R-Verify: Deep Checking of Embedded Code

James Ezick†
Donald Nguyen†
Richard Lethin†
Rick Pancoast*

(†) Reservoir Labs
(⁎) Lockheed Martin

The Eleventh Annual Workshop on
High Performance Embedded Computing

September 2007
R-Verify Design

Usable
• Simplifies software development process
  – Find bugs earlier (debugging)
  – Static identification of bug patterns (regression testing)
  – Verify entire application with components (integration testing)
• Integrates with existing workflows
• Visual counter-example presentation
• Extensible to any domain
• Verification expertise not required
  – No new language to learn
  – Properties written in source language

Powerful
• SAT based model checking technology
  – Modeling and deep semantic checking of complex systems
  – Aggressively optimized search
• Works with any programming style

Scalable
• Uses reasonable time and space
• Applies abstraction and refinement

R-Verify Supports:
• Pre- and post-condition checking including documented VSIPPL rules
• Memory safety of embedded device drivers, interrupt handlers, and VSIPPL “admitted” blocks
• Numerical precision of arithmetic pipelines with an emphasis on VSIPPL pipeline implementations
R-Verify Deep Checking Cycle

Development/Verification Cycle

Developer IDE

Program Error

Execution Trace

R-Stream IR

Constraint Encoding

Salt

Alef Input

Alef Solution
Built on Existing Reservoir Labs Technologies

- **R-Stream 3.0**
  - Can plug into existing development environments such as Eclipse
  - Industry-standard C front end
  - Parses source code and generates intermediate representation (IR)
  - Robust toolkit of compiler algorithms
    - Infer known loop bounds
    - Unroll loops and inline entire function calls
  - Supports classical optimizations
    - Constraint propagation
    - Dead code elimination
    - Partial subexpression elimination
    - Many more
  - Integrated IR visualization and reflection tools to aid reporting
  - Rich library of useful analyses and representations
    - Points-to analysis
    - Reachability analysis
    - Program dependency graphs
    - Augmented post-dominator tree

- **Salt 1.5**
  - Constraint intermediate language
    - Easy to target
  - Optimizes constraint representations
  - Supports logical, pseudo-Boolean, fixed point arithmetic and set constraints
  - Extensible to other constraint logics

- **Alef SAT Solver**
  - Parallel satisfiability solver
  - Targets large problem sizes
  - Improved performance by exploiting data and cooperative search parallelism

By basing R-Verify on mature, robust technologies we were able to:

- **Manage Risk**
- **Reduce development time, cost**
- **Focus on applications and usability rather than technology**