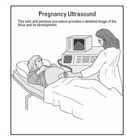
Applying Model Checking to Large Software Specifications

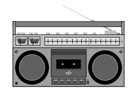
David Notkin
Dept. of Computer Science & Engineering
University of Washington
10 March 1997
www.cs.washington.edu/homes/notkin

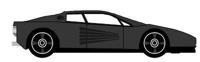
Joint with:
- Richard Anderson
- Paul Beame
- William Chan
- Steve Burns
- Francesmary Modugno
- Jon Reese

Embedded systems

- Software is increasingly pervasive
- No complex system is any longer built without software
  - Avionics, medical imaging and treatment, consumer electronics, appliances, ...



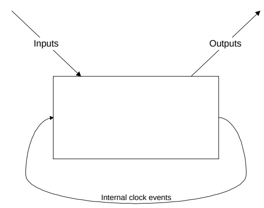




Problems

- Getting embedded systems to work properly is important
  - Safety-critical systems [comp.risks; Leveson, *Safeware*]
  - Pressures of the marketplace
- Getting them to work right is hard
  - Hard to clarify requirements
    - » Problems that appear later cost far more to fix
  - Difficulties at the interfaces

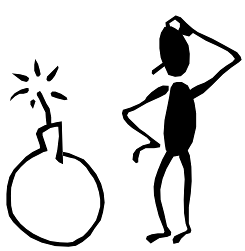
Reactive systems



- Reactive systems are often specified (in part) by state machines that describe the actions that the system should make in response to an external event

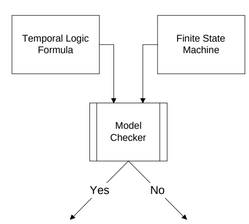
Key question

- How do we increase our confidence that the requirements specification has the properties we want?
  - It does what it is supposed to do
  - It doesn't do things it isn't supposed to do



Symbolic model checking

- Evaluate temporal properties of finite state systems
- Extremely successfully for hardware verification
- Open question: applicable to large software specifications for reactive systems?



Software model checking

- Finite state software specifications
  - Reactive systems (avionics, automotive, etc.)
  - Hierarchical state machine specifications
    - » Statecharts (Harel), RSML (Leveson)
- Goal: increase confidence in the correctness of the specification

Why might model checking fail?

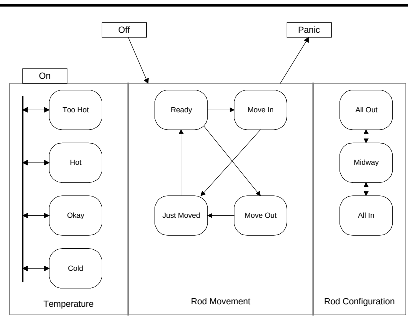
- Software is often specified with infinite state descriptions
  - We don't address those specifications
    - » Jackson, Damon, Jha; Wing, Vaziri-Farahani; etc.
- Software specifications may be structured differently from hardware specifications
  - Hierarchy
  - Representations and algorithms for model checking may not scale

Our approach—try it!

- Applied model checking to the specification of TCAS II
  - **T**raffic **A**lert and **C**ollision Avoidance system
    - » In use on U.S. commercial aircraft
  - FAA adopted specification
  - Initial design and development by Leveson
- Translation process (RSML to SMV)
- Model checking (dealing with BDD's)
- Analyzing TCAS properties

Outline

- Hierarchical state machine specifications
- Symbolic model checking
- TCAS
- Our experiences in analyzing TCAS using model checking



Events

Temp_Reading	External	Temperature
Initiate_Move	Rod_Move	Rod_Config
Move_Finished	External	Rod_Config
Rod_Updated	Rod_Config	Rod_Move
Clock_Event	External	Rod_Move
Temp_Update	Temperature	Rod_Move

Sample transitions

On → Panic

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

Ready → Move In

```
Trigger_Event: Temp_Update
Condition: Rod_Movement in Ready and Temperature in Hot
Output Action: Initiate_Move
```

Just Moved → Ready

```
Trigger_Event: Clock_Event
Condition: Rod_Movement in Just_Moved and
          t > t(entered(Just_Moved))+ Move_Delay
```

Events

- External—interactions with environment
- Synchrony hypothesis
  - External event arrives
  - Triggers cascade of internal events (micro steps)
  - Stability reached before next external event
- Technical issues with micro steps
  - Harel, Pnueli, Leveson

Properties to check: examples

- If `Temperature` is in `Hot`, then eventually `Temperature` is in `Okay` or `Rod_Configuration` is in `All_In`
- `Rod_Configuration` only changes in response to a `Move_Finished` event

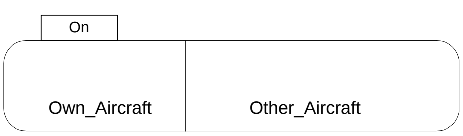
TCAS

- http://www.faa.gov/and/and600/and620/newtcas.htm
- Warn pilots of traffic
  - Plane to plane, not through ground controller
  - On essentially all commercial aircraft
- Issue resolution advisories only
  - Vertical resolution only
  - Relies on transponder data

TCAS specification

- Irvine Safety Group (Leveson et al.)
  - Specified in RSML as a research project
  - FAA adopted RSML version as official
- Specification is about 400 pages long
- This study uses: Version 6.00, March 1993
  - Not the current FAA version

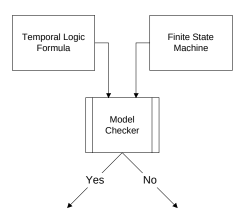
TCAS—high-level structure



- Own_Aircraft
  - Sensitivity levels, Alt_Layer, Advisory_Status
- Other_Aircraft
  - Tracked, Intruder_State, Range_Test, Crossing, Sense Descend/Climb

Model checking

- Does a temporal logic formula hold for a finite state machine?
  - If not, find counterexample
- Temporal logic
  - until, eventually, always, etc.
- For many logics, checking can be done in linear time in the size of the space state
  - Explicit model checking does this, exploiting symmetries for performance

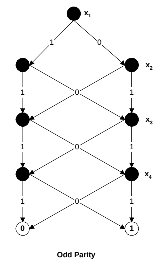


Symbolic model checking

- State space can be huge (>2^{1000}) for many systems
- Use implicit representation
  - Data structure to represent transition relation as a boolean formula
- Algorithmically manipulate the data structure to explore the state space
- Key: efficiency of the data structure

Binary decision diagrams (BDDs)

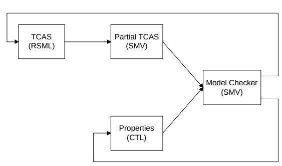
- "Folded decision tree"
- Fixed variable order
- Many functions have small BDDs
  - Multiplication is a notable exception
- Can represent
  - State machines (transition functions)
  - Temporal queries



Due to Randy Bryant

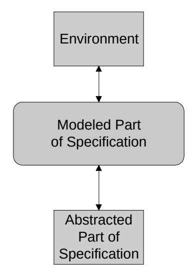
Using SMV

- SMV is a BDD-based model checker
- It checks CTL formulas
  - A specific temporal logic



Iterative process

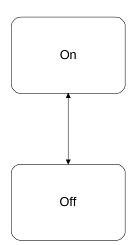
- Iterate SMV version of specification
- Clarify temporal formula
- Model environment more precisely
- Refine specification



Use of non-determinism

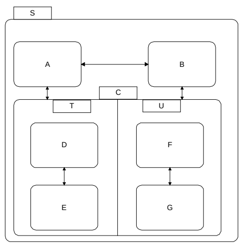
- Inputs from environment
  - `Altitude := {1000…8000}`
- Simplification of functions
  - `Alt_Rate := 0.25*(Alt_Baro-ZP)/Delta_t`
  - `Alt_Rate := {-2000…2000}`
- Unmodelled parts of specification
  - States of `Other_Aircraft` treated as non-determinstic input variables

Translating RSML to SMV




```
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;
```

State encoding



- Flatten nested AND and nested OR states
- One variable for each OR state
  - An enumerated type of the alternatives
- VAR

```
S: {A,B,C};
T: {D,E};
U: {F,G};
```


Synchrony hypothesis

- Handling an external event

```
DEFINE
  Stable := !Initiate_Move &
            !Move_Finished &
            !Rod_Updated & !Clock_Event
ASSIGN
  next(Move_Finished) := case
    Stable : {0,1};
    1      : 0;
esac;
…for other external events…
```


Transitions

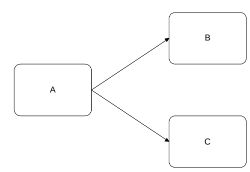
```
VAR RC: {Out, Mid, In};

ASSIGN
  T_Out_Mid : Mid;    T_Mid_In : In;
  T_Mid_Out : Out;    T_In_Mid : Mid;
  1 : RC;
esac;
```


Non-deterministic transitions

- A machine is deterministic if at most one of `T_A_B`, `T_A_C`, etc. can be true
- Else non-deterministic
- Can encode non-deterministic transitions
-

```
next(S) := case
   T_A_B & T_A_C: {B,C};
   T_A_B : B; T_A_C : C;
   1 : S;
esac;
```



Checking properties

- Initial attempts to check any property generated BDDs of over 200MB
- First successful check took 13 hours
  - Has been reduce to a few minutes
- Partitioned BDDs
- Reordered variables
- Implemented better search for counterexamples

Property checking

- Domain independent properties
  - Deterministic state transitions
  - Function consistency
- Domain dependent
  - Output agreement
  - Safety properties
- We used SMV to investigate some of these properties on TCAS' `Own_Aircraft` module

Disclaimer

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

Deterministic transitions

- Do the same conditions allow for non-deterministic transitions?
- Inconsistencies were found earlier by other methods [Heimdahl and Leveson]
  - Identical conditions allowed transitions from Sensitivity Level 4 to SL 2 or to SL 5
- Our formulae checked for all possible non-determinism; we found this case, too

Earlier version of TCAS spec


```
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 & V524a);
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);
```

Tradeoffs

- Our approach was slower than the Heimdahl & Leveson approach
- Their approach reported some false positives

Function consistency

- Many functions are defined in terms of cases
- A function is inconsistent if two different conditions C_i and C_j and be true simultaneously

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

AG ! ((C_1 & C_2) | (C_1 & C_3) | (C_2 & C_3))

`Display_Model_Goal`

- Tells pilot desired rate of altitude change
- Checking for consistency gave a counterexample
  - `Other_Aircraft` reverse from an `Increase-Climb` to an `Increase-Descend` advisory
  - After study, this is only permitted in our non-deterministic modelling of `Other_Aircraft`
  - Modelling a piece of `Other_Aircraft`'s logic precludes this counterexample

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 and 3000 ft/min
  - Presumably, on a climb advisory, `Display_Model_Goal` should be positive

Output agreement check

- `AG (RA = Climb -> DMG > 0)`
  - If Resolution Advisory is `Climb`, then `Display_Model_Goal` is positive
- Counterexample was found
  - t_0 : `RA = Descend, DMG = -1500`
  - t_1 : `RA = Increase-Descend, DMG = -2500`
  - t_2 : `RA = Climb, DMG = -1500`

Where may formulae come from?

"There have been two pilot reports received which indicated that TCAS had issued Descend RA's at approximately 500 feet AGL even though TCAS is designed to inhibit Descent RAs at 1,000 feet AGL. All available data from these encounters are being reviewed to determine the reason for these RAs."

--TCAS Web site

Performance results

Property	Time (secs)	#BDD nodes	Memory (MB)
Transition Consistency	387	717K	16.4
Function Consistency	289.5	387K	11.5
Step Termination	57.2	142K	7.4
Descend Inhibit	166.8	429K	11.8
Increase-Descend	193.7	282K	9.9
Output Agreement	325.6	376K	11.6

- Sun SPARCStation 10 with 128MB
- SMV Release 2.4.4

Discussion

- A positive data point for applying model checking to state based software specifications
- Iterative use of model checking promising
  - Refine and debug specification
  - Explicit clarification of interfaces
  - Regression testing of specifications

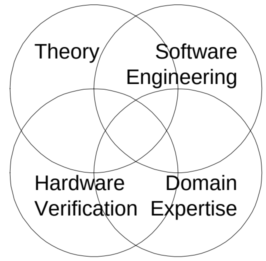
Discussion

- What are the limits?
  - Specification size
    - We produce around 200 boolean variables, about the edge of what SMV can handle
  - Numerical issues (multiply, divide, etc.)
    - Needed to refine `Other_Aircraft`
  - Desirable properties to check?
- Domain expertise is critical
  - Thanks, Jon!

Discussion

- Differences in applying to software and to hardware?
  - Word-level vs. bit-level?
  - Event structure? Synchrony hypothesis?
  - Properties to check?
  - Timing properties?

Model checking software



Displayed_Model_Goal =

0	**if** Composite_RA **not in state** Positive
Max(Own_Track_Alt_Rate, PREV(Displayed_Model_Goal), 1500 ft/min)	**if** (New_Climb **or** New_Threat) **and not** New_Increase_Climb **and not** (Increase_Climb_Cancelled **or** Increase_Descend_Cancelled) **and** Composite_RA **in state** Climb
Min(Own_Track_Alt_Rate, PREV(Displayed_Model_Goal), −1500 ft/min)	**if** (New_Descend **or** New_Threat) **and not** New_Increase_Descend **and not** (Increase_Climb_Cancelled **or** Increase_Descend_Cancelled) **and** Composite_RA **in state** Descend
2500 ft/min	**if** New_Increase_Climb
−2500 ft/min	**if** New_Increase_Descend
Max(Own_Track_Alt_Rate, 1500 ft/min)	**if** Increase_Climb_Cancelled **and not** New_Increase_Climb **and** Composite_RA **in state** Positive
Min(Own_Track_Alt_Rate, −1500 ft/min)	**if** Increase_Descend_Cancelled **and not** New_Increase_Descend **and** Composite_RA **in state** Positive
PREV(Displayed_Model_Goal)	Otherwise