Formal Methods for SPIDER

Lee Pike¹

Reporting joint work with
Paul Miner¹ Jeffrey Maddalon¹
Alfons Geser² Radu Siminiceanu²

¹Formal Methods Group NASA Langley Research Center {lee.s.pike, paul.s.miner, j.m.maddalon}@larc.nasa.gov

²National Institute of Aerospace {geser, radu}@nianet.org

September 24, 2004



Introduction

Safety-Critical Systems

Overview

SPIDER: A Fault-Tolerant Communications Bus

Formal Methods

What are Formal Methods and Why Use Them? Tools of the Trade
Formal Methods for SPIDER

Conclusion





The failure of a *safety-critical system* has the potential to cause the loss of life or serious injury.



The failure of a *safety-critical system* has the potential to cause the loss of life or serious injury.

Such systems are found in

Aircraft







The failure of a *safety-critical system* has the potential to cause the loss of life or serious injury.

Such systems are found in

- Aircraft
- Automobiles







The failure of a *safety-critical system* has the potential to cause the loss of life or serious injury.

Such systems are found in

- Aircraft
- Automobiles
- ► Trains and Rail Systems









The failure of a *safety-critical system* has the potential to cause the loss of life or serious injury.

Such systems are found in

- Aircraft
- Automobiles
- ► Trains and Rail Systems
- Medical Devices













Safety-critical system should have catastrophic failure rates no higher than 10^{-7} to 10^{-12} per hour of operation.



Safety-critical system should have catastrophic failure rates no higher than 10^{-7} to 10^{-12} per hour of operation.

Just how low is that?



Safety-critical system should have catastrophic failure rates no higher than 10^{-7} to 10^{-12} per hour of operation.

Just how low is that?

▶ 10⁹ hours is about 114,000 years.



Safety-critical system should have catastrophic failure rates no higher than 10^{-7} to 10^{-12} per hour of operation.

Just how low is that?

- ▶ 10⁹ hours is about 114,000 years.
- ▶ Odds of being hit by lightning in a year: 1 in 240,000 $(1/2.4 \times 10^{-5}/\text{year})$.





Safety-critical system should have catastrophic failure rates no higher than 10^{-7} to 10^{-12} per hour of operation.

Just how low is that?

- ▶ 10⁹ hours is about 114,000 years.
- ▶ Odds of being hit by lightning in a year: 1 in 240,000 $(1/2.4 \times 10^{-5}/\text{year})$.
- ▶ Odds of dying in a car wreck per trip: 1 in 4 million $(1/4 \times 10^{-6}/\text{trip})$.







► Increased deployment.



- ► Increased deployment.
- Increased sophistication.





- Increased deployment.
- Increased sophistication.
- Increased integration.





- Increased deployment.
- Increased sophistication.
- Increased integration.

Leading to increased risk of catastrophic failure.





- Increased deployment.
- Increased sophistication.
- Increased integration.

Leading to increased risk of catastrophic failure.

Every day, our lives are in the hands of computers.





SPIDER: What Is It?

Scalable Processor-Independent Design for Enhanced Reliability



A means to safely integrate embedded systems and partition faults.



Integrating interdependent applications of differing criticality in extreme safety-critical environments.



Integrating interdependent applications of differing criticality in extreme safety-critical environments.

For use in

► Long-term environments (e.g., space-exploration craft).





Integrating interdependent applications of differing criticality in extreme safety-critical environments.

For use in

- Long-term environments (e.g., space-exploration craft).
- Highly-automated environments (e.g., unpiloted air vehicles).







Integrating interdependent applications of differing criticality in extreme safety-critical environments.

For use in

- ▶ Long-term environments (e.g., space-exploration craft).
- ▶ Highly-automated environments (e.g., unpiloted air vehicles).
- Highly-integrated environments (e.g., tomorrow's commercial aircraft).

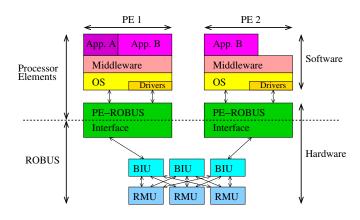








Architecture





ROBUS Protocols

Highly fault-tolerant protocols implemented for

- Initial Clock Synchronization
- Clock Resynchronization
- Distributed Diagnosis
- Interactive Consistency
- Reintegration





What are Formal Methods?

Formal methods is a field in which formal mathematical techniques are applied to ensure the correctness of digital systems.

$$\frac{\varepsilon_3}{\varepsilon_1} = \frac{A'}{A^2} \beta^2$$

$$\varepsilon_1 = \left(\frac{A}{A+1}\right)^2 E_1$$

$$\mu_3 = \mu$$

$$\frac{\varepsilon_4}{\varepsilon_1} = \frac{A'}{A+1-A'} \frac{\varepsilon_3}{\varepsilon_1}$$

$$\mu_4 = \mu$$







▶ Design errors could dramatically and unexpectedly raise the failure rate of a system.



- ▶ Design errors could dramatically and unexpectedly raise the failure rate of a system.
- ▶ To determine whether a system meets a failure rate of 10^{-12} /hour by testing, you'd need to test 1 trillion systems in parallel for an hour.





- ▶ Design errors could dramatically and unexpectedly raise the failure rate of a system.
- ▶ To determine whether a system meets a failure rate of 10^{-12} /hour by testing, you'd need to test 1 trillion systems in parallel for an hour.

Design assurance via testing is infeasible for safety-critical systems.





A mechanical theorem-prover is a tool supporting formal specification and mathematical proofs about the specifications.

As powerful as mathematics, in principle.



- As powerful as mathematics, in principle.
- As difficult as mathematics, and exacerbated by





- As powerful as mathematics, in principle.
- As difficult as mathematics, and exacerbated by
 - ► Low-level proof rules.
 - ► The complexity of specifying digital systems.



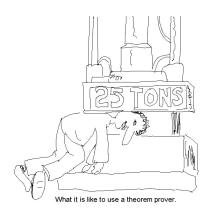


- As powerful as mathematics, in principle.
- As difficult as mathematics, and exacerbated by
 - Low-level proof rules.
 - ► The complexity of specifying digital systems.
- Little feedback provided for why a proof fails (which is a common occurance).





In Other Words...





A model-checker is a tool that automatically explores the state-space generated from a system specification.



A model-checker is a tool that automatically explores the state-space generated from a system specification.

Automated!



A model-checker is a tool that automatically explores the state-space generated from a system specification.

- Automated!
- Can only handle relatively small systems (or sparse abstractions of real systems).





A model-checker is a tool that automatically explores the state-space generated from a system specification.

- Automated!
- ► Can only handle relatively small systems (or sparse abstractions of real systems).
- Model-checking languages are constrained.



(Interactive and Automated) Compilation

Interpreters and compilers are so ubiquitous that they are rarely thought of as formal methods, but they are.



(Interactive and Automated) Compilation

Interpreters and compilers are so ubiquitous that they are rarely thought of as formal methods, but they are.

► An interactive compiler allows the user to guide the compilation (e.g., optimizations for a specific architecture).



(Interactive and Automated) Compilation

Interpreters and compilers are so ubiquitous that they are rarely thought of as formal methods, but they are.

- ► An interactive compiler allows the user to guide the compilation (e.g., optimizations for a specific architecture).
- ► The design space of compilation is more restrictive than theorem-proving.





All three methods are used for SPIDER. The specific tools used are

▶ <u>PVS</u>, a mechanical theorem-prover developed at SRI.



- ▶ <u>PVS</u>, a mechanical theorem-prover developed at SRI.
- ► <u>SAL</u>, a (family of) model-checkers developed at SRI.





- PVS, a mechanical theorem-prover developed at SRI.
- ► <u>SAL</u>, a (family of) model-checkers developed at SRI.
- ► <u>SMART</u>, a model-checker developed at William & Mary.





- <u>PVS</u>, a mechanical theorem-prover developed at SRI.
- SAL, a (family of) model-checkers developed at SRI.
- ► <u>SMART</u>, a model-checker developed at William & Mary.
- <u>DRS</u>, an interactive hardware compiler developed at Derivation Systems, Inc. (based on research by Steve Johnson, Indiana Univ., Bloomington).





PVS Screenshot

```
PVS File Edit Cotions Buffers Tools Complete In/Out Stateds Help
       8 x 0 6 3 7 7 0 8 6 6 6 8 8
        subset_filter: LEMMA subset?(filter(rel)(select)(f)(i).select)
         \begin{array}{lll} filter\_sus\_oard: LEMMs \\ & cord(filter\_(rel.)(colect)(f)(i)) = \\ & cord(filter\_(rel.)(colect)(f)(i)) = \\ & cord(select) - cord(filter\_(LAMM)s (x,y; real) : NOT rel(x,y)) (colect)(f)(i)) \end{array} 
    t le complement: LTMMA (LAMENA (x, y real) NOT x (= y) = ())
t ge complement: LTMMA (LAMENA (x, y real) NOT x (= y) = ()
       middle value(select)(f)(i): bool - & returns true if f(i) is a middle value
           iddle mana(mast., (r(s))
select() {
    select() {
    cord(filter(*)(select)(f)(i)) > card(filter(*)(select)(f)(i)) @
    card(filter(**)(select)(f)(i)) >= card(filter(*)(select)(f)(i))
      middle_value_mejority_ther: 12888
middle_value(selecti(f)(i) PP
select(i) &
2*sard(filter(**)(selecti(f)(i)) > card(select) &
2*sard(filter(**)(selecti(f)(j)) >= card(select)
       middle value exists: ATION
TRISTS i: middle_value(select)(f)(i)
       middle_value_translation: LERGA (Fronti i: f1(i) >= f2(i)) a middle_value(select) (f1) (j) a middle_value(select) (f2) (i) = 11(i) >= f2(i)
    - mid value index select.ovs 7 1796 1 30 Emil (PVS ready CVS 1 1 1 1)-L09-C7-576------
    middle value majority char
    Fule? (skemimp*)
Empeatedly Solemining and flattening,
this simplifies to
middle_value_majority_char :
     Rule? (assert)
Simplifying, rewriting, and recording with decision procedures,
this simplifies to:
     middle value majority char
      | | middle value(select(1)(f(1)(i(1) IFF
               select[iii] &

2 * card(filter(=) (select[i)(f|i)(i|1)) > card(select[i) &

2 * card(filter(=) (select[i)(f|i)(i|1)) >= card(select[i) &

2 * card(filter(=) (select[i)(f|i)(i|1)) >= card(select[i) &
    ** ***
                           7 17PM 1 30 Nail (ELISP ceads) -- LSD -- C6--Bat--
```







SAL Screenshot

```
File Edit Options Buffers Tools Help
    0 8 × 0 6 × × 0 0 0 0 0 0 0 0 0
          [ pc = 1 AND (6 status = good (6 6_status = symmetric) -- 6_out = ([s:NEEs] correct_v]
              pc = 1 AND 0_status = henign -->
0_out' = [[i:NNTe] rec_err]
              po = 1 800 0 status = asymmetrio -->
0 out IN (a. RWEC | 750R)
        II ---
          DD:
            CONTROL CONTRO
          CHU[1: NMTH]: HUULE -
HEGIN
INGUT
                po STAIR.
T in WALS.
TT PWILTS.
            T ACC ACC
            TOOK WED
              pc = 2 AND rf = qued AND r_in = rec_err -->
r_out' = [[p:BIWe] src_err]
          % below is the IC protocol fix. Gamment out the above statement % and uncommunent the 4 lines below to obtain the new protocol.
          t po = 2 AND cf = good AND (c_in = res_err (R r_sco = scoused) -->
t c_out' = [ip #20s] sco ecc|
            Do = 2 AND rf = good AND r_in /= rec_err
          % AND r_sec /* secused
                      r_out' = [[p:8100] r_in]
              pc = 2 AND rf = benign -->
r_out' = [[p:BIWe] rec_err]
          ([) (x.mars): bo = 5 vs. it = elementic -->
              pc = 2 AND rf = asymmetric -->
r_out' IN (a: WWEC | THEME)
            II ELSE -->
        count(s: EMEC. v: WALS. h_soc: B_ACC): ALL = count_h(s. v. R. h_soc):
          -- robus_io.sal 7 21PH 2 02 NS11 (SML)--LT1--ED--18%---
```







SMART Screenshot

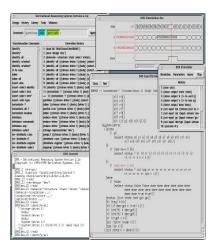
```
Elle Edit View Terminal Go Help
    FILE: phils.sn
   int("*** The Dining Ph
   int("using MDDs (set t
                               ining philosophers model. 2 philosophers/partition.
                                866,560,404,888,397,142,632,592,186,940,314,984,533,562,080,744,056,513,742,
                                 20,646,066,531,852,701,443,005,360,123,819,468,884,232,667,787,581,404,156,
   bigint count := nun s
                               .171.592.409.137.345.858.300.659.147.113.999.038.258.653.453.423.060.811.808.0
                               radu@seahorse Progs]$ [
```







DRS Screenshot







How are Formal Methods Applied to the Design of SPIDER?

- ► Theorem-Proving is used to specify and verify fault-tolerant protocols.
- Model-checking is used to specify and verify fault-tolerant protocols.
- Interactive compilation is used to derive hardware for individual system nodes.





► A fault-tolerance library for theorem-proving.





- ► A fault-tolerance library for theorem-proving.
- Better abstractions for theorem-proving/model-checking.





- ► A fault-tolerance library for theorem-proving.
- ▶ Better abstractions for theorem-proving/model-checking.
- ▶ Formal methods tool integration.





- ► A fault-tolerance library for theorem-proving.
- Better abstractions for theorem-proving/model-checking.
- ► Formal methods tool integration.
- Technology transfer to government and industry.





Summary

Safety-critical systems are becoming more ubiquitous, integrated, and complex.



Summary

- Safety-critical systems are becoming more ubiquitous, integrated, and complex.
- ► Formal methods can greatly facilitate the design and validation of safety-critical systems.





Summary

- Safety-critical systems are becoming more ubiquitous, integrated, and complex.
- ► Formal methods can greatly facilitate the design and validation of safety-critical systems.
- ► Formal methods are not a panacea, but they increase the assurance of correct design.





Resources

SPIDER Project

http://shemesh.larc.nasa.gov/fm/spider/

Google: formal methods spider

NASA Langley Research Center Formal Methods Group

http://shemesh.larc.nasa.gov/fm/

Google: nasa formal methods





Resources (Cont.)

Formal Methods Virtual Library

http://www.afm.sbu.ac.uk/

Google: formal methods

Formal Methods and the Certification of Critical Systems by John Rushby

http://www.csl.sri.com/papers/csl-93-7/

Google: rushby certification formal methods





Formal Methods Tool Websites

PVS

```
http://pvs.csl.sri.com/
```

Google: pvs sri

SAL

```
http://sal.csl.sri.com/
```

Google: sal sri

SMART

http://www.cs.wm.edu/~ciardo/SMART/

Google: smart model checking

DRS

http://www.derivation.com/

Google: dsi derivation

