Will This Be Formal?

Dr. Steven P. Miller
July 15, 2008
Presentation Overview

What Problem are We Solving?

Who Are We?

What are Formal Methods?

Examples of Using Formal Methods

Challenges and Future Directions
What Problem Are We Solving?

- **Increasing Size and Complexity of Critical Systems**
  - Safety critical, security critical, and mission critical
  - Exponential growth in size and complexity

- **Rapidly Growing Cost of Verification**
  - Exponential growth in cost
  - Becoming the limiting factor in deployment
Airborne Software Doubles Every Two Years

Similar Growth Has Been Seen by Boeing

Complexity

<table>
<thead>
<tr>
<th>Object Code</th>
<th>Year</th>
<th>No. of Signals</th>
</tr>
</thead>
<tbody>
<tr>
<td>747-200</td>
<td>1970</td>
<td>0</td>
</tr>
<tr>
<td>757/767</td>
<td></td>
<td></td>
</tr>
<tr>
<td>747-400</td>
<td>1995</td>
<td>230K</td>
</tr>
</tbody>
</table>

Size

<table>
<thead>
<tr>
<th>Object Code</th>
<th>Year</th>
<th>Object Code (Mbytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>747-200</td>
<td>1970</td>
<td>0</td>
</tr>
<tr>
<td>757/767</td>
<td></td>
<td></td>
</tr>
<tr>
<td>747-400</td>
<td>1995</td>
<td>100</td>
</tr>
</tbody>
</table>
DoD software is growing in size and complexity

Total Onboard Computer Capacity (OFP)


Robert Gold, OSD
Emerging Software Size and Complexity

- Advanced system attributes (on-board *intelligence* and *adaptive control laws*) will be required to accommodate emerging functional requirements.
- This will increase the size and complexity of control systems beyond the capability of current V&V practices.

*Projected Exponential Increase in SW Size and Complexity*
Criteria for Formal Verification

• Is the Problem Important?

• Are High Fidelity Models Available?

• Can the Properties of Interest be Formalized?

• Are the Right Analysis Tools Available?
Presentation Overview

What Problem are We Solving?

Who Are We?

What are Formal Methods?

Examples of Using Formal Methods

Challenges and Future Directions
Rockwell Collins

- Headquartered in Cedar Rapids, Iowa
- 20,000 Employees Worldwide
- 2007 Sales of $4.42 Billion
Rockwell Collins’ core business is based on the delivery of High Assurance Systems

- Commercial/Military Avionics Systems
- Communications
- Navigation & Landing Systems
- Flight Control
- Displays

“Working together creating the most trusted source of communication and aviation electronic solutions”
Advanced Technology Center

*Identify, acquire, develop and transition value-driven technologies to support the continued growth of Rockwell Collins.*

Automated Analysis Section

**Technologists:** 10  
**Administrators:** 1

Applies mathematical tools and reasoning to the production of high assurance systems.
Formal Methods at Rockwell Collins

- AAMP5 Microcode Verification (PVS)
- AAMP-FV Microcode Verification (PVS)
- AAMP5 Partitioning (PVS)
- FGS Mode Confusion Study (PVS)
- AAMP7 Microcode (ACL2)
- FCP 2002 Microcode (PVS)
- JEM Java μProc (PVS)

NASA Aviation Safety
- FGS Mode Confusion (RSML-e, PVS)
- FGS Safety Analysis (RSML-e, NuSMV)
- ADGS 2100 (Simulink, NuSMV)

AFRL
- CerTA FCS (NuSMV, Prover)
- Mixed Criticality Architectures
- Greenhills Integrity RTOS (ACL2)
- Greenhills Integrity Gen4 (ACL2)

NSA
- vFaat (ACL2, PVS)
- Turnstile (SPARK)
- Guardol (ACL2, Prover)
- SHADE (ACL2)

© Copyright 2008 Rockwell Collins, Inc. All rights reserved.
Presentation Overview

What Problem are We Solving?

Who Are We?

What are Formal Methods?

Examples of Using Formal Methods

Challenges and Future Directions
What are Formal Methods?

Mathematically-based techniques for the specification, development and verification of software and hardware systems.

Wikipedia, 8 April 2008

• **Specification**
  - Textual notations (Z, B, VDM, CSP, ...)
  - Tabular notations (Parnas Tables, SCR, RSML, ...)
  - Graphical notations (SCADE, Simulink, Statecharts ...)

• **Development**
  - Stepwise refinement with proofs of correctness
  - Model-Based Development
  - Automated code generation

• **Verification**
  - Lightweight static analysis
  - Theorem proving (ACL2, PVS, HOL, ...)
  - Model-checking (SMV, SAL, Prover, ...)
node Thrust_Required(
    FG_Mode : FG_Mode_Type ;
    Airborne : bool ;
    In_Flare : bool ;
    Emergency_Descent : bool ;
    Windshear_Warning : bool ;
    In_Eng_Accel_Zone : bool ;
    On_Ground : bool )
returns (IsTrue : bool ) ;

let

IsTrue =
    (FG_Thrust_Mode(FG_Mode) and Airborne)
or
    (Airborne and Emergency_Descent)
or
    Windshear_Warning
or
    ((FG_Mode = ThrottleRetard) and In_Flare)
or
    (In_Eng_Accel_Zone and On_Ground) ;
tel ;
# Model-Based Development

<table>
<thead>
<tr>
<th>Company</th>
<th>Product</th>
<th>Tools</th>
<th>Specified &amp; Autocoded</th>
<th>Benefits Claimed</th>
</tr>
</thead>
</table>
| Airbus                           | A340                           | SCADE With Code Generator    | • 70% Fly-by-wire Controls  
• 70% Automatic Flight Controls  
• 50% Display Computer  
• 40% Warning & Maint Computer | • 20X Reduction in Errors  
• Reduced Time to Market         |
| Eurocopter                       | EC-155/135 Autopilot           | SCADE With Code Generator    | • 90% of Autopilot                                                               | • 50% Reduction in Cycle Time                                                   |
| GE & Lockheed Martin             | FADEDC Engine Controls         | ADI Beacon                   | • Not Stated                                                                       | • Reduction in Errors  
• 50% Reduction in Cycle Time  
• Decreased Cost                |
| Schneider Electric               | Nuclear Power Plant Safety     | SCADE With Code Generator    | • 200,000 SLOC Auto Generated from 1,200 Design Views                             | • 8X Reduction in Errors while Complexity Increased 4x                         |
| Schneider Electric               | Engine Controls                |                              |                                                                                     |                                                                                  |
| US Spaceware                     | DCX Rocket                     | MATRIXx                      | • Not Stated                                                                       | • 50-75% Reduction in Cost  
• Reduced Schedule & Risk        |
| PSA                              | Electrical Management System   | SCADE With Code Generator    | • 50% SLOC Auto Generated                                                         | • 60% Reduction in Cycle Time  
• 5X Reduction in Errors         |
| CSEE Transport                   | Subway Signaling System        | SCADE With Code Generator    | • 80,000 C SLOC Auto Generated                                                    | • Improved Productivity from 20 to 300 SLOC/day                                  |
| Honeywell Commercial Aviation    | Primus Epic Flight Control     | MATLAB Simulink              | • 60% Automatic Flight Controls                                                   | • 5X Increase in Productivity  
• No Coding Errors  
• Received FAA Certification  |
| Systems                          | System                         |                              |                                                                                     |                                                                                  |

© Copyright 2008 Rockwell Collins, Inc.  
All rights reserved.
Verification - Rockwell Collins Translation Framework

- **Simulink** → **SCADE**
- **StateFlow** → **Safe State Machines**
- **Reactis**

**Lustre**
- **NuSMV**
- **Prover**
- **ACL2**
- **PVS**
- **Design Verifier**

**SAL**
- **SAL Symbolic Model Checker**
- **SAL Bounded Model Checker**
- **SAL Infinite Model Checker**

**Rockwell Collins/U of Minnesota**
- **Esterel Technologies**
- **SRI International**
- **MathWorks**
- **Reactive Systems**

© Copyright 2008 Rockwell Collins, Inc.
All rights reserved.
# Translators Optimize for Specific Analysis Tools

<table>
<thead>
<tr>
<th>Model</th>
<th>CPU Time (For NuSMV to Compute Reachable States)</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Before</td>
<td>After</td>
</tr>
<tr>
<td>Mode1</td>
<td>&gt; 2 hours</td>
<td>11 sec</td>
</tr>
<tr>
<td>Mode2</td>
<td>&gt; 6 hours</td>
<td>169 sec</td>
</tr>
<tr>
<td>Mode3</td>
<td>&gt; 2 hours</td>
<td>14 sec</td>
</tr>
<tr>
<td>Mode4</td>
<td>8 minutes</td>
<td>&lt; 1 sec</td>
</tr>
<tr>
<td>Arch</td>
<td>34 sec</td>
<td>&lt; 1 sec</td>
</tr>
<tr>
<td>WBS</td>
<td>29+ hours</td>
<td>1 sec</td>
</tr>
</tbody>
</table>
Presentation Overview

What Problem are We Solving?

Who Are We?

What are Formal Methods?

Examples of Using Formal Methods

Challenges and Future Directions
FCS 5000 Flight Control Mode Logic

Mode Controller A

Modeled in Simulink
Translated to NuSMV
6.8 \times 10^{21} \text{ Reachable States}

Mode Controller B

Example Requirement
Mode A1 \implies Mode B1

Counterexample Found in Less than Two Minutes

Found 27 Errors in Early Requirements Models
ADGS-2100 Adaptive Display & Guidance System

Modeled in Simulink
Translated to NuSMV
4,295 Subsystems
16,117 Simulink Blocks
Over $10^{37}$ Reachable States

Example Requirement:
Drive the Maximum Number of Display Units
Given the Available Graphics Processors

Counterexample Found in 5 Seconds

Checked 573 Properties -
Found and Corrected 98 Errors
in Early Design Models
AAMP7G Certified Microprocessor

- Rockwell Collins proprietary microprocessor
- Formal proof of the MILS security partitioning implemented in the AAMP7G microprocessor
- Example of the industrial use of theorem proving using ACL2
- Developed formal description of separation for uniprocessor, multipartition system (GWV)
- Modeled trusted AAMP7G microcode in ACL2
- Constructed machine-checked proof of separation of the AAMP7G model using ACL2
- Model subject of intensive code-to-spec review with AAMP7G microcode
- Satisfied formal methods requirements for NSA AAMP7G certification awarded in May 2005
  - “capable of simultaneously processing unclassified through Top Secret Codeword Information”
  - “verified using Formal Methods techniques as specified by the EAL-7 level of the Common Criteria”
Greenhills Integrity-178B Real-Time OS Evaluation

- Formal proof of the MILS security partitioning implemented in the Integrity-178B Real-Time OS
- Example of the industrial use of theorem proving using ACL2
- Generalized the formal description of separation to describe the more dynamic scheduling managed by the OS (GWVr2)
- Modeled in ACL2 the target-independent C code implementing the Integrity-178B kernel.
- Constructed machine-checked proof of separation for the Integrity-178B kernel
- Model, analysis approach and proofs subject to intensive multi-national review
- Satisfied US Government SKPP (EAL6+), as well as Common Criteria v2.3 EAL7 ADV requirements
  - Final certification pending NSA penetration testing
Turnstile High Integrity Guard

- High-assurance cross domain platform that provides secure communication between different security classification domains ranging from top secret to unclassified.

- Core guard application is based on the NSA certified AAMP7G.

- I/O processing is relegated to Offload Engines (OE) that do not have to be as highly trusted.

- System integrator can add function to the OE without compromising the guard function.

- Certification based on ACL2 theorem prover
CerTA FCS Phase I

- **Sponsored by the Air Force Research Labs**
  - Air Vehicles (RB) Directorate - Wright Patterson

- **Investigate Roles of Testing and Formal Verification**
  - Can formal verification complement or replace some testing?

- **Example Model – Lockheed Martin Adaptive UAV Flight Control System**
  - Redundancy Management Logic in the Operational Flight Program (OFP)
  - Well suited for verification using the NuSMV model-checker

---

### Lockheed Martin Aero

- Based on Testing
- Enhanced During CerTA FCS
  - Graphical Viewer of Test Cases
  - Support for XML/XSLT Test Cases
  - Added C++ Oracle Framework
- Developed Tests from Requirements
- Executed Tests Cases on Test Rig

---

### Rockwell Collins

- Based on Model-Checking
- Enhanced During CerTA FCS
  - Support for Simulink blocks
  - Support for Stateflow
  - Support for Prover model-checker
- Developed Properties from Requirements
- Proved Properties using Model-Checking

© Copyright 2008 Rockwell Collins, Inc.
All rights reserved.
CerTA FCS Phase I - OFP Redundancy Management Logic

For Each of Ten Control Surfaces

- **Triplex Voter**
  - Input monitor, sensor fusion, and failure isolation
- **Failure Processing**
  - Logs failures into a data store
- **Reset Manager**
  - Reset logic for sensors and control surfaces (not shown)

<table>
<thead>
<tr>
<th>Subsystems / Blocks</th>
<th>Charts / Transitions</th>
<th>Truth Table Cells</th>
<th>Reachable State Space</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Triplex voter</td>
<td>10 / 96</td>
<td>3 / 35</td>
<td>198</td>
<td>6.0 * 10^{13}</td>
</tr>
<tr>
<td>Failure processing</td>
<td>7 / 42</td>
<td>0 / 0</td>
<td>0</td>
<td>2.1 * 10^{4}</td>
</tr>
<tr>
<td>Reset manager</td>
<td>6 / 31</td>
<td>2 / 26</td>
<td>0</td>
<td>1.32 * 10^{11}</td>
</tr>
<tr>
<td>Total</td>
<td>23 / 169</td>
<td>5 / 61</td>
<td>198</td>
<td>N/A</td>
</tr>
</tbody>
</table>
CerTA FCS Phase I -
Testing and Model Checking Recurring Costs

- **Test:** time to run the tests.
  - **MC:** running the tools, analyzing and explaining counter-examples to LM Aero, and creating a revised model.

- **Spent ~50% more time testing than model-checking.**

- **Test:** time spent fixing errors in test cases.
- **MC:** time to repeat analysis.

---

<table>
<thead>
<tr>
<th>Recurring Costs</th>
<th>Preparation</th>
<th>Initial Test</th>
<th>Rework</th>
<th>Grand Total</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Testing</strong></td>
<td>40</td>
<td>20</td>
<td>10</td>
<td>70</td>
</tr>
<tr>
<td><strong>Model-Checking</strong></td>
<td>20</td>
<td>50</td>
<td>30</td>
<td>120</td>
</tr>
</tbody>
</table>
CerTA FCS Phase I – Errors Found

<table>
<thead>
<tr>
<th></th>
<th>Model Checking</th>
<th>Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Triplex Voter</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>Failure Processing</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Reset Manager</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>12</strong></td>
<td>0</td>
</tr>
</tbody>
</table>

- Model-Checking Found 12 Errors that Testing Missed
- Spent More Time on Testing than Model-Checking
  - 60% of total on testing vs. 40% on model-checking

Model-checking was more **cost effective** than testing at finding **design** errors.
CerTA FCS Phase II

- **Sponsored by the Air Force Research Labs**
  - Air Vehicles (RB) Directorate - Wright Patterson

- **Can Model-Checking be Used on Infinite State Systems?**
  - Large, numerically intensive, non-linear systems

- **Example Model**
  - Lockheed Martin Adaptive UAV Flight Control System
  - Effector Blender (EB)
  - Generates actuator commands for aircraft control surfaces
  - Matrix arithmetic of floating point numbers

- **Challenges**
  - Identifying the right properties to verify
  - Verification of floating point numbers
  - Verification of Stateflow *flowcharts* with cyclic transition paths
  - Compositional verification to scale to entire Effector Blender
CerTA FCS Phase II – Effector Blender

- **Generates Actuator Commands**
  - Six control surfaces
  - Adapts its behavior as aircraft state changes
  - Iterative algorithm that repeatedly manipulates a 3 x 6 matrix of floating point numbers

- **Large Complex Model**
  - **Inputs**
    - 32 floating point inputs
    - 3 x 6 matrix of floating point values
  - **Outputs**
    - 1 x 6 vector of floating point values
  - **166 Simulink subsystems**
  - **2000+ basic Simulink blocks**
  - Huge reachable state space

- **Completely Functional**
  - No internal state
CerTA FCS Phase II – What to Verify?

- No Explicit Requirements for the Effector Blender Model
  - Requirements defined for Effector Blender + aircraft model
  - Addition of aircraft model pushes verification beyond current tools

- Avoid Properties Verifiable by Other Means
  - Control theory – stability, tracking performance, feedback design ...
  - Simulation – design validation
  - Implementation – code generation/compilation, scheduling, ...

- Focus on the Consistency of the Effector Blender Model
  - Relationships the model should always maintain
  - Partial requirements specification

- Preservation of Control Surface Limits
  - EB computes upper and lower limits for each control surface command
  - Function of aircraft design, aircraft state, and max extension per cycle
  - Commanded extension should always be between these limits
CerTA FCS Phase II – Verification of Floating Point Numbers

• Floating Point Numbers
  – Fixed number of bits with a movable decimal (radix) point
  – No decision procedures for floating point numbers available

• Real Numbers
  – Real numbers have unbounded size and precision
  – Would hide errors caused by limitations of floating point arithmetic
  – Control theory problems are inherently non-linear
  – Decision procedures for non-linear real numbers have exponential cost

• Solution - Translate Floating Point Numbers into Fixed Point
  – Extended translation framework to automate this translation
  – Convert floating point to fixed point (scaling provided by user)
  – Convert fixed point into integers (use bit shifting to preserve magnitude)
  – Shift from NuSMV (BDD-based) to Prover (SMT-solver) model checker

• Advantages & Issues
  – Use bit-level integer decision procedures for model checking
  – Results unsound due to loss of precision
  – Highly likely to find errors – very valuable tool for debugging
CerTA FCS Phase II – Verification of Stateflow Flowcharts

- Stateflow Flowcharts
  - No explicit states
  - Stateflow junctions
  - Cyclic paths
  - Transitions modify local state variables
  - Imperative programming

- Solution
  - Extend translator to support flowcharts
  - Require a parameter that specifies the maximum times any cycle will be executed
CerTA FCS Phase II – Compositional Verification

Typical Specification
- Models are typically organized in a hierarchy of subsystems
- Subsystems are often nested several levels deep
- Most of the complexity is in the leaf subsystems
- Leaf subsystems can often be verified through model checking

Composition of Subsystems
- Tends to be simple
- Lends itself well to theorem proving

Issues
- Need to avoid circular reasoning to ensure soundness
- Can be ensured by eliminating cyclic dependencies between atomic subsystems
- Identifying the right leaf level invariants to support composition
- Complexity of the proof obligations for the intermediate levels
- Lack of a unified automated verification system
CerTA FCS Phase II - Results

- Can Model-Checking be Used on Infinite State Systems?
  - Large, numerically intensive, non-linear systems

- Effector Blender
  - Inputs
    - 32 floating point inputs
    - 3 x 6 matrix of floating point values
  - Outputs
    - 1 x 6 vector of floating point values
    - 166 Simulink subsystems
    - 2000+ basic Simulink blocks

- Errors Found
  - Five previously unknown errors that would drive actuators past their limits
  - Several implementation errors were being masked by defensive programming

- Areas for Future Research
  - Decision procedures for floating point arithmetic
  - Interval arithmetic
  - Automation for compositional verification
Presentation Overview

What Problem are We Solving?

Who Are We?

What are Formal Methods?

Examples of Using Formal Methods

Challenges and Future Directions
Extending the Verification Domain

• Theorem Provers
  – Deal with arbitrary models
  – Concerns are ease of use and labor cost

• Large Finite Systems (<10^{200} States)
  – Implicit state (BDD) model checkers
  – Easy to use and very effective

• Very Large or Infinite State Systems
  – SMT-Solvers
  – Large integers and reals
  – Limited to linear arithmetic
  – Ease of use is a concern

• Floating Point Arithmetic
  – Most modeling languages use floating point (not real) numbers
  – Decision procedures

• Non-Linear Arithmetic
  – Multiplication/division of real variables
  – Transcendental functions (trigonometric, …)
  – Essential to navigation systems
Compositional Verification

Typical Model-Based Specification
- Models are organized in a hierarchy of subsystems several levels deep
- Most of the complexity is in the leaf models
- Leaf models can often be verified through model checking

Composition of Subsystems
- Tends to be simple
- Well suited for theorem proving

Issues
- Lack of a unified automated verification system
  - Use model-checking to verify leaf models and theorem proving for composition
- Avoid circular reasoning to ensure soundness
  - Can be ensured by eliminating cyclic dependencies between atomic subsystems
- Identifying the right leaf level invariants to support composition
- Complexity of the proof obligations for the intermediate levels
System Architectural Modeling & Analysis

System Architecture Development
Conclusions

- **Formal Methods Are Practical and Are Being Widely Used**
  - Model Based Development is the industrial face of formal methods
  - The engineers get to pick the modeling tools!
  - Semantics of some of the commercial tools could be improved

- **Formal Verification Tools Are Being Used in Industry**
  - Key is to verify the models the engineers are already building
  - Large portions of existing systems can be verified with model checkers
  - Model checkers are only going to get better
  - Theorem proving can be used on stable industrial systems

- **Directions for the Future Work**
  - Making verification tools more powerful and easier to use
  - Addressing scalability through compositional verification
  - Integration of theorem proving and model checking
  - Modeling and analysis of system architectural models
For More Information

http://shemesh.larc.nasa.gov/fm/fm-collins-intro.html


