Integration of Tools and Techniques

John Rushby
Computer Science Laboratory
SRI International

The Center's Role In Integration

- Do the things that are beyond the scope and interest of individual research groups
- External
  - Help present our collective field to the outside world
  - And the outside world to us
- Internal
  - Achieve a more unified understanding of the field
  - Leverage each others' experience, capabilities, and tools
    - So that each individual group can achieve more
    - And so that we can build integrated systems and undertake collaborative examples that go beyond individual capabilities
- Collectively we will achieve more than individually

Why So Many Systems, Notations, Methodologies?

- Perception
  - The field is Balkanized into competing camps
- Result
  - A barrier to adoption, discourages entry
- Reality
  - Different techniques have different strengths and weaknesses
  - Different problems require different techniques
  - But most require some combination of techniques
- Center Response
  - Support a variety of techniques, and guide their integration

Explain and Support a Variety of Tools and Techniques

- Educational resources

  About the different methods, and how they can work together

- Web-based access to tools

  In a systematic manner (ETI... but for intensive interaction?)

- Common resources

  Shared examples, libraries, complete applications, tool components

- Common formats/languages

  That several tools can use for interoperation

- Integrated approaches

  That allow several tools to work together

Shared Examples: Experience

- Great benefit from informal "competition" on certain hot problems
  - SRT division (following Pentium FDIV bug)
  - Tomasulo algorithm (for superscalar dispatch)
  - Cache coherence
  - Bounded Retransmission Protocol (BRP)
- And from multiple analyses of common specifications
  - TCAS (airborne collision avoidance)
  - Rockwell autopilot
  - Safer
- Spurs methodology development, hones tools

Shared Examples: Center Activity

- Develop repository of substantial examples
  - As problem statements
  - In forms used by specific tools
  - In intermediate format that can be translated to many tools
- In a manner useful to both users and developers
  - Develop benchmarks and calibrate progress
  - And challenge problems to promote cross-fertilization of ideas

Shared Examples: Research Challenge

- The intermediate format (see later)

Shared Libraries: Experience

PVS example

- (Undirected) graph theory library developed at NASA Langley

  By Butler & Sjogren (AFOSR) for proof of Menger's Theorem

- Modified for directed graphs at University of Utah

  By Mokkedem, Hosabettu, and Gopalakrishnan for Formalization and Proof of a Solution to the PCI 2.1 Bus transaction Ordering Problem (FMCAD 98)

- And subsequently re-adopted/generalized by NASA

Shared Libraries: Additional Experiences

- ACL2 example: Integer Hardware Specification (IHS) developed for Motorola, reused by Rockwell
- Another PVS example: Analysis library developed by Dutertre used by Martin in a sound interface to symbolic integration routines
  - Mathematica et al. make hideous errors in this domain
  - There are several analysis libraries

    ...but which ones are correct, useful?
- Different kind of benefit: incorporation of HOL and PVS libraries in NuPrl
- COSPAN/FormalCheck property library also used by VIS

Shared Libraries: Center Activity

- Collect, commission, reconcile libraries
  - Both foundational (e.g., set theory, algebra)
  - And application-oriented (e.g., quaternions, voters)
- In forms used by specific tools
- And in generic format that can be translated to many tools
  - Or used to help validate the libraries of individual tools

Shared Libraries: Research Challenges

- Libraries have both declarative and pragmatic content
- So can we develop libraries that are both portable and effective?
- How can libraries be indexed effectively?

Shared Applications: Experiences

- Algorithm for Byzantine FT clock synchronization mechanically verified by Rushby and von Henke at SRI (in EHDM, 1989)
- Improved in "competition" with Young at CLI (using Nqthm)
- Extended to hybrid fault model by Rushby at SRI
- Generalized treatment for many algorithms by Shankar at SRI
- Significantly improved by Miner at NASA Langley, instantiated for different algorithms, carried down to circuit implementation
- Extended (for non-averaging algorithms), and applied to TTP (safety-critical automobile control) by Pfeifer, Schwier, von Henke at Ulm (using PVS, 1999)
- Ten person-years of cumulative effort, four institutions, three tools

Shared Applications: Center Activity

- Host selected complete applications
- Refine them as examples, libraries, useful to others

Shared Applications: Research Challenges

- Extracting reusable content from large verifications
- Reducing tours-de-force to routine applications
- Developing, supporting collaborative approaches to verification

  In AAMP verification, huge amount of time was spent doing diffs

Shared Components: Experiences

- Most systems use standard parser generators (Ergo, JavaCC), graphics (Tcl/Tk), editors, and window managers
  - But not much beyond that
- Our own systems are monolithic, not built from components, provide limited access to internal functions
- Example: use of VIS model checker by PVS
  - VIS Input is assumed to be a BLIF-MV design
  - Whereas we want access to the basic model checking functions
- Example: use of PVS by SCR
  - PVS assumes interactive operation
  - Whereas they just want decision procedures and automation

Shared Components: Center Activity

- Repository of useful building blocks
- Especially interface components
  - Input: front ends for RSML, Verilog, Java
  - Output: MSC generators, graphics

Shared Components: Engineering and Research Challenges

- Extracting deductive components from existing tools
- APIs for these components

  Best considered under integration

Integrated Approaches: Experiences

Front Ends

- Improved interfaces
  - IBM interface to SMV
- Customizations and embeddings
  - PC/DC: Duration calculus in PVS
  - TAME: IO automata in PVS
- Language support
  - TCAS (RSML) to SVC

Integrated Approaches: Experiences

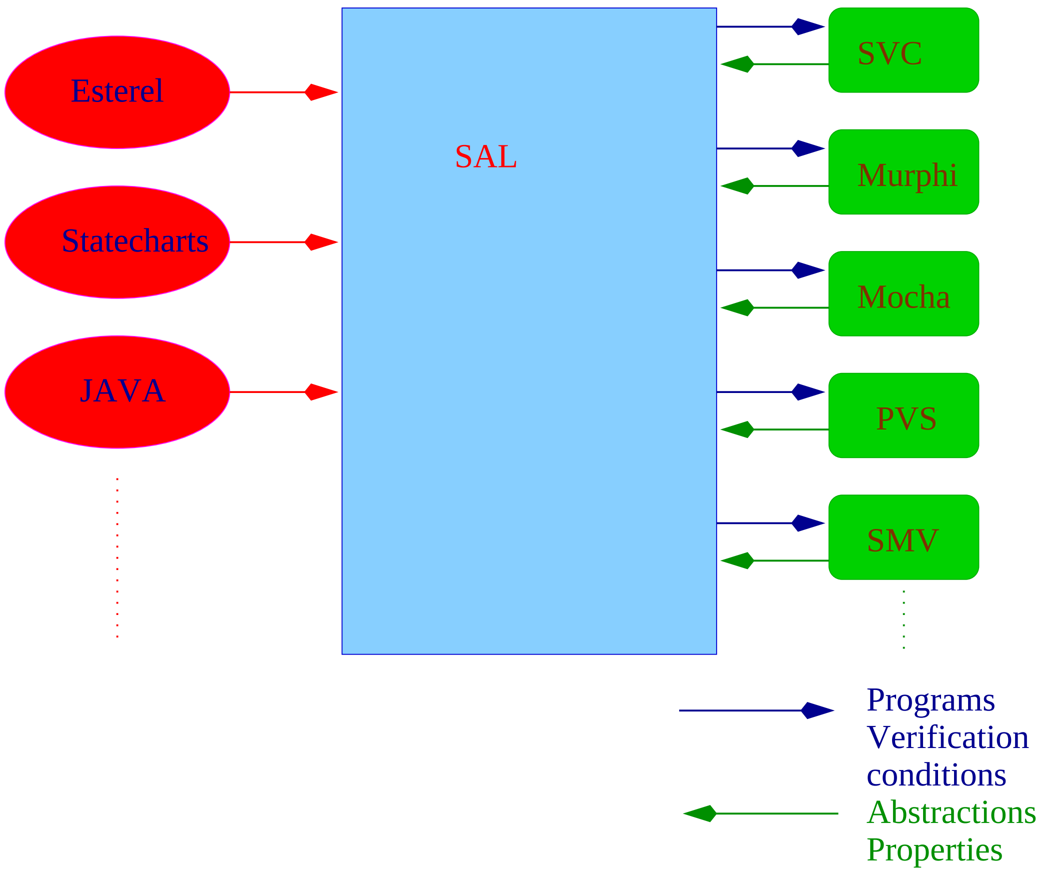
Back Ends

- General deductive support
  - SCR/PVS
  - Java/Spin
- Specialized deductions
  - BDDs, model checker, Mona, SVC, Lisp invoked from PVS

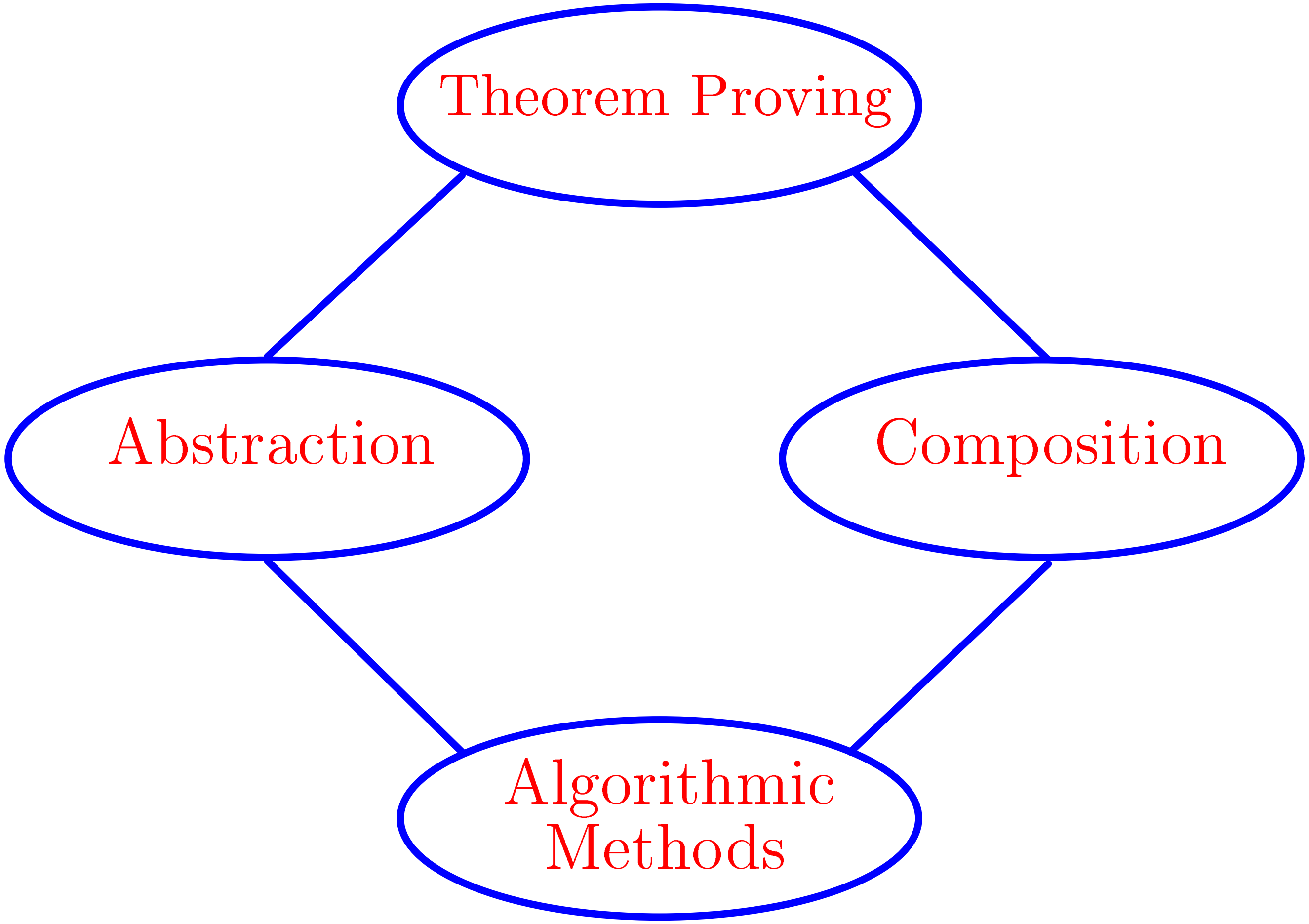
Cooperative use of multiple tools

- SAL (Symbolic Analysis Laboratory)
- VeriTech (Grumberg, Katz at The Technion)
- Newton Institute (Hoare, Martin, Paulson proposal)

Example of Cooperative Integration (SAL)



Another View of Cooperative Integration



Integration: Engineering and Research Challenges

- Open systems, component-based designs
- The right APIs
- Translating syntax (to common intermediate language(s))
- Connecting semantics (e.g., via abstract interpretation)
- Interpreting returned values (e.g., counterexamples)
- Access to intermediate values, not just verification outcomes
  - E.g., reachable stateset
- Ancillary tools (invariant generators, abstractors, concretizers)

Integrated Approaches: Center Activity

- Coordinate development of common APIs
  - E.g., to model checkers, SAT procedures, decision procedures, rewrite engines (C.f. BDDs for model checking)
- Coordinate development of common intermediate representations
  - E.g., for transition relations
  - E.g., for standard logics
- Customize/develop suitable translators and tool components to use these APIs and representations, and general support software
  - Counterexample and trace animators/simulators
  - Real world (Verilog, UML) to intermediate language translators
  - User-interface tools, language front-ends, VC-generators

Integration of Tools and Techniques: Benefits

- Currently
  - Few problems are handled satisfactorily by any single technique
  - Methodology lacking to guide appropriate combinations
  - Tools do not work together
  - Tools do not share components
  - Researchers must often reinvent, reconstruct existing stuff
  - Users are confused
- The Center will shift the focus of researchers and users
  - To development of components for integration in larger systems
  - To methodologies that employ a wide range of techniques

So people will build and use FM tools very differently than today