Lock Inference for Systems Software

John Regehr    Alastair Reid
University of Utah

March 17, 2003
Embedded Systems

- Properties:
  - Important
  - Resource constrained
  - Evolve significantly
  - Contain multiple execution environments
Execution Environments

- Sets of
  - Idioms and abstractions for structuring software
  - Rules for sequencing actions
  - Rules for sharing information

- Examples
  - Low-level: Cyclic executive, interrupts, threads, event loop
  - High-level: Dataflow graph, time triggered system, hierarchical state machines
Diversity in Execution Environments is...

◆ Good:
  - Diversity can be exploited
    - To create efficient systems
    - To match design problems

◆ Bad:
  - Environments have rules
  - Interacting environments have rules
Concurrency

- Embedded software is fundamentally concurrent
  - Interrupt driven
  - Response time requirements
- Critical sections are a functional aspect
  - But choice of lock implementation can be a non-functional aspect
Task Scheduler Logic (TSL)

- Formalizes locking concerns across execution environments
  - Currently unchecked
- Finds races and other errors
- Generates mapping from each critical section in a system to an appropriate lock
  - Lock inference
Why Infer Locks?

- Locking rules are hard to learn, hard to get right
- Sometimes no lock is needed
- Components can be agnostic with respect to execution environments
- Global side effects can be managed
TSL Concepts

- Tasks – units of computation
- Asymmetric preemption
  - $A \prec B$ means “$B$ may preempt $A$”
- Schedulers
  - $S \downarrow B$ means “$S$ schedules $B$”
- Locks
  - $S \downarrow L$ means “$S$ provides $L$”
  - $A \preceq L B$ means “$B$ may preempt $A$ while $A$ holds $L$”
Resources and Races

- **Resources**
  - \( A \rightarrow_{L} R \) means “A holds L while accessing R”

- **Race** (A, B, R) = \( A \rightarrow_{L_1} R \)
  \[ \wedge B \rightarrow_{L_2} R \]
  \[ \wedge A \neq B \]
  \[ \wedge A \ll_{L_1 \cap L_2} B \]
Applying TSL

- Applied to embedded monitoring system with web interface
  - 116 components
  - 1059 functions
  - 5 tasks
  - 2 kinds of locks + null lock
Summary

- Contributions
  - Reasoning about concurrency across execution environments
  - Automated lock inference
- Future work: Optimal lock inference
  - Minimize run-time overhead
  - Maximize chances of meeting real-time deadlines
More info and papers here:
http://www.cs.utah.edu/~regehr/