Early Time-Budgeting in Distributed Embedded Control Systems

Manoj G. Dixit,

Jointly with,

Prof. Pallab Dasgupta, IIT KGP and Dr. S. Ramesh, GM R&D
Background

- Modern embedded control systems are component based and have large number of features
  - E.g. AUTOSAR based development

- Safety features have strict real-time end-to-end requirements

- Many components interact together to meet system level requirements

- System is distributed in nature

**How to do a timing layout of entire system to meet end-to-end real-time requirements?**
Illustrative Example

Functional Specification

In a crash scenario, the airbags blow up to enable passenger safety

Crash Scenario → Airbags blown

Crash Detect → Disable Power Steering → Collapse Steering → Blow Airbags

Safety Requirement

In a crash scenario, the airbags must blow up within 20ms to enable passenger safety

Crash Scenario → Airbags blown

< 20 ms

Crash Detect → Disable Power Steering → Collapse Steering → Blow Airbags
**Illustrative Example**

In a crash scenario, the airbags blow up to enable passenger safety.

- Crash Scenario → Airbags blown
- Crash Detect → Disable Power Steering → Collapse Steering → Blow Airbags

**Functional Specification**

**Standards/Statistical data**

**Safety Requirement**

**Actual Implementation**

**Time Budgeting**

In a crash scenario, the airbags *must* blow up within 20ms to enable passenger safety.

- Crash Scenario
- < 20 ms
- Airbags blown
- Crash Detect → Disable Power Steering → Collapse Steering → Blow Airbags
Emerging Challenges

- Increasing complex features
- Multiple functions in a single computational unit, e.g. AUTOSAR
- More component sharing promoted by the smaller component sizes
- Need for advance planning of resources for extensibility

This is leading to...

- Increasing real-time interdependencies between components
Prevalent Approaches

- Ad-hoc estimates about component response time
- Architecture exploration to do component-task mapping, component-ECU mapping etc.
- On failure, difficult to trace the culprit component

How to budget time for each component is not clear.
Our Proposal

- Early time-budgeting for embedded control-systems
- Component have parametric timing requirements
- Use formal specification and analysis methodology to generate constraints on parameter valuations
Example

In a crash scenario, the airbags **must** blow up **within 20ms** to enable passenger safety.

- **Crash Scenario** → **< 20 ms** → **Airbags blown**
- **Crash Detect** → **Disable Power Steering** → **Collapse Steering** → **Blow Airbags**

**Real-time Specifications**

**Parametric -time Specifications**

**Time-Budgeting**: What values of x, y, w, z are good-enough?
Industrial relevance

- Shift towards early specification of timing requirements
  - Large component integration, multiple suppliers
  - Important to know, how a specific choice of timing specification for one component affects the other

- AUTOSAR meta model allows specifying timing specifications at different levels of software hierarchy – components to network
  - EAST-ADL and TIMMO2 provides higher level of abstractions for specifying functional and product line requirements
  - Timing requirements are refined across different levels
    - Event models – periodic, sporadic etc.
    - Delay, synchronization constraints

Source: Autosar timing spec from http://www.autosar.org

Source: TIMMO Methodology presentation by Stephan Kuntz, Continental Automotive GmbH, 2010
The Problem

- We are given a set of features and their real-time requirements

- We are given a set of components and their parametric-time requirements for implementing these features

- Propose Early stage Time-Budgeting Methodology
  - Find constraints over parameter values
  - Design space exploration to select suitable valuation
  - Scalable
Scalability and Usability Considerations -1

- In practice, component decompositions are hierarchical: DAG
  - Methodology aligned for hierarchical specifications

- Large decompositions:
  - Each feature component has 10s of requirements
  - Simultaneous budgeting does not seem to be scalable
  - Linear constraints are preferred

- Requirements become finer and more complex (or detailed) as we move down the hierarchy

- Handling large hierarchical decompositions:
  - Split component time-budgeting into smaller sub-problems and repeat
  - Compositional approach
  - DF-traversal with back-tracking takes care of component re-use case

- Specialized methods to analyze requirements patterns
Time-Budgeting Single Step

- New Algorithms
- Constraint Solving, Optimization: Well known

- Novelty: System level optimization gets converted to constraint solving, scalability is much better this way

Feature Requirements (known-timing)
Component Requirements (unknown-timing)
Formalized Feature Requirements (Real-time)
Formalized Component Requirements (Parametric-time)
Design Constraints
Optimization Decisions

Formal Analyzer of Parametric Specs
Linear Constraints On Parameters
Constraint Optimizer

Component Time-Budget

CMI Workshop: Making Formal Verification Scalable and Usable
Manoj Dixit 10th Jan 2013
Formalization of Requirement Decomposition

- Requirements decomposition step is formalized as a collection of *requirement decomposition pairs*

- $f$: feature requirement and let $g_1, ..., g_k$ are component requirements identified for $f$

- $(f, \{g_1, ..., g_k\})$ is a requirement decomposition pair

- Verification check:
  - Informally: component requirements put-together should satisfy all feature requirements
  - We have the following reduction:

**Theorem:**
*It is enough to analyze each pair separately, compute validity constraint. Any solution to conjoined constraint defines a suitable time-budget*
### Parametric Temporal Logic (PLTL)

- Extends well known Linear Temporal Logic
- Semantics is defined by using a parameter valuation

<table>
<thead>
<tr>
<th>Feature/Component Name</th>
<th>PLTL Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC Feature</td>
<td>$\phi_1 : \square(\text{lead_slow} \Rightarrow \Diamond_{\leq 500} \text{apply_brake})$</td>
</tr>
<tr>
<td>Sensor Component</td>
<td>$\psi_1 : \square(\text{lead_slow} \Rightarrow \Diamond_{\leq x_1} \text{lead_kinematics_info})$</td>
</tr>
<tr>
<td>ACC Controller Component</td>
<td>$\psi_2 : \square(\text{lead_kinematics_info} \Rightarrow \Diamond_{\leq x_2} \text{apply_brake})$</td>
</tr>
</tbody>
</table>

$(\phi_1, \{\psi_1, \psi_2\})$ is a Requirement Decomposition Pair
Validity of a Requirement Decomposition Pair

<table>
<thead>
<tr>
<th>Feature/Component Name</th>
<th>PLTL Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC Feature</td>
<td>$\phi_1 : \square (lead_slow \Rightarrow \Diamond \leq_{500} apply_brake)$</td>
</tr>
<tr>
<td>Sensor Component</td>
<td>$\psi_1 : \square (lead_slow \Rightarrow \Diamond \leq_{x_1} lead_kinematics_info)$</td>
</tr>
<tr>
<td>ACC Controller Component</td>
<td>$\psi_2 : \square (lead_kinematics_info \Rightarrow \Diamond \leq_{x_2} apply_brake)$</td>
</tr>
</tbody>
</table>

The requirement decomposition pair is valid if and only if PLTL formula $\psi_1 \land \psi_2 \Rightarrow \phi_1$ is valid.

Due to the parameters, this reduces to constraint computation.
Abstractly...

Given a PLTL formula $\phi$, we want to find the representation of the solution region in the form of a constraint.

<table>
<thead>
<tr>
<th>Feature/Component Name</th>
<th>PLTL Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC Feature</td>
<td>$\phi_1 : \Box(\text{lead_slow} \Rightarrow \Diamond_{\leq 500} \text{apply_brake})$</td>
</tr>
<tr>
<td>Sensor Component</td>
<td>$\psi_1 : \Box(\text{lead_slow} \Rightarrow \Diamond_{\leq x_1} \text{lead_kinematics_info})$</td>
</tr>
<tr>
<td>ACC Controller Component</td>
<td>$\psi_2 : \Box(\text{lead_kinematics_info} \Rightarrow \Diamond_{\leq x_2} \text{apply_brake})$</td>
</tr>
</tbody>
</table>
Scalability and Usability Considerations - 2

- **Scalable decision procedures for PLTL**
  - Emptiness, universality conditions for a formula
  - Closed form representation of validity region possibly using linear constraints
  - Constraint computation involves dealing with large search space to compute boundary

- **Pattern specific scalable constraint computation techniques**
  - Requirement decomposition pairs modeled using bounded-response pattern
  - Suitable for specifying end-to-end response

- **Developing modeling guidelines for enhancing the usability**
  - Model requirements at top level of hierarchy using bounded-response pattern
  - At lower level have more complex pattern
  - Whenever scalability issues are encountered in constraint computation, perform bounded-response based decomposition and then refine

Bounded-response pattern

Model top requirements using bounded-response and then refine
Emptiness, Universality and Finiteness of Validity Region

+ 

Unsuitability of Linear Predicates for Representing Validity Region
# Emptiness, Universality and Finiteness Problems

<table>
<thead>
<tr>
<th>Formula $\Phi$</th>
<th>Emptiness</th>
<th>Universality</th>
<th>Finiteness</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$F$</td>
<td>$S_\Phi$</td>
<td>$V_\Phi$</td>
</tr>
<tr>
<td>PLTL $\square$</td>
<td>${\Phi(0_0)}$</td>
<td>$\leftrightarrow$, sat</td>
<td>$\rightarrow$, val</td>
</tr>
<tr>
<td>PLTL $\diamond$</td>
<td>${\widetilde{\Phi}}$</td>
<td>$\leftrightarrow$, sat</td>
<td>$\rightarrow$, val</td>
</tr>
<tr>
<td>PLTL</td>
<td>${\widetilde{\Phi(0_Y)}}$</td>
<td>$\leftrightarrow$, sat</td>
<td>$\rightarrow$, val</td>
</tr>
</tbody>
</table>

- $\alpha_0$: Parameter valuation assigning 0 to all parameters.
- $0_X$: Partial parameter valuation assigning 0 to all members of $X$.
- $0_Y$: Partial parameter valuation assigning 0 to all members of $Y$.
- $0_Y \setminus \{y\}$: Partial parameter valuation assigning 0 to all members of $Y \setminus \{y\}$.
- $\leftrightarrow$: Solution provides both necessary and sufficient condition.
- $\Rightarrow$: Solution provides only sufficient condition.
- sat: Satisfiability check required of a member formula.
- val: Validity check required of a member formula.

- **Defined** parameter abstraction operation for PLTL

\[
\Phi = \square_{\leq y} p_1 \land \square_{\leq 10} p_2
\]

\[
\widetilde{\Phi} = \square p_1 \land \square_{\leq 10} p_2
\]

- **Improved Complexity**

- Our complexity: $O(\frac{c_\Phi}{2^{n_\Phi}})$. Earlier: $O(\frac{k_\Phi}{c_\Phi} + 1) 2^{n_\Phi} (k_\Phi + 1)$
Unsuitability of Linear predicates

- Negative result

- Earlier known for a wider class of PLTL formulae

- We have further restricted this to a subclass of PLTL

- However.. all is not lost... many nice properties still fall in decidable fragment
Bounded-Response Constraint Extraction Method
A Scalable Method for a widely used Requirement Pattern

- At higher levels most of the requirements are based on a specific pattern... \textit{bounded-response}

- $\phi$: Boolean formula

- $\psi$: Boolean formula

- $x$: Parameter or constant

- We consider validity checks of requirement decomposition pairs using this pattern
**Constraint Extraction Method**

- Reasoning over temporal formulae reduced to Boolean reasoning... hence scalable

- And-or tree constructed from formulae
  - We have defined a notion of an irreducible cover for Boolean formulae

- Assign path constraints

- Final constraint: conjunction/disjunction of path constraints

---

Let $\gamma$ and $\gamma_1, \cdots, \gamma_n$ be Boolean formulae. An irreducible cover of $\gamma$ is a minimal subset $H$ of $\{\gamma_1, \cdots, \gamma_n\}$ such that $\gamma \Rightarrow \bigvee_{\gamma' \in H} \gamma'$ is valid.
Example: Bounded-Response Tree

\[ \Phi = \Box (a_1 \lor a_2 \Rightarrow \Diamond \leq_{10} b_1 \lor b_2) \]

\[ \begin{align*}
\psi_1 &= \Box (a_1 \Rightarrow \Diamond \leq_{x_1} c_1 \lor c_2 \lor b_2) \\
\psi_2 &= \Box (a_2 \Rightarrow \Diamond \leq_{x_2} c_1) \\
\psi_3 &= \Box (c_1 \Rightarrow \Diamond \leq_{x_3} b_1) \\
\psi_4 &= \Box (c_2 \Rightarrow \Diamond \leq_{x_4} b_1) \\
\psi_5 &= \Box (c_2 \Rightarrow \Diamond \leq_{x_5} b_2) \\
\psi_6 &= \Box (c_2 \Rightarrow \Diamond \leq_{x_6} c_4) \\
\psi_7 &= \Box (a_1 \land a_2 \Rightarrow \Diamond \leq_{x_7} c_3)
\end{align*} \]

\[ \{ [x_1 + x_3 \leq 10] \land (x_1 + x_4 \leq 10) \} \lor \{ [(x_1 + x_3 \leq 10) \land (x_1 + x_5 \leq 10)] \lor [(x_1 + x_3 \leq 10) \land \text{false}] \} \land (x_2 + x_3 \leq 10) \]
Corner Point Constraint Extraction Method
A General Constraint Extraction Method for PLTL

- Suitable for complex temporal properties
- We focus on PLTL fragments and their geometric properties
- PLTL Global fragment is downward-closed
- Downward Closed Region have finite number of corner-points
  - Include Point-at-infinity
  - Constraint definition using Corner-Points
**Algorithm Overview**

- **Prune and Search Approach**
  - Find a corner point
  - Partition the further search

- **Search Step**
  - Obtain a farthest useful valuation along diagonal starting from a base
  - Decide sub-set of parameters for which max limit is reached
  - Fix them and re-iterate till all parameters are over

- **Prune Step**
  - Partition and identify the region(s) where no corner-point can lie
  - Adjust new base valuation so that those regions get ignored from later search
  - Repeat Search step recursively for them

---

Max limit for $x_1$ reached

Iteration 1

Iteration 2

New base
Demonstration of the Methodology
Integrated Time-Budgeting Methodology

- NuSMV for LTL checks
- Yices for constraint solving
- Eclipse, Java
Case Studies

- Adaptive Cruise Control, Collision Mitigation
- 120+ feature and component properties
- 100+ add-on constraints in the design space exploration
- Budgeting for 3 feature combinations: ACC only, CM only and ACC-CM
Some Results

<table>
<thead>
<tr>
<th>Component</th>
<th>Parameter</th>
<th>Only ACC</th>
<th>Only CM</th>
<th>ACC-CM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brake Controller Component</td>
<td>$w_1$</td>
<td>200</td>
<td>160</td>
<td>160</td>
</tr>
<tr>
<td></td>
<td>$w_2$</td>
<td>70</td>
<td>-</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td>$w_3$</td>
<td>100</td>
<td>-</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>$w_4$</td>
<td>100</td>
<td>-</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>$w_5$</td>
<td>10</td>
<td>-</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>$w_6$</td>
<td>150</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Electronic Brake Control</td>
<td>$w_{11}$</td>
<td>35</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>$w_{12}$</td>
<td>100</td>
<td>80</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td>$w_{13}$</td>
<td>30</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>$w_{14}$</td>
<td>10</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>$w_{15}$</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>$w_{16}$</td>
<td>60</td>
<td>-</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td>$w_{17}$</td>
<td>60</td>
<td>-</td>
<td>60</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Decomposition Type</th>
<th>Num Components</th>
<th>RDP Nos</th>
<th>Num. SAT Checks (Average)</th>
<th>Constraint Computation Time (Average) (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Level</td>
<td>12</td>
<td>15</td>
<td>314 (LTL)</td>
<td>1280</td>
</tr>
<tr>
<td>Multi Level</td>
<td>22</td>
<td>11 (Bounded-Resp), 20 (General)</td>
<td>83 (Boolean SAT), 57 (LTL)</td>
<td>2.36 (Boolean SAT), 19.25 (LTL)</td>
</tr>
</tbody>
</table>
Many Challenges still remain ...

- Validity checking of parameter-free PLTL formulae
  - Presence of large constants lead to scalability of model checkers

- More scalable decidability algorithms for PLTL

- Seamless integration with architecture exploration phase to align with existing development flow
Summary

- **A Hierarchical Time-Budgeting Methodology**
  - Integrates all of the below techniques

- **Emptiness, Universality and Finiteness Problems for PLTL**
  - Non-triviality of the solution region

- **Bounded-Response Constraint Extraction Method**
  - A specially tuned method for a widely used requirements pattern

- **Corner Point Constraint Extraction Method**
  - Complex temporal relationships

- **Case Studies**
  - Tool framework and demonstration on automotive features ACC and CM
Publications


References


Thank You
The relevance

- AUTOSAR defines a meta-model for specifying component based distributed systems in automotive domain
- Specifications for components, middleware and higher level properties

Source: Autosar timing and VFBspecs from http://www.autosar.org