Specification, Design and Verification of Distributed Embedded Systems

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

Eric Klavins
U. Washington

Pablo Parrilo
MIT

Dynamics and Control Program Review
12 August 2010

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

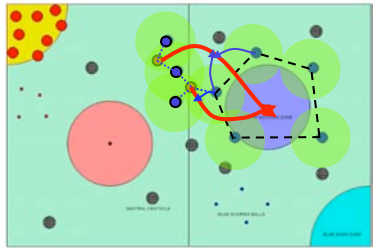
Problem Scope

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

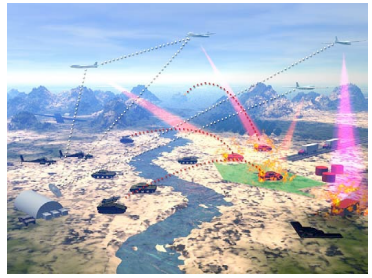
Specification

- 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?



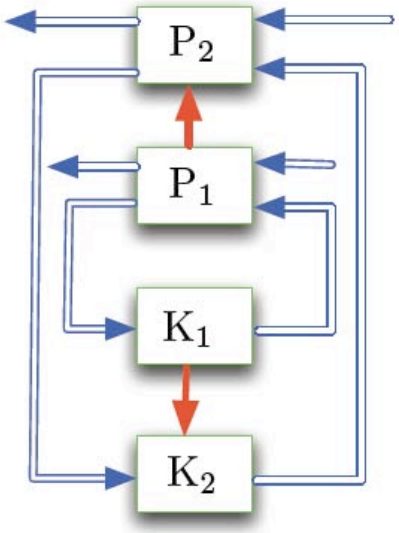
Implementation and verification

- 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

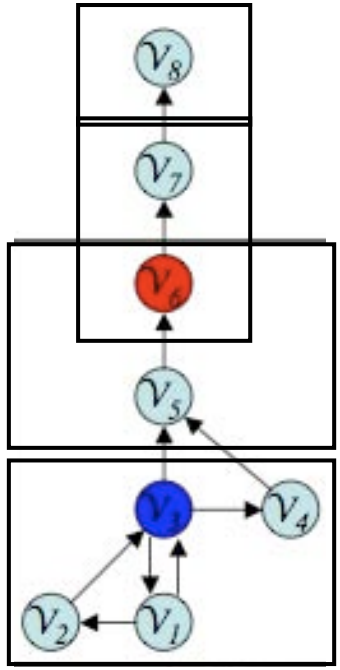
Lyapunov (-like) functions continue to be a powerful tool

- Allows us to reason about entire sets of continuous variables
  - system properties → algebraic conditions
- Can also capture problems in discrete transition systems
  - lexicographically-ordered Lyapunov fcns 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

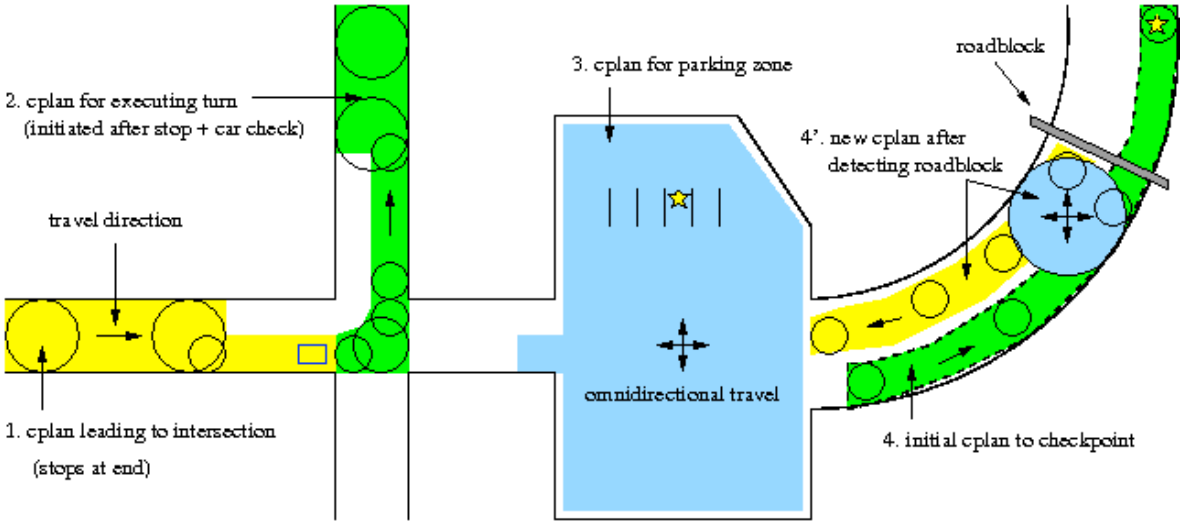
- Allows descriptions of proper behavior on execution *sequences*
- Model checking/theorem proving provide tools for verifying behavior
  - PVS, SPIN, TLC, SBT Checker/Inveriant, TLV, ...
- "LTL should be part of every control engineer's knowledge basis"

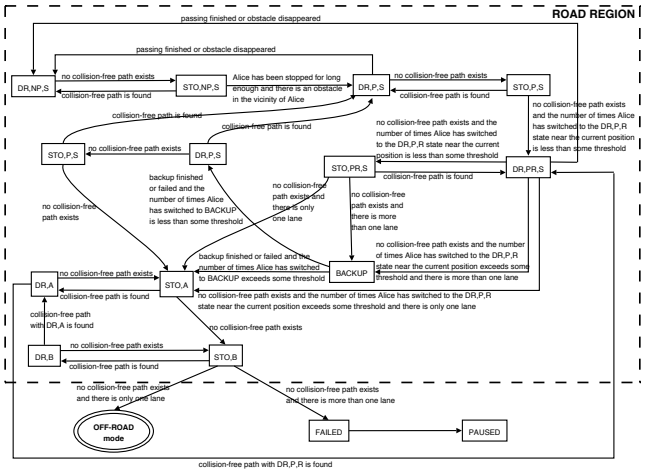


Asynchronous behavior via guarded command languages

- 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



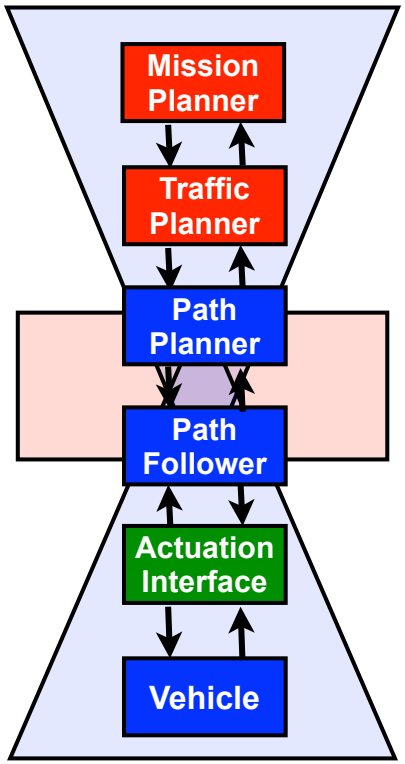


How do we *design* control protocols that manage behavior

- 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

$$(\varphi_{init} \wedge \Box\varphi_e) \Longrightarrow (\Box\varphi_s \wedge \Diamond\varphi_g)$$

- φ_{init} = init conditions
- φ_e = envt description
- φ_s = safety property
- φ_g = planning goal

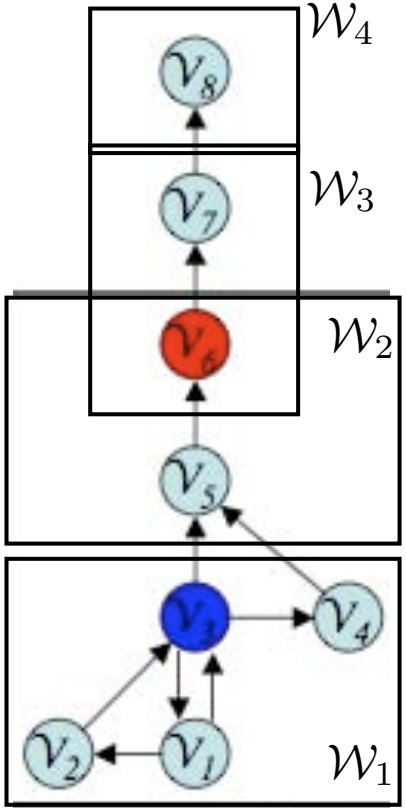
- Can find automaton to satisfy this formula in $O((nm|\Sigma|^3)$ time (!)

Basic idea

- Discretize state space into regions $\{\mathcal{V}_i\}$ + 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 $\mathcal{A}_i$ satisfying

$$\Psi_i = ((v \in \mathcal{W}_i) \wedge \Phi \wedge \Box\varphi_e) \Longrightarrow (\Box\varphi_s \wedge \Diamond(v \in \mathcal{W}_{g_i}) \wedge \Box\Phi)$$

  - Φ describes receding horizon invariants (eg, no collisions)
  - Automaton states describe sequence of regions we transition through; $\mathcal{W}_{g_i} \preceq_{\phi_g} \mathcal{W}_i$ is intermediate (fixed horizon) goal
  - Planner generates trajectory for each discrete transition
  - Partial order condition guarantees that we move closer to goal



Properties

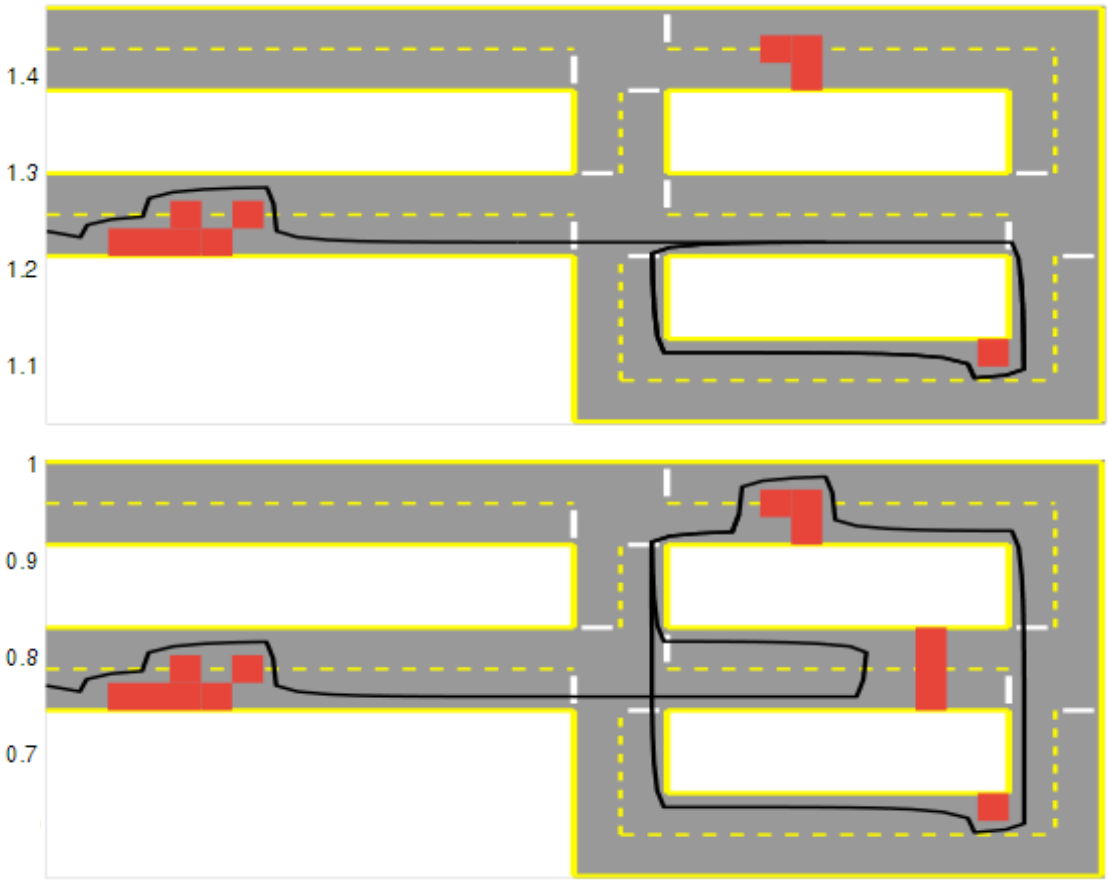
- Provably correct behavior according to spec

Comments and Example

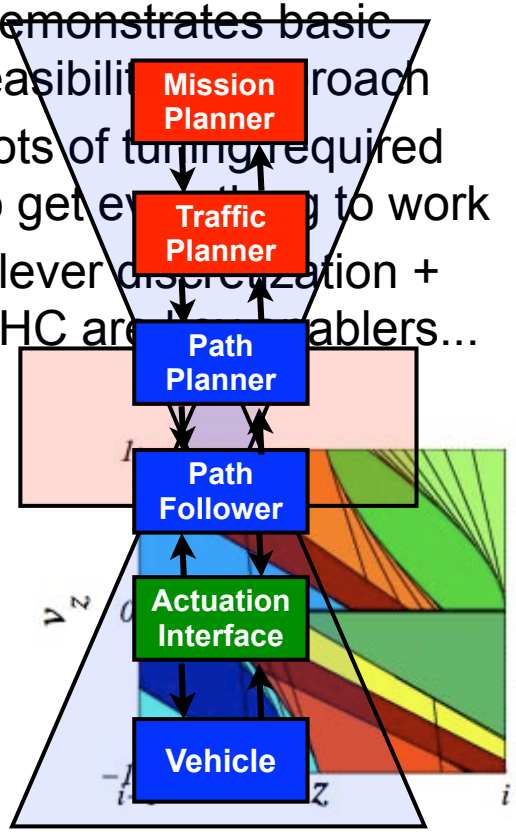
Comments and caveats

- 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...



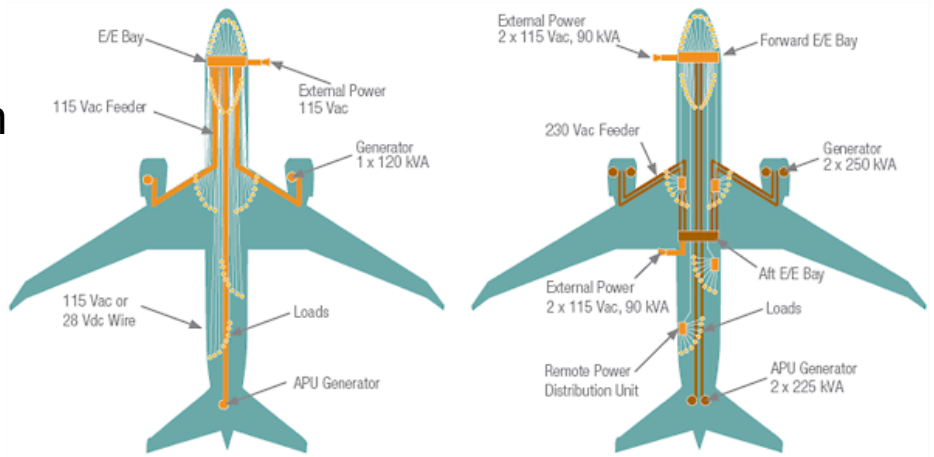
Transition Paths and Next Steps

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?

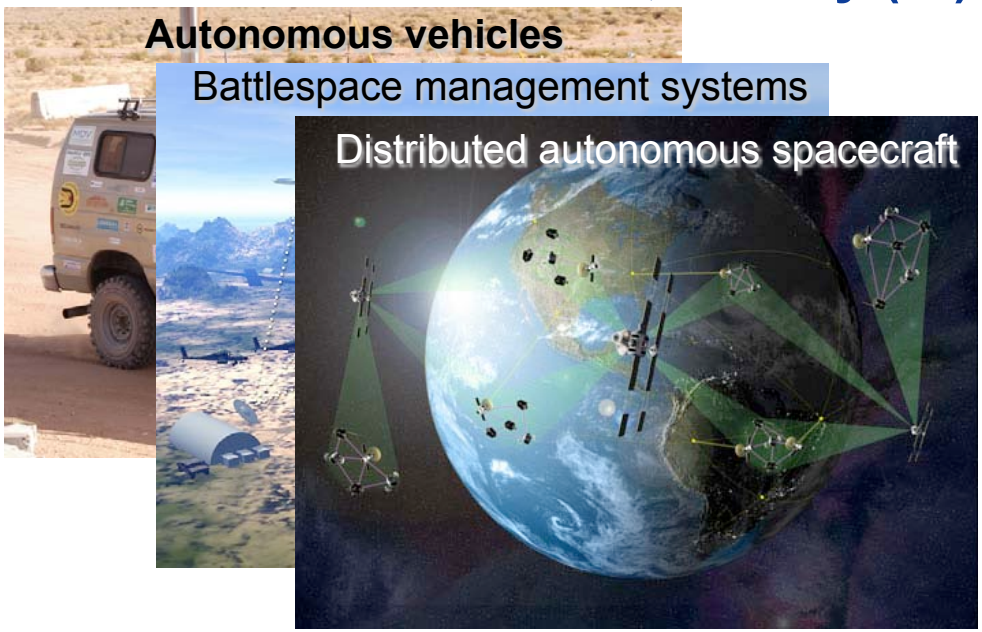


Centralized Distribution:
Circuit Breakers, Relays, and Contactors

Remote Distribution:
Solid-State Power Controllers and Contactors

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

APPROACH/TECHNICAL CHALLENGES

- Specification and reasoning using guarded command languages, temporal logic and graph grammars
- Sum of squares analysis for certificates, invariants
- Model checking/theorem proving for hybrid systems
- Extensions to probabilistic, adversarial and networked operations

ACCOMPLISHMENTS/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

	FY06	FY07	FY08	FY09	FY10	FY11
AFOSR Funds	417	1000	1000	1000	1000	593
Boeing	310	390	390	370	390	[390]
DARPA GC		1200				

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