Programatica Summary

James Hook
OGI School of Science & Engineering
Oregon Health & Science University

Programatica Team

- James Hook
- Mark Jones
- Richard Kieburtz
- Tim Sheard
- John Launchbury
- Peter White
- Bill Harrison
- Sylvain Conchon
- Thomas Hallgren
- Mark Tullsen
- Iavor Diatchki
- Nathan Linger

PacSoft

- Focus on Domain-specific language technology
  - DSLs for Hardware design (Hawk)
  - Spin-off Galois Connections continues to apply DSLs in high-assurance domains
- Programatica leverages Haskell enriched by properties for modeling and implementing software “as if correctness matters”

Programatica Vision

- Concise, Executable (Formal) Models and Systems expressed in Haskell
- Systematic identification of domain-relevant properties
- Integration of evidence for properties from external tools (some new, some existing) including testimonial ("Mark says so"), test, model checking and theorem provers
- Query and Navigation of evidence

Status: Modeling

- Modeling a non-trivial secure system
  - Developed a model of Spook: a POSIX compliant operating system supporting strict separation
  - White has coded over 12k lines of Haskell
  - Separation property can be expressed in Programatica
  - Spook architecture isolates “tricky bits” so that separation in 90% of the model is established via Haskell type checking
  - White presentation following lunch

Status: Semantics

- Folklore:
  - Haskell has a straightforward semantics; all of the hard parts have been addressed in published papers
- Reality:
  - While the hard parts were well studied, the integration was not
  - Particularly critical is the characterization of the fine control of evaluation (laziness)
- Consequence:
  - We will deliver a formal definition of Haskell (See supplementary materials: Harrison; to appear MPC02)

Status: Logic

- Haskell is a powerful modeling language
- Pun between inductive lists and coinductive streams is powerful, natural
- Expressive power is sufficient to encode concurrency (among other things)
- Modeling idioms in hardware and security exploit this power
- Support for these idioms has led to a modal mu calculus for Haskell (Kieburtz, talk to follow, supplemental materials)

Status: Tools

- Extensible tools for parsing, type checking, and navigating Haskell
- First implementation of recursive modules fully compliant with report
- Prototype integration with Alfa proof editor
- Free theorem generator (theorems from types via parametricity)
- Prototype P-logic proof engine in Stratego

Poirot:
The evidence manager

- An Herculean task
- Evidence manager integrates certificates from various sources (heterogeneous, auditable evidence)
- Supports queries on the heterogeneous evidence base (traceability)
- Core tool for achieving the “programatica vision” of software development
- Evidence manager gives “the knob”
- Status: Early stand-alone prototype; not yet integrated with P-logic or other tools

Next Steps

- Continue Spook development: focus on having non-trivial interprocess communication
- Complete Haskell definition
- Improve support of P-logic
- Integrate P-logic with Haskell front-end
- Integrate P-logic into Poirot

Long Term

- Programming as if properties matter
  - Developing the properties is part of developing the code
- Spook
  - Properties of interest are domain-relevant, high-level, global properties
- Poirot
  - Integrating and evaluating evolving, heterogeneous evidence
- Poirot for other languages
  - http://www.cse.ogi.edu/PacSoft/conf/jvw02/
- Tools