Formal Analysis of a Triplex Sensor Voter at Rockwell Collins France

Michael Dierkes

TAPAS 2010 workshop

September 17, 2010
Perpignan
Agenda

1) Presentation Rockwell Collins

2) The Rockwell Collins Translator Framework

3) Analysis of a Triplex Sensor Voter
Rockwell Collins France

- Rockwell Collins France (RCF) is an **electronic systems manufacturer**

- 700+ employees, mainly located in Toulouse, France, subsidiary of Rockwell Collins Inc. (20,000 empl.)

- Systems and equipments for aircraft and rotary wing manufacturers (Airbus, Eurocopter, Augusta,…)
  - Communication, Navigation, Radar, Surveillance, Cockpit equipments

- We provide communication systems for European MODs (radio, networks)
  - Software define radio, Data Links (Link11, Link 16,…), Localization and SAR (Search And Rescue) equipments
Formal Methods at Rockwell Collins

• In the US: team of ~10 research engineers (mostly PhD)

• Work on
  – model checking (MATLAB Simulink© translator framework)
  – Theorem proving (especially ACL2)

• For 1.5 years, 1 research engineer in France

• Starting in October 2010, a PhD in France (CIFRE with ONERA)
The Rockwell Collins Translator Framework

- **Purpose**: Verification of SCADE™ and MATLAB Simulink® models

- **Long term effort** in the domain of formal methods

- Used on several projects (see articles by Steven Miller and Michael Whalen, e.g. *Software model checking takes off*, CACM 53(2), 2010)

- Based on an extension of Lustre as intermediate language

- Can output optimized descriptions in input languages of several different analyzers
The Rockwell Collins Translator Framework (2)

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

- **Lustre**
  - **NuSMV**
  - **Prover**
  - **ACL2**
  - **PVS**
  - **C, Ada**

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

---

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

Proprietary Information
The Triplex Sensor Voter

- Compute an output from input of **three redundant sensors**
- Able to detect and eliminate one **faulty sensor**
- User **reset** possible
- Implemented in **Simulink**
- Several blocks:
  - Value computation (arithmetic)
  - Fault detection (mainly boolean)
  - Reset (purely boolean)
Industrial Context of the Analysis

- **Legacy** model (~20 years old)
- Reverse engineering – why and how does it work?
- Finding right **parameters** is very time consuming
- Has been **qualified**, high confidence
- **Modifications** are made now
  - Better usage of Simulink
  - 4th input?
- **New application** areas
Normal Operation Mode of the Voter (no fault)

- From each of the three inputs, subtract an equalization value
- Output is middle value of equalized values
- Equalization based on integration – 3 memories of rational type
Simulation: Input Values
Simulation: Equalized Values
Simulation: Output Value
Equations of the Normal Operation Mode

\[ \text{Equalization}A_0 = 0.0 \]
\[ \text{Equalization}B_0 = 0.0 \]
\[ \text{Equalization}C_0 = 0.0 \]

\[ \text{Centering}_t = \text{middleValue}(\text{Equalization}A_t, \text{Equalization}B_t, \text{Equalization}C_t) \]

\[ \text{Equalized}A_t = \text{Input}A_t - \text{Equalization}A_t \]
\[ \text{Equalized}B_t = \text{Input}B_t - \text{Equalization}B_t \]
\[ \text{Equalized}C_t = \text{Input}C_t - \text{Equalization}C_t \]

\[ \text{VoterOutput}_t = \text{middleValue}(\text{Equalized}A_t, \text{Equalized}B_t, \text{Equalized}C_t) \]

\[ \text{Equalization}A_{t+1} = \text{Equalization}A_t + 0.05 \times (\text{sat}_{0.5}(\text{Equalized}A_t - \text{VoterOutput}_t) - \text{sat}_{0.25}(\text{Centering}_t)) \]
\[ \text{Equalization}B_{t+1} = \text{Equalization}B_t + 0.05 \times (\text{sat}_{0.5}(\text{Equalized}B_t - \text{VoterOutput}_t) - \text{sat}_{0.25}(\text{Centering}_t)) \]
\[ \text{Equalization}C_{t+1} = \text{Equalization}C_t + 0.05 \times (\text{sat}_{0.5}(\text{Equalized}C_t - \text{VoterOutput}_t) - \text{sat}_{0.25}(\text{Centering}_t)) \]
Sensors and their Faults

• Non-faulty sensors furnish a value within an interval around true value determined by constant $\text{MaxSensorError}$

\[
\text{abs}(\text{SensorValue} – \text{TrueValue}) \leq \text{MaxSensorError}
\]

• Fault detection is based on equalization values
Objectives of the Analysis

- Analyse output to show that transient peaks cannot occur
- Find good parameters for fault detection and prove that a non-faulty sensor is never eliminated
- Correct behaviour: output tends to middle input value
- No overflows
- In general, what can we do with our translator framework?
Approach of the Analysis

• Check on **model level**

• Handle **real** values in model as **rational** values

• Proof by induction -> **invariants** necessary
Example Properties

- What is the **maximal output error** for a given maximal sensor error?
  
  \[ \text{abs(VoterOutput} - \text{TrueValue)} \leq \]

- What is the **maximal difference** of two equalization values for a given maximal sensor error?
  
  \[ \text{abs(EqualizationA} - \text{EqualizationB)} \leq \]
Inductive Invariant

- \( \text{Abs}(\text{Equalization}_A - \text{Equalization}_B) \leq 0.4 \)
- \( \text{Abs}(\text{Equalization}_A - \text{Equalization}_C) \leq 0.4 \)
- \( \text{Abs}(\text{Equalization}_B - \text{Equalization}_C) \leq 0.4 \)
- \( \text{Abs}(\text{Equalization}_A + \text{Equalization}_B + \text{Equalization}_C) \leq 0.66 \)
- \( \text{Abs}(\text{Equalization}_A) \leq 0.4 \)
- \( \text{Abs}(\text{Equalization}_B) \leq 0.4 \)
- \( \text{Abs}(\text{Equalization}_C) \leq 0.4 \)
- \( \text{Abs}(\text{middle}(\text{Equalization}_A, \text{Equalization}_B, \text{Equalization}_C)) \leq 0.24 \)
Simulation and Proof
Inductive Octagonal Invariant

- \( \text{Abs}(\text{EqualizationA}) \leq 0.4 \)
- \( \text{Abs}(\text{EqualizationB}) \leq 0.4 \)
- \( \text{Abs}(\text{EqualizationC}) \leq 0.4 \)

- \( \text{Abs}(\text{EqualizationA} - \text{EqualizationB}) \leq 0.4 \)
- \( \text{Abs}(\text{EqualizationA} - \text{EqualizationC}) \leq 0.4 \)
- \( \text{Abs}(\text{EqualizationB} - \text{EqualizationC}) \leq 0.4 \)

- \( \text{Abs}(\text{EqualizationA} + \text{EqualizationB}) \leq 0.6 \)
- \( \text{Abs}(\text{EqualizationA} + \text{EqualizationC}) \leq 0.6 \)
- \( \text{Abs}(\text{EqualizationB} + \text{EqualizationC}) \leq 0.6 \)
Inductive Octagonal Invariant
Simple Automatic Generation of Inductive Invariants

Choose a set of expressions $\text{expr}_1$, ..., $\text{expr}_n$ over state variables $v_1, ..., v_n = 0.0$

Repeat

Check if $(\text{abs} (\text{expr}_1) \leq v_1 \text{ and } ... \text{ and } \text{abs} (\text{expr}_n) \leq v_n)$

is inductive invariant

For all i

If step counter example exists with $\text{abs} (\text{expr}_i) > v_i$

$v_i += 0.01$

Until no counter example exists
Analysis with Astrée

- Implementation of the (reduced) voter in C
- Astrée casts **false alarms** on **overflow** of equalization values
- Communicated to AbsInt
- Confirm the inductive invariant on code level?
Lessons Learnt from Analysis

- Inductive proof was used, finding invariants was very **time consuming**

- All terms in invariant are **linear** (sums and differences)

- For max output error: still **gap** between value found by simulation and value proved

- **BMC not helpful**: too many steps necessary
Ongoing Work

• Extension to \textit{fault case}

• \textbf{Speed up analysis} by adding lemmas

• Try to find \textit{closer approximation} of state space

• Experiment \textit{different proof engines} (e.g. new version of Kind)
Future Directions

• Can invariants be found *automatically*? By abstract interpretation?

• **Other forms** of invariants (non-linear, combined with boolean conditions, etc.)?

• Potential case study for *combining* model checking and abstract interpretation (CMACS, PhD RCF/ONERA)

• Relevance for implementation with *floating point numbers*?
Thank you!