Problem

CPS engineering combines diverse model-based analyses from various engineering domains. Differences in domain abstractions lead to integration issues:

- If an assumption of analysis is violated by another, the outputs of the former may be invalid.
- Specification of such implicit assumptions and detection of their violation is left to human designers, who are often unable to cope with complexity.
- Analysis integration problems discovered late in development lead to expensive changes to the system.

Hence the research question:

- How to specify analysis compositions and verify their correctness?

Example System

Consider an autonomous aircraft as an example CPS. It operates data with different classes of security, from normal to top secret (ThSecCl). Periodic threads (T) execute on several processors (O, A). The aircraft is powered using multi-cell reconfigurable batteries (B). The system's architecture shown below is specified in AADL.

A battery has a matrix of cells, and each cell has a current and voltage output requirement.

Thermally, each cell exchanges heat with its neighboring cells (thermal neighbor, TN) through an electrical connector, affecting the risk of a thermal runaway.

Example Analyses

Scheduling verification domain σ_sched

**Verification Domains**

A verification domain $\sigma = (\mathcal{A}, \mathcal{S}, \mathcal{R}, \mathcal{T}, \Pi)$ formalizes domain-specific constructs for several related analyses.

- $\mathcal{A}$ — a set of sorts, comprised of system elements and standard sorts. E.g., integers $\mathbb{Z}$, threads $T$, or scheduling policies $\text{SchedPol}$.
- $\mathcal{S}$ — a set of static functions that encode design-time properties. E.g., thread period $\text{Per}$, thread-to-CPU binding $\text{CPUBind}$, and system-wide Voltage.
- $\mathcal{R}$ — a set of runtime functions that encode dynamic properties. E.g., preemption relation $\text{canPrmt}(t_1, t_2)$ and number of cells in a battery $b$ with $i$ thermal neighbors $\text{TN}(b, i)$.
- $\mathcal{T} = \{\text{execution semantics of } \sigma \text{ — a set of sequences of assignments to } \mathcal{R}\}. \text{We use Promela programs to implement the semantics.}$

$\Pi = \{\mathcal{A}, \mathcal{S}, \mathcal{R}, \mathcal{T}\}$, E.g., $\{\text{SchedPol} = \{\text{RMS, DMS, EDF}\}$.

Formally, an AADL architectural model $m$ is an interpretation $\Pi$ of $\mathcal{A}, \mathcal{S},$ and $\mathcal{T}$.

- $\Pi = \{\mathcal{A}, \mathcal{S}, \mathcal{R}, \mathcal{T}\}$, E.g., $\{\text{SchedPol} = \{\text{RMS, DMS, EDF}\}$.

$\Pi = \{\mathcal{A}, \mathcal{S}, \mathcal{R}, \mathcal{T}\}$, E.g., $\{\text{SchedPol} = \{\text{RMS, DMS, EDF}\}$.

Let $\Pi$ and $\Pi_0$ form a full interpretation of $\mathcal{A}, \mathcal{S},$ and $\mathcal{T}$.

**Analysis Contracts**

Each analysis assigns a contract to a tuple $(\Pi, O, A, G)$. Records:

- Inputs $I \subseteq \mathcal{A} \cup \mathcal{S}$ declare elements that the analysis reads.
- Outputs $O \subseteq \mathcal{A} \cup \mathcal{S}$ declare elements that the analysis writes.
- Assumptions $\mathcal{A} \subseteq \mathcal{F}_a$ are logical statements that must be satisfied by every input model to the analysis: $m \models A$.
- Guarantees $\mathcal{G} \subseteq \mathcal{F}_a$ are logical statements that must be satisfied by every output model of the analysis: $m \models G$.

Assumption and guarantee formulas have the following syntax:

$\mathcal{F}_a : = \{\forall \phi \cdot \exists \psi \cdot \phi \lor \exists \phi \cdot \exists \psi \cdot \phi \land \psi\}$

where $\phi$ is a predicate logic formula over $\mathcal{A} \cup \mathcal{S}$, $\psi$ is an LTL formula over $\mathcal{A} \cup \mathcal{S} \cup \mathcal{T}$.

**Analysis Ordering**

Correct execution of analyses requires satisfaction of all input-output dependencies for each analysis. Formally, contract $G$ depends on contract $C_j$ if $G \cap C_j \cap \sigma \neq \emptyset$.

Am ordering $\langle C, \ldots, C_n \rangle$ of contracts is sound if and only if predecessors are not dependent on successors:

$\forall i \in [1, n] \cdot \forall j \in [1, i] \cdot C_j \cap C_i \cap \sigma \neq \emptyset$.

Consider a graph with vertices being contracts and edges being contract dependencies. There exists a sound ordering of contracts if and only if the graph is not cyclic. If it is not cyclic, any topological ordering is sound.

**Contract Verification**

The goal of contract verification is to decide $m \models F_a$.

For purely first-order formulas that contain only $\phi$, we decide satisfiability via SMT solving. An SMT program is generated based on $\mathcal{A}$ and $\mathcal{S}$ mentioned in $\phi$, and an SMT solver is invoked on $\phi$ (or $\phi$ for existential quantification). A universally (existentially) quantified contract is satisfied if and only if UNSAT (SAT) is returned.

For formulas combining predicate formula $\phi$ and LTL formula $\psi$, we first generate an SMT program for $\phi$ and find all valuations of $v_{\phi, \psi}$ that satisfy $\phi$. For each such valuation we call Spin on a Promela program that implements $T$ for $m$ in the domain of $\phi$. Formula $\psi$ is transformed into an LTL property specification in Promela. A universally (existentially) quantified contract is satisfied if and only if the LTL property holds for all (at least one) valuations. The architecture of our verification tool is shown below.

**Experimental Results**

Efficiency: We have been able to detect analysis integration errors and verify their absence for each analysis in the example.

*Scalability: the results of scalability experiments with our implementations of $\mathcal{T}$ are shown in the tables below.*

<table>
<thead>
<tr>
<th>Threads</th>
<th>$T_{\text{Sch}}$</th>
<th>$T_{\text{LTV}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0.01</td>
<td>0.33</td>
</tr>
<tr>
<td>4</td>
<td>0.13</td>
<td>0.75</td>
</tr>
<tr>
<td>5</td>
<td>0.81</td>
<td>2.35</td>
</tr>
<tr>
<td>6</td>
<td>1.66</td>
<td>1.37</td>
</tr>
<tr>
<td>7</td>
<td>3.33</td>
<td>2.18</td>
</tr>
<tr>
<td>8</td>
<td>6.66</td>
<td>3.42</td>
</tr>
<tr>
<td>9</td>
<td>9.99</td>
<td>6.85</td>
</tr>
<tr>
<td>10</td>
<td>22.22</td>
<td>12.43</td>
</tr>
<tr>
<td>11</td>
<td>33.33</td>
<td>16.66</td>
</tr>
</tbody>
</table>

Copyright © 2014 ACM. Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice and copyright notice on all copies. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.