Analysis Contracts for Cyber-Physical Systems

Presenter: Dionisio de Niz
  dionisio@sei.cmu.edu

AADL Std Meeting. Sept 30th, 2014

Team: Ivan Ruchkin, Sagar Chaki, David Garlan

1 Software Engineering Institute &
2 School of Computer Science
Analysis Contracts for Cyber-Physical Systems

Dionisio de Niz Sagar Chaki, Ivan Ruchkin, David Garlan

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Approved for public release, distribution unlimited

The original document contains color images.
Motivation

The development of Cyber-Physical Systems (aircrafts, cars, trains, robots, etc.) increasingly relies on many types of analyses from different disciplines for assurance purposes

• Control stability, scheduling, logic, thermal, power, aerodynamics, etc.

Large CPS are integrated out of components developed by suppliers that use their own analysis methods and make their own assumptions

Analysis assumption mismatches are discovered late in the system integration phase

• Difficult and costly to solve
Boeing 787 Suppliers

Boeing Australia
Kawasaki
Mitsubishi
Spirit
Saab
Alenia
Latecoere
Rolls Royce
Goodrich
KAL-ASD
Boeing US

Source: Boeing / Reuters
Analyses Interactions

Scheduling + Frequency Scaling

Battery Recharge Scheduling

Selected Voltage

Cell Interconnects

Thermal Runaway Analysis

Source: National Renewable Energy Laboratory
Analysis Contracts

- Frequency Scaling
  - Power
  - Exec Time

- Schedulability
  - Read/write

- Model checking
  - deadlock

- Control Stability
  - Read/write

- Battery Sched
  - Read/write

- Assumption
  - Thermal runaway

- Guarantee
  - Read/write

- Frequency Scaling
  - Power
  - Exec Time

- Schedulability
  - Read/write

- Model checking
  - deadlock

- Control Stability
  - Read/write

- Battery Sched
  - Read/write

- Assumption
  - Thermal runaway

- Guarantee
  - Read/write

- AADL
Analysis Contract Scheme

Model

Analysis 1
Contract 1
Domain 1

Analysis 2
Contract 2
Domain 2

Analysis 3
Contract Semantic Basis

Domain \((\mathcal{A}, S, \mathcal{R}, \mathcal{T}, [\cdot]_\sigma)\)

- Sorts \(\mathcal{A} = \{A_1, A_2, \ldots\}\), e.g. Booleans, Integers, Threads, Priorities
- Static Properties : \(S = \{S_i\}, \ S_i: A_1 \times \cdots \times A_j \rightarrow A_k\)
  - Design-time invariants, Regular operators: \((\land, \mathcal{B} \times \mathcal{B} \rightarrow \mathcal{B})\)
- Runtime Properties : \(\mathcal{R} = \{R_i\}, \ R_i: A_1 \times \cdots \times A_j \rightarrow A_k\)
  - Evolving valuation at different states \(q: q(R_i)\)
- Domain execution semantics: \(\mathcal{T}\)
  - Infinite sequence of assignments to runtime properties (executions)
- Domain interpretation of sorts/static properties: \([\cdot]_\sigma\)
  - E.g. allowed schedulers, some left uninterpreted

Architectural Model Interpretation: \([\cdot]_M\) on \(\mathcal{A}, S, \mathcal{T}\)

- E.g. threads and periods: \([T]_M = \{t_1, t_2\}; \ [Per]_M = \{t_1 \mapsto 10, t_2 \mapsto 20\}\)

Executions of system defined by \(M:\) \([\mathcal{T}]_M\)

- Combining \([\cdot]_\sigma, [\cdot]_M\) into \([\cdot]\)
- Each state \(q\) in possible states \(Q\) maps \(R_i\) to function \(q(R_i): [A_1] \times \cdots \times [A_j] \mapsto [A_k]\)
- With all infinite sequence of states \(Q^\omega\)
- \([\mathcal{T}]_M \subseteq Q^\omega\)
Contract Language

Contract formulas

- Given domain $\sigma = (\mathcal{A}, \mathcal{S}, \mathcal{R}, \mathcal{T}, \llbracket \cdot \rrbracket_\sigma)$,
- $\mathcal{F}_\sigma ::= \forall v_1, \ldots, v_j \cdot \phi \mid \exists v_1, \ldots, v_j \cdot \phi \mid \forall v_1, \ldots, v_j \cdot \phi : \psi \mid \exists v_1, \ldots, v_j \cdot \phi : \psi$
  - $v_i: A_i$, $\phi$: static (first order) formula
  - $\psi$: LTL formula

Contract $C = (I, O, A, G)$

- $I \subseteq (\mathcal{A} \cup \mathcal{S})$: Sorts and properties read by the analysis
- $O \subseteq (\mathcal{A} \cup \mathcal{S})$: Sorts and properties written by the analysis
- $A \subseteq \mathcal{F}_\sigma$: assumptions: must be true in input
- $G \subseteq \mathcal{F}_\sigma$: guarantees: must be true in output
Contract Verification

Given model M and set of analyses $\mathcal{AN} = \{An_i\}$:

- For $An_i. C = (I, O, A, G)$: M application to $An_i$ iff
  
  $\forall a \in A \cdot M \models a, \forall g \in G \cdot An_i(M) \models g$

Valid analysis ordering: no dependencies from later analysis

- Contract (& analysis) dependency: $d(C_i, C_j): C_i. I \cap C_j. O \neq \emptyset$

Contract Formulas

- First order: in SMT (Z3)
- LTL: Model checker
- FOL + LTL: Generate all solutions for FOL, check LTL in each
Example: Surveillance Aircraft

Analysis

**Security**: tasks of different level to different processor

**Scheduling**: meet all deadlines

**Freq. Scaling**: minimize power

**Logic**: no deadlocks or race conditions

**Battery scheduling**: meet battery lifetime

**Battery thermal**: no runaways
Surveillance Aircraft Contracts

Security Analysis

- $A_{sec}.C: I = \{T, ThSecCl\}, O = \{NotColoc\}, A = \emptyset, G = \{g\}
  - $g: \forall t_1, t_2 \cdot ThSecCl(t_1) \neq ThSecCl(t_2) \Rightarrow t_1 \in NotColoc(t_2)$

Multiprocessor scheduling: (Binpacking + scheduling)

- $A_{sched}.C: I = \{T, C, NotColoc, Per, WCET, Dline\}, O = \{CPUBind\}, A = \emptyset, G = \{g\}
  - $g: \forall t_1, t_2 \cdot NotColoc(t_2) \Rightarrow CPUBind(t_1) \neq CPUBind(t_2)$

Frequency Scaling

- $A_{freqsc}.C: I = \{T, C, CPUBind, Dline\}, O = \{CPUFreq\}, G = \emptyset, A = \{a\}
  - $a: \forall t_1, t_2 \cdot CPUBind(t_1) = CPUBind(t_2) \Rightarrow Dline(t_1) < Dline(t_2)$

Model checking periodic program (REK):

- $A_{rek}.C: I = \{T, C, Per, Dline, WCET, CPUBind\}, O = \{ThSafe\}, G = \emptyset, A = \{a_1, a_2\}$
  - $a_1: \forall t \cdot Per(t) = Dline(t), a_2: \forall t_1, t_2 \cdot G(\text{CanPrmpt}(t_1, t_2) \Rightarrow G \neg \text{CanPrmpt}(t_2, t_1))$

Thermal runaway:

- $A_{therm}.C: I = \{B, BatRows, BatCols, Voltage\}, O = \{K\}, A = \emptyset, G = \emptyset$

Battery Scheduling

- $A_{bsched}.C: I = \{B, BatRows, BatCols\}, O = \{BatConnSchedPol, HasReqLifetime, SeriqlReq, ParalRea\}, A = \emptyset, G = \{g\}$
  - $g: G(K(0) \times TN(0) + K(1) \times TN(1) + K(2) \times TN(2) + K(3) \times TN(3) \geq 0)$
Frequency Scaling Assumption

\[ \forall t_1, t_2 \cdot \text{CPUBind}(t_1) = \text{CPUBind}(t_2) \Rightarrow G(\text{CanPrmt}(t_1, t_2) \Rightarrow Dline(t_1) < Dline(t_2)) \]

- **DMS ≠ RMS**
  - P = D
  - DMS ≡ RMS
  - P = D

- **EDF ≠ RMS**
  - P = D

- **DMS = RMS**
  - P = D

- **EDF = RMS**
  - P = D, Harmonic, Sync
Battery Scheduling Assumption

\[ g: G(K(0) \times TN(0) + K(1) \times TN(1) + K(2) \times TN(2) + K(3) \times TN(3) \geq 0) \]

Ratio of cells with 0, 1, 2, 3 neighbors: \[ 1 \cdot TN(1) - 1 \cdot TN(2) + 10 \cdot TN(3) \geq 0 \]

\[ 1 \cdot 4 - 1 \cdot 10 + 10 \cdot 2 = 14 \geq 0 \]
\[ 1 \cdot 2 - 1 \cdot 14 + 10 \cdot 0 = -12 < 0 \]
Analyses Dependencies

\[ An_{sched} \rightarrow \text{CPUBind} \rightarrow An_{REK} \]

\[ An_{sec} \rightarrow \text{NotColoc} \rightarrow An_{freqsc} \]

\[ An_{freqsc} \rightarrow \text{Voltage} \rightarrow An_{bsched} \]

\[ An_{therm} \rightarrow K \rightarrow An_{bsched} \]
Implementation

Models in the Architecture Analysis and Design Language (AADL)

- Supports multiple analysis
- Supports language extensions (subannexes)
- OSATE Implementation

Analysis Contract Annex

- Implement contract language
- Generates model interpretation

Contract formulas verification

- First Order Logic (Static): SMT / Z3
- LTL (Runtime): Model checking / SPIN
Contract Verification Architecture

OSATE Execution Environment

- AADL types
- AADL instances
- OSATE-DB converter

Analysis execution controller

SMT verification engine

- SMT problem
- Z3

Sched verification engine

- Sched Promela model
- Spin

Batt verification engine

- Batt Promela model

Analysis tools

Model DB

Legend

- Data
- Object
- Executable
First-Order Logic Verification (Z3)

(define-sort thread () (Int))
(define-sort processor () (Int))
(declare-fun Actual_Processor_Binding (thread) Int)
(define-fun Not_Collocated (x1 thread) (x2 thread) Bool
  (ite (and (= x1 0) (= x2 1)) true (ite (and (= x1 0)(= x2 2)) true
    (ite (and (= x1 1)(= x2 0)) true (ite (=x1 2)(= x2 0)) true false ))))
(assert (= (Actual_Processor_Binding 0) 0))
(assert (= (Actual_Processor_Binding 1) 1))
(assert (= (Actual_Processor_Binding 2) 1))
(assert (not (forall ((x1 thread) (x2 thread)) (=>
  (and (or (= x1 0) (= x1 1) (= x1 2))
    (or (= x2 0) (= x2 1)(= x2 2))
  (=> (and (not (= x1 x2)) (Not_Collocated x1 x2))
    (not (= (Actual_Processor_Binding x1) (Actual_Processor_Binding x2)))))))
(check-sat)
Model Checking for Scheduling

Periodic Tasks

Multiple Processors Allows Global Scheduling

Priority Based Scheduling

- Fixed-Task Priorities: fixed at configuration time (RMS / DMS)
- Dynamic-Task (fixed job): changes at job arrival (EDF)

Tickless model

- Time advances by scheduling events
  - Deterministically: next event is a deterministic arrival
  - Non-deterministically: multiple possible events

Clock variable resets

- To the earliest event in variables
Model Checking for Battery Scheduler

Model battery cell scheduler

• Schedules cells to discharge, charge, rest
• Match required output through serial connections (efficient)
• Maximizes battery lifetime

Matrix of battery cells

Connections between cells change dynamically

• Reflects needs to provide output voltage (serial connections)
• at certain current (parallel connections)

Cell charge changes to reflect charge discharge
## Evaluation

### Real-Time Scheduling

<table>
<thead>
<tr>
<th>Threads</th>
<th>(R/D)MS Time</th>
<th>EDF Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0.01</td>
<td>0.01</td>
</tr>
<tr>
<td>4</td>
<td>0.01</td>
<td>0.52</td>
</tr>
<tr>
<td>5</td>
<td>0.07</td>
<td>33.4</td>
</tr>
<tr>
<td>6</td>
<td>0.37</td>
<td>2290.0</td>
</tr>
<tr>
<td>7</td>
<td>2.18</td>
<td>Out Mem</td>
</tr>
<tr>
<td>8</td>
<td>12.4</td>
<td>Out Mem</td>
</tr>
<tr>
<td>9</td>
<td>71.2</td>
<td>Out Mem</td>
</tr>
<tr>
<td>10</td>
<td>421</td>
<td>Out Mem</td>
</tr>
<tr>
<td>11</td>
<td>Out Mem</td>
<td>Out Mem</td>
</tr>
</tbody>
</table>

### Battery Scheduling

<table>
<thead>
<tr>
<th>Cells</th>
<th>FGURR</th>
<th>FGWRR</th>
<th>GPWRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>0.13</td>
<td>0.15</td>
<td>0.15</td>
</tr>
<tr>
<td>12</td>
<td>0.61</td>
<td>2.34</td>
<td>3.94</td>
</tr>
<tr>
<td>16</td>
<td>44</td>
<td>31.4</td>
<td>127</td>
</tr>
<tr>
<td>20</td>
<td>1060</td>
<td>619</td>
<td>Out Mem</td>
</tr>
<tr>
<td>25</td>
<td>Out Mem</td>
<td>Out Mem</td>
<td>Out Mem</td>
</tr>
</tbody>
</table>

Time in seconds. Amazon EC2 virtual Machine with 8 cores and 30 GB of mem.
Conclusions

Analyses are key for development of CPS

- But inconsistent assumptions may compromise results

Analysis contracts to automatically verify assumptions

- Analysis contract language & verification framework
- Implementation in AADL sub-annex

Example

- Two domains
- Five analyses

Analysis contracts: sound and scalable

- Single multi-domain analysis intractable