Contract-Based Integration of Cyber-Physical Analyses

Ivan Ruchkin
Dionisio De Niz
Sagar Chaki
David Garlan

October 14, 2014
14th International Conference on Embedded Software
Copyright 2014 ACM

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN “AS-IS” BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

This material has been approved for public release and unlimited distribution.

This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.


DM-0001714
Outline

• Analysis integration problem
• Analysis contracts approach
  – Specification
  – Verification
• Experimental results
Model integration in CPS

- Subtle mismatches between technical domains
- Lead to costly fixes or failures
Analytic aspect of integration

- Frequency scaling is applicable *only* when:
  - used after Bin packing
  - the system is behaviorally deadline-monotonic
- Otherwise, frequency scaling may render the system unschedulable
Frequency scaling assumption

- Behavioral equivalence to deadline-monotonic scheduling

RMS ≠ DMS

EDF ≠ DMS

RMS = DMS

EDF = DMS

P=D, Harmonic, Sync

P=D
Analysis integration problem

- Out-of-order execution
- Invalidation of assumptions
Existing solutions

- Assume-guarantee component composition does not handle analytic integration of tools [1][2].
- Architectural views tackle model consistency, not analytic tool consistency [3][4]
- Meta-level AADL languages do not allow domain-specific semantics [5]
- Previous work on analysis contracts: single domain only, unsound and incomplete verification [6]

Running example

Scheduling

Frequency scaling

Thread model checking

deadlock

Bin packing

Data security

Battery

Battery Scheduling

Discharge

Charge

Thermal runaway

System

Thread

Thread

Thread

CPU

CPU

CPU

Battery

Battery
Outline

- Analysis integration problem
- **Analysis contracts approach**
  - Specification
  - Verification
- Experimental results
Analysis contracts approach

1. Formalize analysis domains
2. Specify dependencies, assumptions, and guarantees of analyses
3. Determine correct ordering of analyses
4. Verify assumptions and guarantees of analyses
Outline

- Analysis integration problem
- Analysis contracts approach
  - Specification
  - Verification
- Experimental results
Running example

Scheduling

- Frequency scaling (Power vs. Exec Time)
- Thread model checking (deadlock)
- Bin packing
- Data security

Battery

- Battery Scheduling (Discharge, Charge)
- Thermal runaway (Temp)

System

- Threads
- CPU
- Battery
Running example

Scheduling domain $\sigma_{\text{sched}}$

Battery domain $\sigma_{\text{batt}}$

System
Verification domain

Domain $\sigma$ is a many-sorted signature $(\mathcal{A}, \mathcal{S}, \mathcal{R}, \mathcal{T}, \{\} \sigma)$:

- $\mathcal{A}$ - set of atoms: $\mathcal{B}$, $\mathcal{Z}$, Threads, Batteries, SchedPol
- $\mathcal{S}$ - static functions: Period, Dline, CPUBind, Voltage
- $\mathcal{R}$ - runtime functions CanPrmpt: Threads $\times$ Threads $\rightarrow$ $\mathcal{B}$
- $\mathcal{T}$ - execution semantics
  - set of sequences of $\mathcal{R}$ assignments
- $\{\} \sigma$ and $\{\} \text{m}$ - domain and model interpretations
  - $\{SchedPol\} \sigma = \{$RMS, DMS, EDF$\}$
  - $\{CPUBind\} \text{m} = \{$(Ctrl_1, CPU_1)$, $(Ctrl_2, CPU_2)$, ... $\}$
Analysis contract

- Given a domain $\sigma$, *analysis contract* $\mathbf{C}$ is a tuple $(\mathbf{I}, \mathbf{O}, \mathbf{A}, \mathbf{G})$
  - Inputs $\mathbf{I} \subseteq \mathbf{A} \cup \mathbf{S}$
  - Outputs $\mathbf{O} \subseteq \mathbf{A} \cup \mathbf{S}$
  - Assumptions $\mathbf{A} \subseteq \mathcal{F}_\sigma$
  - Guarantees $\mathbf{G} \subseteq \mathcal{F}_\sigma$
- Where:
  - $\mathcal{F}_\sigma ::= \{\forall|\exists\} v_1..v_n \cdot \phi \mid \{\forall|\exists\} v_1..v_n \cdot \phi : \psi$
  - $\phi$ is a static predicate formula over $\mathbf{A}$ and $\mathbf{S}$
  - $\psi$ is an LTL formula over $\mathbf{A}$, $\mathbf{S}$, and $\mathcal{R}$
  - E.g.: $\forall t_1, t_2: \text{Threads} \cdot t_1 \neq t_2 \land \text{CPUBind}(t_1) = \text{CPUBind}(t_2):$
    $$\mathbf{G} (\text{CanPrmpt}(t_1, t_2) \Rightarrow \text{Dline}(t_1) < \text{Dline}(t_2))$$
Outline

• Analysis integration problem
• Analysis contracts approach
  – Specification
  – Verification
• Experimental results
Running example

Scheduling domain $\sigma_{\text{sched}}$

Battery domain $\sigma_{\text{batt}}$

System

Thread

Thread

Thread

CPU

CPU

CPU

Battery

Battery

Discharge

Charge

Frequency scaling

Exec Time

Data security

Bin packing

deadlock

Battery Scheduling

Thermal runaway

Temp
Assumption verification

- **Goal:**
  \[ \forall t_1, t_2 : Threads \cdot t_1 \neq t_2 \land CPUBind(t_1) = CPUBind(t_2) : \]
  \[ G (CanPrmpt(t_1, t_2) \Rightarrow Dline(t_1) < Dline(t_2)) \]

  - SMT solver finds solutions for static fragment \( \varphi \)
    \[ \forall t_1, t_2 : Threads \mid t_1 \neq t_2 \land CPUBind(t_1) = CPUBind(t_2) \]

  - Model checking property \( \psi \) in a behavioral Promela model for each SMT solution:
    \[ G (CanPrmpt(t_1, t_2) \Rightarrow Dline(t_1) < Dline(t_2)) \]
Battery modeling

Battery domain $\mathcal{G}_{batt}$

- Abstraction: circuits
- Selects a scheduler for cell connections
- Oblivious of heat: treats any configuration as acceptable heat-wise

- Restrictions on acceptable thermal configurations
- Guarantee: unacceptable ones don't occur

- Abstraction: geometry
- Simulates heat propagation
- Cannot scale to dynamic scheduling: simulates only fixed cell configurations
Battery scheduling guarantee

- $G$: “Bad thermal configurations are not reachable”
- $TN(b, i) \in \mathcal{R}$ – number of cells in $b$ with $i$ thermal neighbors
- $K(b, i) \in \mathcal{S}$ – experimental weight for $TN(b, i)$
- $G = \{ \forall b: \text{Batteries} \cdot G ( \sum_{i=0..3} K(b, i) \cdot TN(b, i) ) \geq 0 \}$
Battery modeling

Battery domain $\sigma_{\text{batt}}$

Selects a battery scheduler

$G: \forall b: \text{Batteries} \cdot G \left( \sum_{i=0..3} K(b, i) \cdot TN(b, i) \right) \geq 0$

Verified with battery Promela/Spin model

Determines $K(b, i)$ via simulation
Outline

• Analysis integration problem
• Analysis contracts approach
  – Specification
  – Verification
• Experimental results
Framework implementation

OSATE Execution Environment

AADL model instances → AADL-DB converter → Analysis tools

AADL types → AADL-DB converter → Analysis tools

Model DB

SMT verification engine

Sched verification engine

Batt verification engine

Z3

Sched Promela model

Spin

Batt Promela model

Legend

Data Object Executable ➔ Dataflow
Scalability evaluation

- SMT solving typically takes less than 0.1 second
- Spin model checking times:

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

<table>
<thead>
<tr>
<th>Cells</th>
<th>FGURR time</th>
<th>FGWRRI time</th>
<th>GPWRRI time</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>

All times are in seconds
Summary

- Analysis integration is error-prone
  - Incorrect ordering
  - Violation of implicit assumptions
- Our solution:
  - Contract specification language
  - Contract verification algorithm
  - Framework implementation
- Effective, extensible, and scalable
Verification domain

- Domain $\sigma$ is a many-sorted signature ($\mathcal{A}$, $\mathcal{S}$, $\mathcal{R}$, $\mathcal{T}$, $\{\}^\sigma$):
  - $\mathcal{A}$: set of sorts – system elements and standard sorts
    - E.g.: $\mathcal{B}$, $\mathbb{Z}$, Threads, Batteries, SchedPol
  - $\mathcal{S}$: $\mathcal{A}_i \times \ldots \times \mathcal{A}_n \rightarrow \mathcal{A}_k$ – static functions that encode design properties
    - E.g.: Period, Dline, CPUBind, Voltage
  - $\mathcal{R}$: $\mathcal{A}_i \times \ldots \times \mathcal{A}_n \rightarrow \mathcal{A}_k$ – runtime functions that encode dynamic properties
    - E.g.: CanPrmpt: Threads $\times$ Threads $\rightarrow \mathcal{B}$
      TN: Batteries $\times$ $\mathbb{Z}$ $\rightarrow$ $\mathbb{Z}$
Verification domain

- Domain $\sigma$ is a many-sorted signature $(\mathcal{A}, \mathcal{S}, \mathcal{R}, \mathcal{T}, \{\}_\sigma)$:
  - $\mathcal{T}$: execution semantics – set of sequences of $\mathcal{R}$ assignments
    - E.g.: thread scheduler state model for $\sigma_{\text{sched}}$
      battery state model for $\sigma_{\text{batt}}$
  - $\{\}_\sigma$: domain interpretation for $\mathcal{A}$ and $\mathcal{S}$
    - E.g.: $\{\text{SchedPol}\}_\sigma = \{\text{RMS, DMS, EDF}\}$
- Architectural model $\mathbf{m}$ is an interpretation $\{\}_\mathbf{m}$ of $\mathcal{A}$, $\mathcal{S}$, and $\mathcal{T}$
  - E.g.: $\{\text{Threads}\}_\mathbf{m} = \{\text{SensorSample, Ctrl}_1, \text{Ctrl}_2\}$
    $\{\text{CPUBind}\}_\mathbf{m} = \{(\text{Ctrl}_1, \text{CPU}_1), (\text{Ctrl}_2, \text{CPU}_2), \ldots\}$
  - $\{\}_\sigma \cup \{\}_\mathbf{m}$ is a full interpretation
Contracts

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

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

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

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

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

Battery Scheduling
- $A_{\text{psched}} \cdot C : I = \{B, \text{BatRows}, \text{BatCols}\}, O = \{\text{BatConnSchedPol}, \text{HasReqLifetime}, \text{SeriqLReq}, \text{ParalRea}\}, A = \emptyset, G = \{g\}$
  - $g : G(K(0) \times TN(0) + K(1) \times TN(1) + K(2) \times TN(2) + K(3) \times TN(3) \geq 0)$