Applied Formal Methods Research at Queen's

Juergen Dingel
Queen's University
Kingston, Ontario, Canada

CSRC'03
May 26, 2003

Applied Formal Methods at Queen's

- **Theme:**

  Develop light-weight methods and tools that support the automatic formal analysis of software artifacts

- **Projects:**
  - **P1:** Assume/Guarantee reasoning with VeriSoft
  - **P2:** Analysis of Publish/Subscribe systems using model checking
  - **P3:** Run-time conformance checking of Java using Alloy

P1: A/G Reasoning with VeriSoft

- In **sequential** programming:
  - Run-time assertions to check invariants
  - Hoare-logic to prove invariants
- In **concurrent** programming:
  - VeriSoft to check invariants automatically, but non-compositionally
  - Assume/Guarantee reasoning to prove invariants compositionally, but not automatically
  - **Our work:** VeriSoft + Assume/Guarantee to check invariants compositionally and automatically

Assume/Guarantee Specifications

$$\underbrace{[P, \Gamma]}_{\text{assumptions}} \quad C \quad \underbrace{[Q, \Delta]}_{\text{guarantees}}$$

where

P = pre-condition

Q = post-condition

Γ = set of predicates to be preserved by environment of C

Δ = set of predicates preserved by C

- For instance:

[y odd, {x even, y odd}]

x := y+1

[x even, {x even, y even, y odd, …}]

Assume/Guarantee Reasoning

- Compositional proof system for A/G specifications [e.g., Jones83, Stirling88]
- For instance,

$$\frac{[P,\Gamma_1] \quad C_1 \quad [Q_1,\Delta_1] \qquad\qquad [P_2,\Gamma_2] \quad C_2 \quad [Q_2,\Delta_2]}{[P_1 \wedge P_2,\ \Gamma_1 \cup \Gamma_2\ ] \quad C_1 \| C_2 \quad [Q_1 \wedge Q_2,\ \Delta_1 \cap \Delta_2]} \textbf{ PAR}$$

where $\underbrace{\Gamma_1 \subseteq \Delta_2 \text{ and } \Gamma_2 \subseteq \Delta_1}$

"assumptions of one process are contained in guarantees of the other"

A/G Reasoning: Cons

Same problems as Hoare logic:

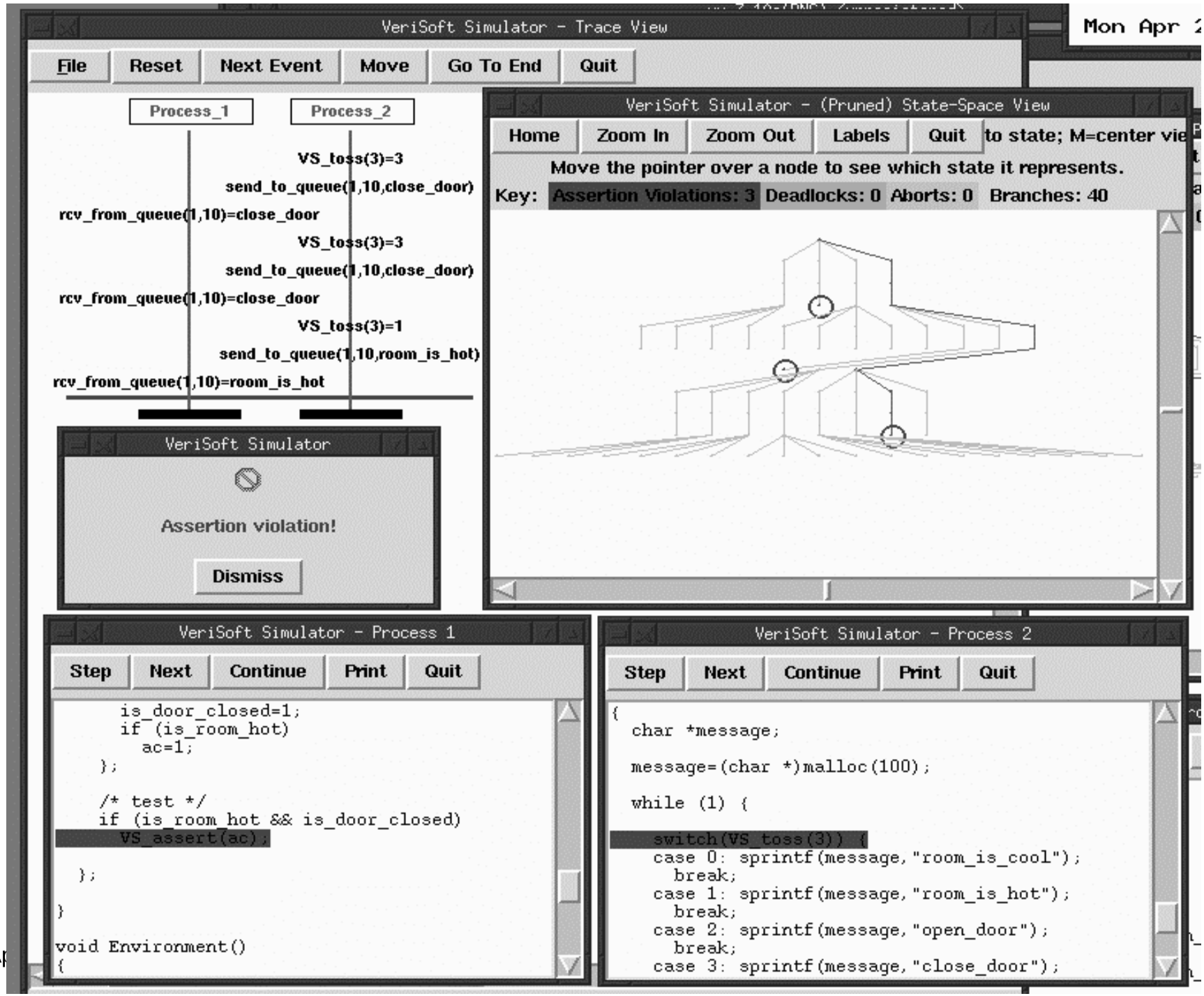
- does **not scale well** to large programs in modern languages
- available tools are **not light-weight**:
  - not fully automatic
  - require user to be expert
  - no gradual learning curve with immediate payoff

Our Work

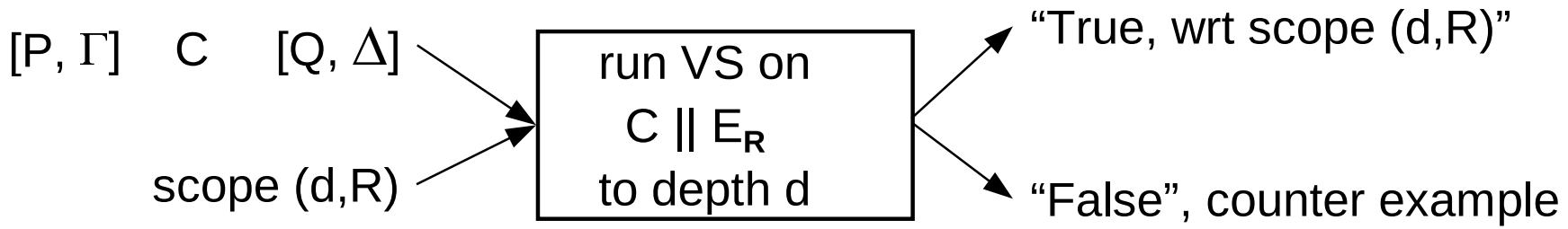
- A/G reasoning largely theoretical, **“pencil & paper” technique**
- Paper shows how
  - state-space exploration tool **VeriSoft** can be used to analyze C/C++ programs wrt **A/G specifications**
  - to make A/G reasoning **more light-weight and practical**

VeriSoft [Godefroid 1996]

- **Systematic testing/state space exploration tool** for concurrent C/C++ programs
- VeriSoft **exhaustively enumerates** all possible sequences of visible actions of concurrent program up to a user-defined depth
- Checks for **deadlocks**, **livelocks**, **divergences**, and **assertion violations**
- Support for **non-determinism**


VeriSoft Simulator - Trace View
File
Reset
Next Event
Move
Go To End
Quit
Process_1
Process_2
VS_toss(3)=3
send_to_queue(1,10,close_door)
rcv_from_queue(1,10)=close_door
VS_toss(3)=3
send_to_queue(1,10,close_door)
rcv_from_queue(1,10)=close_door
VS_toss(3)=1
send_to_queue(1,10,room_is_hot)
rcv_from_queue(1,10)=room_is_hot
VeriSoft Simulator
Assertion violation!
Dismiss
VeriSoft Simulator - (Pruned) State-Space View
Home
Zoom In
Zoom Out
Labels
Quit
to state; M=center vie
Move the pointer over a node to see which state it represents.
Key:
Assertion Violations: 3
Deadlocks: 0
Aborts: 0
Branches: 40
VeriSoft Simulator - Process 1
Step
Next
Continue
Print
Quit
is_door_closed=1;
if (is_room_hot)
ac=1;
};
/* test */
if (is_room_hot && is_door_closed)
VS_assert(ac);
};
}
void Environment()
{
VeriSoft Simulator - Process 2
Step
Next
Continue
Print
Quit
{
char *message;
message=(char *)malloc(100);
while (1) {
switch(VS_toss(3)) {
case 0: sprintf(message,"room_is_cool");
break;
case 1: sprintf(message,"room_is_hot");
break;
case 2: sprintf(message,"open_door");
break;
case 3: sprintf(message,"close_door");

Checking A/G Specs with VS



- d: depth of computation tree generated by VS
- R: range of values that VS can assign to shared vars
- E_R: environment process that
  - picks initial state in R satisfying P non-deterministically, and then
  - continuously
    - picks new state in R non-deterministically such that Γ is preserved, and
    - checks that state changes by C preserve Δ

Analyzing N-Process Tie-Breaker

- A/G analysis of C_i with VeriSoft:
  - for N=3 and depth=20: 35mins
- Monolithic analysis of TIE(n) with VeriSoft:
  - for N=3 and depth=20: >10hrs

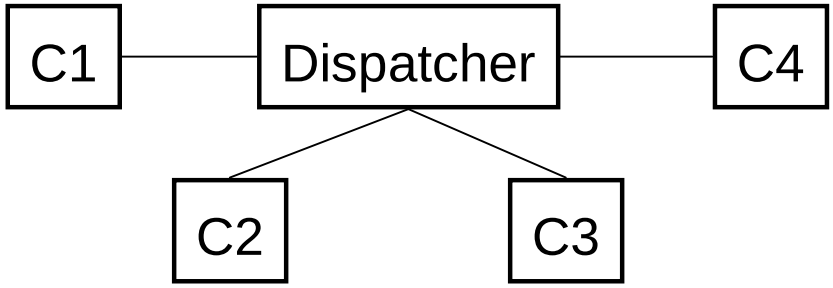
Summary

- Used VeriSoft for **compositional analysis** of concurrent C/C++ programs wrt A/G specifications
- Analysis bound to **user-defined depth and scope**

  $\Rightarrow$ lose completeness, but gain full automation
- **Open problems**:
  - state space explosion
  - supports only invariant-based reasoning
- ICSE'03

P2: Analyzing Pub/Sub Systems using Model Checking

- **Publish/Subscribe** paradigm widely used mechanism for system integration



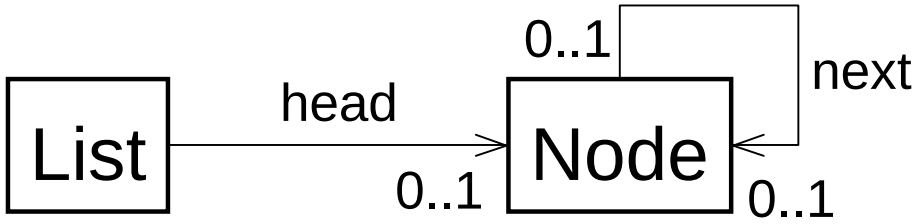
- **Behaviour** of system depends on, for instance:
  - presence/absence of events and event-method bindings
  - event delivery policy

P2: Analyzing Pub/Sub Systems using Model Checking (cont'd)

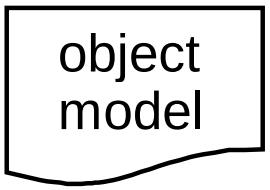
- **Problem**: Little support for analyzing pub/sub systems
- **Approach**:
  - translate description of pub/sub system into FSM
  - analyze FSM with Cadence SMV model checker
  - extension of Garlan & Khersonsky's work
- Joint work with Jeremy Bradbury
- FSE'03

P3: Run-time Conformance Checking of Java using Alloy

- **Object models** widely used in OO development methodologies




```
fact NoCyles {
    all l : List |
        no n : l.head.*next |
            n in n.^next
}
```

- **Question**: Does  satisfy its  ?

- **Problem**: Little support for answering this

P3: Conformance Checking of Java using Alloy (cont'd)

- **Approach**:
  - collect state information at user-specified breakpoints
  - use Alloy Analyzer to see whether state satisfies constraints expressed in the model
- Joint work with Michelle Crane
- submitted to RV'03

More Information

Web sites

- **Applied Formal Methods Group**

  www.cs.queensu.ca/~stl/afmg

- **Software Technology Lab**

  www.cs.queensu.ca/~stl

- **Juergen Dingel**

  www.cs.queensu.ca/~dingel