Goals for today

- Talk about the rest of the "autonomy stack"
- Introduce some concepts in supervisory (feedback) control

Reading

Control System “Standard Model”

Key elements
- Process: input/output system w/ dynamics
- Actuation: mechanism for manipulating process
- Sensing: mechanism for detecting process state
- Compute: compare actual / desired; determine action

Advantages of feedback
- Design of dynamics
- Robustness to uncertainty
- Modularity and interoperability

Disadvantages of feedback
- Increased complexity
- Potential for instability
- Amplification of noise

Feedback
- Other modules
- Com- pare
- Com- pute
- Act- uate

Uncer- tainty
- disturbances
- noise

“Closed loop control”

Other modules
## Different Types of Control Systems

<table>
<thead>
<tr>
<th>System type</th>
<th>Modeling approaches</th>
<th>Specifications</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continuous states</td>
<td>Ordinary and partial differential equations; difference equations</td>
<td>Integrated cost over time/space</td>
<td>Well-studied; excellent tools avail (especially LTI systems)</td>
</tr>
<tr>
<td>Probabilistic systems</td>
<td>Stochastic ODEs, Kolmogorov equations, Markov chains</td>
<td>Expected values and moments</td>
<td>Well-studied; excellent tools avail (especially LTI, MDPs)</td>
</tr>
<tr>
<td>Discrete state systems</td>
<td>Finite state automata, timed automata, Petri nets</td>
<td>Temporal logic formulae</td>
<td>Good tools for verification; design/synthesis is harder</td>
</tr>
</tbody>
</table>
Layered Approaches to Design

Multi-layer Networked Control System

- **Physical System**
  - Sensors
  - Actuators

- **External Environment**
  - Sensor Processing (KF)

- **Online System Model (sys + env)**
  - Online Optimization (MPC, RHC)

- **Feedback Control (PID)**

- **Decision-Making**
  - (mode, contingency and constraint management)

- **Model**

- **Specs**
  - $(\phi_{\text{init}} \land \square \phi_{\text{env}}) \implies (\square \phi_{\text{safe}} \land \square \Diamond \leq T \phi_{\text{live}})$

- **Outer loop**
  - $\dot{x} = f_\alpha(x, u)$
  - $g_\alpha(x, u, z) \leq 0$

- **Inner loop**
  - $y = P_{yu}(s) u + P_{yd}(s) d$

- **Operating Envelope**
  - $\|W(s)d(s)\| \leq 1$

- **Energy Efficiency**
  - $\min J = \int_0^T L_\alpha(x, u) dt + V(x(T))$

- **Actuator Authority**

- **Many tools available for analysis and design and different layers of abstraction**
Specifying Discrete Behavior Using Temporal Logic

Linear temporal logic (LTL)
- "eventually" - satisfied at some point in the future
- "always" - satisfied now and forever into the future
- "next" - true at next step

Signal temporal logic (STL)
- Allow predicates that compare values
- Allow temporal bounds

- \( p \rightarrow \Diamond q \)  \( p \) implies eventually \( q \) (response)
- \( p \rightarrow q \mathcal{U} r \)  \( p \) implies \( q \) until \( r \) (precedence)
- \( 
\boxdot p \)  always eventually \( p \) (progress)
- \( \Diamond \boxdot p \)  eventually always \( p \) (stability)
- \( \Diamond p \rightarrow \Diamond q \)  eventually \( p \) implies eventually \( q \) (correlation)
- \( V < V_{\text{max}} \)  \( V(t) \) less than threshold \( (V_{\text{max}}) \)
- \( \Box [t_1, t_2] \)  \( p \) true for all time in \([t_1, t_2]\)
- \( p \rightarrow \Diamond [0, t] q \) if \( p \) occurs, \( q \) will occur w/in time \( t \)

Verification via Model Checking

Basic idea

- Model systems as a finite transition system
  - Capture system + environment
- Search for executions that violate the spec
  - Provide concrete counterexample

Properties

- Checks all possible execution traces (e.g., interleaving, environmental actions)
- Modern model checkers (SPIN, nuSMV, TLC) check systems with many states (>10⁹)

Goal: leverage advances for system design and control
Formal Methods for System Design

\[ \Box \{ (\checkmark = 1 \land c = 0 \land (x_C < T_{\text{min}})) \rightarrow (\lozenge c = 0 \land \Box x_C = x_C + \delta) \}, \]
\[ \Box \{ (\checkmark = 1 \land c = 0 \land (x_C \geq T_{\text{min}})) \rightarrow (\lozenge c = 1 \lor \Box x_C = x_C + \delta) \}, \]
\[ \Box (x_C \leq T_{\text{min}}). \]

\begin{itemize}
  \item \textbf{System model}
  \item \textbf{Assumptions}
  \begin{itemize}
    \item \textbf{Requirements} (on the unknowns, e.g., environment behavior)
  \end{itemize}
  \item \textbf{Verification}
  \begin{itemize}
    \item \textbf{Satisfied} (+ certificate)
    \item \textbf{Violated} (+ counterexample)
  \end{itemize}
  \item \textbf{Synthesis}
  \begin{itemize}
    \item \textbf{Controller} that satisfies the specs
    \item \textbf{No such controller exists}
  \end{itemize}
\end{itemize}

Ready to be applied now (SoS, SPIN, TLC, nuSMV, PRISM, SMT)

Next generation tools (in progress)
Discrete Abstractions for (Hybrid) Dynamical Systems

Continuous states $\rightarrow$ discrete abstractions

$$\dot{x} = f_\alpha(x, u)$$
$$g_\alpha(x, u, z) \leq 0$$

Use formal tools to create abstractions

- Use reachability analysis (trajectory gener’n) to compute regions, transitions
- Account for disturbances, uncertainty, failures (using, for example, MPT)

- Look for regions such that we can move from one region to another w/out leaving the union of two regions

Solve via trajectory generation algorithm: piecewise linear dynamics w/ disturbances:

$$s[t + 1] = As[t] + Bu[t] + Ed[t]$$
$$s[t] \in \varsigma_i, s[N] \in \varsigma_j, u[t] \in U$$
Synthesis of Reactive (Feedback) Controllers

Reactive Protocol Synthesis
- Find control action that insures that specification is always satisfied
- For LTL, complexity is doubly exponential (!) in the size of system specification

GR(1) synthesis for reactive protocols
- Piterman, Pnueli and Sa’ar, 2006
- Assume environment fixes action before controller (breaks symmetry)
- For certain class of specifications, get complexity cubic in # of states (!)

\[
\begin{align*}
\square\{(\bar{e} = 1 \land c = 0 \land (x_C < T_{\text{init}})) & \rightarrow \\
(\square c = 0 \land x_C = x_C + \delta)\}, \\
\square\{(\bar{e} = 1 \land c = 0 \land (x_C \geq T_{\text{init}})) & \rightarrow \\
(\square c = 1 \lor x_C = x_C + \delta)\}, \\
\square(x_C \leq T_{\text{max}}). \\
\end{align*}
\]

Environment assumption  System guarantee

- GR(1) = general reactivity formula
- Assume/guarantee style specification

A. Pnueli, 2005

Richard M. Murray, Caltech CDS

CDC, 11 Dec 2017
Temporal Logic Planning (TuLiP) toolbox
http://tulip-control.org

Python Toolbox
- GR(1), LTL specs
- Nonlinear dynamics
- Supports discretization via MPT
- Control protocol designed w/ gr1c
- Receding horizon compatible

Applications of TuLiP
- 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: 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 intersection when clear
- No U-turns in the middle of a road

Assumptions
- Obstacle may not block the entire road
- Obstacle is detected before vehicle gets too close
- 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

Solved using receding horizon temporal logic planning
TuLiP returns 900 state FSA in about 1.5 seconds

\[
\left( \phi_{\text{init}}^e \land □ \phi_{\text{safe}}^e \land □ □ \phi_{\text{prog}}^e \right) \\
\rightarrow \left( \phi_{\text{init}}^s \land □ \phi_{\text{safe}}^s \land □ □ \phi_{\text{prog}}^s \right)
\]
Rapprochement Between Formal Methods and Control Theory

Getting more rigorous about control of reactive systems

- Systems are too complex to be tested by trial and error
- Systems are too safety-critical to be tested by trial and error
- Move from “design then verify” to
  - specify then synthesize
  - synthesis of contracts
- Combine data-driven with formal methods to achieve safety, performance and “human-like” interactions
- Incorporate security and privacy as guarantees

Controlling cyberphysical systems requires solving both problems

\[
\|z\|_2 \leq \gamma \|d\|_2 \quad \text{for all} \quad \|\Delta\| \leq 1
\]