Automated Software Analysis and Verification with Separation Logic

Josh Berdine   Cristiano Calcagno   Dino Distefano
Samin Ishtiaq   Peter O'Hearn   John Reynolds
Hongseok Yang

CAV 2016
Toronto

Technical Milestones

- Separation Logic
- Symbolic Heaps
- Symbolic Execution
- Frame Inference
- Syntactic Abstraction
- Interprocedural Localization
- Higher-Order Inductive Definitions
- Approximate Join
- Bi-Abduction

Technical Milestones

- Separation Logic
- Symbolic Heaps
- Symbolic Execution
- Frame Inference
- Syntactic Abstraction
- Interprocedural Localization
- Higher-Order Inductive Definitions
- Approximate Join
- Bi-Abduction

Technical Milestones

- Separation Logic
- Symbolic Heaps
- Symbolic Execution
- Frame Inference
- Syntactic Abstraction
- Interprocedural Localization
- Higher-Order Inductive Definitions
- Approximate Join
- Bi-Abduction

Technical Milestones

- Separation Logic
- Symbolic Heaps
- Symbolic Execution
- Frame Inference
- Syntactic Abstraction
- Interprocedural Localization
- Higher-Order Inductive Definitions
- Approximate Join
- Bi-Abduction

Technical Milestones

- Separation Logic
- Symbolic Heaps
- Symbolic Execution
- Frame Inference
- Syntactic Abstraction
- Interprocedural Localization
- Higher-Order Inductive Definitions
- Approximate Join
- Bi-Abduction

Technical Milestones

- Separation Logic
- Symbolic Heaps
- Symbolic Execution
- Frame Inference
- Syntactic Abstraction
- Interprocedural Localization
- Higher-Order Inductive Definitions
- Approximate Join
- Bi-Abduction

Technical Milestones

- Separation Logic
- Symbolic Heaps
- Symbolic Execution
- Frame Inference
- Syntactic Abstraction
- Interprocedural Localization
- Higher-Order Inductive Definitions
- Approximate Join
- Bi-Abduction

Technical Milestones

- Separation Logic
- Symbolic Heaps
- Symbolic Execution
- Frame Inference
- Syntactic Abstraction
- Interprocedural Localization
- Higher-Order Inductive Definitions
- Approximate Join
- Bi-Abduction

Technical Milestones

- Separation Logic
- Symbolic Heaps
- Symbolic Execution
- Frame Inference
- Syntactic Abstraction
- Interprocedural Localization
- Higher-Order Inductive Definitions
- Approximate Join
- Bi-Abduction

Tools & Impact

- SLAyer
  - Windows kernel drivers
  - 10s of kLOC
  - "Quality Gate" checks when merging branches
  - 10s of bugs
  - "proofs" of a few drivers
- Infer
  - Facebook Android and iOS apps
  - 1000s of kLOC
  - analyze each change submitted to continuous integration
  - 1000s of changes analyzed per month
  - 100s of issues reported per month ($\sim$ 70% fixed)

Influence

- Some significant practical impact
  - But still much less influential on developer efficiency than e.g. build or source control systems
- Also some significant research influence
  - Many verification and analysis methods, tools, etc.
  - Open, robust, rich community working on different implementations of the fundamental ideas
  - Basic ideas have migrated out of separation logic
    - first-order logic encodings, and verification methodologies
  - Most fundamental ideas receding into everyday practice:
    - representation of memory in miTLS: separated hyperheaps

Influence

- Some significant practical impact
  - But still much less influential on developer efficiency than e.g. build or source control systems
- Also some significant research influence
  - Many verification and analysis methods, tools, etc.
  - Open, robust, rich community working on different implementations of the fundamental ideas
  - Basic ideas have migrated out of separation logic
    - first-order logic encodings, and verification methodologies
  - Most fundamental ideas receding into everyday practice:
    - representation of memory in miTLS: separated hyperheaps

Important Choices

- Short human-readable proofs
  - Still seem to be a good guide for scalable implementation
  - Tools gamble: there exists a short proof, or else analysis would explode no matter what
- Simplish proof-theory close to the “right” semantics
  - Implementations need to manipulate a representation
    - proof theory gave a reasonable one
  - Close and direct relation between proof theory and semantics
    - important to avoid complicated, verbose, encodings
  - Semantics plays key role guiding soundness etc. arguments

Important Choices

- Short human-readable proofs
  - Still seem to be a good guide for scalable implementation
  - Tools gamble: there exists a short proof, or else analysis would explode no matter what
- Simplish proof-theory close to the "right" semantics
  - Implementations need to manipulate a representation
    - proof theory gave a reasonable one
  - Close and direct relation between proof theory and semantics
    - important to avoid complicated, verbose, encodings
  - Semantics plays key role guiding soundness etc. arguments

Important Observations

- Goal of verification: Increase developer efficiency
  - Distinct but overlaps with classic reliability/security/…goals
- Deployment model is extremely important
  - precision vs. scalability
    - SLAYER useless no matter how fast if not precise enough
    - INFER useless no matter how precise if not scalable enough
  - developer groups differ
    - how much evidence required to justify code change
    - expectations and tolerance re missed/useless reports
    - batch mode vs. code review

Important Observations

- Goal of verification: Increase developer efficiency
  - Distinct but overlaps with classic reliability/security/...goals
- Deployment model is extremely important
  - precision vs. scalability
    - SLAyer useless no matter how fast if not precise enough
    - Infer useless no matter how precise if not scalable enough
  - developer groups differ
    - how much evidence required to justify code change
    - expectations and tolerance re missed/useless reports
    - batch mode vs. code review

Theoretical Research to Practical Application

Theory to Practice Continuity

- The narrower the gaps between theoretical research, experimental research, and practical application the better
- Practical application seems to take a team of engineers where a large fraction have verification/analysis PhDs
  - Knowing or being able to read the literature not enough
  - Seems to require / benefit hugely from experience exploring the theoretical research design space
  - Hard to scale application of verification in this situation
- It would strengthen the field to
  - Explicitly take continuity from theoretical research through practical application as a goal
  - Consciously evaluate work such that the whole spectrum is incentivized

Theoretical Impact

- The further theoretical results can provide good guidance for applied work the better
  - To optimize exploration of the design space
  - To optimize division of labor
- Simplifying assumptions
  - E.g.: abstractions are finite, transformers are distributive, etc.
  - Making them is necessary to make progress
  - Eventual practical applications usually need to lift them
  - It is “suboptimal” if “last mile” practitioners are the only ones trying to lift them
- Need to be careful not to under-value work that lifts simplifying assumptions or “just applies” an existing theory

Empirical Impact

- Convincing empirical arguments are hard
  - Measuring one variable and controlling the rest is difficult
  - What metrics are accurate predictors of utility?
- How closely correlated is what we measure in experimental papers with limiting factors in practice? (Or what we prove in theoretical papers?)
  - efficiency vs. scalability
  - “small” vs. “large” instances
  - synthetic vs. organic instances
  - worst-case vs. “average”-case vs. observed complexity
  - increased performance vs. increased precision/expressivity

Objective Subjectivism or Subjective Objectivism

- Need to encourage work that is significantly *different*, even if not measurably *better*
- We may not know the metric that shows improvement
- May only find the metric by contrasting different but incomparable approaches

Practical Impact

- People doing practical applications and theoretical or experimental research usually have:
  - different organizations
  - different motivations
  - different incentives
  - different timeframes
  - ...
- Some "impedance matching" to do, effective communication may not be trivial
- Dialogue can enable higher impact across the spectrum

Technical Challenges

Compositional & Differential

- Compositional, bottom-up, analysis
  - absolutely critical to INFER's scalability
  - extremely under-explored in the literature
  - probably every theoretical question except soundness is open
- Differential analysis
  - analyze a code *change*, not an entire code base
  - step further than differential *reporting*
  - cheap way to obtain specifications: use previous version
  - also critical for scalability, and extremely under-explored

Compositional & Differential

- Compositional, bottom-up, analysis
  - absolutely critical to INFER's scalability
  - extremely under-explored in the literature
  - probably every theoretical question except soundness is open
- Differential analysis
  - analyze a code *change*, not an entire code base
  - step further than differential *reporting*
  - cheap way to obtain specifications: use previous version
  - also critical for scalability, and extremely under-explored

Unknown / Incomplete Code

- Code to be analyzed always refers to code that is unknown, impossible to verify, etc.
- How should unknown code be treated?
  - write specifications
  - write models (i.e. specifications)
  - be pessimistic
  - be optimistic
  - be optimistic, but test dynamically
  - synthesize (and inspect?) specifications / models
  - ...
- There is some work, but still lots of progress needed
- Even harder when properties are not global invariants
  - trace properties for taint analysis
  - isolation properties for separation logic

Principled Reporting

- Developers want soundness and completeness
  - in practice, cannot have either one absolutely
- Trade-off: Sound relative to idealized, optimistic model
- Still far too incomplete
  - Analyzer finds many useless/spurious/false issues
- Trade-off: Reduce noise, only report "high-confidence" issues
- "High-confidence" determined by ad-hoc heuristic based on
  - execution history leading to violation
  - providence of values involved
  - known inaccuracies
  - ...
- Principled definitions of such confidence metrics seem under-explored

Conclusion

- Milestones, Choices, Observations
- Many bugs in lots of code, Migration of ideas
- Community Challenges
  - Narrow gaps between theoretical research, experimental research, and practical application
  - Beware systemic under-valuation of points on the spectrum
  - What metrics are accurate predictors of utility?
- Technical Challenges
  - Compositional & Differential verification / analysis
  - Unknown / Incomplete Code
  - Principled Reporting