Specification and Synthesis of Networked Control Systems with Application to Autonomous Vehicles

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

Tichakorn (Nok) Wongpiromsarn
Ministry of Science and Technology
Thailand

Ufuk Topcu
Aerospace Engineering
U. Texas

IEEE International Conference on Automation Science and Engineering (CASE)
26 August 2015

http://www.cds.caltech.edu/~murray/wiki/CASE2015

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

Motivating Example: Alice (2004-2007)

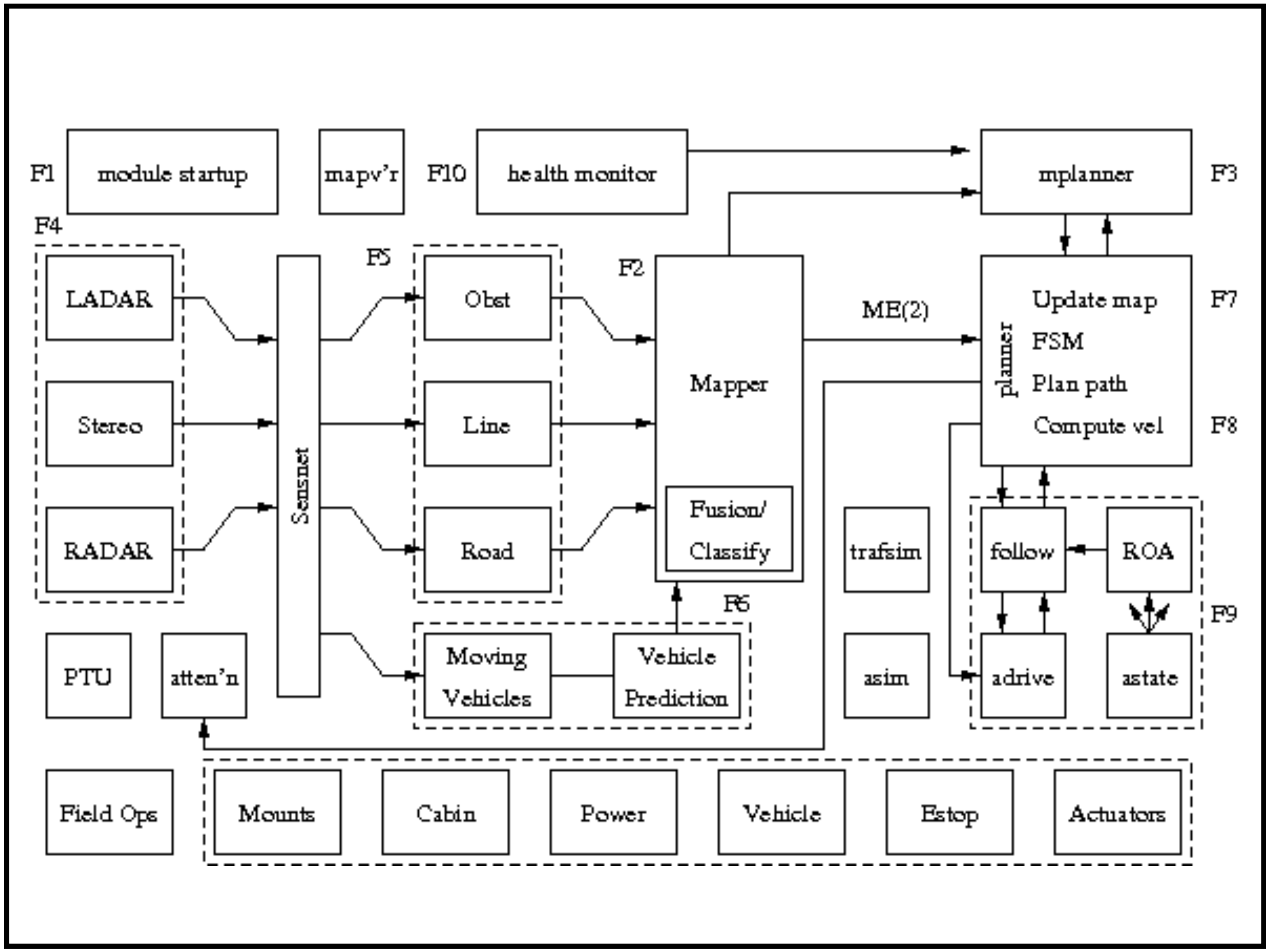
Alice

- 300+ miles of fully autonomous driving
- 8 cameras, 8 LADAR, 2 RADAR
- 12 Core 2 Duo CPUs + Quad Core
- 3 Gb/s data network
- ~75 person team over 18 months (x 2)

Software

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

Networked Control System

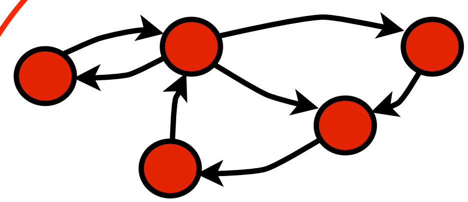


How should we design systems of this complexity?
How do we make sure they function as desired?

Abstractions for Networked Control System Design

Continuous: $\dot{x} = f_\alpha(x, u, d)$ $\quad$ $\min J = \int_0^T L(x, u, \alpha)dt + V(x(T))$

Discrete: $g(x, \alpha) \implies \alpha' = r(x, \alpha)$ $\quad$ if X then Y, never Z, always W, …

	Level	Model	Specification
Supervisory Control (FSM)	Decision-Making		Logical constraints (MLD) Regular languages (DES) Temporal logic (LTL, STL) $(\phi_{\text{init}} \wedge \Box\phi_{\text{env}}) \Longrightarrow (\Box\phi_{\text{safe}} \wedge \Box\Diamond_{\le T}\phi_{\text{live}})$
Online Optimization (RHC)	Trajectory	$\dot{x} = f_\alpha(x, u)$ $g_\alpha(x, u, z) \le 0$	$\min J = \int_0^T L_\alpha(x, u)\, dt + V(x(T))$
Feedback Control (PID)	Tracking	$y = P_{yu}(s)u + P_{yd}(s)d$ $\|W(s)d(s)\| < 1$	$T\|_\infty < \gamma$
System Dynamics (ODE)	Process	$\dot{x}^i = f_\alpha(x^i, u^i, d^i)$ $x \in \mathcal{X}, u \in \mathcal{U}, d \in \mathcal{D}$	Operating Envelope Energy Efficiency Actuator Authority

Outline for remainder of today's talk

- Formal specification using temporal logic (LTL, STL)
- "Design then verify": modeling checking and abstraction
- "Correct-by-construction" synthesis of controllers
- Final thoughts: where we have been, where we might go

Specifying Discrete Behavior Using Temporal Logic

Linear temporal logic (LTL)

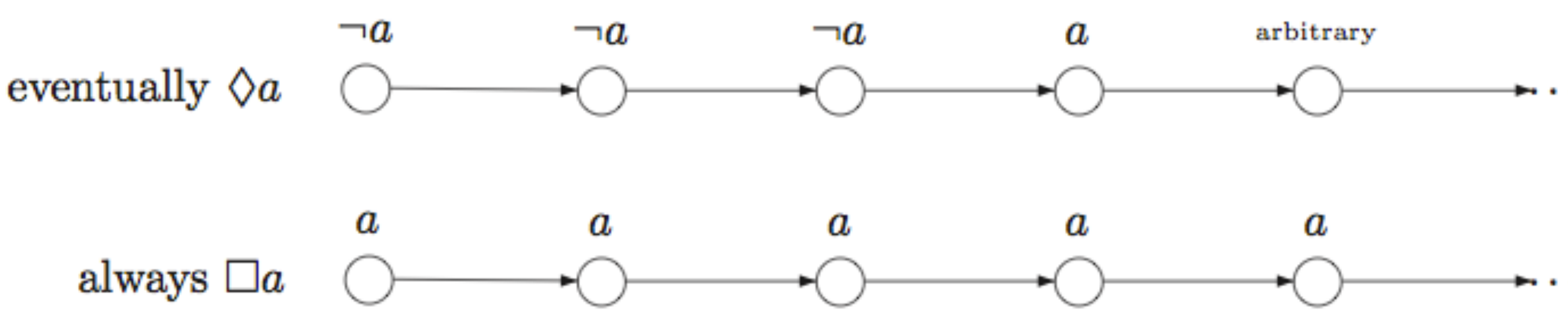
- $\Diamond$ "eventually" - a property is satisfied at some point in the future
- $\Box$ "always" - a property is satisfied now and forever into the future
- $\bigcirc$ "next" - true at next step

• $p \rightarrow \Diamond q$	p implies eventually q (response)
• $p \rightarrow q \, U \, r$	p implies q until r (precedence)
• $\Box\Diamond p$	always eventually p (progress)
• $\Diamond\Box p$	eventually always p (stability)
• $\Diamond p \rightarrow \Diamond q$	eventually p implies eventually q (correlation)

Signal temporal logic (STL)

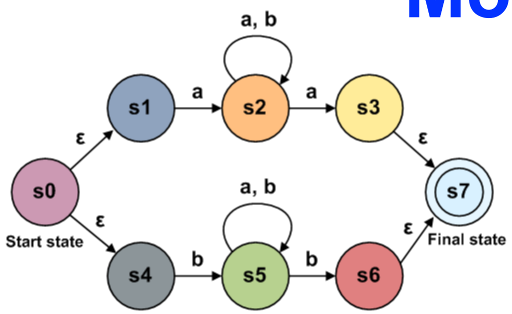
- Allow predicates that compare values (via subsets of state space)
- Allow bounds on temporal operators

• $V < V_{\max}$	$V(t)$ less than threshold ($V_{\max}$)
• $\Box_{[t1,t2]}\, p$	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

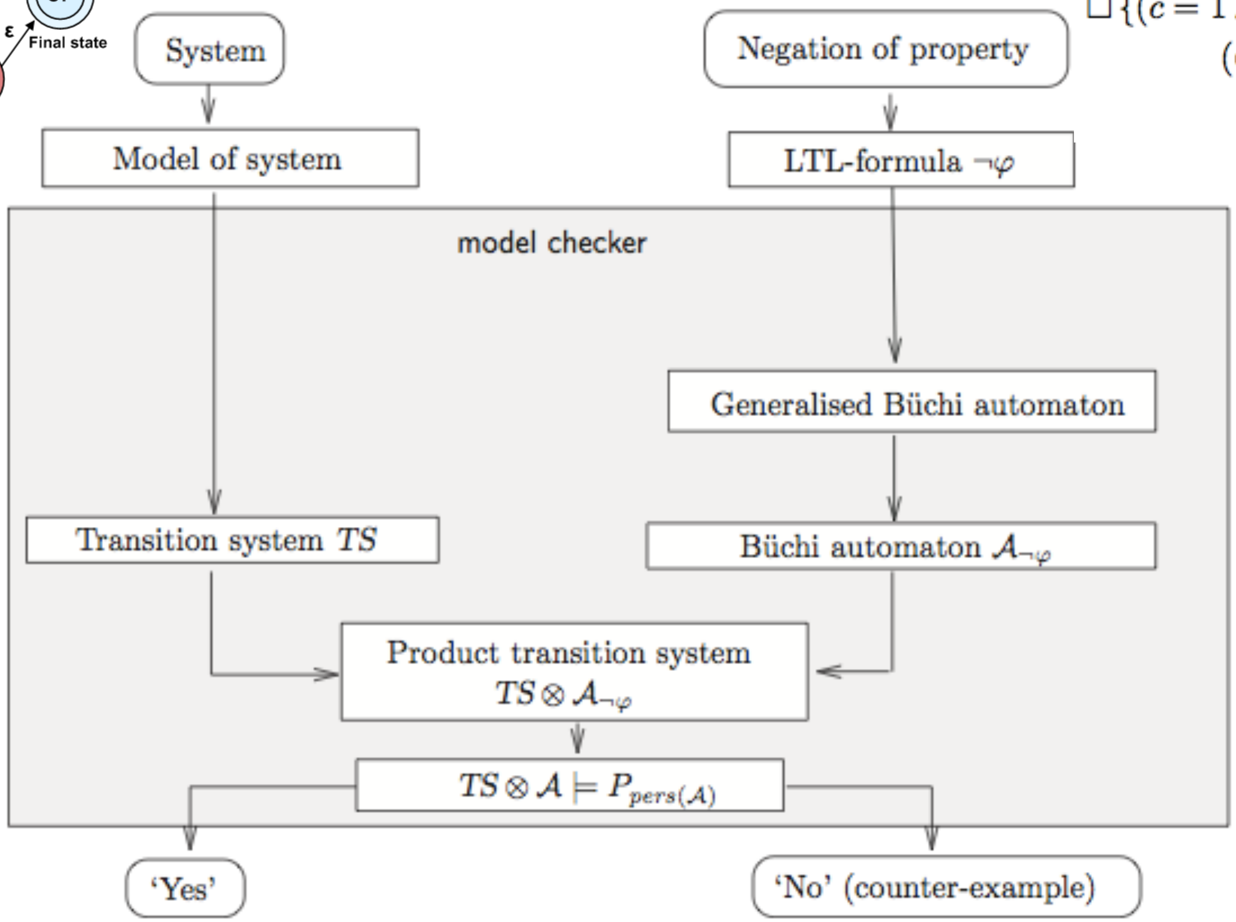


Baier and Katoen, *Principles of Model Checking*, 2007

Model Checking: Design and Verify



$$\Box\{(\tilde{c} = 1 \wedge c = 0 \wedge (x_C < T_{c_{min}})) \rightarrow (\bigcirc c = 0 \wedge \bigcirc x_C = x_C + \delta)\},$$
$$\Box\{(\tilde{c} = 1 \wedge c = 0 \wedge (x_C \geq T_{c_{min}})) \rightarrow (\bigcirc c = 1 \vee \bigcirc x_C = x_C + \delta)\},$$
$$\Box(x_C \leq T_{c_{max}}).$$

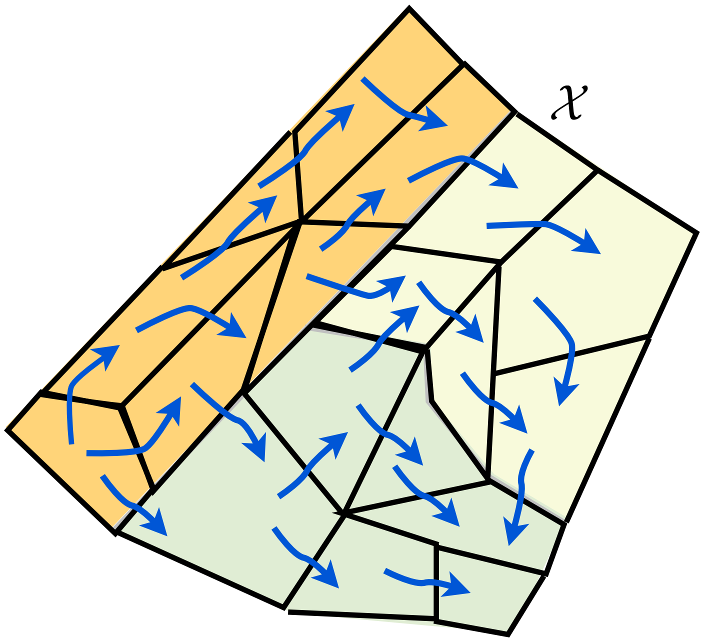


Baier and Katoen, *Principles of Model Checking*, 2007

Approach: enumeration of all possible execution sequences (!)

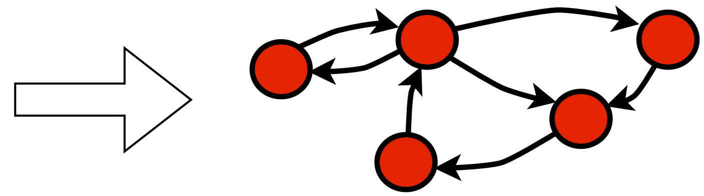
- Can test systems with up to 10^{11} states

Discrete Abstractions for (Hybrid) Dynamical Systems



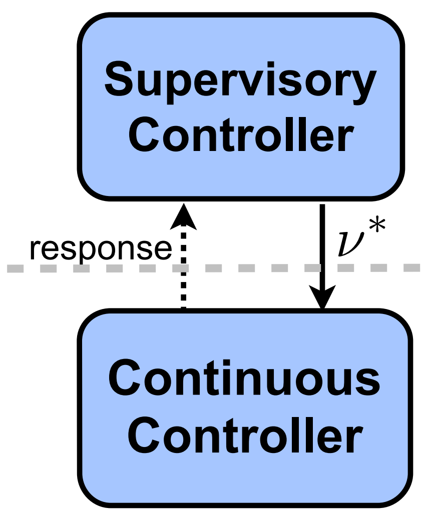
Continuous models to discrete abstractions

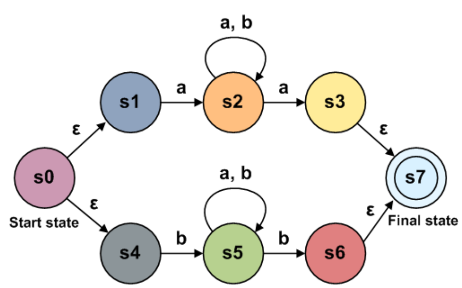
$$\dot{x} = f_\alpha(x, u)$$
$$g_\alpha(x, u, z) \leq 0$$

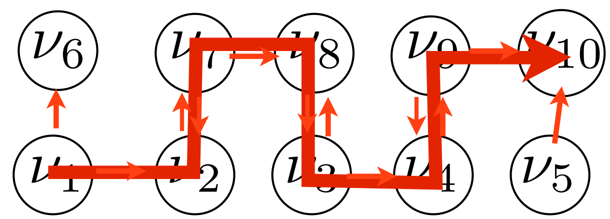


Formal tools available to create abstractions

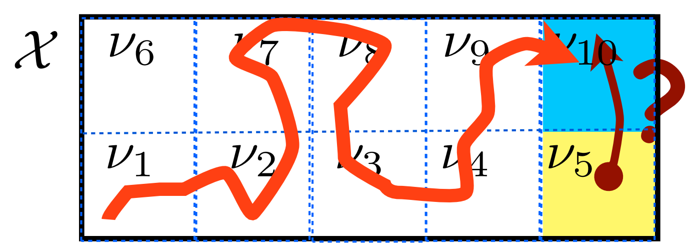
- Use reachability analysis (trajectory generation) to compute regions, transitions
- Account for disturbances, uncertainty, failures (using, for example, MPT toolbox)



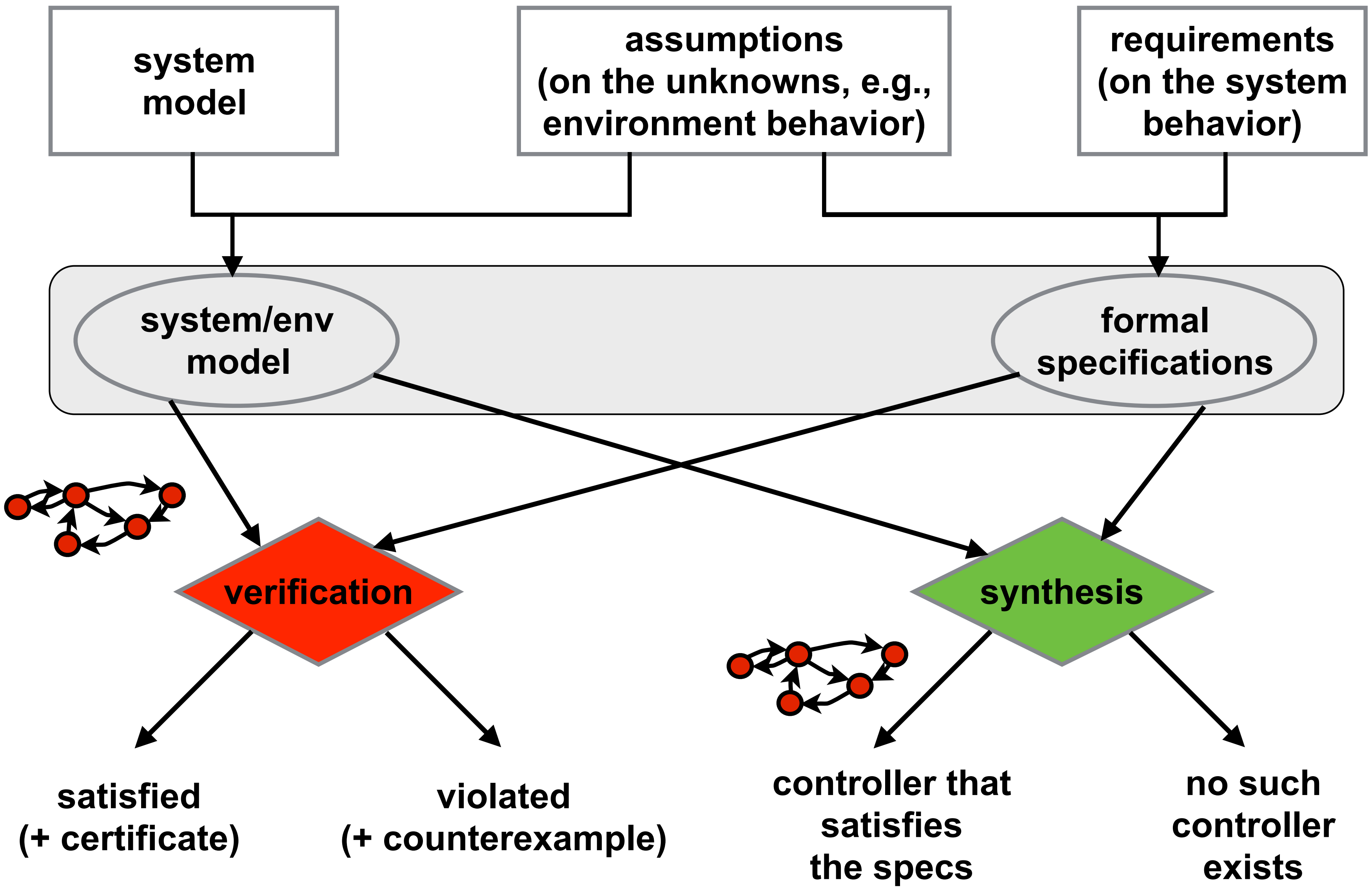




$$\min J = \int_0^T L_\alpha(x, u)\, dt + V(x(T))$$



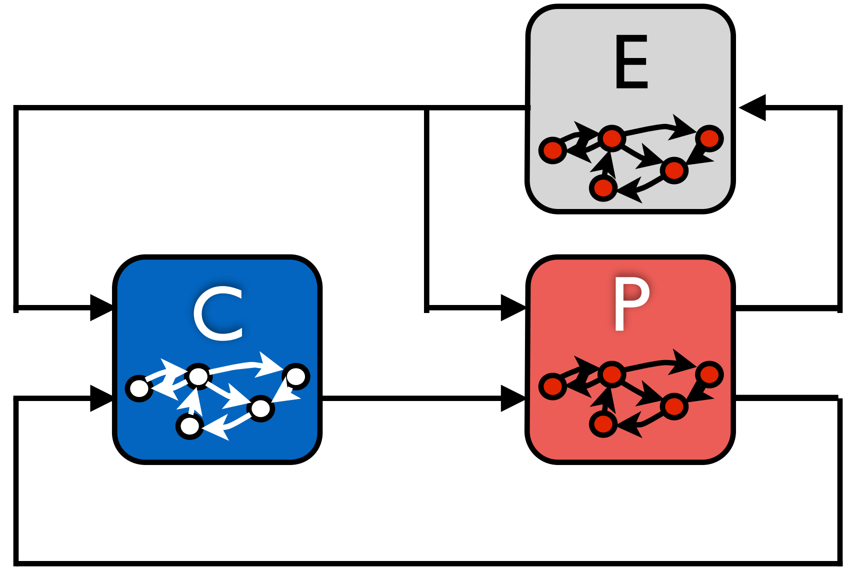
Formal Methods for System Verification & *Synthesis*



"Correct-by-Construction" Controller *Synthesis*

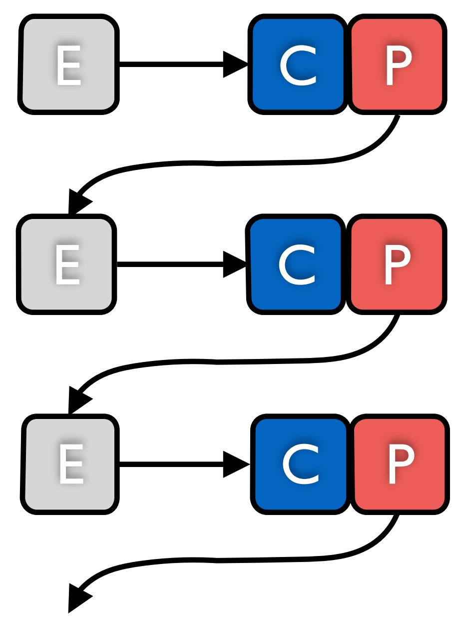
Reactive Protocol Synthesis

- Find control action that insures that specification is always satisfied
- Complexity is doubly exponential (!) in size of the system specification



GR(1) synthesis for reactive protocols

- Piterman, Pnueli and Sa'ar, 2005
- Assume environment fixes action before controller (breaks symmetry)
- For certain class of specifications, get complexity cubic in # of states (!)



$$(\phi^{e}_{\mathrm{init}} \wedge \Box\phi^{e}_{\mathrm{safe}} \wedge \Box\Diamond\phi^{e}_{\mathrm{prog}}) \rightarrow (\phi^{s}_{\mathrm{init}} \wedge \Box\phi^{s}_{\mathrm{safe}} \wedge \Box\Diamond\phi^{s}_{\mathrm{prog}})$$

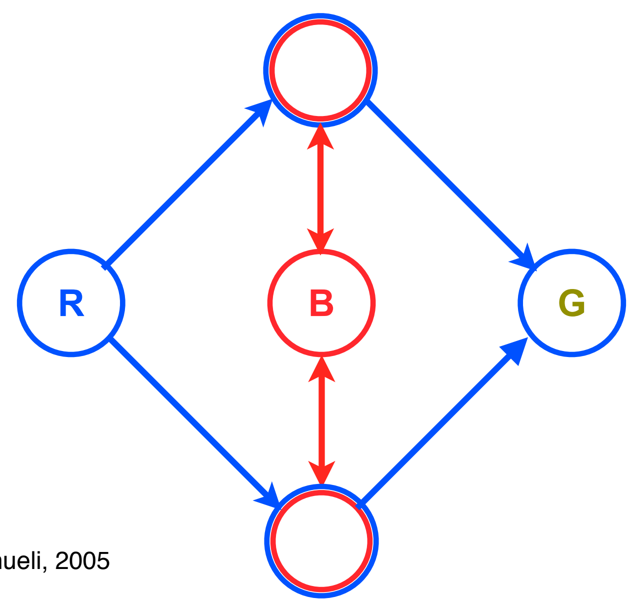
Environment assumption System guarantee

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

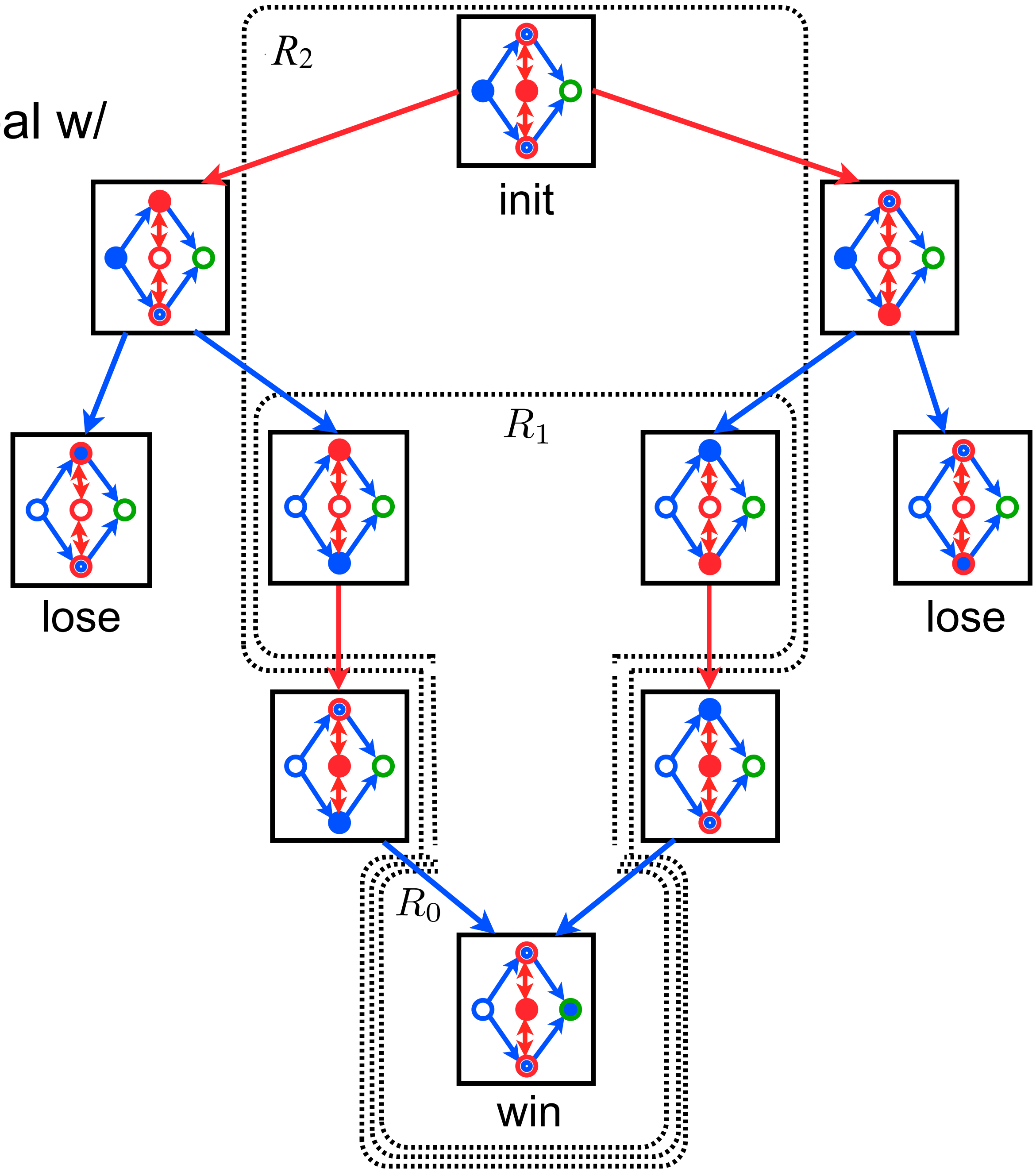
Example: Runner Blocker System

Simple two person game

- Runner attempts to reach goal w/ out being blocked
- Blocker has limited motion
- Each player must move each turn
- Back out strategy from seq- uence of winning sets



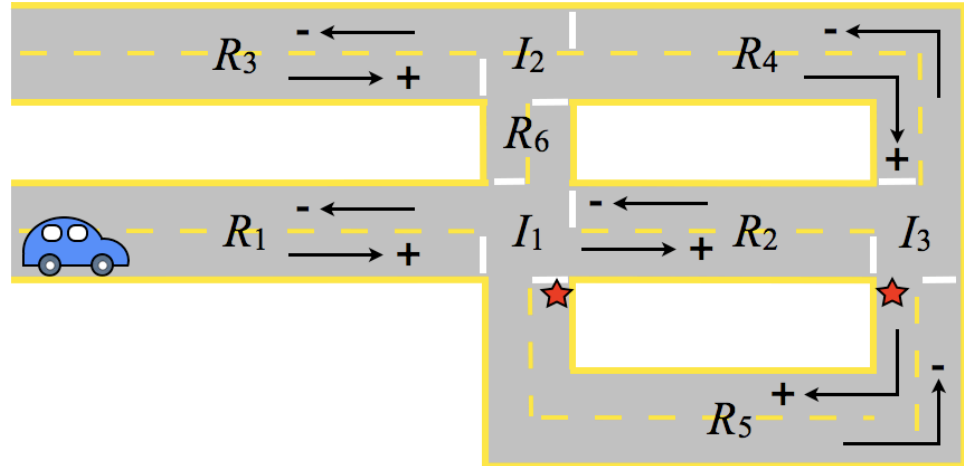
A. Pnueli, 2005



Example: Autonomous Navigation in Urban Environment

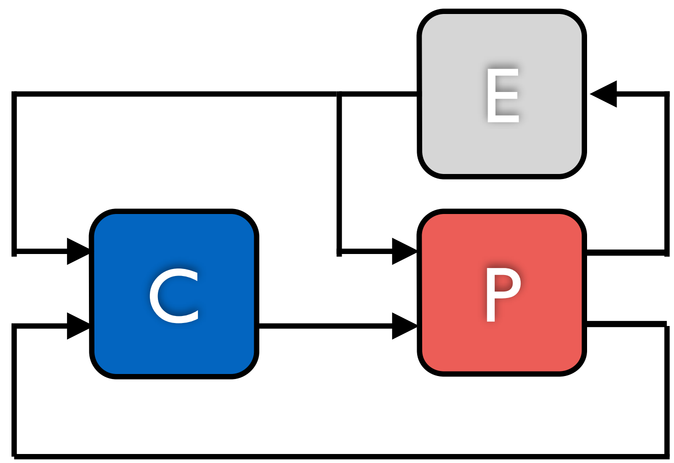
Traffic rules

- 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



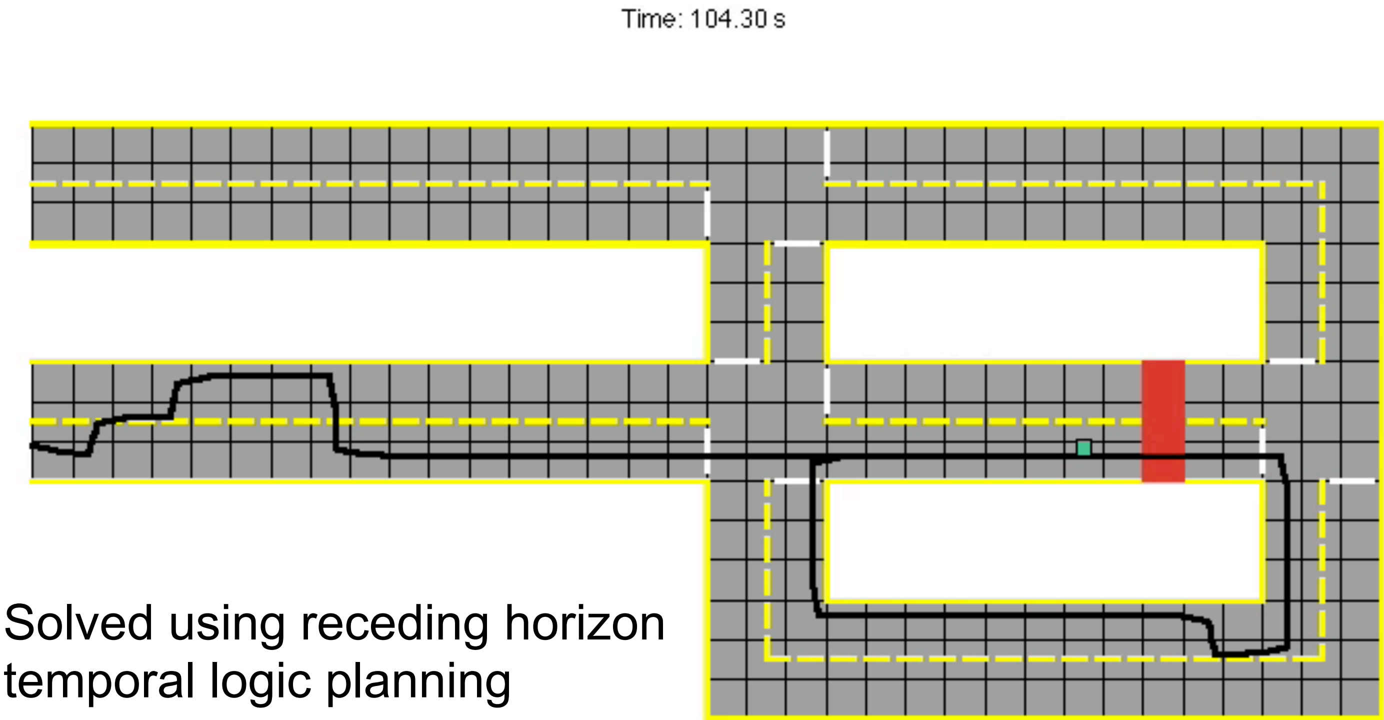
Assumptions

- Obstacle may not block a road
- Obstacle is detected before vehicle gets too close
- 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

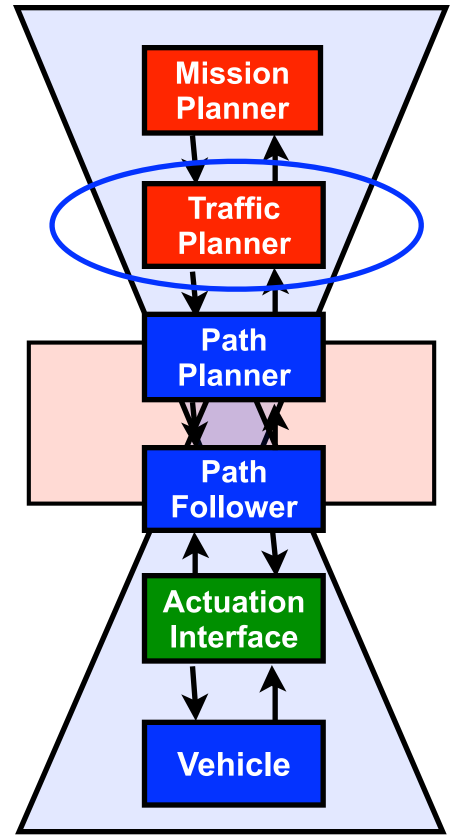


$$(\phi^e_{\text{init}} \wedge \Box\phi^e_{\text{safe}} \wedge \Box\Diamond\phi^e_{\text{prog}}) \rightarrow (\phi^s_{\text{init}} \wedge \Box\phi^s_{\text{safe}} \wedge \Box\Diamond\phi^s_{\text{prog}})$$

Example: Autonomous Navigation in Urban Environment



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



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

Wongpiromsarn, Topcu and M
IEEE TAC 2012

Receding Horizon Control for Linear Temporal Logic

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

$$(\varphi_{init} \wedge \square\varphi_e) \implies (\square\varphi_s \wedge \Diamond\varphi_g)$$

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

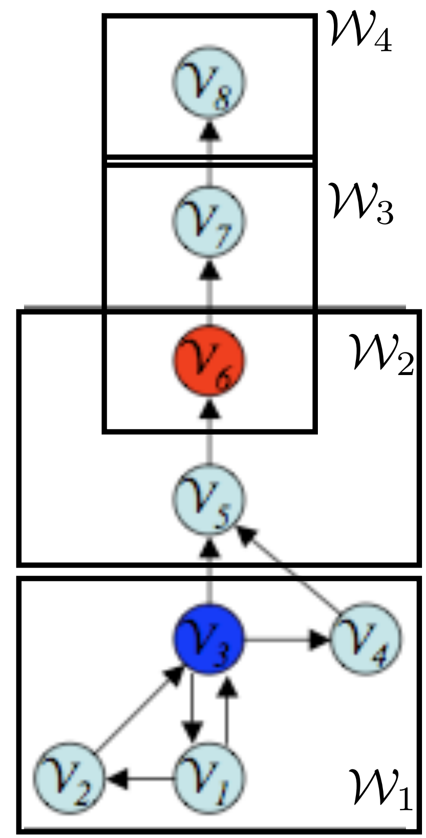
- For discrete system, can find automaton to satisfy this formula in $O((nm|\Sigma|^3)$ time (!)

Basic idea

- Discretize state space into regions $\{\mathcal{V}_i\}$ + interconnection graph
- Organize regions into a partially ordered set $\{\mathcal{W}_i\}$; $\mathcal{W}_j \preceq_{\varphi_g} \mathcal{W}_i$ $\Rightarrow$ 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

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

  - 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

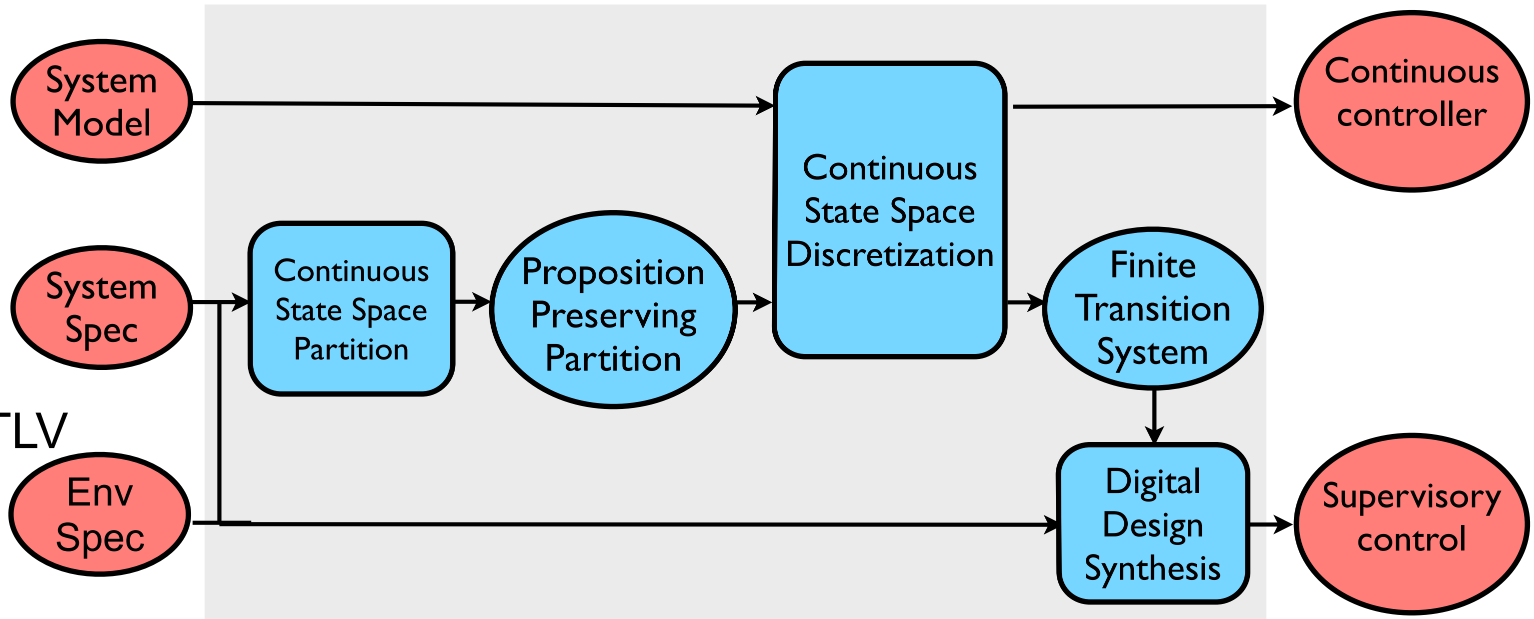
- Provably correct behavior according to spec

Temporal Logic Planning (TuLiP) toolbox

http://tulip-control.org

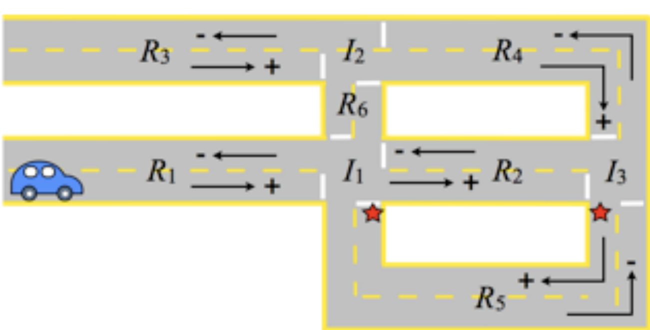
Python Toolbox

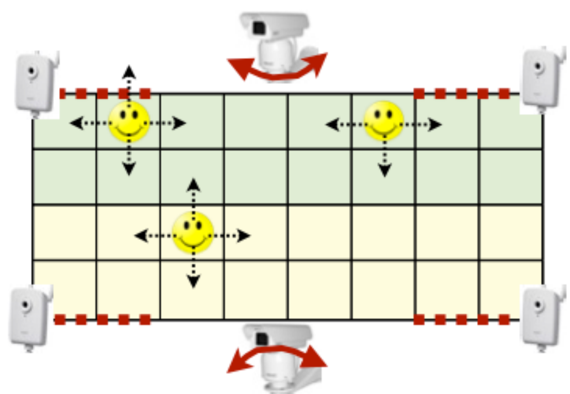
- GR(1), LTL specs
- Nonlin dynamics
- Supports discretization via MPT
- Control protocol designed using JTLV
- Receding horizon compatible

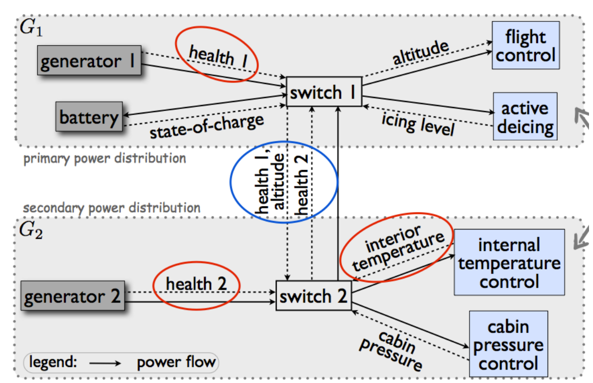


Applications of TuLiP

- 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







Approaches for Correct-By-Construction Synthesis

system model	assumptions (on the unknowns, e.g., environment behavior)	requirements (on the system behavior)

$$\dot{x}^i = f^i(x^i, \alpha^i, y^{\sim i}, u^i) \qquad x^i \in \mathbb{R}^n, u^i \in \mathbb{R}^m$$
$$y^i = h^i(x^i, \alpha^i) \qquad y^i \in \mathbb{R}^q$$

$$J = \int_0^T L(x, \alpha, u)\, dt + V(x(T), \alpha(T)),$$
$$(\varphi_{init} \wedge \Box \varphi_e) \implies (\Box \varphi_s \wedge \Diamond_{\leq T}\, \varphi_g)$$

Type	Discrete abstraction	Layered architecture	Mixed integer solver
Prop-erties	• Continuous dyna-mics $\rightarrow$ discrete transition system	• Break problem into separate layers of abstraction	• Convert temporal logic into integer constraints
Pros	• Exploit SAT, SMT, etc • Compatibility with model checkers	• Use best tools at each layer • Modularity	• Exploit MILP solvers • Rich specification semantics
Cons	• Get very large dimen-sional state spaces • Harder to encode optimality specs	• Requires manual de-composition of layers • No formal proofs of correctness (yet)	• Can get large number of integer variables • Difficult to encode reactivity (& GR(1))

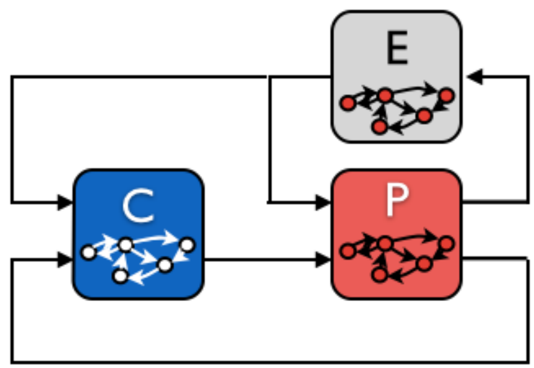
Summary and Future Research

Networked control of autonomous systems

- Requires integration of control, computer science, networking technologies
- Specific focus on *robustness* (to environment, to faults)
- Move from *design-then-verify* to *specify-then-synthesize*

Many open problems remain

- Decomposition of specs between subsystems/agents
- Design of abstraction layers + interfaces
- Extension to more descriptive classes of specifications: timed, probabilistic, etc



$$(\phi_{\text{init}} \wedge \Box\phi_{\text{env}}) \implies (\Box\phi_{\text{safe}} \wedge \Box\Diamond_{\leq T}\phi_{\text{live}})$$