Specification, Design and Verification of Distributed Embedded Systems

Mani Chandy    John Doyle    Richard Murray (PI)
California Institute of Technology

Eric Klavins    Pablo Parrilo
U. Washington    MIT

Dynamics and Control Program Review
12 August 2010
### Specification, Design and Verification of Distributed Embedded Systems

<table>
<thead>
<tr>
<th>1. REPORT DATE</th>
<th>2. REPORT TYPE</th>
<th>3. DATES COVERED</th>
</tr>
</thead>
<tbody>
<tr>
<td>12 AUG 2010</td>
<td></td>
<td>00-00-2010 to 00-00-2010</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4. TITLE AND SUBTITLE</th>
<th>5a. CONTRACT NUMBER</th>
<th>5b. GRANT NUMBER</th>
<th>5c. PROGRAM ELEMENT NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specification, Design and Verification of Distributed Embedded Systems</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>6. AUTHOR(S)</th>
<th>7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)</th>
<th>8. PERFORMING ORGANIZATION REPORT NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>California Institute of Technology, Pasadena, CA, 91125</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)</th>
<th>10. SPONSOR/MONITOR’S ACRONYM(S)</th>
<th>11. SPONSOR/MONITOR’S REPORT NUMBER(S)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>12. DISTRIBUTION/AVAILABILITY STATEMENT</th>
<th>13. SUPPLEMENTARY NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approved for public release; distribution unlimited</td>
<td>presented at 2010 AFOSR Dynamics and Control program meeting</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15. SUBJECT TERMS</th>
<th>16. SECURITY CLASSIFICATION OF:</th>
<th>17. LIMITATION OF ABSTRACT</th>
<th>18. NUMBER OF PAGES</th>
<th>19a. NAME OF RESPONSIBLE PERSON</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>a. REPORT</td>
<td>b. ABSTRACT</td>
<td>c. THIS PAGE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>unclassified</td>
<td>unclassified</td>
<td>unclassified</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>18. NUMBER OF PAGES</th>
<th>19a. NAME OF RESPONSIBLE PERSON</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td></td>
</tr>
</tbody>
</table>

**Standard Form 298 (Rev. 8-98)**
Prescribed by ANSI Std Z9-18
V&V MURI Team

Principal Investigators
• Mani Chandy (Caltech CS)
• John Doyle (Caltech CDS)
• Gerard Holzmann (JPL CS)*
• Eric Klavins (U. Washington, EE/CS)
• Richard Murray (Caltech CDS)
• Pablo Parrilo (MIT EE)

Partners
• Air Force Research Laboratory: (IF), MN, VA, VS
• Boeing Corporation - Systems of Systems Integration
• Honeywell Corporation - Guidance and Control [Glavaski -> Easton -> UTRC]
• Jet Propulsion Laboratory (JPL) - Laboratory for Reliable Software (LARS)
• Julia Braman -> NASA Johnson Space Center
Problem Scope

Overall Goal:
Develop methods and tools for designing control policies, specifying the properties of the resulting distributed embedded system and the physical environment, and proving that the specifications are met.

Specification
- How does the user specify---in a single formalism---continuous and discrete control policies, communications protocols and environment models (including faults)?

Design and reasoning
- How can engineers reason that their designs satisfy the specifications?
- In particular, can engineers reason about the performance of computations and communication, and incorporate real-time constraints, dynamics, and uncertainty into that reasoning?

Implementation and verification
- What are the best ways of mapping detailed designs to hardware artifacts, running on specific operating systems? What languages are suitable for specifying systems so that the specifications can be verified more easily?
(Some) Accomplishments and Lessons to Date

Lyapunov (-like) functions continue to be a powerful tool
- Allows us to reason about entire sets of continuous variables
  - system properties $\rightarrow$ algebraic conditions
- Can also capture problems in discrete transition systems
  - lexicographically-ordered Lyapunov functions for graph grammars
- Powerful new tools (based on SOS) are making reasoning easier
  - non-monotonic Lyapunov functions, ROA estimates, ...

Use temporal logic for specification at higher levels of abstraction
- Allows descriptions of proper behavior on execution sequences
- Model checking/theorem proving provide tools for verifying behavior
  - PVS, SPIN, TLC, SBT Checker/Inveriant, TLV, ...
- “LTL should be part of every control engineer’s knowledge basis”

Asynchronous behavior via guarded command languages
- Guarded command languages allow good description of distributed operation with no globally synchronized clock
- Can reason about asynchronous behavior using LTL formalisms
- CCL with rates to describe stochastic, multi-rate systems
Example: Verification for Autonomous Systems

How do we design control protocols that manage behavior

• Mixture of discrete and continuous decision making
• Insure proper response external events, with unknown timing
• Design input = specification + model (system + environment)
• Design output = finite state machine implementing logic

Approach: rapidly explore all trajectories satisfying specs

• Search through all possible actions and events, discarding executions that violate a set of (LTL) specifications
• Issue: state space explosion (especially due to environment)
• Good news: recent results in model checking for class of specs
**Receding Horizon Control for Linear Temporal Logic**

Find planner (logic + path) to solve general control problem

\[(\varphi_{init} \land \square \varphi_e) \Longrightarrow (\square \varphi_s \land \diamond \varphi_g)\]

- \(\varphi_{init}\) = init conditions
- \(\varphi_e\) = envt description
- \(\varphi_s\) = safety property
- \(\varphi_g\) = planning goal

Can find automaton to satisfy this formula in \(O((nm|\Sigma|^3)\) time (!)

**Basic idea**

- Discretize state space into regions \(\{V_i\}\) + interconnection graph
- Organize regions into a partially ordered set \(\{\mathcal{W}_i\}\); \(\mathcal{W}_j \preceq_{\varphi_g} \mathcal{W}_i\) ⇒ if state starts in \(\mathcal{W}_i\), must transition through \(\mathcal{W}_j\) on way to goal
- Find a finite state automaton \(A_i\) satisfying
  \[\Psi_i = ((v \in \mathcal{W}_i) \land \Phi \land \square \varphi_e) \Longrightarrow (\square \varphi_s \land \diamond (v \in \mathcal{W}_{g_i}) \land \square \Phi)\]
  - \(\Phi\) describes receding horizon invariants (eg, no collisions)
  - Automaton states describe sequence of regions we transition through; \(\mathcal{W}_{g_i} \preceq_{\varphi_g} \mathcal{W}_i\) is intermediate (fixed horizon) goal
  - Planner generates trajectory for each discrete transition
  - Partial order condition guarantees that we move closer to goal

**Properties**

- Provably correct behavior according to spec
Comments and Example

Comments and caveats

- Automaton synthesis is basically searching thru all feasible trajectories (efficiently)
- Complexity is polynomial, but can still get large → receding horizon is a huge help!
- Discretization of the state space is important and non-trivial

Example: driving down a lane with unknown obstacles

- Demonstrates basic feasibility of approach
- Lots of tuning required to get everything to work
- Clever discretization + RHC are key enablers...

\[
\begin{align*}
\dot{z}_{t+1} &= 0.0952 z_t + 0.9048 v_{t+1} \\
\dot{v}_{t+1} &= -0.0048 z_t + 0.0952 v_t + 0.0048
\end{align*}
\]

- Demonstrates basic feasibility of approach
- Lots of tuning required to get everything to work
- Clever discretization + RHC are key enablers...
Transition Paths and Next Steps

Transition activities

- Hands-on workshop: DoD, NASA, industry
- DARPA/industry Multi-Scale Systems Center
  - Working with UTC and Raytheon
  - Python-based toolkit for RHTLP

Next steps

- Optimization-based methods
  - How do we include cost in solutions?
- Structured synchronization
  - Typically assume very little structure in asynchronous processes => hard to verify
  - How do we allow some synchronized behavior, but not completely sync’d?
- Distributed synthesis
  - How do we design provably correct, distributed planners?
Specification, Design and Verification of Distributed Embedded Systems
Caltech/MIT/UW, Murray (PI)/Chandy/Doyle/Klavins/Parrilo

**Long-Term PAYOFF:** Rigorous methods for design and verification of distributed systems-of-systems in dynamic, uncertain, adversarial environments

**OBJECTIVES**
- Specification language for continuous & discrete control policies, communications protocols and environment models (including faults)
- Analysis tools to reason about designs and provide proof certificates for correct operation
- Implementation on representative testbeds

**ACCOMPILMENTS/RESULTS**
- Foundations of local/global properties of computation
- Embedded graph grammars for cooperative control
- Lyapunov-based verification of temporal properties
- Receding horizon temporal logic planning
- New formulations of game theory/stochastic problems

**FUNDING ($K)**—Show all funding contributing to this project
<table>
<thead>
<tr>
<th></th>
<th>FY06</th>
<th>FY07</th>
<th>FY08</th>
<th>FY09</th>
<th>FY10</th>
<th>FY11</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFOSR Funds</td>
<td>417</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>593</td>
</tr>
<tr>
<td>Boeing</td>
<td>310</td>
<td>390</td>
<td>390</td>
<td>370</td>
<td>390</td>
<td>[390]</td>
</tr>
<tr>
<td>DARPA GC</td>
<td>310</td>
<td>390</td>
<td>390</td>
<td>370</td>
<td>390</td>
<td>1200</td>
</tr>
</tbody>
</table>

**TRANSITIONS**
- Application to autonomous driving (DGC07)
- Tools inserted in MuSyC (DoD 6.2 + UTC, Raytheon)
- Software toolkits, workshops, and personnel transfer

**STUDENTS, POST-DOCS**
2006-09: 24 graduate students, 5 postdocs, 4 undergraduates

**LABORATORY POINT OF CONTACT**
Dr. Siva Banda, AFRL/RBCA, WPAFB, OH