Symbolic Model Checking of State Machine Based Software Specifications

Richard Anderson, Paul Beame, Steve Burns, William Chan, Francesmary Modugno, David Notkin, Jon Reese

Department of Computer Science and Engineering
University of Washington
Software Model Checking

System Specification

Reactive Systems
Air Traffic Control
Automotive (Cruise Control, Anti-lock Braking)

Hierarchical State Machine Specification

Statecharts (Harel)
RSML (Leveson)

Goal: Increase confidence in the correctness of the specification
Symbolic Model Checking

Evaluate Temporal Properties of Finite State Systems

Very Important for Hardware Verification

Does it apply to Software???
This work

Applied model checking to the specification of TCAS II

Traffic Alert and Collision Avoidance System

In use on commercial aircraft in US

Translation Process (RSML to SMV)

Model Checking (Dealing with BDD's)

Analyzing TCAS
Outline

Hierarchical State Machine Specifications

Symbolic Model Checking

TCAS

Our Experiences analyzing TCAS
Reactive Systems

- Inputs
- Outputs
- Internal Events
- Clock
State Machine Description Languages

Statecharts (Harel)
RSML (Leveson et al.)

State machines with:
Hierarchical States
Parallel States
Event Driven Transitions
Reactor Model

Temperature: Cold, Okay, Hot, Too Hot

Rod Movement: Ready, Just Moved, Move In, Move Out

Rod Configuration: Panic, All Out, Midway, All In
Reactor Events

Temp_Reading
Initiate_Move
Move_Finished
Rod_Updated
Clock_Event
Temp_Update

External
External
External

Rod_Move
Rod_Config
Rod_Move

Temparature
Rod_Config
Rod_Move
Reactor Transitions

On → Panic

Trigger_Event:
  Temp_Update
Condition:
  Temperature in state Too Hot
Output Action:
  Panic_Event

Ready → Move In

Trigger_Event:
  Temp_Update
Condition:
  Rod_Movement in state Ready AND
  Temperature in state Hot
Output Action:
  Initiate_Move
Events

External

Internal

Synchrony Hypothesis

External event arrives

Triggers cascade of internal events (micro steps)

Stability reached before next external event

Technical issues with micro steps

Differences between Statecharts and RSML
Properties to Check

Temperature is never Too Hot

If Temperature is in state Hot, then eventually temperature is in state Okay or Rod_Configuration is in state All In

Rod_Configuration only changes in response to a Move_Finished event
Warn pilots of traffic

Issue resolution advisories

Vertical resolution only

Relies on transponder data.
TCAS

Irvine Safety Group (Leveson et al.)
Specified in RSML as a research project
FAA adopted RSML version as "Official"

400 pages

This study: Version 6.00, March 1993
TCAS

Own_Aircraft

Sensitivity Levels, Alt_Layer, Advisory_Status

Other_Aircraft

Tracked, Intruder_Status, Range_Test, Crossing, Sense Descend/Climb
Model Checking

Does a temporal logic formula hold for a finite state system

Temporal Logic

until, eventually, next, always . . .

Model checking can be done in linear time (in size of state space) for many logics
Symbolic Model Checking

State space can be HUGE, $2^{1000}$

Implicit representation

Data structure represents transition relation
(boolean formula)

Algorithmically manipulate the data structure to explore the state space

Key: efficiency of the data structure
Binary Decision Diagrams (BDD's)

"Folded decision tree"

Fixed Variable Order

Many functions have small BDD's (notable exception, multiplication)
SMV

BDD based tool to check CTL specifications for finite state systems

TCAS RSML \rightarrow SMV Language \rightarrow Model Checker

CTL Formula
Iteration

Correcting specification (or SMV version of specification)

Clarifying temporal formula

Modelling environment

Refining specification
Non-determinism

Inputs from environment
Altitude := \{1000. \ldots 8000\};

Simplification of functions
Alt\_Rate := 0.25 \times (Alt\_Baro - ZP) / Delta_t
Alt\_Rate := \{-2000 \ldots 2000\};

Unmodelled parts of specifications
States of Other\_Aircraft treated as non-deterministic input variables
MODULE main
VAR
  state: {ON, OFF};
  on_event: boolean;
  off_event: boolean;
ASSIGN
  init(state) := OFF;
  next(state) := case
    state = ON & off_event: OFF;
    state = OFF & on_event: ON;
  1: state;
  esac;
Environment Variables

VAR boolean Initiate_Move, Move_Finished, Rod_Updated, Clock_Event;

ASSIGN
Stable := ! Initiate_Move & ! Move_Finished
     & ! Rod_Updated & ! Clock_Event;
next(Move_Finished) := case
    Stable : {0, 1};
    1 : 0;
esac;
Transitions

VAR RC : {Out, Mid, In};

ASSIGN
T_Out_Mid := Move_Finished & RC = Out &
    &Rod_Move = Move_In;

next(RC) := case
    T_Out_Mid : Mid;
    T_Mid_In : In;
    T_Mid_Out : Out;
    T_In_Mid : Mid;
    1 : RC;
esac;
Deterministic Transitions

Deterministic: at most one of T\_A\_B and T\_A\_C can be true

Non deterministic: both of T\_A\_B and T\_A\_C can be true

To allow non-deterministic transitions

\begin{verbatim}
next(S) := case 
    T\_A\_B & T\_A\_C : {B, C};
    T\_A\_B : B;     T\_A\_C : C;    1 : S;
end;
\end{verbatim}
Obstacles

Initial attempts generated BDDs requiring 200 M memory.

First successful check took 13 hours

(this has been reduced to minutes)

Partitioning BDDs

Ordering Variables:

\[ x_1, x_2, x_3, y_1, y_2, y_3 \text{ vs. } x_1, y_1, x_2, y_2, x_3, y_3 \]

Implementing Search
Software Checking

Domain Independent
- Deterministic state transitions
- Function consistency
- Complete Initialization

Domain Dependent
- Output agreement
- Safety properties
Results

Used SMV to analyze Own_Aircraft module of TCAS

Investigated

  Transition Consistency
  Function Consistency
  Output Agreement
  Safety Properties
Disclaimer

The intent of this work is to evaluate symbolic model checking of state-based specifications, not to evaluate the TCAS II specification. We used a preliminary version of the specification, the version 6.00, dated March, 1993 in our study. We did not have access to later versions, so we do not know if the issues identified here are present in later versions.
Transition Consistency

Same conditions allow multiple transitions (non determinism)

Previously discovered by other methods (Heimdahl & Leveson)

\[ AG \neg(T_{254} \land T_{257}) \]

Identical conditions gave transition from Effective Sensitivity Level 4 to ESL 2 and from ESL 4 to ESL 5
T_254 and T_257

V_254a := MS = TA_RA | MS = TA_only | MS = 3 | MS = 4
| MS = 5 | MS = 6 | MS = 7;
V_254b := ASL = 2 | ASL = 3 | ASL = 4 | ASL = 5
| ASL = 6 | ASL = 7;
T_254 := (ASL = 2 & V_254a) | (ASL = 2 & MS = TA_only)
| (V_254b & LG = 2 & V_254a);
V_257a := LG = 5 | LG = 6 | LG = 7 | LG = none;
V_257b := MS = TA_RA | MS = 5 | MS = 6 | MS = 7;
V_257c := MS = TA_RA | MS = TA_only | MS = 3 | MS = 4
| MS = 5 | MS = 6 | MS = 7;
V_257d := ASL = 5 | ASL = 6 | ASL = 7;
T_257 := (ASL = 5 | V_257a | V_257b)
| (ASL = 5 & MS = TA_only)
| (ASL = 5 & LG = 2 & V_257c)
| (V_257d & LG = 5 & V_257b)
| (V_257d & V_257a & MS = 5);
Function Consistency

Definition by Cases

\[
F := \begin{cases} 
  V_1 & \text{if } C_1 \\
  V_2 & \text{if } C_2 \\
  V_3 & \text{if } C_3 
\end{cases}
\]

Function is inconsistent if \( C_i \) and \( C_j \) can be true simultaneously

\[
AG ! (( C_1 \& C_2) \mid (C_1 \& C_3) \mid (C_2 \& C_3))
\]
Display_Model_Goal

Tells pilot the desired rate of altitude change

Checking for consistency gave a counter example: Other_Aircraft reverses from an Increase-Climb to an Increase-Descend advisory

Other_Aircraft modeled non-deterministically, and Other_Aircraft logic apparently prohibits this.

Otherwise okay
Output Agreement

Related outputs should be consistent

Resolution Advisory
   Increase-Climb, Climb, Descend, Increase-Descend

Display_Model_Goal
   Desired rate of altitude change
   Between -3000 ft/min to 3000 ft/min

Presumably, on a climb advisory, DMG should be positive . . .
Output Agreement Check

If the Resolution Advisory is climb, then Display_Model_Goal is positive
AG (RA = Climb -> DMG > 0)

Counter-example:

\[ t_0 : RA = \text{Descend}, \ DMG = -1500 \]
\[ t_1 : RA = \text{Increase-Descend}, \ DMG = -2500 \]
\[ t_2 : RA = \text{Climb}, \ DMG = -1500 \]
Safety Properties

An airplane at low altitude should not be told to reduce elevation

$$\text{AG} \neg (\text{Alt} \leq 1450 \land \text{RA} = \text{Increase-Descend})$$

FALSE!

Error in preliminary version of spec (typo during reverse engineering, not in other documentation).
## Performance Results

<table>
<thead>
<tr>
<th>Property</th>
<th>Result</th>
<th>Time</th>
<th>No. of BDD Nodes</th>
<th>Mem Mb.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transition Rel.</td>
<td>N/A</td>
<td>46.6</td>
<td>124K</td>
<td>7.1</td>
</tr>
<tr>
<td>Trans. Consist.</td>
<td>false</td>
<td>387.0</td>
<td>717K</td>
<td>16.4</td>
</tr>
<tr>
<td>Funct. Consist.</td>
<td>false</td>
<td>289.5</td>
<td>387K</td>
<td>11.5</td>
</tr>
<tr>
<td>Step Termination</td>
<td>true</td>
<td>57.2</td>
<td>142K</td>
<td>7.4</td>
</tr>
<tr>
<td>Descend Inhibit</td>
<td>true</td>
<td>166.8</td>
<td>429K</td>
<td>11.8</td>
</tr>
<tr>
<td>Increase-Desc.</td>
<td>false</td>
<td>193.7</td>
<td>282K</td>
<td>9.9</td>
</tr>
<tr>
<td>Output Agree.</td>
<td>false</td>
<td>325.6</td>
<td>376K</td>
<td>11.6</td>
</tr>
</tbody>
</table>

Timings on a Sun SPARCstation 10 with 128 Mb Memory. SMV Release 2.4.4.
Discussion I

This study provides a positive data point for applying model checking to software

- Translation and construction of BDDs
- Identified several issues

Iterative use of model checking

- Refine specification
- Use non-determinism to reduce size
- Debugging specification
Discussion II

Limits for checking specifications

Specification size
Numerical Issues (multiply, divide)

To what extent is software model checking different from hardware model checking?

Word level vs. bit level
Event structure

Domain expertise necessary

Thanks, Jon!
Discussion III

Integration with other tools

Automatic translation
Use with state machine editor/simulator

Optimization of Model Checker for Software

Better handling of numbers
Cascades of micro events, stability
Timing and timeouts
Model Checking Software