONTENTS

GENERAL INFORMATION

VIRTUAL VENUE

The 2021 HCSS Conference will be hosted on the Hopin virtual conferencing platform. **Hopin** is a virtual venue with multiple interactive areas that are optimized for connecting and engaging.

During the whole duration of the conference, we have enabled a virtual **Gather Town** for attendees to interact, socialize, meet new people and talk about research. Move your avatar around the 2D world with 8-bit graphics by using your keyboard's arrow keys.

To gain access to the virtual conference platforms please register at: https://cps-hcss.org.

CONFERENCE MATERIALS

Conference presentations will be available online at https://cps-hcss.org.

POLICIES

HCSS is committed to providing a safe and enjoyable event experience for all participants, and a welcoming environment for free discussion of ideas. We do not tolerate harassment of participants in any form during events or in any HCSS online space or social media. If you have any concerns about inappropriate behavior by any participant during the event, please contact the HCSS organizers:

File an Incident Report: https://cps-vo.org/hcss/incident-report

Email: hcss@cps-vo.org

Hopin chat: send a message to Katie Dey

Please review the complete code of conduct at: https://cps-vo.org/group/hcss/policies

PROGRAM AGENDA

PROGRAM AGENDA

MONDAY, MAY 3

THEME: Proof Robustness: decade-long experiences

TIME *(EDT)*	TITLE	SPEAKER	PAGE
1315 - 1330	**WELCOME & OPENING REMARKS**		
1330 - 1430	**KEYNOTE PRESENTATION** Robustness of formal verification of x86 microprocessors	*Anna Slobodova (Centaur)*	13
1430 - 1500	Proof Robustness in ACL2	*Eric Smith (Kestrel Institute)*	15
1500 - 1530	**BREAK**		
1530 - 1600	How to Improve the Robustness of Auto-active Program Proof Through Redundancy	*Yannick Moy (AdaCore)*	18
1600 - 1630	Proof Robustness in the seL4 Verification	**Gerwin Klein & **Rafal Kolanski (*Proofcraft & UNSW Sydney, **Proofcraft & seL4 Foundation)*	20
1630- 1645	**BREAK + FIRESIDE + CLOSING REMARKS FROM SESSION CHAIR**		

TUESDAY, MAY 4

PROGRAM AGENDA

THEME:
Holistic System Reasoning

TIME *(EDT)*	TITLE	SPEAKER	PAGE
1330 - 1430	**KEYNOTE PRESENTATION** Mitigating Emergent Computation: the need for new approaches in systems engineering	*Sergey Bratus (DARPA)*	21
1430 - 1500	Compositional verification of modular C programs using VST and VSU	*Lennart Beringer (Princeton University)*	23
	Formalizing, and Evaluating Checked C	*Michael Hicks (University of Maryland)*	24
	On Computing Relevant Parameters of Decision Functions	*Susmit Jha (SRI International)*	26
1500 - 1515	**Lightning Talk Fireside**		
1515 - 1530	**BREAK**		
1530 - 1600	Knowledge-Assisted Reasoning of Model-Augmented System Requirements	*Brendan Hall (Honeywell Advanced Technology)*	28
1600 - 1630	Safe Composition through Dynamic Feature Interaction Resolution	*Eunsuk Kang (Carnegie Mellon University)*	33
1630 - 1645	**BREAK**		
1645 - 1715	Automating Argumentation (for Overarching Properties) with Goal-directed Answer Set Programming	*Gopal Gupta (UT Dallas)*	35
1715 - 1730	**BREAK + FIRESIDE + CLOSING REMARKS FROM SESSION CHAIR**		

PROGRAM AGENDA

WEDNESDAY, MAY 5

THEME:
Morning: Formal Methods in DevOps
Afternoon: Reasoning About Security, Crypto, & Protocols

TIME *(EDT)*	TITLE	SPEAKER	PAGE
1330 - 1400	Retrofitting a type system onto a real world dynamic expression language	*Vaibhav Sharma, Bilal Khan (AWS)*	40
1400 - 1500	**KEYNOTE PRESENTATION** DoD Enterprise DevSecOps Initiative & Platform One	*Nicolas Challain (USAF)*	42
1500 - 1515	**BREAK**		
1515 - 1545	Continuous Integration and Formal Methods with Muse, Affix, and 5C	*Stephen Magill (MuseDev / Sonatype) Michael Hicks (University of Maryland)*	43
1545 - 1615	**BREAK + FIRESIDE**		
1615 - 1645	Automated Trust Analysis for Layered Attestations	*Ian Kretz (MITRE)*	45
1645 - 1715	CYBORG Cryptographic Security: Bluetooth, Signal, and Beyond	*Britta Hale (NPS)*	46
1715 - 1745	Zero-knowledge Proofs of Binary Exploitability	*Ben Perez (Trail of Bits)*	47
1745 - 1800	**BREAK + FIRESIDE + CLOSING REMARKS FROM SESSION CHAIR**		

THURSDAY, MAY 6

PROGRAM AGENDA

THEME:
Exploring Compositionality

TIME *(EDT)*	TITLE	SPEAKER	PAGE
1330 - 1430	**KEYNOTE PRESENTATION** Enabling Provable Security at Scale	*Neha Rungta* *(AWS)*	49
1430 - 1500	Briefcase Full of Proofs	*Darren Cofer* *(Collins Aerospace)*	50-
1500 - 1530	**BREAK**		
1530 - 1600	HAMR - High-Assurance Modeling and Rapid Engineering for Embedded Systems Using AADL	*John Hatcliff* *(Kansas State University)*	51
1600 - 1630	Verifiable Binary Lifting	*Joe Hendrix* *(Galois)*	53
1630 - 1645	**BREAK + FIRESIDE + CLOSING REMARKS FROM SESSION CHAIR**		
1645	**CONFERENCE ADJOURNED**		

CONFERENCE PRESENTATIONS

bold name denotes presenter

KEYNOTE PRESENTATION
ROBUSTNESS OF FORMAL VERIFICATION OF X86 MICROPROCESSORS

Anna Slobodova
Centaur

Industrial formal verification is challenging due to the continuous changes inherent in commercial design. Validation starts as soon as any part of a new design is available, and it needs to continue smoothly through post-silicon verification. Hence, verification must be adaptable to incomplete design and frequent changes. In addition, specification might change due to a feedback from back-end tools (layout, timing, array organization, power) and changes in the product market. Centaur Technology, an Austin, TX-based company that designs x86 compatible microprocessors, has been using formal verification since 2007. This effort was initiated by Centaur leadership through their contact with UT Professor, Warren Hunt. Hunt with his then PhD student, Sol Swords, proposed to make a case study using ACL2 system to analyze their arithmetic design. Their ability to find a very subtle bug in a mature design convinced Centaur's leadership to invest in a formal verification (FV) team that over time grew to four members. Over the thirteen years of their existence, FV group has become an indispensable part of Cenatur's design team. Unlike other hardware companies, Centaur analyzes their designs primarily using theorem proving, instead of relying on model-checking and SMT tools only. Although surprising to some, the ACL2 theorem prover provides a fine balance between proof automation and user guidance. The Centaur FV approach is fully integrated into their design flow. A suite of regression proofs can be trigerred by changes in the design, specification and/or proofs, or can be scheduled (nightly, weekly, etc.). FV team is then alerted by a failure that can be investigated and remediated. Formal microprocessor verification has been shown to be resilient to the continuous changes of the design under

development. We will describe the challenges of using formal verification in an industrial setting, and how we handle them using open-source and proprietary tools.

Anna Slobodova is the Lead of the Formal Verification group at Centaur Technology, Inc. Centaur's focus is on designing high-performance low-cost x86 microprocessors https://www.centtech.com/. Dr. Slobodova is interested in technology that can improve validation of complex industrial design (from hands-on theorem proving to automated procedures). She obtained a Masters degree and a PhD in Theoretical Computer Science (with a focus on Complexity Theory) at Comenius University, Bratislava, Slovakia. She started her career as a Research Scientist at Comenius University. She later moved to became an Assistant Professor at the University of Trier, Germany, and she was also appointed in the Fraunhoffer Institue for Telematik. In 1998, she joined Digital Equipment Corporation (later Compaq) in Massachussetts, USA, where she worked in a small group on one of the first sequential equivalence checkers in industry. When Compaq was purchased by Intel, she joined the formal verification group known as "Dragon Team". After seven years at Intel, Dr. Slobodova joined Centaur Technology, where she assumed responsibility for Centaur's formal verification team that had been started by a University of Texas professor. Additional professional staff has been added, and the FV team's work is now an indispensable part of the company's design and validation process. Dr. Slobodova has served on multiple program committees focused on formal methods including CAV, CPP, DATE and FMCAD. For FMCAD 2009, she served as the, organizational co-chair, and as the conference general co-chair in 2011. She presently serves on the FMCAD Steering Committee. She was a Test and Verification track co-chair of ICCD 2011, and FV topic co-chair for DATE 2020 and 2021.

PROOF ROBUSTNESS IN ACL2

Eric Smith
Kestrel Institute

This experience report builds on 20 years of maintaining proofs for the ACL2 theorem prover.

Keeping old proofs working has many benefits. It let us start using, extending, or adapting them any time, without painful modernization to get them working with the latest version of the prover. And proofs about new versions of software and hardware artifacts are easier if one has access to working proofs of the old versions. Having a large suite of working proofs also helps evaluate proposed changes to supporting libraries and to the prover itself.

The ACL2 Community maintains a large and growing repository of proof libraries on GitHub (github.com/acl2/acl2). It contains, very roughly, about 90,000 function definitions and about 174,000 theorems. These libraries are actively being extended and improved, at the rate of about 10 new commits per day. Kestrel Institute and other major users of ACL2 maintain their own large proof libraries. (Kestrel's internal proofs include many derivations of correct-by-construction software, many proofs about Java programs, etc.)

We endeavor to keep all of these proofs working, even as changes are made to 1) the precise definitions supporting the proofs (e.g., formal models, specifications, and deeply or shallowly embedded software or hardware artifacts), 2) the libraries of rules used in the proofs, 3) any supporting proof tools (connections to SMT solvers, custom theorem provers), and 4) ACL2 itself. Keeping proofs working requires infrastructure support, such as single commands to re-build large sets of proofs. Changes to the ACL2 libraries on Github must pass automated regression tests (building almost everything in the repository) before they are automatically merged into the master branch. Replaying all of our proofs takes a few hours, even on a machine with dozens of cores, but file-level dependency analysis prevents unaffected material from being rebuilt.

Automated build systems prevent "bit rot" (proof rot?), allowing new proof failures to be detected quickly. Diagnosing a proof that worked yesterday but fails today is often easy, since not much has changed. It can be much harder to un-break a proof after hundreds of changes have been made to the prover and its libraries.

Given all this, the ACL2 Community has developed an informal set of best practices for writing and maintaining robust proofs. ACL2 proofs are done by supplying hints to the prover, but certain constructs lead to brittle proofs and should be avoided when possible. These include 1) hints that depend on the detailed structure of the proof, and 2) hints that depend on the detailed structure of the logical terms involved. The former include hints attached to subgoals

(such as "Subgoal 3.4", indicating the fourth subgoal generated from the third top-level goal generated from the conjecture). These are brittle because the precise decomposition of the proof into subgoals is likely to change. In many cases, such hints can simply be attached to "Goal" instead. Examples of #2 above include hints that mention particular pieces of syntax, such as terms given to instantiate a lemma, or to split a proof into cases. These will break if function names change, or if arguments are added, removed, or reordered. Similarly, Proof Builder instructions (like "go to the third argument of the fifth hypothesis") can also be brittle. Theory hints ("turn this rule on for the entire proof") tend to be a bit more robust. It is often helpful to replace detailed hints with explicit rewrite rules that will simplify patterns everywhere they appear, in the current proof and in all future proofs. ACL2 encourages this style of development.

We also seek robustness in proofs that are generated by tools, ensuring they will work in any future context. Here, the key techniques include very tightly controlling the set of rules and definitions used, making induction schemes explicit, and disabling ACL2 proof techniques that are not expected to contribute to the proof.

On a higher level, we advocate a lightweight approach to structuring files and directories of proofs. This involves limiting what is included into a development, to limit the effect of changes. Crucially, we also try to limit what is exported by each file of proofs. For example, if a development requires arithmetic reasoning but is not itself about arithmetic, it should not force its users to include the arithmetic library that it happens to use. ACL2's "local include" construct supports this.

Still, sometimes a proof fails after a change. In such cases, we can almost always get it working again. Having access to a working version of the proof (in an older checkout of the repository), or at least to the prover's output from a working version, can be a great help. One can compare the human-readable commentary printed for the working and failing proofs. Things to check include the induction scheme used, whether the same definitions were expanded, and whether the same rules were used. If more detail is required, the proofs can be re-run in a verbose mode. Individual subgoals from the failing proof can be even submitted into the old session, to see if they can be proved in the old environment. If rules that applied in the old proof now fail to fire, ACL2's "monitor" feature can help determine why. If there are rules that fire in the new proof but did not fire in the old one, one might need to disable them, but a better approach might be to formulate additional rules that get the proof working again.

Dr. Eric W. Smith is the CEO of Kestrel Institute, where he does research in formal methods. He co-leads Kestrel's APT (Automated Program Transformations) project, which is developing proof-emitting program transformations for the ACL2 theorem prover. APT is currently being used to synthesize network protocol implementations and high-assurance monitors for machine learning algorithms. Dr. Smith also works on the Axe tool originally written for his Stanford Ph.D. thesis. Axe can lift Java bytecode and x86 binary programs into a logical representation which is then verified using Axe's high-performance prover and equivalence checker. A newly released version of Axe can verify large rank-1 constraint systems used in zero-knowledge proofs. Dr. Smith led Kestrel's DerivationMiner effort, which used program synthesis techniques (derivations and refinements) to construct correctness proofs of code found in large online repositories. He also led Kestrel's DARPA APAC effort to find malware in Android apps and formally prove the correctness of apps without malware.

HOW TO IMPROVE THE ROBUSTNESS OF AUTO-ACTIVE PROGRAM PROOF THROUGH REDUNDANCY

Yannick Moy
AdaCore

SPARK is a program proof environment for Ada programs. It is typically used to guarantee simple properties of medium-size critical programs, from thousands to tens of thousands of lines of code. The typical properties of interest range from absence of run-time errors to data invariants and correct sequencing of calls. The scope of a proof in SPARK is an individual function, which is in general much smaller than the complete program, from tens to hundreds of lines of code. While this reduced analysis scope greatly facilitates the initial establishment of proofs and their maintenance over time, in particular because it avoids scalability problems, it does not completely solve the issue of proof maintenance. Restricting our investigation to changes that don't impact specifications, there are two root causes for proof regressions: either the code has changed, or the tool has changed. Code changes that don't impact the specification are typical of small code fixes or code optimizations. Tool changes are incurred when upgrading to a newer version of the tools that offers new functionality and/or fixes problems in the tools. As part of developing the SPARK tools, we are well familiar with the latter, as it is a recurring cost to maintain the proof baseline of our testsuites. In both cases, restoring the proof may require changing the tool settings and/or modifying the ghost code used to nudge automatic provers towards a proof. Our accumulated experience over the years is that the redundancy provided by using multiple automatic provers and careful use of ghost code increases the robustness of such auto-active proofs. We will present experiments that we conducted to question these impressions, which provide quantitative results on how each type of redundancy contributes to the overall proof robustness. In these experiments, we ran public versions of the SPARK tool released between 2018 and 2020 on algorithms from the SPARK-by-Example[1] education project, the SPARKNaCl[2] rewrite of the cryptographic library TweetNaCl in SPARK and a functionally proved library of imperative red-black-trees in SPARK[3]. We enabled different provers and/or ghost code, to see how each type of redundancy was contributing to the robustness of the proof as we're changing the version of the tool. All results will be made public for others to reproduce our results or perform similar experiments.

References

1 https://github.com/tofgarion/spark-by-example
2 https://github.com/rod-chapman/SPARKNaCl
3 http://toccata.lri.fr/gallery/spark_red_black_trees.en.html

Yannick Moy is Static Analysis Lead at AdaCore. Yannick leads the development of SPARK, a software source code analyzer aiming at verifying safety/security properties of programs. He frequently talks about SPARK in articles, conferences, classes and blogs (in particular blog.adacore.com). Yannick previously worked on source code analyzers for PolySpace (now The MathWorks) and at Universite Paris-Sud.

PROOF ROBUSTNESS IN THE SEL4 VERIFICATION

***Gerwin Klein, **Rafal Kolanski**

**Proofcraft & UNSW Sydney, **Proofcraft & seL4 Foundation*

The seL4 proofs have seen sustained development, maintenance, and extension for over 15 years. They now measure more than a million lines of Isabelle/HOL proof script and therefore likely represent the largest machine-checked interactive proof on the planet. This talk will report on past and ongoing efforts to maintain robustness of this proof and code base. We will give examples of where these efforts were successful and the presented techniques worked well. We will also show examples of where proofs lacked robustness and these efforts failed or did not apply, together with our thoughts of what can be done in such cases. With the creation of the seL4 Foundation and increasing contributions to both code and proofs from open source collaborators as well as companies from the US and Europe, the importance of proof robustness in seL4 continues to increase. We will discuss the measures already in place as well as planned efforts to make verified seL4 development more agile and more suitable for a vibrant open source community. These include adjustments to our automatic continuous integration and deployment pipeline for the high-assurance artifact of the seL4 kernel itself, as well as the ecosystem of components and platforms around it, which vary in the degree of assurance they provide.

Gerwin Klein, Chief Scientist and co-founder at Proofcraft, and Conjoint Professor at UNSW Sydney, is the architect of the seL4 microkernel verification. He planned, led, and executed the initial verification of seL4 as well as many of its extensions, and continues to be actively involved in all aspects of the project.

Rafal Kolanski Chief Engineer and co-founder at Proofcraft, has been leading the seL4 Proof Engineering team since 2015. He is responsible for many of the modern proof engineering techniques and mechanisms introduced into the seL4 verification since then. He was a member of the initial seL4 verification and brings with him more than 15 years of experience in large-scale formal verification

KEYNOTE PRESENTATION

MITIGATING EMERGENT COMPUTATION: THE NEED FOR NEW APPROACHES IN SYSTEMS ENGINEERING

Sergey Bratus
DARPA

Modern computing systems demonstrate strong propensity for unintended, emergent computations and the related unintended, emergent programming models that enable or amplify cyber-attacks. Computing mechanisms built for a particular purpose and with particular intended models of execution in mind prove to be capable of executing unintended computing tasks outside of their original specification and their designers and programmers' mental models. The Spectre family of attacks on CPU microarchitectures is one recent example, with more examples provided by operating systems kernels and standard runtime environments.

Today, we start examining systems for signs of emergent behavior, with methods such as fuzz-testing, only after they are fully built. However, recent research strongly suggests that a system's exploitability and propensity for emergent execution arise, and can also therefore be mitigated, already at the design stage.

The talk will discuss the challenges in systems engineering, formal methods, as well as in program analysis and modeling to construct systems of systems in which emergent computation is mitigated.

Dr. Sergey Bratus joined DARPA as a program manager in January 2018. His research interests include computer security and intrusion analysis.

Dr. Bratus joins DARPA from Dartmouth College, where he has worked since 2002, initially as a post-doctoral research associate and most recently as a research associate professor in the Computer Science department. He has made significant technical contributions in the area of language-theoretic security, which aims to eliminate broad classes of input-driven exploitable vulnerabilities. Dr. Bratus began his professional career at BBN Technologies, where he worked on natural language processing.

Dr. Bratus holds a Doctor of Philosophy degree in mathematics from Northeastern University. He has published over 100 technical papers, and his research has received several awards, most notably the 2012 BlackHat Pwnie award for Most Innovative Research. Dr. Bratus also earned best paper awards at the WOOT 2011, IJSSE 2010, and TRUST 2008 conferences.

COMPOSITIONAL VERIFICATION OF MODULAR C PROGRAMS USING VST AND VSU

Lennart Beringer
Princeton University

Modularity and representation hiding are key principles of modern software engineering. Type theoretic notions such as parametricity and existential abstraction provide conceptual underpinnings, but have hitherto found limited application outside the world of functional programming. Our goal is to realize these notions in the setting of C, for procedural programming styles using abstract data types as well as object-oriented programming styles that are based on function pointers and dynamic dispatch. The talk will outline our experience in extending the Verified Software Toolchain, a Coq-embedded interactive verification framework for CompCert-Clight, with support for modular data abstraction at the granularity level of compilation units, complemented by a mostly-automated linking system that allows individually verified components to be assembled to fully linked C programs that are subject to VST's foundationally proven soundness guarantee.

Lennart Beringer is a Research Scholar at Princeton University and Associate Director of the NSF Expedition in Computing "The Science of Deep Specification". His main research focus over the past decade has been the Verified Software Toolchain (VST), a separation-logic based foundational program verification framework for the C language. He is particularly interested in compositionality aspects of verification methodologies; as a participant of DARPA-HACMS, he developed Compositional CompCert and top-to-bottom proofs of cryptographic primitives that connect formal verification of cryptographic security of abstract primitives with proofs of implementation correctness for concrete open-source code bases. His present research topics include reasoning about of object-oriented programming styles and formal verification of software-defined networking using the P4 language.

FORMALIZING AND EVALUATING CHECKED C

Michael Hicks
University of Maryland

The C programming language remains extremely popular despite the emergence of new, modern languages. Unfortunately, C programs lack spatial memory safety, which has long made them susceptible to a host of devastating vulnerabilities, including buffer overflows and out-of-bounds reads/writes. Checked C[1], developed open-source by Microsoft starting in 2015, extends the C language with new types and annotations whose use can ensure a program's spatial safety. Importantly, Checked C supports development that is incremental and compositional. Individual code regions (e.g., functions or whole files) designated as checked are sure to enforce spatial safety, a property which is preserved via composition with other checked regions. But not all regions must be checked: Checked C's checked pointers are binary-compatible with legacy pointers, and may coexist in the same code, which permits a deliberate refactoring process. In prior work[2], we developed a core formalization of Checked C in which we proved that *checked code cannot* be blamed: any spatial safety violation can only be attributed to code that is not yet fully checked.

Recently, we have been developing a *NULL-terminated (NT) checked pointer type* in Checked C, whose syntax is `nt_array_ptr<T>` (for any type *T*). Programmers familiar with C's standard string library will be quite familiar with NT pointers. But *spatially safe* NT pointers are not present in modern type-safe languages (such as Java, Scala, Haskell, Rust, etc.), which typically pair dynamically-sized buffers with metadata to indicate the buffer's length and capacity. While some prior safe-C implementations (such as CCured, Cyclone[3], and Deputy) have supported NT pointers, no prior work of which we are aware has formalized and proved them safe, nor has it assessed their performance against non-NT alternatives. We fill these gaps in our work.

Normal pointers to arrays have a fixed length, e.g., `array_ptr<char> p:count(n)` indicates that `p` has length `n` (where `n` is another variable). Declaration `nt_array_ptr<char> p:count(n)` indicates that `p`' s length is at least `n`, but further capacity is present if `p[n] != NULL`. As such, in statements if `(*p) { ... }` and `p`' s type is `nt_array_ptr<T> count(0)`, then the compiler type-checks the branch ... with `p` typed as `nt_array_ptr<T> count(1)`, i .e., 1 more than it was, since the character at the current-known length is non-NULL. Such *flow sensitivity* supports checking common idioms involving pointer arithmetic, e.g., as in the implementation of `strlen`. An `nt_array_ptr<T> count(n)` can be safely converted to an array_ptr<T> count(n-1),

for better library code reuse, as well as the reuse of the bounds-check elimination optimization in the Checked C compiler itself.

In our work we extend our earlier formalism with checked NT pointers and re-prove our compositional proof of blame (done i n the Coq proof assistant); our formalism also includes lightweight dependent and function types, and other extensions. We have also implemented checked NT pointers in the Checked C compiler, and evaluated their effectiveness on a series of benchmarks; we use these benchmarks to show the advantages of our support compared to refactoring the code to use normal arrays. Our implementation, code, and benchmarks are (or will be) available i n the Checked C Github repository[4].

References

1 https://www.microsoft.com/en-us/research/project/checked-c/
2 https://www.cs.umd.edu/~mwh/papers/ruef18checkedc-incr.html
3 http://cyclone.thelanguage.org/
4 https://github.com/microsoft/checkedc-clang)

Michael W. Hicks is a Professor in the Computer Science Department at the University of Maryland, where he has been since 2002, and the CTO of Correct Computation, Inc. (CCI), a role he has held since late 2018.

He is a leading member of the software security community, having carried out diverse and award-winning research published in top venues covering computer security, programming languages, systems, and software engineering. He was the first Director of the University of Maryland's Cybersecurity Center (MC2), and was the elected Chair of ACM SIGPLAN, the Special Interest Group on Programming Languages; he now edits its blog, PL Perspectives, at https://blog.sigplan.org. Over his career, he has published more than 125 peer-reviewed scientific articles, and his research has twice won the NSA's Best Scientific Cybersecurity Paper competition; he is the only two-time winner. A key recent focus at UMD and CCI has been to explore how to make legacy software written in low-level languages, like C, more secure. He has been collaborating on the Checked C project, carrying on a nearly 20-year arc of work that started with the Cyclone safe programming language, which itself was a strong influence on Rust, a next-generation language. He has also currently looking at synergies between cryptography and programming languages; techniques for better random (fuzz) testing and probabilistic reasoning; and high-assurance tools and languages for quantum computing.

ON COMPUTING RELEVANT PARAMETERS OF DECISION FUNCTIONS

Susmit Jha, Patrick Lincoln
SRI International

To facilitate reasoning about modern cyber-physical systems, we wish to determine which inputs or parameters of a black-box function are relevant to the output. Adaptive software components rely on a plethora of decision-making and control functions for reconfiguration and adaptation to different contexts. Such functions can be manually written or obtained using machine learning. Here we use only oracle (black box) access to the decision function, making our approach more widely applicable, scalable, and free from the classical requirement of meticulous formal modeling. Our approach addresses two central challenges of high-assurance design: robust proofs where the correctness argument for a single context or configuration can be extended to a set of contexts or configurations, and compositional design where contexts for different configuration decisions can be automatically identified. We present two different algorithms using Hamming-distance based search and random-walk in Boolean hypercube for solving the problem of computing relevance and present theoretical results on the runtime complexity and probabilistically approximately correct (PAC) correctness guarantees. Finally, we present experimental results from compositional design of cyberphysical systems to demonstrate the value of our approach.

Dr. Susmit Jha, a Principal Computer Scientist in the Computer Science Laboratory at SRI International, is an expert in neurosymbolic approaches to Artificial Intelligence (AI) with applications to high-assurance autonomous cyberphysical systems. Dr. Jha is currently PI/co-PI on several NSF, DARPA, IARPA, and ARL projects on building creative, trustworthy, and resilient artificial intelligence methods, including DARPA Assured Autonomy, DARPA Symbiotic Design of Cyberphysical Systems, DARPA Intent-driven Design of Adaptive Systems, IARPA TrojAI, NSF Self-improving Cyberphysical Systems, NSF Duality-based Program Synthesis, and ARL Principles of Robust Learning in IoBTs CRA. Before joining SRI, Dr. Jha was a Staff Scientist at United Technologies Research Center (Raytheon), Berkeley, and a Research Scientist at Intel. At UTRC, Dr. Jha performed on several DoD projects on AI and autonomy, including DARPA Aircraft Labor In-cockpit Automation Systems, DARPA Communications under Contested Environments, and ONR Human-Machine Systems. At Intel, Dr. Jha developed approaches to AI-driven architecture design, which influenced the on-chip-interconnect design on a 14nm SoC design. Dr. Jha completed his Ph.D. from UC Berkeley in 2011. His thesis work on the automated formal synthesis of safe controllers and programs received Leon O. Chua Award for outstanding achievement in nonlinear sciences. His oracle-guided approach to program synthesis, which combines formal reasoning with machine learning from samples, received the 10-year Most Influential Paper award at IEEE/ACM International Conference on Software Engineering (ICSE), 2020. Dr. Jha has over 55 peer-reviewed publications and over 2000 citations.

Dr. Patrick Lincoln, Ph.D., is Vice President, Information and Computing Sciences and the director of the Computer Science Laboratory at SRI International. He is also the executive director of SRI's program for the Department of Homeland Security's Cyber Security Research and Development Center and director of the SRI Center for Computational Biology. Dr. Lincoln leads research in the fields of formal methods, computer security and privacy, computational biology, scalable distributed systems, and nanoelectronics. He has led multidisciplinary groups conducting high-impact research projects in symbolic systems biology, scalable anomaly detection, exquisitely sensitive biosensor systems, strategic reasoning and game theory, and privacy-preserving data sharing. He has published dozens of influential papers, holds several patents, has served on scientific advisory boards for private and publicly held companies, nonprofits, and government agencies and departments. Dr. Lincoln holds a Ph.D. in computer science from Stanford University and a B.Sc. in computer science from MIT. He has previously held positions at MCC, Los Alamos National Laboratory, and ETA Systems.

KNOWLEDGE-ASSISTED REASONING OF MODEL-AUGMENTED SYSTEM REQUIREMENTS

*__Brendan Hall__, †Sarat Chandra Varanasi, ††Jan Fiedor, §Joaquin Arias, †Kinjal Basu, †Fang Li, *Devesh Bhatt, *Kevin Driscoll, †Elmer Salazar, †Gopal Gupta

**Honeywell Advanced Technology, †The University of Texas at Dallas,*
††Honeywell International s.r.o & Brno Univ. of Technology, §Universidad Rey Juan Carlos

Developing effective requirements is crucial for success in building a system. The more complete, consistent, and feasible the requirements, the fewer problems system developers will encounter later. Current automation of requirements engineering tasks attempt to ensure completeness, consistency, and feasibility. However, such automated support remains limited. In this work, we present novel automated techniques for aiding the development of model-augmented requirements that are complete (to the extent possible), consistent, and feasible--where consistency and feasibility are validated. Thus, we can have more confidence in the requirements. We limit ourselves to requirements for cyber-physical systems, particularly those in avionics. We assume that requirements are generated within the MIDAS (Model-Assisted Decomposition and Specification)[4] environment, and are expressed in CLEAR (Constrained Language Enhanced Approach to Requirements) [3], a constraint natural requirement language.

Our main contribution in this work is to show how the Event Calculus[5] (EC) and Answer Set Programming (ASP)[2] can be used to formalize constrained natural language requirements for cyber-physical systems and perform knowledge assisted reasoning over them. ASP is a logic-based knowledge representation language that has been prominently used in AI. Our work builds upon recent advances made within the s(CASP) system[1], a query-driven (or goal-directed) implementation of predicate ASP that supports constraint solving over reals, permitting the faithful representation of time as a continuous quantity. The s(CASP) system permits the modeling of event calculus elegantly and directly. A major advantage of using the event calculus--in contrast to automata and Kripke structure-based approaches--is that it can

directly model cyber-physical systems, thereby avoiding "pollution" due to (often premature) design decisions that must be made in the other modeling formalisms. The event calculus is a formalism--a set of axioms--for modeling dynamic systems and was proposed by artificial intelligence researchers to solve the *frame problem*[5]. The primary goal of this work is to explore how constrained natural language requirements, specified within MIDAS [4] using the CLEAR notation, can be automatically reasoned about and analyzed using the event calculus and query-driven answer set programming. Specifically, we explore:

1. How to systematically capture design and intent within the MIDAS framework.
2. How ASP-based model checking (over dense time) can validate specified system behaviors wrt system properties.
3. How application of *abductive reasoning* can extend ASPbased model checking to incorporate domain knowledge and real-world/environmental assumptions/ concerns.
4. How knowledge-driven analysis can identify typical requirement specification errors, and/or requirement constructs which exhibit areas of potential/probable risk.

The talk will be organized as follows. We will give motivation for our work and discuss the importance of writing requirements that are consistent and complete. We will present how MIDAS enables a formal flow-down of functional intent through different stages of design refinement. We will summarize the two faces of requirements (outward and inward facing), as they support validation and verification objectives (respectively). We will then discuss the enabling background technologies (EC and ASP), before presenting how they can be integrated within the MIDAS platform to support our goals. We will illustrate our approach using an altitude alerting case study from an actual aerospace system and discuss adjacent real-world examples to show how one can use generalized knowledge within other system contexts. We will illustrate requirement defect discovery using s(CASP) for property-based model-checking as well as discuss how more general knowledge of potential requirements defects may detect defects that traditional techniques may not be able to find.

References

[1] Joaqu'in Arias et al. "Constraint answer set programming without grounding". In: *TPLP* 18(3-4):337-354 (2018).

[2] M. Gelfond and Y. Kahl. Knowledge representation, reasoning, & design of intelligent agents: The answer-set programming approach. *Cambridge Univ. Press*, 2014.

[3] B. Hall, D. Bhatt, et al. "A CLEAR Adoption of EARS". In: *IEEE EARS Workshop* . 2018, pp. 14-15.

[4] B. Hall, J. Fiedor, and Y Jeppu. "Model Integrated Decomposition and Assisted Specification (MIDAS)". In: *INCOSE Int'l Symp.* Vol. 30(1):821-841. Wiley.

[5] Murray Shanahan. "The event calculus explained". In: *Artificial intelligence today* . Springer, 1999, pp. 409-430.

Brendan Hall is an Engineer Fellow with Honeywell Aerospace Advanced Technology. He has over 25 years of experience within safety-relevant domains and has successfully executed internal and externally funded (NASA, AFRL, and FAA) research programs. His research interests include fault-tolerant distributed system architectures, model-based system and safety engineering, formal and property-based design assurance, and the industrial application of formal methods and logic programming. Mr. Hall holds over 45 patents and has published numerous technical papers within the fields of fault-tolerant distributed systems, requirement engineering, applied formal methods, and model-based engineering.

Sarat Chandra Varanasi is a PhD student in the CS Department at UT Dallas. Sarat Chandra's research work involves automatically synthesizing concurrent programs using commonsense knowledge about concurrency. Given a sequential program for a pointer data structure (e.g., insert a node, delete a node), the goal of this research is to automatically synthesize its concurrent version, just like a human would. To accomplish this, knowledge that models concurrency, shared memory, and how pointer data structures behave is used (represented as axioms in answer set programming). Sarat also has several years of industry experience and interests are in distributed computing, concurrency, computational logic, and automated reasoning.

Jan Fiedor is a Model-based development engineer at Honeywell International s.r.o. and an external researcher at Brno University of Technology. He obtained his Ph.D. in the area of software safety analysis, focusing on dynamic analysis of concurrent programs, where he is continuing his work in various European projects. At Honeywell, he is broadening his focus to model-based system and safety engineering, applying his knowledge from the software world to the general system context. He strongly believes that the collaboration between academia and industry will move the world forward.

Joaquin Arias is an assistant professor at Universidad Rey Juan Carlos in Madrid (URJC). He obtained his M.Arch. in Architecture in 2002, his B.Sc./M.Sc. degrees in Informatics in 2014/2015, and received a Ph.D. in Computer Science in 2020, from the Universidad Politecnica de Madrid. His principal area of expertise is the development and application of advanced evaluation techniques for (non-)monotonic reasoning using rules and constraints. His background includes the implementation of an abstract interpreter based on tabled constraint logic programming, and the development (in collaboration with the IMDEA Software Institute and the University of Texas at Dallas) of s(CASP) a novel non-monotonic reasoner that evaluates constraint answer set programs. He is currently a fellow of the Artificial Intelligence Research Group at the URJC and his research interests are in the areas of Event Calculus, XAI, and Building Information Modeling.

Kinjal Basu is a computer science Ph.D. student working under Prof. Gopal Gupta on closed domain question-answering and commonsense reasoning. Currently, he is leading the CASPR team from UT Dallas at the 'Alexa Prize: Social Bot Grand Challenge 4' competition organized by Amazon. His research interest lies in Logic Programming, Natural Language Understanding, and Machine Learning, and his recent papers are in the area of textual and visual question answering and conversational agents. Kinjal also acquired real-world data science project experience while working as an intern at Intuit. During his Bachelor's in Information Technology, he worked on several NLP projects (e.g., Text Similarity) and also interned at IIT Kharagpur for NLP research.

Fang Li is a Computer Science PhD candidate in Prof. Gopal Gupta's lab. Fang's research interests are explainable AI, commonsense reasoning, answer set programming, and logic programming. Fang got his Master' in Applied Mathematics & Computer Science and graduated with honors from University of Central Oklahoma. In the past few years he has conducted research on human computer interaction, IoT, autonomous vehicles, AI planning, and answer set programming implementation. He cofounded a company as a CTO called Turning Systems that won an award in an entrepreneurship competition.

Devesh Bhatt received a B.Tech. in electrical engineering from Indian Institute of Technology Kanpur in 1975 and Ph.D. in EE/CS from Stony Brook University in 1980. He is currently a Senior Fellow at Honeywell Aerospace Laboratories. His research interests include verification of complex safety-critical systems and software, including requirements engineering, analysis, and testing techniques, and certification approaches for use of new technologies in autonomous applications.

Kevin Driscoll is a Fellow at Honeywell Labs with 49 years' experience in safety and security critical systems -- including the aspects of hardware, software, and system design. He currently is leading an $8M effort to create assurance cases. Other recent programs include "Considerations in Assuring Safety of Increasingly Autonomous Systems", "Validation and Verification of Safety-Critical Integrated Distributed Systems", and "Cyber Safety and Security for Reduced Crew Operations". He was a main author of several embedded system network standards and the FAA network selection handbook. Kevin is a consultant to other Honeywell divisions for dependable computing, electronic architectures, and data networks. Kevin has over 50 patents and numerous publications covering safety and security critical real-time systems.

Elmer Salazar is an assistant professor of instruction at the University of Texas at Dallas. Though teaching full time, he dedicates time to research, and works with the ALPS research group. His research interests include non-monotonic computational logic, the application of computational logic to AI and common sense reasoning, and the interaction of machine learning techniques (neural networks) and computational logic. He is the co-designer of the s(ASP) language (a non-monotonic, ungrounded, goal-directed logic programming language based on stable model semantics), and recently has been working on using proof trees generated by proof-theoretic execution to automatically find possible problems in ASP code based on some requirements encoded as predicates.

Gopal Gupta is a professor of computer science at the University of Texas at Dallas where he holds the Erik Jonsson chair. His areas of research interest are in automated reasoning, logic programming, machine learning, programming languages, and assistive technology. He has published extensively in these areas. His group has also authored many software systems, many of which are publicly available. His research work has also resulted in commercial software systems that have formed the basis of two startup companies. His group has won several best-paper awards as well as the ICLP 2016 most influential paper award for their work on co-inductive logic programming. Prof. Gupta's current research is funded by the NSF and DARPA. His lab is one of nine labs worldwide currently competing in the Amazon Alexa Socialbot competition where the goal is to build an Alexa skill that can hold a casual conversation with a random user for at least 20 minutes. He obtained his MS & PhD degrees from UNC Chapel Hill and his B.Tech. in Computer Science from IIT Kanpur, India.

SAFE COMPOSITION THROUGH DYNAMIC FEATURE INTERACTION RESOLUTION

Eunsuk Kang
Carnegie Mellon University

The *feature interaction* problem occurs when two or more independently developed components interact with each other in unexpected ways, causing undesirable effect on the system performance and safety. For instance, a pair of safety features in a vehicle may attempt to send conflicting acceleration commands to the engine controller, potentially violating a safety requirement that would have been satisfied if each feature had existed in isolation. Feature interactions are major obstacles to building large, complex systems out of heterogenous components, and pose new challenges in emerging cyber-physical systems (CPS), such as intelligent vehicles, unmanned aerial vehicles (UAVs), and the Internet-of-Things (IoT).

In this talk, I will describe our on-going work on techniques for safely managing and resolving undesirable interactions between CPS components [1, 2]. I will first introduce the state-of-the-art methods for managing feature interactions and argue that the existing approaches are not sufficient to deal with the highly dynamic, evolving nature of modern CPS domains. I will then describe a new type of approach called *context-driven resolution* that leverages techniques in runtime verification and synthesis to dynamically detect and resolve undesirable interactions. I will describe how our approach is capable of resolving interactions even when (1) the system evolves over time with newly added or modified features, and (2) none of the conflicting features may be satisfactory with respect to the overall system safety. I will demonstrate our approach using case studies on safety features in autonomous drones and intelligent vehicles. I will also present some of the remaining challenges and future directions towards enabling safe, seamless composition of heterogenous CPS.

References:

[1] Gafford, B., Durschmid, T., Moreno, G.A., Kang, E.: Synthesis-based resolution of feature interactions in cyber-physical systems. In: IEEE/ACM International Conference on Automated Software Engineering (ASE) (2020)

[2] Raghavan, S.G., Watanabe, K., Kang, E., Lin, C., Jiang, Z., Shiraishi, S.: Property-driven runtime resolution of feature interactions. In: International Conference on Runtime Verification (RV) (2018)

Eunsuk Kang is an Assistant Professor in the Institute for Software Research, School of Computer Science at Carnegie Mellon University. His research interests include software engineering and formal methods, with applications to system safety and security. His expertise is in leveraging formal modeling techniques, design methodologies, and automated verification to construct secure and reliable software and cyber-physical systems (CPS), and he has applied his work to a diverse range of systems, including intelligent vehicles, unmanned aerial vehicles (UAVs), medical devices, water treatment plants, and mobile applications. For his work, he has received two ACM Distinguished Paper Awards (FSE 2016, ICSE 2015) and a Best Paper Award (at IEEE/ACM Internet of Things Design and Implementation Conference, 2017).

AUTOMATING ARGUMENTATION (FOR OVERARCHING PROPERTIES) WITH GOAL-DIRECTED ANSWER SET PROGRAMMING

***Gopal Gupta,** †Joaquíin Arias, *Kinjal Basu, *Zhuo Chen, §Kevin Driscoll, *Serdar Erbatur, §Brenden Hall, *Fang Li, *Elmer Salazar, *Farhad Shakerin, *Sarat Varanasi, *Huaduo Wang

**The University of Texas at Dallas, †Universidad Rey Juan Carlos, §Honeywell Corp*

Formalizing the human thought process has been considered very difficult. The field of *informal logic* has been developed in recognition of this difficulty where an argument is interpreted as an attempt to present evidence for a conclusion. Holloway and Wasson have developed a primer to establish the terms, concepts, principles, and uses of arguments to reason about overarching properties [4]. We argue that recent advances in automated reasoning, especially incorporation of *negation as failure* (NAF), facilitate the formalization of the human thought process. These advances help formalize concepts that were hitherto thought of as impossible to formalize. We show how the paradigm of *answer set programming* (ASP), that incorporates NAF, can be used to formalize *all* the concepts presented in Holloway and Wasson's primer on argument. The main application of our work is in automated software systems certification and assurance. The goal is to represent the thought process of a human software certifier using ASP. ASP has been shown to be a highly suitable paradigm for knowledge representation and reasoning. It has been applied to many intelligent tasks including visual and natural language question answering [2] and emulating advanced human expertise (where it can outperform the human expert) [3].

In the overarching properties method, a human certifier starts with the goal of establishing intent, correctness and innocuity. Next, the certifier will have to construct *arguments* to prove these goals. An example argument maybe the following: a software is correct, if each of its components are correct and all interactions between the components are correct. Arguments will have to be made recursively then to establish the correctness of each component/interaction. Eventually, this chain of argument ends in a *belief* where a belief could be

based on hard *evidence* (fact) or an *assumption*. Arguments may also be defeated: a defeater provides support for not believing an argument. Thus, we use reasoning process that reaches a conclusion using arguments and beliefs. We show how a conclusion can be mapped to an ASP *query*, an argument to an ASP *rule*, an evidence to an ASP *fact*, an assumption to an *abducible*, and a defeater to a *negated goal*. An argument thus can be coded in ASP and the resulting program can be run on our goal-directed s(CASP) system [1] to *automatically* establish a conclusion in presence of evidence, assumptions and defeaters. In the talk we will present our general methodology and illustrate it with several non-trivial examples.

References

[1] Joaquín Arias, Manuel Carro, Elmer Salazar, Kyle Marple & Gopal Gupta (2018): Constraint answer set programming without grounding. TPLP 18(3-4), pp. 337-354.

[2] Kinjal Basu, Sarat Chandra Varanasi, Farhad Shakerin, Joaquin Arias & Gopal Gupta: Knowledge driven Natural Language Understanding of English Text and its Applications. In: Proc. of AAAI 2021.

[3] Zhuo Chen, Elmer Salzar & Others (2018): An AI-Based Heart Failure Treatment Adviser System. IEEE journal of translational engineering in health and medicine 6, 2800810.

[4] C. Michael Holloway & Kimberly S. Wasson (2020): A Primer on Argument. Available at https://shemesh.larc.nasa.gov/people/cmh/cmhpubs.html.

Gopal Gupta is a professor of computer science at the University of Texas at Dallas where he holds the Erik Jonsson chair. His areas of research interest are in automated reasoning, logic programming, machine learning, programming languages, and assistive technology. He has published extensively in these areas. His group has also authored many software systems, many of which are publicly available. His research work has also resulted in commercial software systems that have formed the basis of two startup companies. His group has won several best-paper awards as well as the ICLP 2016 most influential paper award for their work on co-inductive logic programming. Prof. Gupta's current research is funded by the NSF and DARPA. His lab is one of nine labs worldwide currently competing in the Amazon Alexa Socialbot competition where the goal is to build an Alexa skill that can hold a casual conversation with a random user for at least 20 minutes. He obtained his MS & PhD degrees from UNC Chapel Hill and his B.Tech. in Computer Science from IIT Kanpur, India.

Kinjal Basu is a computer science Ph.D. student working under Prof. Gopal Gupta on closed domain question-answering and commonsense reasoning. Currently, he is leading the CASPR team from UT Dallas at the 'Alexa Prize: Social Bot Grand Challenge 4' competition organized by Amazon. His research interest lies in Logic Programming, Natural Language Understanding, and Machine Learning, and his recent papers are in the area of textual and visual question answering and conversational agents. Kinjal also acquired real-world data science project experience while working as an intern at Intuit. During his Bachelor's in Information Technology, he worked on several NLP projects (e.g., Text Similarity) and also interned at IIT Kharagpur for NLP research.

Zhuo Chen's Ph.D. work focused on the development of a system for recommending heart failure treatment based on automating commonsense reasoning with answer set programming. The system was developed in collaboration with cardiologists. He subsequently worked as a postdoctoral researcher at the University of Texas at Dallas where he continued his work on automation of commonsense reasoning. In the past, Zhuo has also worked in mission-critical software system development and implementation. Aside from Answer Set Programming, Zhuo also enjoys programming in Scala and Go and currently works as a backend developer for Kinzoo.

Elmer Salazar is an assistant professor of instruction at the University of Texas at Dallas. Though teaching full time, he dedicates time to research, and works with the ALPS research group. His research interests include non-monotonic computational logic, the application of computational logic to AI and common sense reasoning, and the interaction of machine learning techniques (neural networks) and computational logic. He is the co-designer of the s(ASP) language (a non-monotonic, ungrounded, goal-directed logic programming language based on stable model semantics), and recently has been working on using proof trees generated by proof-theoretic execution to automatically find possible problems in ASP code based on some requirements encoded as predicates.

Farhad Shakerin is an experienced software engineer from the railway signaling industry. He contributed to the development of safety-critical systems for ten years before joining the graduate program at The University of Texas at Dallas. In 2020, he received a Ph.D. degree in Computer Science. His dissertation contributed to the field of explainable AI and logic-based natural language understanding. He is currently a senior software engineer at Microsoft Azure Cognitive Service.

Serdar Erbatur is an assistant professor of instruction at UT Dallas. He received his PhD degree from University at Albany (SUNY) in 2012. He has held postdoc positions in University of Verona (Italy), LMU Munich (Germany) and IMDEA Software Institute (Spain). He also visited INRIA Nancy (France) and Technical University of Valencia (Spain) for short terms. His current research is centered on automated reasoning and type-based program analysis with the overarching goal of applying it to formal verification (of programs and protocols). Currently, he is working on two ongoing research projects; (i) logical problems for protocol verification and (ii) the GuideForce project, funded by the German Research Foundation, focused on developing a program analysis tool for checking if a given Java program follows desired best programming practices (guidelines).

Fang Li is a Computer Science PhD candidate in Prof. Gopal Gupta's lab. Fang's research interests are explainable AI, commonsense reasoning, answer set programming, and logic programming. Fang got his Master' in Applied Mathematics & Computer Science and graduated with honors from University of Central Oklahoma. In the past few years he has conducted research on human computer interaction, IoT, autonomous vehicles, AI planning, and answer set programming implementation. He cofounded a company as a CTO called Turning Systems that won an award in an entrepreneurship competition.

Huaduo Wang is a PhD student in the ALPS lab led by Prof. Gopal Gupta. His current research focus on textual based QA, commonsense reasoning. His research interest lies in logic programming, natural language processing and understanding. His past work and research experience were in computer systems, distributed systems and autonomous vehicles. For his Master's research, he worked on system kernel project. He has interned as a backend SDE at Baidu as well as worked as a full-time software engineer for 3 years in the banking industry.

Joaquín Arias is an assistant professor at Universidad Rey Juan Carlos in Madrid (URJC). He obtained his M.Arch. in Architecture in 2002, his B.Sc./M.Sc. degrees in Informatics in 2014/2015, and received a Ph.D. in Computer Science in 2020, from the Universidad Politecnica de Madrid. His principal area of expertise is the development and application of advanced evaluation techniques for (non-)monotonic reasoning using rules and constraints. His background includes the implementation of an abstract interpreter based on tabled constraint logic programming, and the development (in collaboration with the IMDEA Software Institute and the University of Texas at Dallas) of s(CASP) a novel non-monotonic reasoner that evaluates constraint answer set programs. He is currently a fellow of the Artificial Intelligence Research Group at the URJC and his research interests are in the areas of Event Calculus, XAI, and Building Information Modeling.

Brendan Hall is an Engineer Fellow with Honeywell Aerospace Advanced Technology. He has over 25 years of experience within safety-relevant domains and has successfully executed internal and externally funded (NASA, AFRL, and FAA) research programs. His research interests include fault-tolerant distributed system architectures, model-based system and safety engineering, formal and property-based design assurance, and the industrial application of formal methods and logic programming. Mr. Hall holds over 45 patents and has published numerous technical papers within the fields of fault-tolerant distributed systems, requirement engineering, applied formal methods, and model-based engineering.

Kevin Driscoll is a Fellow at Honeywell Labs with 49 years' experience in safety and security critical systems -- including the aspects of hardware, software, and system design. He currently is leading an $8M effort to create assurance cases. Other recent programs include "Considerations in Assuring Safety of Increasingly Autonomous Systems", "Validation and Verification of Safety-Critical Integrated Distributed Systems", and "Cyber Safety and Security for Reduced Crew Operations". He was a main author of several embedded system network standards and the FAA network selection handbook. Kevin is a consultant to other Honeywell divisions for dependable computing, electronic architectures, and data networks. Kevin has over 50 patents and numerous publications covering safety and security critical real-time systems.

RETROFITTING A TYPE SYSTEM ONTO A REAL WORLD DYNAMIC EXPRESSION LANGUAGE

Vaibhav Sharma, Bilal Khan
Amazon Web Services

Customers use AWS IoT Events to monitor their equipment for failures. To do so, they create an event detector, modeled as a state machine, in AWS IoT Events to trigger actions when such failures occur. The state machine consists of states and transitions predicated on conditions written in the IoT Events expression language. Each state machine takes an input value sent by a customer's IoT device, substitutes input values in its expressions, evaluates them, and triggers other AWS services as its output.

Customers of AWS IoT Events create and edit their detector model in the console. As they update the detector model, their work is checked for common errors via a troubleshooting "Analysis" feature integrated into the console. This feature allows customers to catch syntactic, missing data, as well as type errors directly on the console, where the customer is editing their detector model. Each reported error and warning guides the customer to the location in the detector model where the potential issue exists. This allows customers to understand and fix the issue in the same console page where they are editing the detector model.

One category of errors and warnings reported during this edit/analyze/edit cycle is type errors. The type checker used by the Analysis feature dynamically creates type inference rules from the type rules of the IoT Events expression language. It infers types for terms in the customer's detector model and reports not only type errors but also inferred types for terms in their detector model. The Analysis feature helps customers understand and fix type errors without asking them to (1) declare types for their inputs, (2) understand the type rules for the expression language, (3) understand the type inference rules used by the type checker. Customers spend no additional effort to run the type checker or fix its reported issues, thereby making the type checker "disappear" in the customer's debugging experience.

Vaibhav Sharma is an Applied Scientist at Amazon Web Services (AWS). Over the last one year, Vaibhav has worked on applying automated reasoning to the Internet of Things (IoT) domain. Prior to AWS, Vaibhav obtained a PhD in Computer Science from the University of Minnesota during which he worked on program synthesis using scalable symbolic execution.

Bilal Khan is a Software Development Manager in the AWS IoT team, leading the software development for AWS IoT Events, a managed service that enables companies to continuously monitor their equipment and fleets of devices for failure or changes in operation and trigger alerts to respond when such events occur. Prior to Amazon, he was in different roles ranging from leading strategic initiatives for a mid-size embedded systems manufacturing company building remote monitoring solutions, and before that he was with Nokia and Motorola focusing on building O&M solutions for large & complex cellular communication networks. Outside of work he enjoys the outdoors in the beautiful PNW.

KEYNOTE PRESENTATION
DoD ENTERPRISE DEVSECOPS INITIATIVE & PLATFORM ONE

Nicolas Chaillan
USAF

The DoD Enterprise DevSecOps Initiative (DSOP) is a joint effort of the DOD's Chief Information Officer (DCIO), Office of the Undersecretary of Defense for Acquisition and Sustainment, and the Military Services. The purpose of the DSOP is to bring automated software tools, services and standards to DOD programs so that warfighters can create, deploy, and operate software applications in a secure, flexible, and interoperable manner. Platform One is the DoD-wide DevSecOps Enterprise Level Service that provides software factories and program offices with managed IT services capabilities, on-boarding, and support. Platform One enables users to start their software work with a 90% solution instead of reinventing the wheel. Other benefits include avoiding vendor lock-in, baked-in Zero Trust security, and delivering capabilities to the warfighter at the speed of relevance.

Mr. Nicolas Chaillan serves as the first Air Force Chief Software Officer and is the Co-lead for the DoD Enterprise DevSecOps Initiative along with the DoD CIO. As the Air Force's senior software czar, Mr. Chaillan is responsible for enabling Air Force programs in the transition to Agile and DevSecOps to establish force-wide DevSecOps capabilities and best practices, including continuous Authority to Operate (c-ATO) processes and faster streamlined technology adoption. Prior to his current position, Mr. Chaillan was the Special Advisor for Cloud Security and DevSecOps at the Department of Defense, OSD, A&S and Special Advisor for Cybersecurity and Chief Architect for Cyber.gov at the Department of Homeland Security. In addition to his public service, Mr. Chaillan is a technology entrepreneur, software developer, cyber expert and inventor. Mr. Chaillan is recognized as one of France's youngest entrepreneurs after founding, WORLDAKT at 15 years of age. Mr. Chaillan founded 12 companies, including AFTER-MOUSE.COM, Cyber Revolution, Prevent Breach, anyGuest.com, and more. Over the last eight years alone, he created and sold over 180 innovative software products to 45 Fortune 500 companies.

CONTINUOUS INTEGRATION AND FORMAL METHODS WITH MUSE, AFFIX, AND 5C

***Stephen Magill**, **Michael Hicks
**MuseDev / Sonatype, **University of Maryland*

Software systems now pervade most aspects of modern life, and as such their security and reliability are increasingly important: penetration or failure of a buggy component could precipitate the downfall of an entire system. Static program analysis, a kind of formal method, is becoming increasingly effective at identifying bugs in source code, and this success has spurred an interest in greater use of static analysis during development. While some analyzers can be run during code authoring (e.g., as part of an IDE), a "deep" static analysis, which looks closely at a program's semantics and not just surface-level patterns, is too slow to be used interactively. Work at Facebook found that deploying such analyzers in a "batch mode" to run overnight was ineffective, with bug reports essentially ignored by developers. They found that integrating static analysis i nto a continuous integration (CI) process worked far better. In such a CI process, a developer checks i n their code for review, and the static analyzer flags issues for the developer to consider, alongside i ssues flagged by human reviewers. Facebook found that with such a process, 70% of flagged errors were fixed[1], whereas 0% were fixed when analysis was run in a batch mode, outside of the standard CI workflow. While Facebook's experience shows that integrating static analysis tools into a CI-based code review system works well, static analysis is not the only valuable automated reasoning (i.e., formal methods) tool. We might like to also integrate annotation inference tools, which propose changes to the underlying source code. Doing so is different from leaving the code alone and submitting comments for review. We might also like to employ binary analysis tools to reason about binary-only libraries, to assist source code analyzers so they can better reason about how a project's source code uses those libraries. What might be the best ways to integrate these sorts of tools into the development workflow, alongside static analysis? In this presentation, we report on work to explore integration of tools such as these into the CI-based development process. In particular, we consider how to integrate annotation-inference and binary-analysis tools into the GitHub pull request system using the Muse platform[2] to manage interactions with the user. We describe new APIs that allow a dialog between the developer and the tool that: 1) is stateful and context-aware, 2) allows the developer to stay entirely within the GitHub UI, 3) lowers requirements of the developer's tool knowledge to account for the fact that most developers

will not be formal methods experts and will be only sporadically interacting with these tools. As two case studies we present ongoing work to integrate Muse with (1) 5C[3], an annotation inference tool that automates conversion of C code to Microsoft's Checked C extension; and (2) Affix[4], a model generation tool for binary code. These integrations provide access to advanced formal methods-based capabilities without requiring developers to directly install, configure, or maintain the tools and without intrusive changes to developers' workflows. Doing so expands the set of developers that can benefit from these technologies and expands the set of interactions for using the tools effectively.

References

[1] https://cacm.acm.org/magazines/2019/8/238344-scaling-static-analyses-at-facebook/fulltext

[2] https://www.muse.dev/

[3] https://correctcomputation.com/products/5c/

[4] https://correctcomputation.com/products/affix/

Dr. Stephen Magill was the CEO and co-founder of MuseDev, and is now VP of Product Innovation at Sonatype. He has spent his career developing tools to help developers identify errors, gauge code quality, and detect security issues. Stephen's research has focused on automated program analysis and has spanned the spectrum from analyses for proving security of low-level code to privacy analyses for high-level query languages. Stephen has led multiple large-scale research initiatives including DARPA projects on privacy, security, and code quality. He also served as research lead for the 2020 and 2021 State of the Software Supply Chain reports. Dr. Magill earned his Ph.D. in CS from Carnegie Mellon University, and his BS from the University of Tulsa. He is a member of the University of Tulsa Industry Advisory Board and has served on numerous program committees and funding panels.

AUTOMATED TRUST ANALYSIS FOR LAYERED ATTESTATIONS

Ian Kretz, John D. Ramsell, Paul D. Rowe
MITRE

In distributed systems, trust decisions are often based on remote attestations in which evidence is gathered about the integrity of subcomponents. Layered attestations leverage hierarchical dependencies among the subcomponents to bolster the trustworthiness of evidence. Complex dependency relationships in production systems can lead to equally complex layered attestations. The power of human analysts to reason about the correctness of an attestation in the presence of an active adversary becomes greatly diminished amid such complexity. We present an automated toolchain for reasoning about the trustworthiness of attestations and provide a formal proof of its correctness. Copland is a domain-specific language for specifying complex layered attestations. A Copland phrase expresses an attestation as a composition of the local activities of subcomponents -- requesting evidence, performing measurements and cryptographic operations, bundling evidence and replying to requests. Phrases themselves may be composed: if phrase P gives evidence of A's runtime state assuming B's state is pristine, and P' gives evidence of B's runtime state, then composing P' with P gives evidence of A's runtime state. In the absence of an adversary, the trust properties of phrases P and P' compose additively regardless of the order in which they are executed. However, an active adversary can undermine the additive nature of this composition. How phrases are composed bears directly on the trustworthiness of the evidence they produce. We introduce a method for analyzing executions of attestations specified by Copland phrases in an adversarial setting. We develop a general theory of executions with adversarial corruption and repair events. Our approach is to enrich the Copland semantics according to this theory. Using the model finder Chase, we characterize all executions consistent with a set of initial assumptions. From this set of models, an analyst can discover all ways an active adversary can corrupt subcomponents without being detected by the attestation. These efforts afford trust policymakers the ability to compare attestations expressed as Copland phrases against trust policy in a way that encompasses both static and runtime concerns.

Ian Kretz is a Cyber Security Engineer at the MITRE Corporation. He was educated at Rice University and Northeastern University. His work has focused on cryptographic protocol analysis, verified implementation and layered attestation.

CYBORG CRYPTOGRAPHIC SECURITY: BLUETOOTH, SIGNAL, AND BEYOND

Britta Hale
Naval Postgraduate School

Cryptographic analyses frequently focus on the device-to-device channel - a perspective often sufficient for network security protocols. However, in many cases the end user is an active participant in the protocol. In Bluetooth Passkey Entry, Signal Authentication, WhatsApp Authentication, and some ISO protocols, the user acts as a trusted third party that interacts directly with the cryptographic protocol. Common analysis techniques ignore such interaction as out-of-scope. In this presentation, we will outline the CYBORG compositional approach to cryptographic modeling, namely the capturing of user-to-device channels, as well as the typical device-to-device channels. The CYBORG model enables realistic real-world security analysis in light of published attacks on user-mediated protocols such as Bluetooth that leverage malware and device displays. We highlight how such attacks have been overlooked by ignoring user interaction in protocol analysis. Moreover, we will present analysis results on Bluetooth Passkey Entry that point to improved design approaches for user-mediated protocols.

This talk addresses the conference theme of *compositional security*, in that it looks beyond a simple device -to-device analysis approach and extends to the user interface, pointing to security risks associated with that and the interlocking effects of the two. Moreover, this presentation has particular future-looking relevancy to the theme of *Continuous Development*: interface interaction with cryptographic protocols have wide implications, and this talk points to integration and expansion of analysis techniques for a wholistic system view.

Dr. Britta Hale is a cryptographer and Assistant Professor in Computer Science at the Naval Postgraduate School in Computer Science. Dr. Hale has a PhD from the Norwegian University of Science and Technology (NTNU), and a Master's of Science in the Mathematics of Cryptography and Communications from Royal Holloway University of London (RHUL). Her focus areas include cryptographic key exchange and authentication protocols, protocol self-healing, post-quantum hybrids, unmanned vehicle security, and secure channels within constrained settings. Dr. Hale is currently a member of the Message Layer Security (MLS) working group of the Internet Engineering Task Force (IETF) and International Association for Cryptologic Research (IACR).

ZERO-KNOWLEDGE PROOFS OF BINARY EXPLOITABILITY

Ben Perez
Trail of Bits

Background: We introduce the first use of zero-knowledge (ZK) proofs to prevent proprietary knowledge and techniques from leaking during software vulnerability disclosures. These proofs allow researchers to prove that they possess a software exploit in a known program binary without disclosing information about the nature of the vulnerability or the structure of its corresponding exploit. Our work builds on the techniques of Ben-Sasson et al., which demonstrate how to efficiently represent statements about program execution on Von Neumann architectures in a ZK context. Unlike previous work, ours supports real-world microprocessor architectures such as the MSP430. We also provide a ZK proof system which reduces proving time by several orders of magnitude compared to the one used by Ben-Sasson et al.

Methods: We equip vulnerability researchers with tools that allow them to unambiguously demonstrate they possess a binary exploit against a program on a real-world processor architecture, without sharing the exploit itself. Since the underlying ZK proof systems reason about statements represented as Boolean or arithmetic circuits, we develop techniques for creating circuits that accept if and only if they are provided with a valid software exploit. This is done by adapting and optimizing prior work on reducing the correctness of RAM program execution to circuit satisfiability. Our approach to representing statements about program execution as circuits eliminates a logarithmic factor present in previous work by replacing the routing network used during input validation with a pseudorandom function (PRF). This PRF requires the proof system to facilitate switching between Boolean and arithmetic circuits mid-proof. While field switching incurs some slight overhead per conversion, it is still substantially more efficient than a quasilinear reduction with large constant factors.

Prior work on ZK proofs of program execution focuses on a specific subclass of proving systems known as Succinct Non-Interactive ARguments of Knowledge (SNARKs), which require a trusted setup phase and optimize proof size and verification time at the expense of prover complexity, which is quasilinear in the circuit size. To optimize our techniques for practical software exploits, we have implemented a prover-optimized ZK proof system based on the MPC-in-the-head paradigm of Ishai et al. This system produces larger proofs than SNARKs, but its prover runtime is several orders of magnitude faster and supports proof streaming so that the verifier never needs to load the entire proof into memory. For example, libSNARK takes roughly 23 minutes to compute 511 iterations of the SHA256 hash function, whereas

our proof system only takes 23 seconds. Due to the asymptotically worse prover complexity of SNARKs, this performance gap becomes untenable for circuit sizes relevant to vulnerability disclosure. e equip vulnerability researchers with tools that allow them to unambiguously demonstrate they possess a binary exploit against a program on a real-world processor architecture, without sharing the exploit itself. Since the underlying ZK proof systems reason about statements represented as Boolean or arithmetic circuits, we develop techniques for creating circuits that accept if and only if they are provided with a valid software exploit. This is done by adapting and optimizing prior work on reducing the correctness of RAM program execution to circuit satisfiability. Our approach to representing statements about program execution as circuits eliminates a logarithmic factor present in previous work by replacing the routing network used during input validation with a pseudorandom function (PRF). This PRF requires the proof system to facilitate switching between Boolean and arithmetic circuits mid-proof. While field switching incurs some slight overhead per conversion, it is still substantially more efficient than a quasilinear reduction with large constant factors.

Results: We focus on practical software exploits by demonstrating that our tools can be used to prove knowledge of solutions to the Microcorruption CTF, a series of binary exploit challenges that involve breaking into a smart lock controlled by an MSP430 processor. Microcorruption covers many common binary exploits such as stack and heap overflows, bypassing memory protections such as ASLR, and the use of return-oriented programming (ROP) gadgets. To validate our techniques we have developed an efficient RAM reduction for MSP430 programs and ported the challenge set into our framework.

Since both prover time and proof size scale linearly with the size of the execution trace, performance depends exclusively on the processor circuit size and the number of instructions required to trigger the exploit. For a simple stack-based buffer overflow bug with a trace length of 1k instructions, we were able to generate an 85MB proof in 30 seconds on a Digital Ocean machine with 8 cores and 64 GB of RAM. When running more complex heap vulnerabilities with a trace length of 12k instructions, our system produced 1.1GB proofs in roughly 6 minutes.

Acknowledgement: This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA). The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

Ben Perez is a senior security engineer at Trail of Bits. He brings expertise in formal methods and applied cryptography to his work on blockchain security tools, cryptography research, and software assessments. He holds a BA in mathematics from St. Olaf College (2014) and an MS in computer science from the University of California, San Diego (2018), where his master's thesis focused on computationally efficient dimension reduction algorithms.

KEYNOTE PRESENTATION
ENABLING PROVABLE SECURITY AT SCALE

Neha Rungta
Amazon Web Services

The cloud model can be viewed as a contract between customers and providers: customers program against the cloud model, and the cloud provider faithfully implements the model. In this talk, we focus on the problem of access control. Many users are moving sensitive workloads to the cloud and require that their access control policies comply with their governance requirements and security best practices. In this talk, I will present an overview of Zelkova, an SMT-based analysis engine, for verification of Amazon Web Services (AWS) access control policies that govern permissions across entire applications in the cloud. It uses traditional formal verification techniques such as language transformation and off-the-shelf SMT solvers. zelkova can be queried to explore the properties of policies e.g. whether some resource is "publicly" accessible, and is leveraged in features such as Amazon S3 Block Public Access, AWS Config Managed Rules, and Amazon Macie. Our experience shows that to leverage formal policy analysis users must have sufficient technical sophistication to realize the criteria important to them and be able to formalize the properties of interest as zelkova queries. In 2019, we launched a publicly available service IAM Access Analyzer that leverages Zelkova to automatically identify resources such as S3 buckets and IAM roles that are shared with an external entity. We help users understand whether their policy is correct, by abstracting the policy into a compact set of positive and declarative statements that precisely summarize who has access to a resource. Users can review the summary to decide whether the policy grants access according to their intentions.

Neha Rungta is a senior principal applied scientist in the Automated Reasoning Group with Amazon Web Services (AWS), working on formal verification techniques for cloud security. Prior to joining AWS, Neha was well-known for her work on symbolic execution, automated program analysis, and airspace modeling at the NASA Ames Research Center. She earned a PhD in computer science from Brigham Young University in 2009.

BRIEFCASE FULL OF PROOFS

Darren Cofer
Collins Aerospace

As part of DARPA's Cyber Assured Systems Engineering (CASE) program, we are developing an engineering environment based on AADL for building cyber-resilient systems. Our tools provide verification of cyber requirements via a formal model of the system architecture with integrated model checking and information flow analysis. We track the satisfaction of cyber requirements throughout the design process by creating an assurance case that is integrated with the AADL system architecture model. We help developers to address cyber requirements through a library of design patterns, implemented via tool-assisted model transformations utilizing verified cyber-resilient components. Cyber-resilient implementations are automatically generated from the verified design model using property-preserving transformations, verified components, and the formally verified seL4 microkernel. The tools have been demonstrated on an example system based on the Unmanned Systems Autonomy Services (UxAS) framework developed by researchers at AFRL. We are currently applying the tools to the mission computing system in a military helicopter to demonstrate effectiveness of the approach on a real avionics system.

Darren Cofer is a Fellow in the Trusted Systems group at Collins Aerospace. He earned his PhD in Electrical and Computer Engineering from The University of Texas at Austin.

His principal area of expertise is developing and applying advanced analysis methods and tools for verification and certification of high-integrity systems. His background includes work with formal methods for system and software analysis, the design of real-time embedded systems for safety-critical applications, and the development of nuclear propulsion systems in the U.S. Navy.

He has served as principal investigator on government-sponsored research programs with NASA, NSA, AFRL, and DARPA, developing and using formal methods for verification of safety and security properties. He is currently the principal investigator for Collins teams working on DARPA's Cyber Assured Systems Engineering (CASE) and Assured Autonomy programs.

Dr. Cofer served on RTCA committee SC-205 developing new certification guidance for airborne software (DO-178C) and was one of the developers of the Formal Methods Supplement (DO-333). He is a member of SAE committee G-34 on Artificial Intelligence in Aviation, the Aerospace Control and Guidance Systems Committee (ACGSC), and a senior member of the IEEE.

HAMR - HIGH-ASSURANCE MODELING AND RAPID ENGINEERING FOR EMBEDDED SYSTEMS USING AADL

John Hatcliff
Kansas State University

Many domains rely on real-time embedded systems that require high levels of assurance. Assurance of such systems is challenging due to the need to support compositionality related to platform-based development, software product lines, interoperability, and system-of-system concepts. The Architecture and Analysis Definition Language (AADL) provides a modeling framework that emphasizes componentbased development, modeling elements with semantics capturing common embedded system threading and communication patterns, and standardized run-time service interfaces. Through its annex languages and tool plug-in extensibility mechanisms, it also supports a variety of architecture specification and analyses including component behavioral contracts, hazard analysis, schedulability analysis, dependence analysis, etc. The AADL vision emphasizes being able to prototype, assure, and deploy system implementations derived from the models. This talk will present an overview of HAMR (High-Assurance Modeling and Rapid Engineering) -- a multipleplatform code-generation, development, and verification tool-chain for AADL-specified systems. HAMR's architecture factors code-generation through AADL's standardized run-time services (RTS). HAMR uses the AADL RTS as an abstract platform-independent realization of execution semantics which can be instantiated by backend translations for different platforms. Current supported translation targets are: (1) Slang (a safety-critical subset of Scala with JVM-based deployment as well as C code generation designed for embedded systems), (2) a C back-end for Linux with communication based on System V inter-process communication primitives, and (3) a C back-end for the seL4 verified micro-kernel being used in a number of US Department of Defense research projects. The C generated by HAMR is also compatible with the CompCert verified C compiler. HAMR supports integrations with other languages (e.g., CakeML) through its generated AADL RTS foreign-function interface facilities.

The talk will describe experience with HAMR on the DARPA CASE (Cyber-Assured System Engineering) project. On CASE, AADL is used to model defense systems and to enable component-based model-based transformations that enhance system architecture to achieve greater cyber-resiliency. HAMR is then used to support development and deployment on

the seL4 microkernel. SeL4's formally-verified partitioning and memory security properties provide the foundation for cyber-resiliency assurance arguments. The talk addresses the HCSS 2021 theme of "Exploring Compositionality". The HAMR work explores how domain-specific threading and communication abstractions ("reasoning in domain-specific settings" from the HCSS announcement) can be used as compositional building blocks for embedded systems ("horizontal", i.e., component-to-component composition) and how formally verified partitioning platforms such as seL4 can support compositional system assurance. HAMR also utilizes AADL's standardized RTS to form an abstract semantic reference model of computation -- enabling components to be "vertically" composed onto different platforms as the RTS abstraction layer is instantiated to conforming backend platforms. HAMR is available under an open-source license, and the project web-site includes an example repository and collection of videos, tutorials, and classroom lecture materials (also suited for workforce training).Dr. Cofer served on RTCA committee SC-205 developing new certification guidance for airborne software (DO-178C) and was one of the developers of the Formal Methods Supplement (DO-333). He is a member of SAE committee G-34 on Artificial Intelligence in Aviation, the Aerospace Control and Guidance Systems Committee (ACGSC), and a senior member of the IEEE.

Dr. John Hatcliff is a University Distinguished Professor and Lucas-Rathbone Professor of Engineering at Kansas State University working in the areas of safety-critical systems, software architectures, and software verification and certification. He leads the Laboratory on Specification, Analysis, and Transformation of Software (SAnToS Lab), whose research has been funded by national funding agencies and companies including Department of Defense, National Science Foundation, DARPA, Department of Homeland Security, US Army, NASA, NIH, ARO, Air Force Office of Scientific Research, SEI, Collins Aerospace, Adventium Labs, and Lockheed Martin.

Dr. Hatcliff recently led the primary technical working group for the AAMI / UL 2800 Joint Committee that is developing safety and security standards for medical device interoperability. The focus of Dr. Hatcliff's current funded research includes developing model-based verification and hazard analysis techniques for safety/security-critical embedded systems, and automated software contract verification for modern languages for embedded systems.

VERIFIABLE BINARY LIFTING

Joe Hendrix
Galois

This talk describes work at Galois on building a verifiable decompiler from machine code to LLVM. The goal is to support new uses of binary analysis such as program recompilation for optimization and security improvements, security patching, and retrofitting legacy applications for new architectures or mission requirements. These new applications have different requirements than traditional uses of decompilation tools such as reverse engineering or malware analysis. In particular, these new applications place less emphasis on helping people understand binaries, and more on helping compilers and verification tools analyze binaries. In particular with these new applications, new binaries may be generated that are intended to run critical programs, and it is imperative that the decompilation framework does no harm by introducing new bugs. Without the source and support engineering team, it will be very difficult to recertify a translated program, and skipping certification is not acceptable in safety critical environments. In this talk, we describe our experience building our decompiler to LLVM and how we are working on providing assured binary lifting. In our experience, there are always improvements and changes to be made in terms of supporting new features for ISAs, operating systems, and compilers, and it would be extremely difficult to build and maintain a **verified decompiler** with proofs robust to changes. Instead, we have focused on a two phased approach:

- Our **decompiler** produces **untrusted** annotations that relate the machine code to the LLVM.
- We have an independently written **decompiler verifier** that uses the annotations and a **compositional strategy** and SMT solving to automatically discharge a large number of small program equivalence checks.

We will describe our compositional approach for automated verification of the generated programs, and how we have modified our decompiler to record the relationships between machine code and LLVM at the level of individual basic blocks. This allows us to decompose proofs about overall decompilation of a program into many efficiently solvable SMT-queries about operations within individual basic blocks. An additional benefit of our approach is that the decompiler and verifier are written independently. The decompiler and verifier are written in different languages and use machine code semantics obtained from different sources. This

provides an informal form of additional assurance that a mistake in either semantics must be present in precisely the same way in the other to successfully invalidate a successful verification result. Our verifier is written within the Lean theorem prover, and the eventual goal is to formally show that the compositional strategy used is sound for all programs. The techniques described in this presentation are implemented in the Galois open source reopt program recompilation tool. We will show this tool, and discuss how the HCSS 2021 composition and proof robustness themes have been addressed in the design of the reopt. The project depicted is sponsored by the Office of Naval Research under Contract No. N68335-17-C-0558. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Office of Naval Research.

Dr. Joe Hendrix is a Principal Researcher at Galois where he focuses on helping transition formal methods technologies to address real world problems. His technical areas of interest include binary analysis, programing language semantics, decision procedures and interactive theorem proving.