Using model checking to improve the quality of cyber-physical systems

Jeroen Keiren
Open University of the Netherlands & Radboud Universiteit Nijmegen
11 July 2016

@jkeiren
jeroenkeiren.nl
Cyber-Physical Systems

Cyber-Physical Systems or “smart” systems are co-engineered interacting networks of physical and computational components. – NIST

Examples

- Smart grid
- Autonomous automotive systems
- Medical monitoring
- Robotics

Networks of embedded systems with physical input and output
Modelling and analysis of cyber-physical systems

Two types of behaviour in CPS:

- Continuous dynamics
- Discrete dynamics

This talk: discrete dynamics
Case study 1: IEEE 11073-20601

Health informatics — Personal health device communication —
Part 20601: Application profile — Optimized exchange protocol

Informatique de santé — Communication entre dispositifs de santé personnels —
Partie 20601: Profil d’application — Protocole d’échange optimisé

and many more
What are personal health devices?

Agent devices

Manager devices
What is the purpose of IEEE 11073-20601?

“[... ] define[... ] a common framework for making an abstract model of personal health data available in transport-independent transfer syntax required to establish logical connections between systems and to provide presentation capabilities and services needed to perform communication tasks.” [IEEE std 11073-20601]
Protocol (sketch)

Agent and manager cycle through states:
- Unassociated
- Associating
- Sending configuration
- Waiting approval
- Operating
- Disassociating
Objectives & Approach

- Assess understandability, consistency, and completeness
  - Formalise protocol in mCRL2 (ACP-like process algebra)
  - Based on standards document
- Establish correctness
  - Formulate desired properties in temporal logic ($\mu$-calculus)
  - Use model checker to verify the properties
- Fix bugs
Observations on the standard
Understandability, consistency and completeness

- **Requirements** are not explicit
- **Formalisms** not introduced
- **Information** inconsistently duplicated between representations
  - Inconsistent **terminology/abbreviations**
  - Different **state changes** for same event
- **Unexpected messages** not fully treated
Deadlocks caused by incompleteness

Diagram:
- Agent
- Buffer
- Manager
- AssocReq
- AssocRsp(accepted_unknown_config)
- AssocRelReq
- ConfigEventReportReq
- AssocAbort
Correctness

Data shall not be transmitted in inconsistent operating states.
Observations

- Omissions and inconsistencies cause easy to fix bugs
- Session setup contains a severe bug
- Agent and manager devices can transfer data in inconsistent configurations
- Rest of the protocol should be verified
Lessons learned

▶ Formal modelling allows detection of problems in short timespan
▶ Standards should provide clear requirements
▶ Bugs cannot be fixed without breaking standard conformance
▶ Formal verification should be part of the development process of every communication standard

Unverified standards contain subtle, hard to find errors!
Case study 2: CERN

source: CERN
Large Hadron Collider

source: CERN
Large Hadron Collider

source: CERN
CMS detector: Scale

source: CERN
Control software

source: CERN
Control software: Global structure

- Top Control
- Software (FSMs)
- Devices

Status/Alarms
- Commands
Control software: Local structure

Wheel subsystem

- Wheel (2)
  - Sector (3)
    - Chamber (5)
  - Sector (4)
    - Chamber (6)
    - Chamber (7)
    - Chamber (8)
Control software: Complexity
Problem: Unresponsive subsystems
Methodology

1. Understand/define semantics SML
2. Identify desirable properties
3. Verify properties
4. Automate verification
5. Optimise verification (dedicated tooling)
6. Integrate tooling into IDE
Example (SML)

class: $FWPART_$TOP$RPC_Chamber_CLASS
  state: OFF
    when (($ANY$FwCHILDREN in_state ERROR) or
           ($ANY$FwCHILDREN in_state TRIPPED))
    move_to ERROR

... 

action: STANDBY
  do STANDBY $ALL$RPC_HV
  do ON $ALL$RPC_LV
Processing in a state machine

When phase

- receive state-update
- all guards false

- evaluating when clauses
- command queue empty

Action phase

- waiting for command or state-update
- executed last statement

- executing statements
- received command

- emptying command queue
Stabilisation: Livelocks

state: ANALOG_ON_RED

... when ( $ANY$TkPowerGroup not_in_state DIGITAL_ON_RED ) move_to LVMIXED

...

state: LVMIXED

... when ( $ALL$FwCaenChannelCtrl in_state ON and
$ALL$TkPowerGroup in_state ANALOG_ON_RED ) move_to ANALOG_ON_RED

...
Stabilisation

- TkControlGroup (ANALOG_ON_RED)
  - TkPowerGroup (ANALOG_ON_RED)
  - FwCaenChannelCtrl (ON)
state: ANALOG_ON_RED
...
when ( $ANY$TkPowerGroup not_in_state DIGITAL_ON_RED )
move_to LVMIXED
...
state: LVMIXED

... when ( $ALL$FwCaenChannelCtrl in_state ON and $ALL$TkPowerGroup in_state ANALOG_ON_RED ) move_to ANALOG_ON_RED

...
Stabilisation

state: ANALOG_ON_RED
...
when ( $ANY$TkPowerGroup not_in_state DIGITAL_ON_RED )
move_to LVMIXED
...

TkControlGroup
(ANALOG_ON_RED)

TkPowerGroup
(ANALOG_ON_RED)

FwCaenChannelCtrl
(ON)
Stabilisation

state: LVMIXED

... when ($\text{ALL}$$\text{FwCaenChannelCtrl}$ in_state ON and
$\text{ALL}$$\text{TkPowerGroup}$ in_state ANALOG_ON_RED )
move_to ANALOG_ON_RED

...
Livelocks
Formalisation

\[ \mathcal{F} + \text{children} \rightarrow \mathcal{M} \]

Lemma

\[ \mathcal{F} \text{ contains a loop of } \text{move\_to} \text{ actions iff } \mathcal{M} \text{ contains loops} \]
Livlocks
Translation to SAT

Existence of loop in $\mathcal{F}$ as satisfiability formula $\varphi_\mathcal{F}$:

1. state constraints
   - each FSM is always in exactly one state
   - children do not change state in when-phase

2. transition relation
   - move-to steps parents can take

3. loop condition
   - parent must be able to return to its starting state
Liveloacks
Translation to SAT

Theorem
There is a loop in $\mathcal{F}$ iff $\varphi_{\mathcal{F}}$ is satisfiable

SAT encoding will find child states if loop exists!
Results: Livelocks

- Full system checking in 79 seconds
- 1302 FSMs have looping potential
- Most not observed/short lived
- Outages of control system traced back to detected problems:

... 
... 
...
Unreachable states

- ERROR
- STANDBY
- ON
- BUSY
- OFF
Results: Reachability

- Full system checked in 18 minutes
- 903 FSMs have reachability issues
- Partly due to clever programmer tricks
- Real problems typically due to copy/paste
Implementation

- Full automated translation to mCRL2
- Dedicated translations to SMT for described problems
- Integration of dedicated tools in IDE
Lessons learned

- Generic tools needed to develop understanding
- Huge system, yet verifiable due to careful analysis
- Specialised tools needed for effective verification
- Real-life problems detected
- Diagnostics ensure quick fixing

“We should have had these tools at the start of the LHC project” — CMS engineer
Verification of Cyber Physical Systems

Cyber physical systems
Networks of embedded systems with physical input and output

- Modelling and verification well-understood
- Possible for huge systems
- Modelling and verification using differential equations
- Well-studied
- Combining discrete- and continuous dynamics in single formalism
- Modelling and verification using: hybrid automata
- Currently studying this with Prof. Cleaveland
Thank you

Joint work with:

▶ Martijn Klabbers (LaQuSo)
▶ Yi Ling Hwong (CERN)
▶ Vincent Kusters (TU/e, CERN, ETH-Z)
▶ Sander Leemans (TU/e, CERN)
▶ Tim Willemse (TU/e)

source: http://xkcd.com/401