Formal Methods at the National Science Foundation

Nina Amla

Program Director

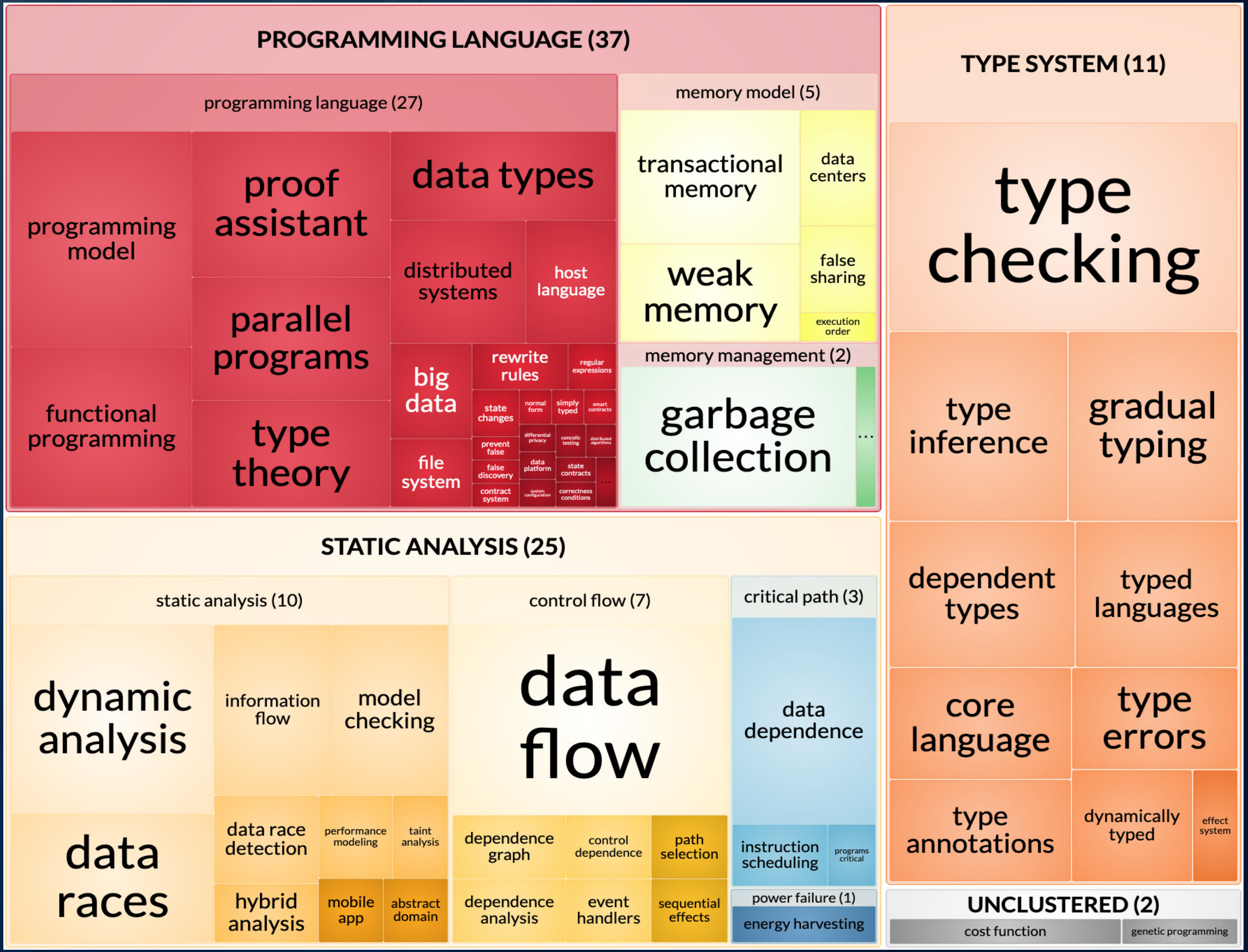
National Science Foundation

Sept 25 2019

NSF programs that support Formal Methods

- Software and Hardware Foundations (SHF) core program
- CISE-wide cross programs
  - Formal Methods in the Field (FMitF)
  - Scalable Parallelism in the Extreme (SPX)
  - Expeditions in Computing
- NSF-wide cross-directorate and cross-agency programs
  - Secure and Trustworthy Cyberspace (SaTC)
  - Cyber Physical Systems (CPS)

Programming Languages Portfolio



PROGRAMMING LANGUAGE (37)

programming language (27)

programming model

functional programming

proof assistant

parallel programs

type theory

data types

distributed systems

host language

big data

rewrite rules

regular expressions

state changes

normal form

simply typed

smart contracts

differential privacy

concolic testing

distributed algorithms

prevent false

false discovery

data platform

state contracts

contract system

system configuration

correctness conditions

...

file system

memory model (5)

transactional memory

data centers

weak memory

false sharing

execution order

memory management (2)

garbage collection

...

TYPE SYSTEM (11)

type checking

type inference

gradual typing

dependent types

typed languages

core language

type errors

type annotations

dynamically typed

effect system

STATIC ANALYSIS (25)

static analysis (10)

dynamic analysis

information flow

model checking

data races

data race detection

performance modeling

taint analysis

hybrid analysis

mobile app

abstract domain

control flow (7)

data flow

dependence graph

control dependence

path selection

dependence analysis

event handlers

sequential effects

critical path (3)

data dependence

instruction scheduling

programs critical

power failure (1)

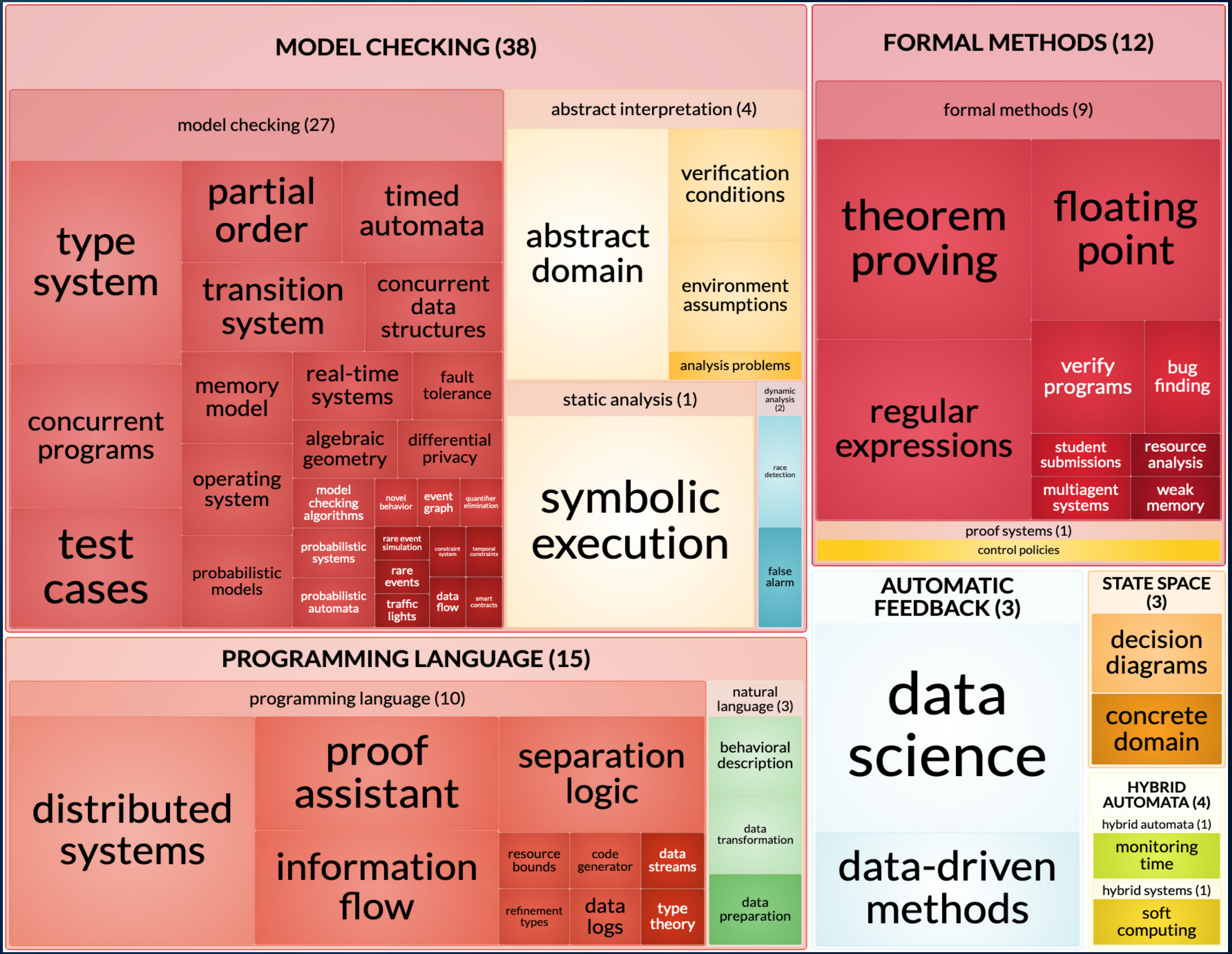
energy harvesting

UNCLUSTERED (2)

cost function

genetic programming

Formal Methods Portfolio



MODEL CHECKING (38)

model checking (27)

type system

partial order

timed automata

transition system

concurrent data structures

concurrent programs

memory model

real-time systems

fault tolerance

algebraic geometry

differential privacy

operating system

model checking algorithms

novel behavior

event graph

quantifier elimination

test cases

probabilistic models

probabilistic systems

rare event simulation

constraint system

temporal constraints

rare events

probabilistic automata

traffic lights

data flow

smart contracts

abstract interpretation (4)

abstract domain

verification conditions

environment assumptions

analysis problems

static analysis (1)

symbolic execution

dynamic analysis (2)

race detection

false alarm

FORMAL METHODS (12)

formal methods (9)

theorem proving

floating point

regular expressions

verify programs

bug finding

student submissions

resource analysis

multiagent systems

weak memory

proof systems (1)

control policies

PROGRAMMING LANGUAGE (15)

programming language (10)

distributed systems

proof assistant

separation logic

information flow

resource bounds

code generator

data streams

refinement types

data logs

type theory

natural language (3)

behavioral description

data transformation

data preparation

AUTOMATIC FEEDBACK (3)

data science

data-driven methods

STATE SPACE (3)

decision diagrams

concrete domain

HYBRID AUTOMATA (4)

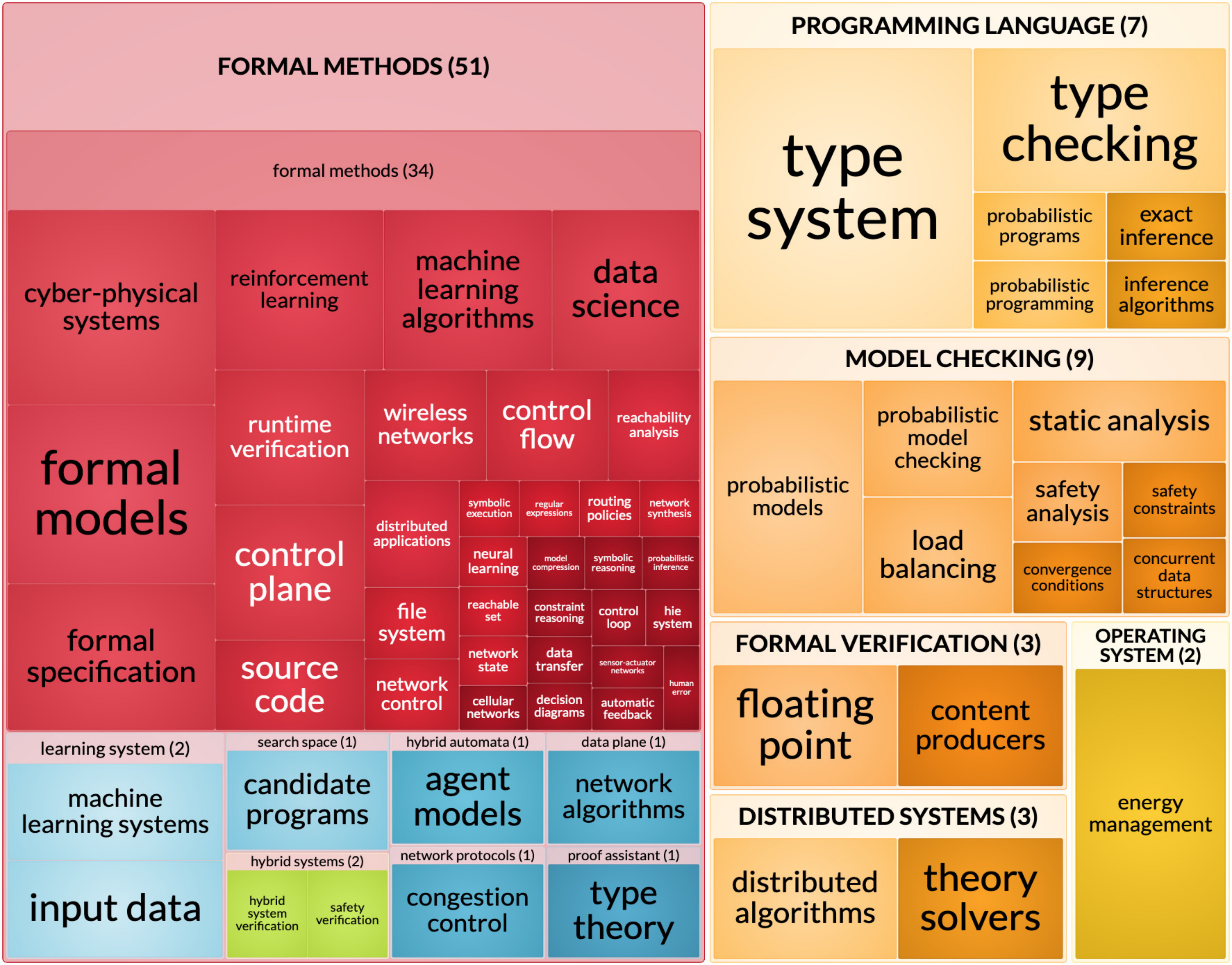
hybrid automata (1)

monitoring time

hybrid systems (1)

soft computing

Formal Methods in the Field Portfolio



A Snapshot of Topic Areas

- Semantics
- Types
- Domain Specific Languages
- SAT and SMT
- Model Checking
- Theorem Proving
- Synthesis

- Security & Privacy
- Operating & Distributed Systems
- Networking
- Cyber Physical Systems
- AI and Machine learning
- Concurrency and parallelism

Semantics

- Semantics-based techniques for compilation of multilingual software
  - *1816837/Ahmed, SHF: Principled Compiling and Linking for Multi-Language Software*
- Denotational models for specifying programming languages and verifying compiler correctness
  - *1814460/Siek, SHF: Revisiting Elementary Denotational Semantics*
- Categorical foundations of indexed programming (for both polymorphism and dependent types)
  - *1713389/Johann, SHF: New Foundations for Indexed Programming*

Types

- Fundamental principles that underlie sound and performant gradual typing systems.
  - *1763922/Tobin-Hochstadt, SHF: Performant Sound Gradual Typing*
- Simplify reasoning about properties of Haskell programs by using dependent types directly in the verification process.
  - *1703835/Weirich, SHF: The Theory and Practice of Dependent Types in Haskell*
- Logical foundations for message-passing concurrency, based on session types, application to Rust
  - *1718267/Pfenning, SHF: Enriching Session Types for Practical Concurrent Programming*

Domain specific languages/tools

- Verification and synthesis tools for system configuration language (Puppet)
  - *1717636/Guha, SHF: Formal Methods for Modern System Configuration Languages*
- Interactive programming environments for scalable web development
  - *1651794/Chugh, CAREER: Direct Manipulation Programming Systems*
- Enhance extant DSL tools with automatic verification and synthesis
  - *1651225/Torlak, SHF: The Next 700 Solver-Aided Languages*

Satisfiability (SAT) and Satisfiability Modulo Theories (SMT)

- Enhancing Reluplex to scale and give correctness guarantees
    - *1814369/Barrett, SHF: Certifiable verification of large neural networks*
- Solving open math problems via better encodings and parallel SAT solving.
    - *1813993/Heule, SHF: MaPaMaP: Massively Parallel Solving of Math Problems*
- High-level modeling of tensor models & data-aware reasoning and optimization techniques for both linear and non-linear models
    - *1816936/Jovanovic, SHF: SMT Reasoning for Tensors and Data*

Model Checking

- Rectification of finite-field arithmetic circuits using Groebner basis techniques and Craig interpolants
  - *1911007/Kalla, SHF: Rectification of Arithmetic Circuits with Craig Interpolants in Algebraic Geometry*
- Theory and model checking for hyper temporal logic for expressing security and privacy policies
  - *1813388/Bonakdarpour, SaTC: Techniques for Software Model Checking of Hyperproperties*
- Paradigms for the exact verification of differential privacy
  - *1901069/Sistla, SHF: Medium: Collaborative Research: Verification of Differential Privacy Mechanisms*

Theorem Proving

- Incorporate the universal composability (UC) framework for analyzing cryptographic systems into EasyCrypt
  - *1801564/Stoughton, SaTC: Towards Mechanized Proofs of Composable Security Properties*
- Build a deductive synthesis framework for deriving mechanically verified program analyzers directly from their induced specifications
  - *1900563/Darais, SHF: Synthesizing Verified Analyzers for Critical Software*
- Coq-based practical verification framework that enables formally reasoning about distributed system implementations
  - *1749570/Tatlock, CAREER: Verifying Distributed System Implementations*

Synthesis and Repair

- Type system for resource aware refinement types and resource guided synthesis
  - *1812876/Hoffmann, SHF: Resource-Guided Program Synthesis*
- Scalable synthesis algorithms based on the idea of counterexample-guided abstraction refinement
  - *1811865/Dillig, SHF: Scalable Program Synthesis using Counterexample-Guided Abstraction Refinement*
- Verifying program fairness , explaining & repairing unfair programs
  - *1749664/Albargouthi, SHF: Formal Methods for Program Fairness*

Security and Privacy

- Design methodology for a fully-verified, functionally-correct hypervisor that satisfies confidentiality and integrity.
  - *1918400/Nieh, FMitF: A Secure and Verifiable Commodity Hypervisor*
- Machine checked verification for proving confidentiality in file systems and mail server
  - *1812522/Zeldovich, SaTC: Verifying security for data non-interference*
- Programming environment (DevDP) to develop programs that behave correctly wrt differential privacy policies
  - *1702760/Kifer, SaTC: CORE: Medium: Developing for Differential Privacy with Formal Methods and Counterexamples*

Networking

- New programming and verification abstractions for distributed network and control planes
  - *1837030/Gupta, FMitF: OpenRDC: A Framework for Implementing Open, Reliable, Distributed, Network Control*
- Synthesize code from user-provided sketches and specifications into low-level switch configurations
  - *1837023/Qiu, FMitF: Transplanting Syntax-Guided Synthesis to Computer Networks*
- Methodology for formal specification and testing of complex Internet protocols (QUIC) using Ivy
  - *1918429/Zuck, FMitF: Injecting Formal Methods into Internet Standardization*

Operating and Distributed Systems

- Investigate how Rust's type system interacts with SMT-style verification (Boogie) to build a verified OS
  - *1837051/Rakamaric, FMitF: RedLeaf: Verified Operating Systems in Rust*
- A framework for synthesis-aided development of efficient, reliable, and secure OS components
  - *1836724/Torlak, FMitF: A Framework for Synthesis of Efficient, Reliable, and Secure Operating System Components*
- A new symbolic execution system (based on KLEE) that is extensible and modular and easier for OS developers to use
  - *1918573/Stefan, FMitF: Finding and Eliminating Bugs in Operating Systems*

Artificial Intelligence and Machine Leaning

- Methods for developing verifiably safe Deep Neural Networks (DNNs)
  - *1900676/Dwyer, SHF: Rearchitecting Neural Networks for Verification*
- Automatically construct simple, coherent, human-readable explanations (programs) of a ML model or its decisions.
  - *1918211/D'Antoni, FMitF: Track I: Formal Methods for Explainable Machine Learning*
- Inference algorithms for probabilistic programming that leverage model checking and model counting techniques.
  - *1837129/Millstein, FMitF: Opening Up the Black Box of Probabilistic Program Inference*

Cyber Physical Systems

- Bounded model-checking via reduction to satisfiability modulo convex (SMC) programming
  - *1845194/Shoukry, CAREER: Decision Procedures for High-Assurance, AI-Controlled, Cyber-Physical Systems*
- Reasoning about predictive data-driven models that consider noise and uncertainties
  - *1815983/Sankaranarayanan, Rigorous Synthesis and Verification of Decisions Using Data-Driven Models*
- Methods for state estimation, online model identification and runtime verification for V2V connected vehicles
  - *1918531/Mitra, FMitF: Predictive Online Safety Analysis from Multi-hop State Estimates for High-autonomy on Highways*

Concurrency and Parallelism

- A library of reusable, high-performance persistent data structures to simplify NVM programming
  - *1717712/Scott, SHF: Data Structures and Transactions for Emerging Nonvolatile Memory*
- Systematize the implementation of scalable applications written in DSLs that target GPUs and DSPs
  - *1919197/Kulkarni , SPX: Write Once, Run on Anything: Verified, Tuned Accelerator Kernels from High Level Specifications*
- New abstractions and verification for traditional processor cores & accelerators
  - *1628926/Malik, XPS: FULL: Hardware Software Abstractions: Addressing Specification and Verification Gaps in Accelerator-Oriented Parallelism*

FM @ Scale

- What is scale in this context?
  - Size (LOC, netlist)
  - Performance (time, memory)
  - Generality vs Domain specific
  - Usability
  - Computing platform
  - Others?

- Key factors
  - Design for correctness (which includes security)
  - Domain expertise
  - Automation
  - Performance
  - Usability

FM @ Scale

- What is needed?
  - Continue to push foundational advances on new methods and tools
  - Engage with domain experts & industry to identify new applications
  - Need methodology that can integrate FM into actual design processes/flows
    - E.g. hardware, SLAM
  - Engage internationally
- What lessons have we learned about scalability of FM in practice?
  - e.g. static analysis, concolic testing, hardware verification, certifiable compilation