Darpa FM PI meeting, Ithaca NY May 12, 13 1999

Symbolic Analysis Laboratory (SAL)

Joint project with David Dill, Stanford

John Rushby (representing Shankar)

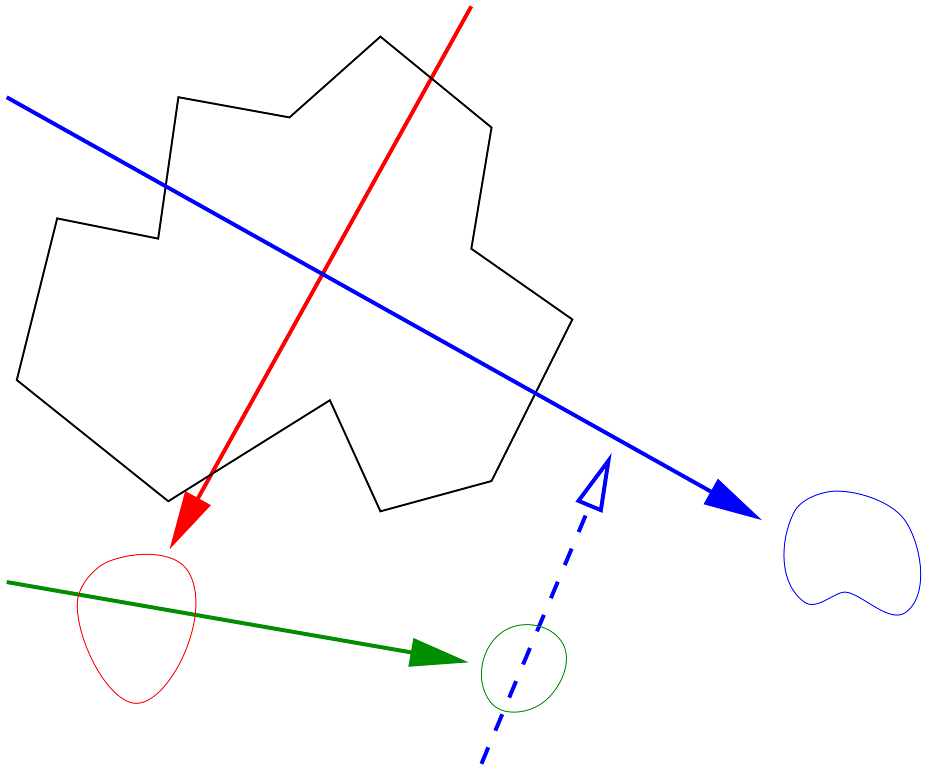
Computer Science Laboratory
SRI International
Menlo Park, CA

Recap of the Goals

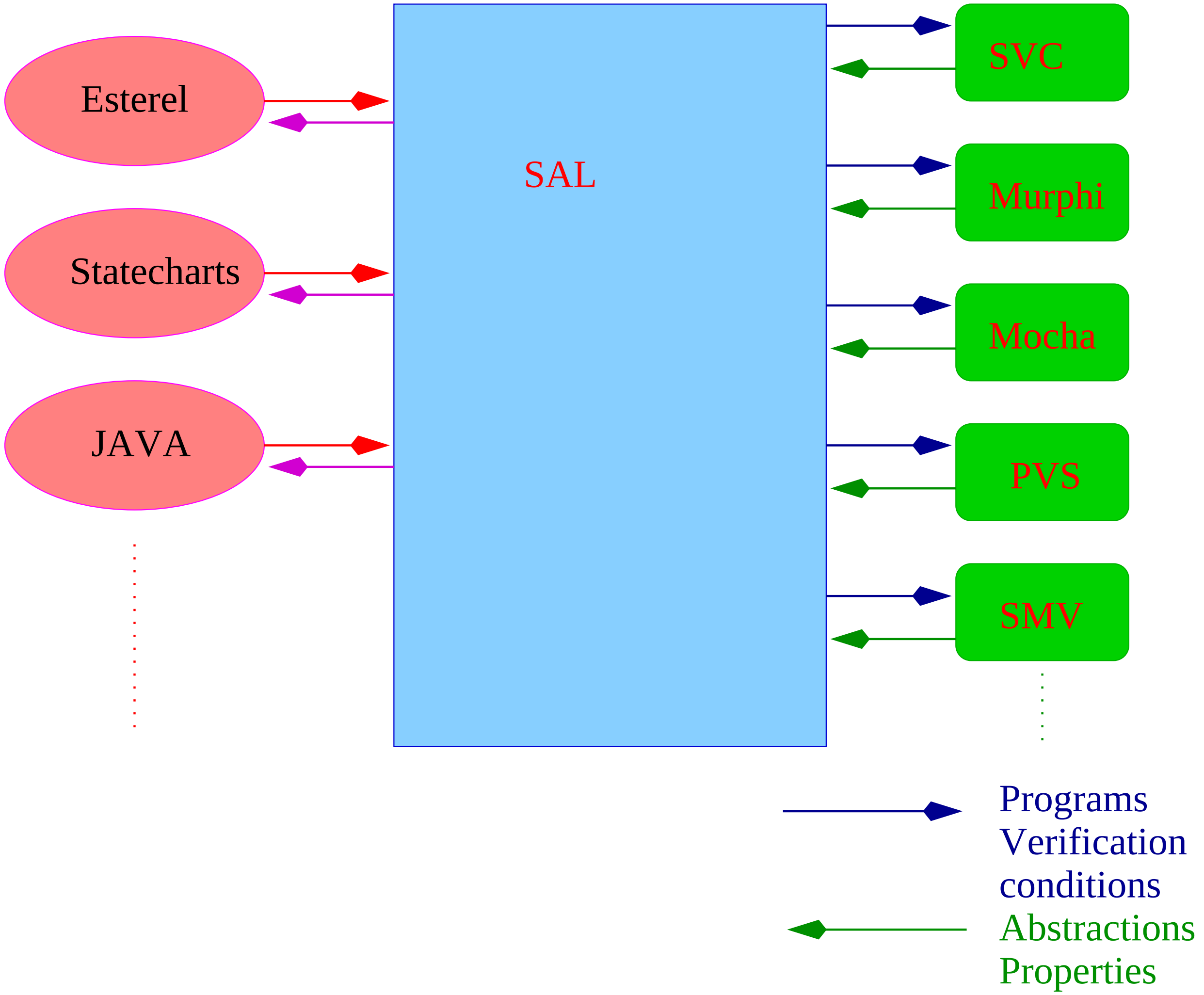
- Handle much bigger problems, provide much more automation, much greater efficiency, than currently
- Cut problems down to sizes that tools can handle efficiently
- And allow multiple tools to interoperate
- Through ubiquitous use of automated abstraction
- The SAL Language serves as a hub
- Abstractors create abstractions that tools can handle
- Concretizers return results that enable further analysis
    - Not just true/false but symbolic values
    - E.g., reachable stateset discovered by a model checker

Ubiquitous Automated Abstraction

- Generate models appropriate to different analyses and different tools from a single description
- Abstraction can be iterated
- Results from one analysis can assist further abstractions



SAL Architecture



Accomplishments: Languages and Translators

- SAL language definition (still undergoing refinement)
- Core of SAL language semantics formalized in PVS
- SAL front end developed in Java (also runs as an applet)
  - Parser
  - Automated translation to SMV
  - Automated translation to PVS
  - Experimental automated translation to Java (for animation)

  Will demo these

Accomplishments: Methodology

- Iterated abstraction and model checking during proof
  - Rusu and Singerman (TACAS 99)
- Verification of infinite-state systems by combining abstraction and reachability analysis
  - Abdulla, Annichini, Bensalem, Bouajjani, Habermehl and Lakhnech (CAV 99)
  - Collaboration with visitors from VERIMAG
  - Builds on InVeSt (Bensalem, Lakhnech, Owre, CAV 98)

Accomplishments: Abstractors

Developed efficient predicate abstractors

- New theory and implementations
- Saïdi and Shankar (CAV 99)
  - Available as a strategy in PVS 2.3
  - Solves BRP in 60 seconds
- Das, Dill and Park (CAV 99)
  - Integrated with SAL front end
  - Verifies on-the-fly GC

  Will demo this

Plans

- Compute, return, and exploit symbolic values
- Perform challenging examples that exploit iterative analysis
  - Multiple round trips
- Verifications that were *tours de force* should become routine
- Papers available at
  - `http://www.csl.sri.com/fm.html`
  - `http://sprout.stanford.edu`