Formal Methods for SPIDER

Lee Pike[1]

Reporting joint work with

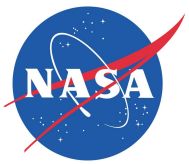
Paul Miner[1] Jeffrey Maddalon[1]

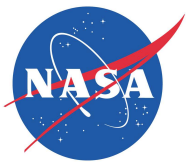
Alfons Geser[2] Radu Siminiceanu[2]

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

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

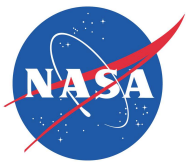
September 24, 2004





What is a Safety-Critical System?

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

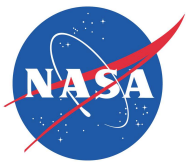


What is a Safety-Critical System?

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



What is a Safety-Critical System?

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

What is a Safety-Critical System?

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

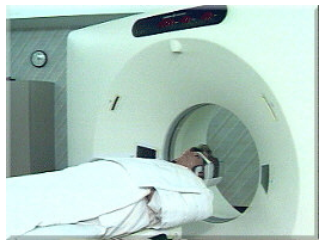
What is a Safety-Critical System?

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

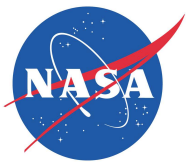
Such systems are found in

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



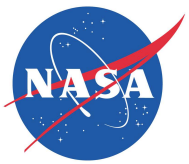


Quantifying "Safety-Critical"



Quantifying "Safety-Critical"

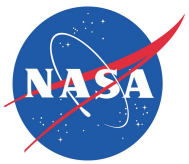
Safety-critical system should have catastrophic failure rates no higher than 10^{-7} to 10^{-12} per hour of operation.



Quantifying "Safety-Critical"

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?

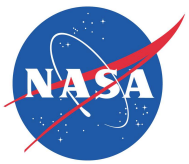


Quantifying "Safety-Critical"

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?

- 10^9 hours is about 114,000 years.

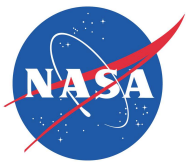


Quantifying "Safety-Critical"

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?

- 10^9 hours is about 114,000 years.
- Odds of being hit by lightning in a year: 1 in 240,000 ($1/2.4 \times 10^{-5}$/year).

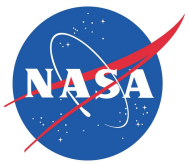


Quantifying "Safety-Critical"

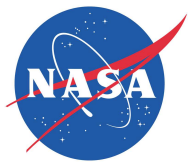
Safety-critical system should have catastrophic failure rates no higher than 10^{-7} to 10^{-12} per hour of operation.

Just how low is that?

- 10^9 hours is about 114,000 years.
- Odds of being hit by lightning in a year: 1 in 240,000 ($1/2.4 \times 10^{-5}$/year).
- Odds of dying in a car wreck per trip: 1 in 4 million ($1/4 \times 10^{-6}$/trip).

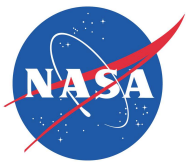


Current Trends in Safety-Critical Embedded Systems



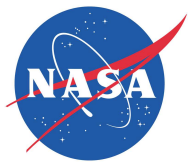
Current Trends in Safety-Critical Embedded Systems

- Increased deployment.



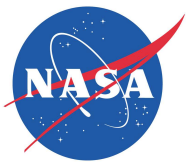
Current Trends in Safety-Critical Embedded Systems

- Increased deployment.
- Increased sophistication.



Current Trends in Safety-Critical Embedded Systems

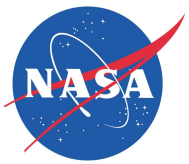
- Increased deployment.
- Increased sophistication.
- Increased integration.



Current Trends in Safety-Critical Embedded Systems

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

Leading to increased risk of catastrophic failure.

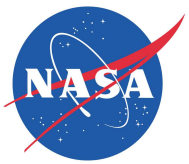


Current Trends in Safety-Critical Embedded Systems

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

Leading to increased risk of catastrophic failure.

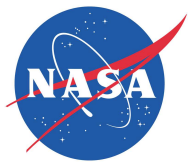
Every day, our lives are in the hands of computers.



SPIDER: What Is It?

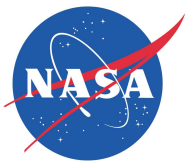
Scalable Processor-Independent Design for Enhanced Reliability

A means to safely integrate embedded systems and partition faults.



The Need for SPIDER

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

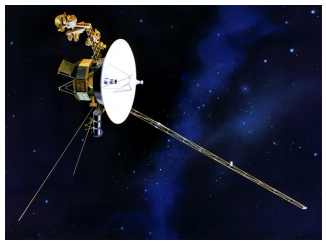


The Need for SPIDER

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

For use in

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

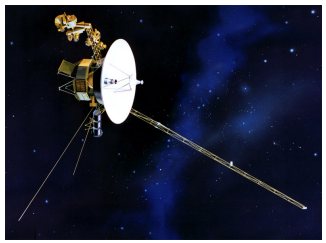


The Need for SPIDER

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).

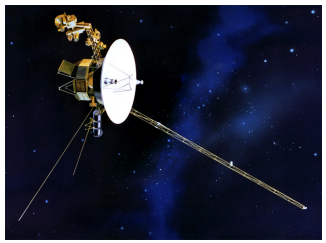


The Need for SPIDER

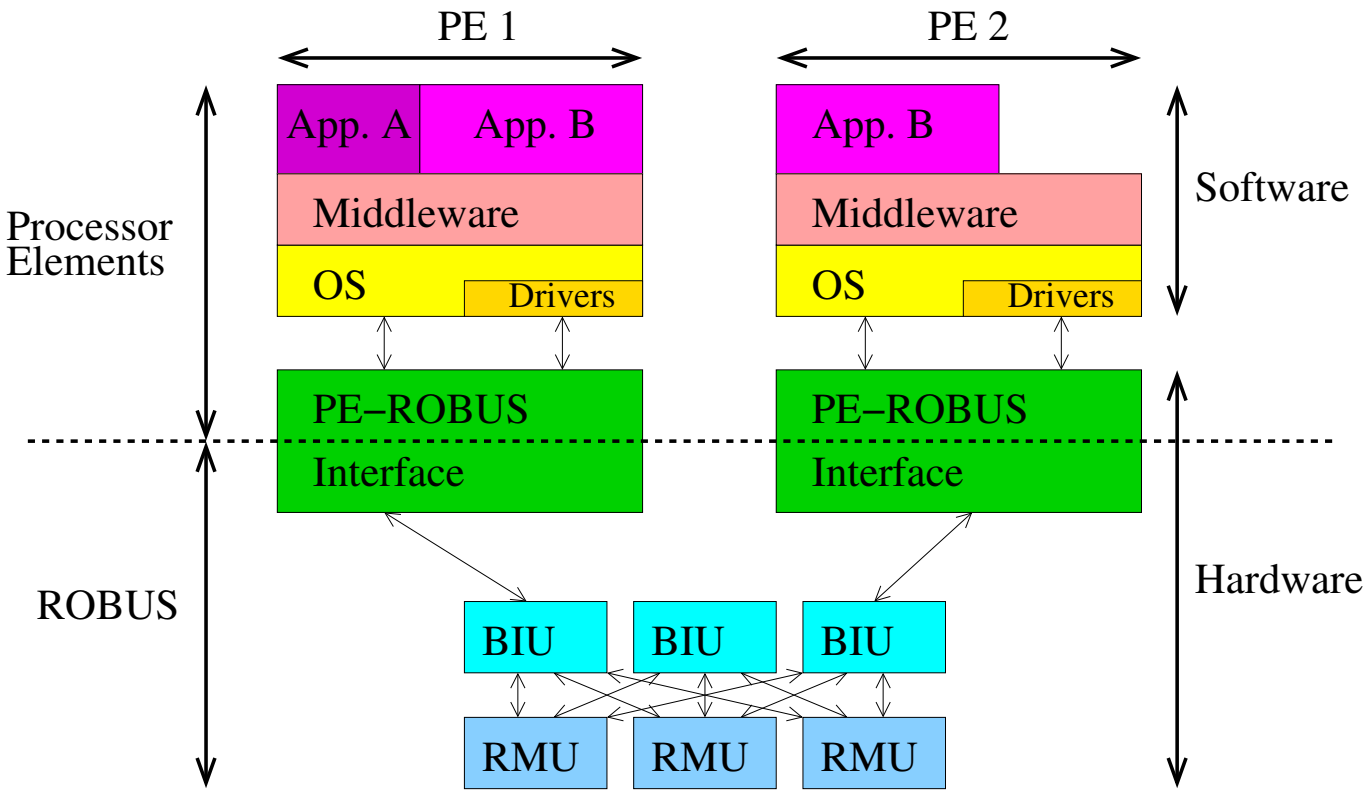
Integrating interdependent applications of differing criticality in extreme safety-critical environments.

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).



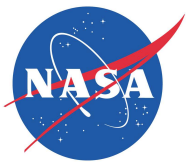
Architecture



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.

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

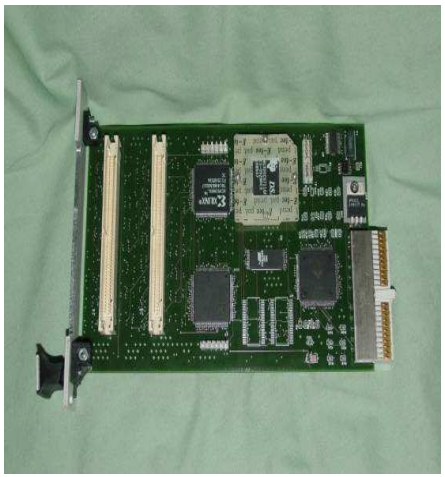
$$\varepsilon_1 = \left(\frac{A}{A+1}\right)^2 E_1$$

$$\mu_3 = \mu$$

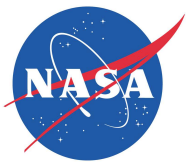
$$\frac{\varepsilon_4}{\varepsilon_1} = \frac{A'}{A+1-A'}\frac{\varepsilon_3}{\varepsilon_1}$$

$$\mu_4 = \mu$$

$\Longleftrightarrow$

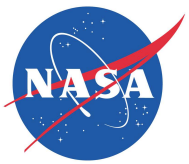


Why Use Formal Methods in the Design of Safety-Critical Systems?



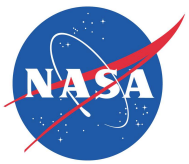
Why Use Formal Methods in the Design of Safety-Critical Systems?

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



Why Use Formal Methods in the Design of Safety-Critical Systems?

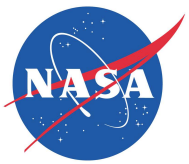
- Design errors could dramatically and unexpectedly raise the failure rate of a system.
- To determine whether a system meets a failure rate of 10^{-12}/hour by testing, you'd need to test 1 trillion systems in parallel for an hour.



Why Use Formal Methods in the Design of Safety-Critical Systems?

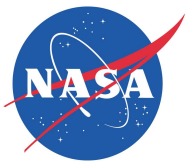
- Design errors could dramatically and unexpectedly raise the failure rate of a system.
- To determine whether a system meets a failure rate of 10^{-12}/hour by testing, you'd need to test 1 trillion systems in parallel for an hour.

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



Mechanical Theorem-Proving

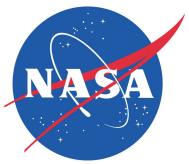
A mechanical theorem-prover is a tool supporting formal specification and mathematical proofs about the specifications.



Mechanical Theorem-Proving

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

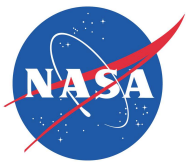
- As powerful as mathematics, in principle.



Mechanical Theorem-Proving

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

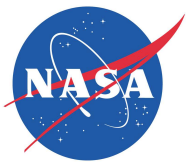
- As powerful as mathematics, in principle.
- As difficult as mathematics, and exacerbated by



Mechanical Theorem-Proving

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

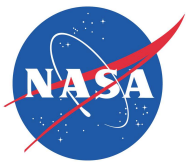
- As powerful as mathematics, in principle.
- As difficult as mathematics, and exacerbated by
  - Low-level proof rules.
  - The complexity of specifying digital systems.



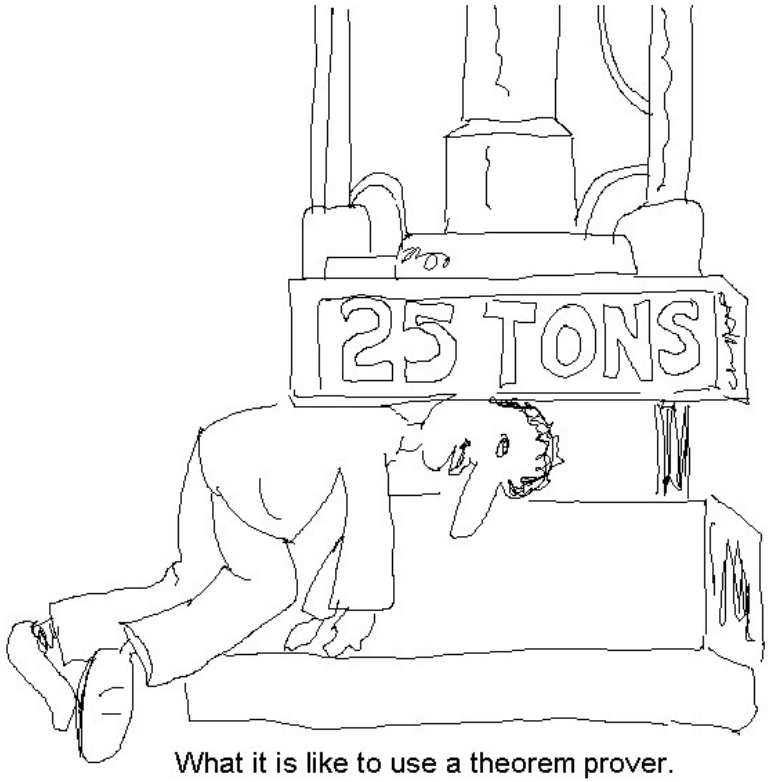
Mechanical Theorem-Proving

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

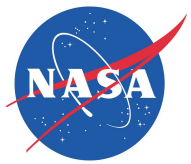
- As powerful as mathematics, in principle.
- As difficult as mathematics, and exacerbated by
  - Low-level proof rules.
  - The complexity of specifying digital systems.
- Little feedback provided for why a proof fails (which is a common occurance).



In Other Words...

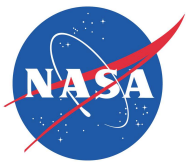


What it is like to use a theorem prover.



Model-Checking

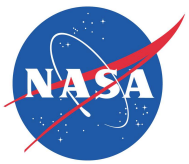
A model-checker is a tool that automatically explores the state-space generated from a system specification.



Model-Checking

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

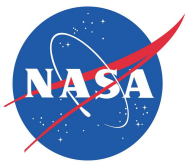
- Automated!



Model-Checking

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

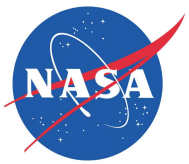
- Automated!
- Can only handle relatively small systems (or sparse abstractions of real systems).



Model-Checking

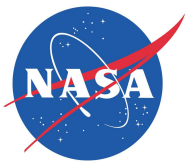
A model-checker is a tool that automatically explores the state-space generated from a system specification.

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



(Interactive and Automated) Compilation

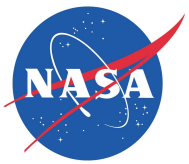
Interpreters and compilers are so ubiquitous that they are rarely thought of as formal methods, but they are.



(Interactive and Automated) Compilation

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

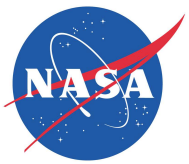
- An interactive compiler allows the user to guide the compilation (e.g., optimizations for a specific architecture).



(Interactive and Automated) Compilation

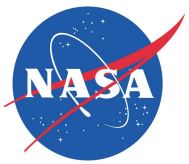
Interpreters and compilers are so ubiquitous that they are rarely thought of as formal methods, but they are.

- 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.



Which methods?

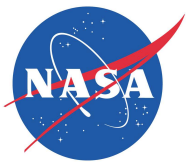
All three methods are used for SPIDER. The specific tools used are



Which methods?

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

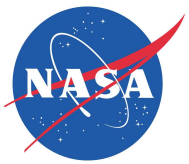
- PVS, a mechanical theorem-prover developed at SRI.



Which methods?

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

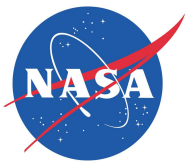
- PVS, a mechanical theorem-prover developed at SRI.
- SAL, a (family of) model-checkers developed at SRI.



Which methods?

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

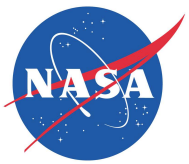
- PVS, a mechanical theorem-prover developed at SRI.
- SAL, a (family of) model-checkers developed at SRI.
- SMART, a model-checker developed at William & Mary.



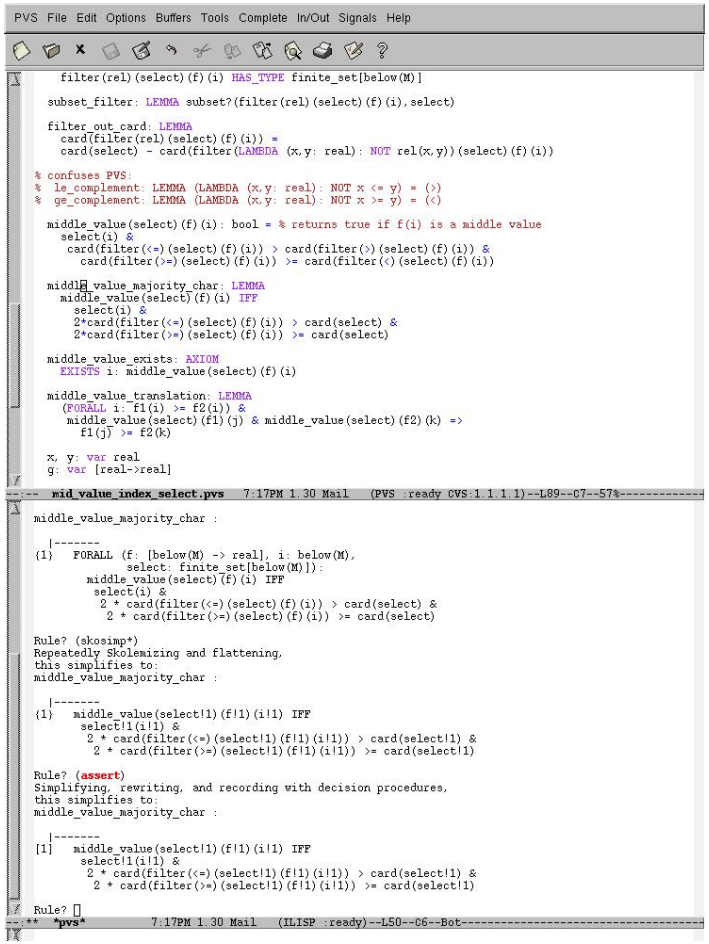
Which methods?

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

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

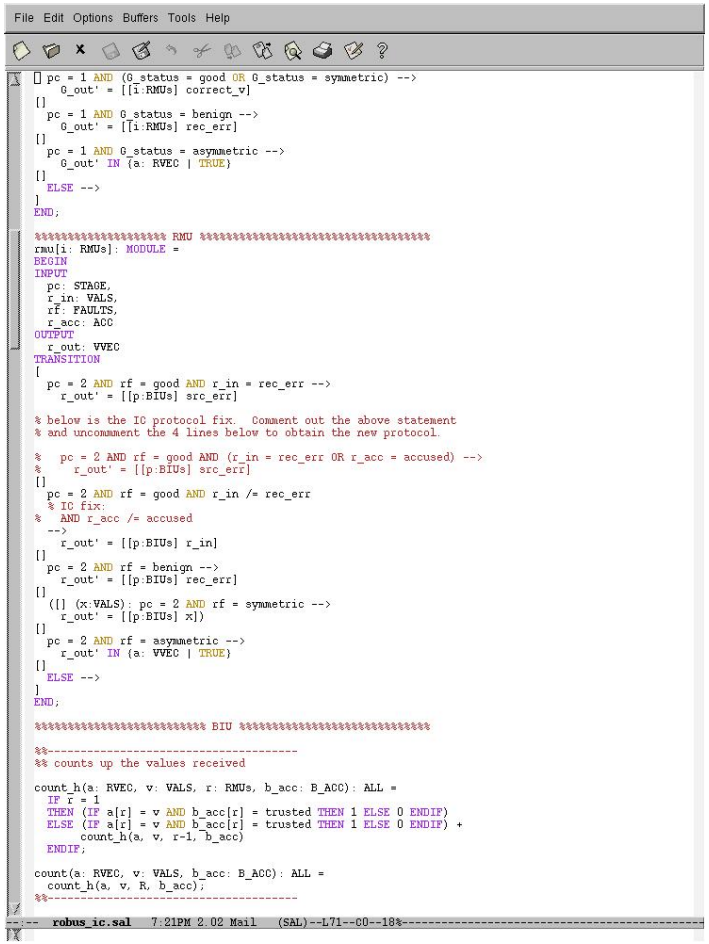


PVS Screenshot



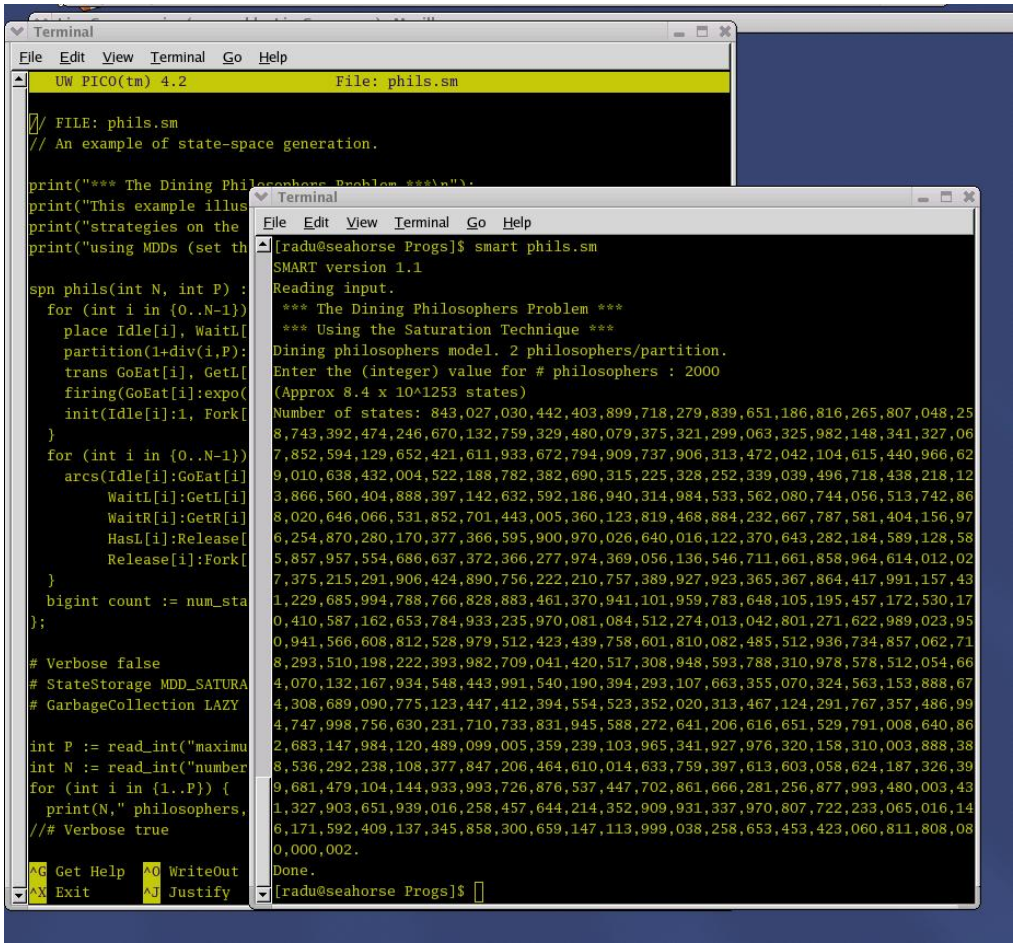
▶ Return

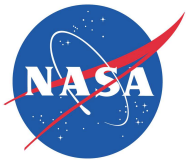
SAL Screenshot



▶ Return

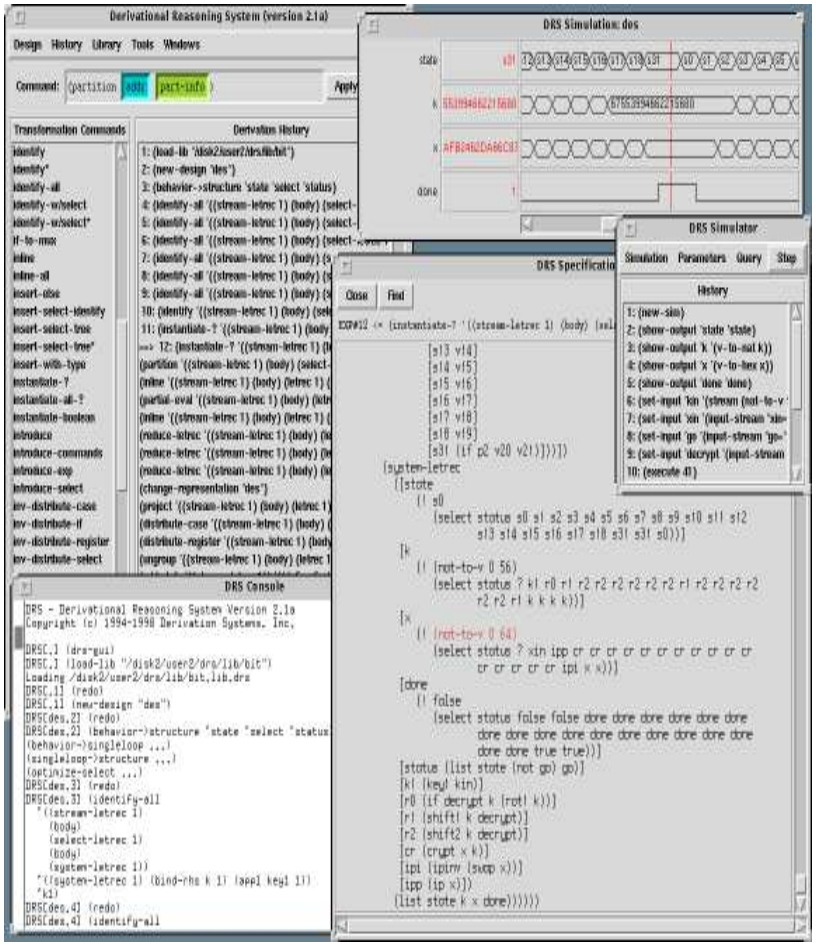
SMART Screenshot





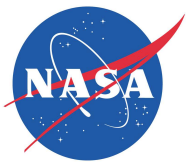
Return

DRS Screenshot



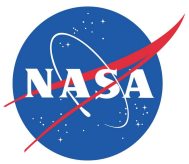
How are Formal Methods Applied to the Design of SPIDER?

- 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.



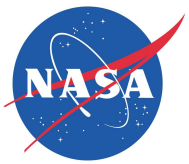
Ongoing Work in Formal Methods

- A fault-tolerance library for theorem-proving.



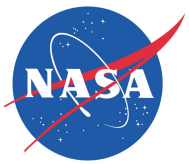
Ongoing Work in Formal Methods

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



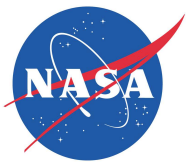
Ongoing Work in Formal Methods

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



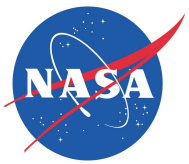
Ongoing Work in Formal Methods

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



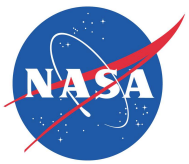
Summary

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



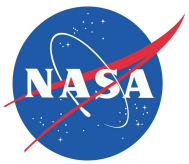
Summary

- 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

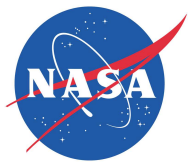
`http://shemesh.larc.nasa.gov/fm/spider/`

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

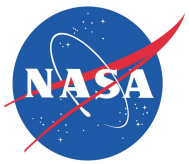
`http://www.afm.sbu.ac.uk/`

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