Reactive Protocol Synthesis for
for Embedded Control Software

Richard M. Murray
Control & Dynamical Systems
California Institute of Technology

Necmiye Ozay (Michigan)  Ufuk Topcu (Penn)  Nok Wongpiromsarn
Scott Livingston  Stephanie Tsuei  Eric Wolff  Mumu Xu

EXCAPE Summer School
13 June 2013

Research support by AFOSR, Boeing, DARPA (FCRP), IBM and United Technologies Corp.
Long Term Research Objectives and Goals

Design of networked control systems

- Interconnected subsystems/agents working to achieve a common goal
- How do we describe complex temporal tasks in a manner that allows rigorous proof of correctness?

Correct-by-construction synthesis

- Given a specification, how do we *synthesize* a control protocol that satisfies the specs

Application areas

- Mission management & flight controls
- Unmanned, autonomous systems
- Safety critical, networked embedded systems (cars, planes, factories)
  - “Internet of things”
- Electrical power distribution networks
- (Cyber-security for embedded systems)
Hybrid, Multi-Component System Description

Subsystem/agent dynamics - continuous

\[
\dot{x}^i = f^i(x^i, \alpha^i, y^i, u^i) \quad x^i \in \mathbb{R}^n, u^i \in \mathbb{R}^m \\
y^i = h^i(x^i, \alpha^i) \quad y^i \in \mathbb{R}^q
\]

Agent mode (or “role”) - discrete

- \(\alpha \in A\) encodes internal state, relationship to current task, environment state
- Discrete transition \(\alpha' = r(x, \alpha)\)
- Environment: discrete transition system with temporal logic specs (eg, LTL)

Communications graph \(\mathcal{G}\)

- Encodes the system information flow
- Neighbor set \(\mathcal{N}^i(x, \alpha)\)

Communications channel

- Communicated information can be lost, delayed, reordered; rate constraints
  \[y^i_j[k] = \gamma y^i(t_k - \tau_j) \quad t_{k+1} - t_k > T_r\]
- \(\gamma\) = binary random process (packet loss)

Task

- Encode task as finite horizon optimal control + temporal logic (LTL/GR(1))
  \[J = \int_0^T L(x, \alpha, u) \, dt + V(x(T), \alpha(T)) \]
  \[(\varphi_{\text{init}} \land \square \varphi_e) \implies (\square \varphi_s \land \Diamond_{\leq T} \varphi_g)\]
- Assume/guarantee contract structure

Strategy = control law + control protocol

- Control action for individual agents
  \[u^i = \gamma(x, \alpha) \quad \{g^i_j(x, \alpha) : r^i_j(x, \alpha)\}\]
  \[\alpha'^i_c = \begin{cases} r^i_j(x, \alpha) & g^i_j(x, \alpha) = \text{true} \\ \text{unchanged} & \text{otherwise.} \end{cases}\]

Decentralized strategy

\[u^i(x, \alpha) = u^i(x^i, \alpha^i, y^{-i}, \alpha^{-i})\]
\[y^{-i} = \{y^j_1, \ldots, y^j_{m_i}\}\]
\[j_k \in \mathcal{N}^i \quad m_i = |\mathcal{N}^i|\]

- Similar structure for role update
Motivating Example: Alice (DGC07)

Alice

- 300+ miles of fully autonomous driving
- 8 cameras, 8 LADAR, 2 RADAR
- 12 Core 2 Duo CPUs + Quad Core
- ~75 person team over 18 months

Software

- 25 programs with ~200 exec threads
- 237,467 lines of executable code
**Planner Stack**

**Mission Planner** performs high level decision-making
- Graph search for best routes; replan if routes are blocked

**Traffic Planner** handles rules of the road
- Control execution of path following & planning (multi-point turns)
- Encode traffic rules - when can we change lanes, proceed thru intersection, etc

**Path Planner/Path Follower** generate trajectories and track them
- Optimized trajectory generation + PID control (w/ anti-windup)
- Substantial control logic to handle failures, command interface, etc

---

Burdick et al, 2007
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
Solve finite time optimization over $T$ seconds and implement first $\Delta T$ seconds

$$u_{[t,t+\Delta T]} = \arg\min \int_{t}^{t+\Delta T} L(x(\tau), u(\tau)) d\tau + V(x(t + T))$$

$$x_0 = x(t) \quad x_f = x_d(t + T)$$

$$\dot{x} = f(x,u) \quad g(x,u) \leq 0$$

Requires that computation time be small relative to time horizons
- Initial implementation in process control, where time scales are fairly slow
- Real-time trajectory generation enables implementation on faster systems
Receding Horizon Control for Linear Temporal Logic

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

\[(\varphi_{\text{init}} \land \Box \varphi_e) \implies (\Box \varphi_s \land \Diamond \varphi_g)\]

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

- For discrete system, 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 \(\{W_i\}\); \(W_j \preceq_{\varphi_g} W_i\)
  \(\Rightarrow\) if state starts in \(W_i\), must transition through \(W_j\) on way to goal
- Find a finite state automaton \(A_i\) satisfying
  \[\Psi_i = ((v \in W_i) \land \Phi \land \Box \varphi_e) \implies (\Box \varphi_s \land \Diamond (v \in W_{g_i}) \land \Box \Phi)\]
  - \(\Phi\) describes receding horizon invariants (eg, no collisions)
  - Automaton states describe sequence of regions we transition through; \(W_{g_i} \preceq_{\varphi_g} W_i\) 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
Constructing Finite State Abstraction

Find discretization of state space and represent connectivity using a graph

- Look for regions such that we can move from one region to another w/out leaving the union of two regions
- If we can start from any point in first region and get to some point in second, the regions are connected
- Approach: guess at initial regions, then refine based on solving for reachability sets

Can solve for feasible regions via finite-time trajectory generation problem

- Fix the horizon length N. Given \( \mathcal{S}_i \leadsto \mathcal{S}_j \), compute a sequence of control signals \( u[0], \ldots, u[N-1] \) such that for any \( s[0] \in \mathcal{S}_i \)

\[
\begin{align*}
s[t + 1] &= A s[t] + B u[t] + E d[t] \\
s[t] &\in \mathcal{S}_i, s[N] \in \mathcal{S}_j, u[t] \in U
\end{align*}
\]

- Assume \( U, D, \mathcal{S}_i, \mathcal{S}_j \) are polyhedral sets. The above equation can be written as

\[
L \begin{bmatrix} s[0] \\ u[0] \\ \vdots \\ u[N - 1] \end{bmatrix} \leq M - G \begin{bmatrix} d[0] \\ \vdots \\ d[N - 1] \end{bmatrix}
\]

- Set \( S_0 \) of all \( s[0] \) satisfying this gives refinement of initial guess
- Multi-Parametric Toolbox (MPT) solves this problem and returns the set \( S_0 \) as polyhedron (!)
**Example: Autonomous Navigation in Urban Environment**

**Traffic rules**
- No collisions with other vehicles
- Stay in the travel lane unless there is an obstacle blocking the lane
- Only proceed through an intersection when it is clear

**Assumptions**
- Obstacle may not block a road
- Obstacle is detected before the vehicle gets too close to it
- Limited sensing range
- Obstacle does not disappear when the vehicle is in its vicinity
- Obstacles may not span more than a certain number of consecutive cells in the middle of the road
- Each intersection is clear infinitely often
- Each of the cells marked by star and its adjacent cells are not occupied by an obstacle infinitely often
Example: Autonomous Navigation in Urban Environment

- JTLV returns 900 state FSA in about 1.5 seconds
- $\Phi = \text{start in proper lane if no obstacle present} \land \text{no collision}$

Use response mechanism to replan if no feasible solution exists
- Trajectory planner sees blockage and fails to find strategy satisfying specification
- Trajectory planner reports failure to goal generator
- Goal generator re-computes a (high level) path to the goal state
Temporal Logic Planning (TuLiP) toolbox
http://tulip-control.sourceforge.net

Python Toolbox

- GR(1), LTL specs
- Nonlin dynamics
- Supports discretization via MPT
- Control protocol designed using JTLV
- Receding horizon compatible

Applications of TuLiP in the last year

- Autonomous vehicles - traffic planner (intersections and roads, with other vehicles)
- Distributed camera networks - cooperating cameras to track people in region
- Electric power transfer - fault-tolerant control of generator + switches + loads
**Example: Electrical Power Management for Aircraft**

**Power management of three VMS subsystems**
- Flight control (actuation) - highest priority
- Active de-icing - elevation dependent demand
- Environmental control - slower timescale

**Specifications**
- Constraint on maximum total power
- Prioritization: actuation, de-icing, environment
- Safety: ice accumulation, altitude change
- Performance: desired altitude and environmental conditions
- External environment: wind gusts, outside temperature, generator health

![Diagram of power management system](image)
Open (Research) Issues

Optimality: “language-constrained, optimal trajectory generation”

\[ J = \int_0^T L(x, \alpha, u) \, dt + V(x(T), \alpha(T)), \]

Partial order computation and hierarchical structure

- How do we determine the partial order for RHTLP and link to “supervisory” levels?

Verification and synthesis with (hard) real-time constraints

- How do we incorporate time in our specifications, verification and synthesis tools?
- Note: time automata and timed temporal logic formulas available...

Contract-based design: automate search interfaces for distributed synthesis

- How do we decompose a larger problem into smaller pieces?
- Especially important for large scale projects with multiple teams/companies

Uncertainty and robustness

- How to specify uncertainty for transition systems, robustness for controllers, specs
- New methods for describing robustness by Tabuada et al: look at how much the specifications must be enlarged to capture new behaviors based on uncertainty

Many other directions: incremental, probabilistic, performance metrics, ...

- Identify problems where knowledge of dynamics, uncertainty and feedback matter
How do we design software-controlled systems of systems to insure safe operation across all operating conditions (including failures)?
Summary and Future Work

Specification, Design, Verification for Autonomous Vehicles

- Most of the actual design was ad hoc; with lots of testing
- Starting to develop tools for systematic design, verification

Synthesis techniques for LTL specifications using receding horizon planning

- Convert the specification into a design criterion
- Use fast solvers to find trajectories that satisfy constraints (including temporal logic specifications)
- Manage complexity using receding horizon approach
- Python toolbox: http://tulip-control.sf.net

Current work

- More systematic design of regions, lattices, invariants
- Better integration of trajectory planning and logic planning
- Application to (aircraft) vehicle management systems:
  - Many heterogeneous systems w/ coupled function, physics
  - Mixed criticality operations, depending on flight mode
  - Distributed resource allocation and decision-making, coupled by (loosely) asynch communications network