Specification, Verification and Synthesis of Networked Control Systems

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

Tichakorn (Nok) Wongpiromsarn
Ministry of Science and Technology
Thailand

Ufuk Topcu
Electrical & System Engineering
U. Pennsylvania

IFAC World Congress
Cape Town, South Africa
29 August 2014

TerraSwarm

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)

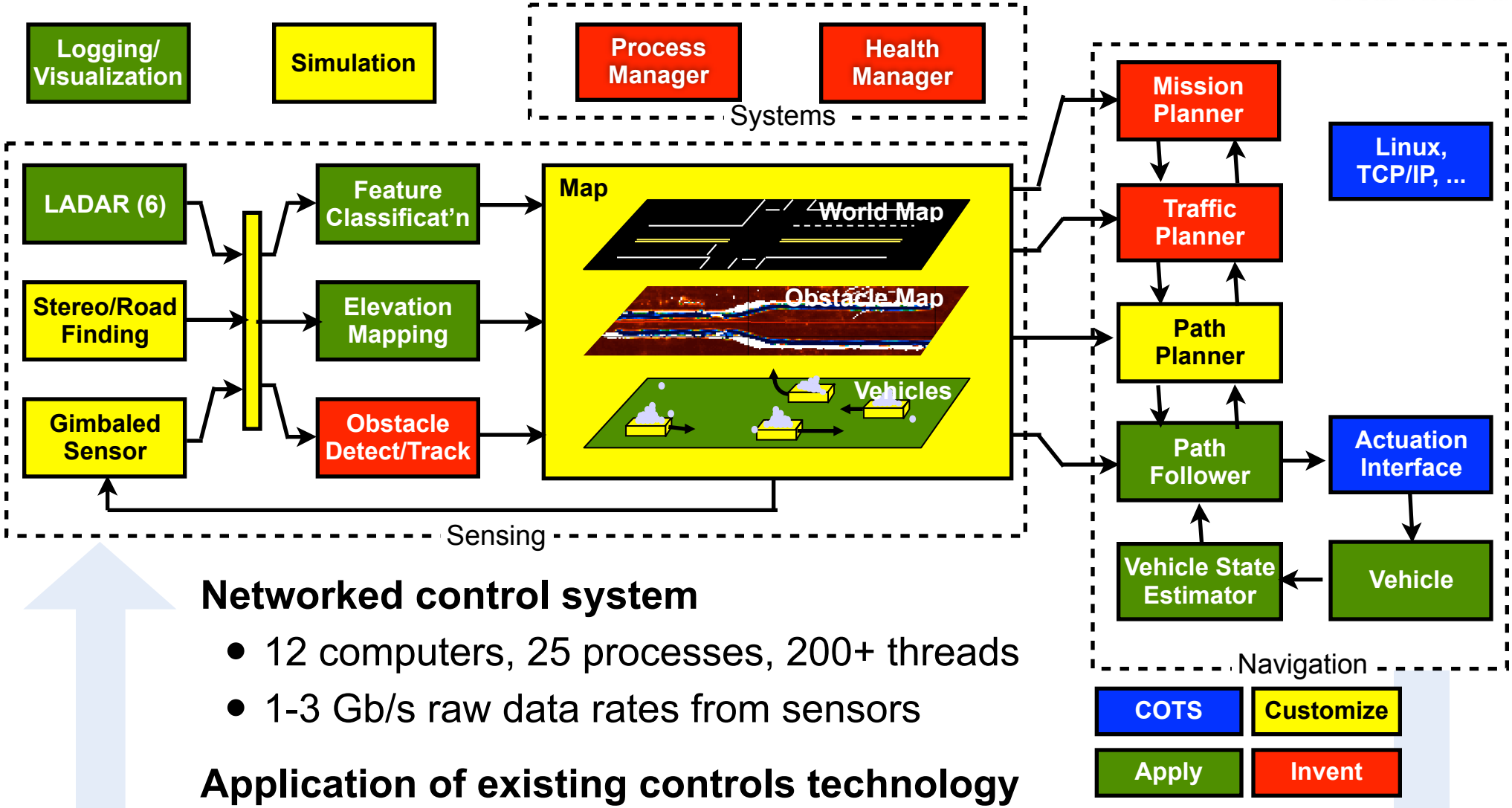
Software

- 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?

DGC07 System Architecture



Networked control system

- 12 computers, 25 processes, 200+ threads
- 1-3 Gb/s raw data rates from sensors

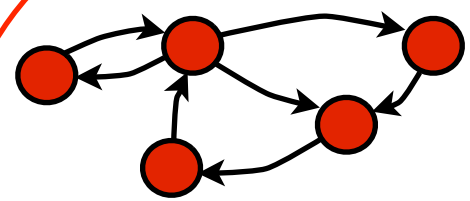
Application of existing controls technology

- Receding horizon control
- Multi-layer sensor fusion
- Real-time trajectory generation
- PID w/ anti-windup

Abstractions for Networked Control System Design

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

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

	Level	Model	Specification
Supervisory Control (FSM)	Decision-Making		$(\phi_{\text{init}} \wedge \Box\phi_{\text{env}}) \implies (\Box\phi_{\text{safe}} \wedge \Box\Diamond_{\leq T}\phi_{\text{live}})$ Logical constraints (MLD) Regular languages (DES) Temporal logic (LTL, STL)
Online Optimization (RHC)	Trajectory	$\dot{x} = f_\alpha(x, u)$ $g_\alpha(x, u, z) \leq 0$	$\min J = \int_0^T L_\alpha(x, u)\, dt + V(x(T))$
Feedback Control (PID)	Tracking	$u = P_{yr}(s)y + P_{ud}(s)d$ $\|W(s)d(s)\| < 1$	$\|W_2 T\|_\infty < \gamma$
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

(Selected) Prior Results to Build On

Discrete Event Systems

- Ramadge and Wonham, “Supervisory control of a class of discrete event processes”, SIAM J. on Control and Optimization, 1987
- Casandras and Lafortune, *Intro to Discrete Event Systems*, 1993/2009 [Chestnut ’99]

Hybrid Systems

- Bemporad, Morari, “Control of systems integrating logic, dynamics, and constraints”, *Automatica*, 1999 [IFAC High Impact ’14]
- Alur, Henzinger, Lafferriere, Pappas, “Discrete abstractions of hybrid systems”, *Proceedings of the IEEE*, 2000

Model Checking

- Clarke, Emerson, Sistla, "Automatic verification of finite-state concurrent systems using temporal logic specifications", *ACM T. Programming Lang & Systems*, 1986.
- Baier and Katoen, *Principles of Model Checking*, 2007

***Reactive* Synthesis (of control protocols)**

- Pnueli and Rosner, “On the synthesis of a reactive module”, *SIGPLAN/SIGACT*, 1989
- Piterman, Pneuli, Sa’ar, “Synthesis of Reactive(1) Designs”, LNCS, 2006.
- Kress-Gazit, Fainekos, Pappas, “Temporal-logic-based reactive mission and motion planning”, *IEEE T. Robotics*, 2009

Specifying Discrete Behavior Using Temporal Logic

Linear temporal logic (LTL)

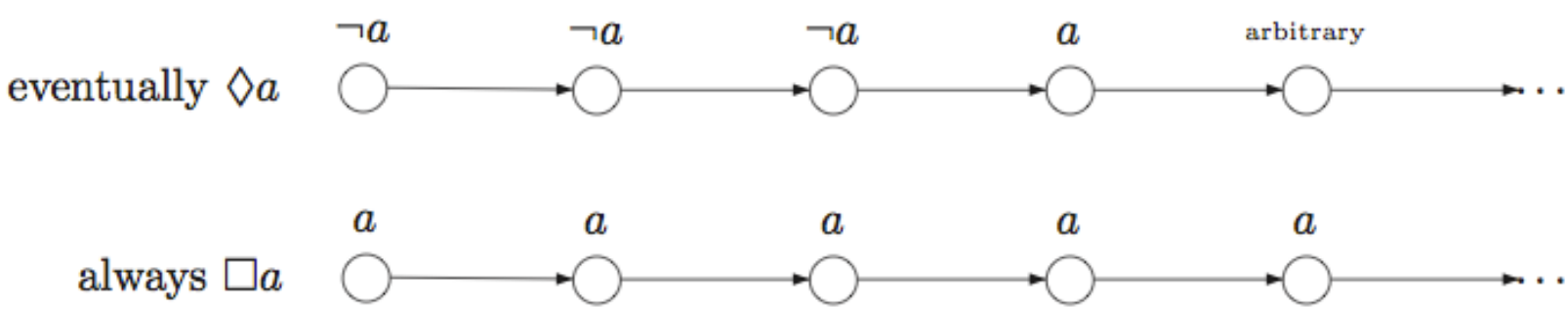
- ◊ "eventually" - a property is satisfied at some point in the future
- □ "always" - a property is satisfied now and forever into the future
- ○ "next" - true at next step

- $p \rightarrow \Diamond q$ — 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

Example: Traffic Light

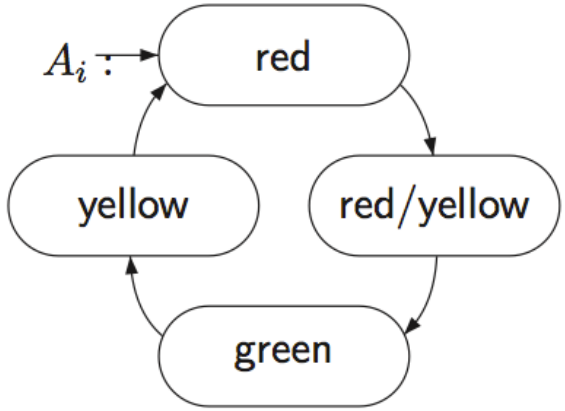
Ordering specifications

- Liveness: “traffic light is green infinitely often”

$$\Box\Diamond \text{green}$$

- Chronological ordering: “once red, the light cannot become green immediately”

$$\Box\ (\text{red} \rightarrow \neg \bigcirc \text{green})$$



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

- More detailed: “once red, the light always becomes green eventually after being yellow for some time”

$$\Box(\text{red} \rightarrow (\Diamond\ \text{green} \wedge (\neg\ \text{green}\ \mathrm{U}\ \text{yellow})))$$

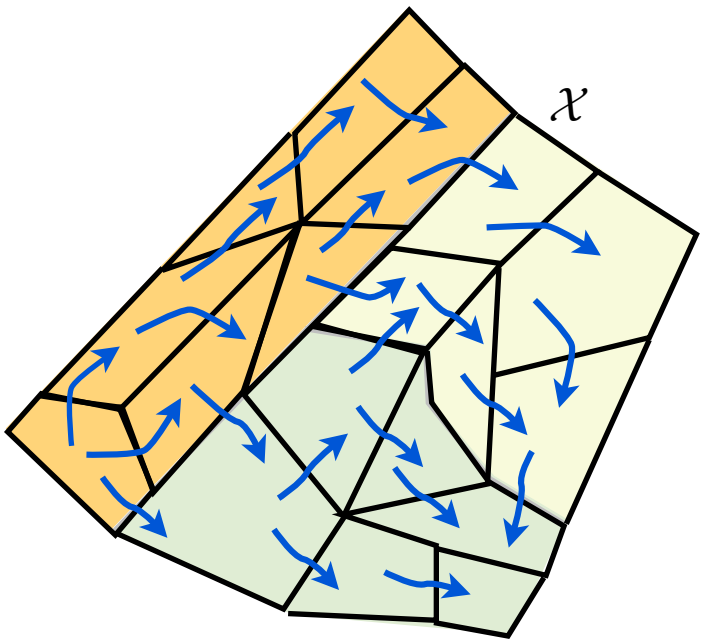
$$\Box(\text{red} \rightarrow \bigcirc\ (\text{red}\ \mathrm{U}\ (\text{yellow} \wedge \bigcirc\ (\text{yellow}\ \mathrm{U}\ \text{green}))))$$

Progress property

- Every request will eventually lead to a response

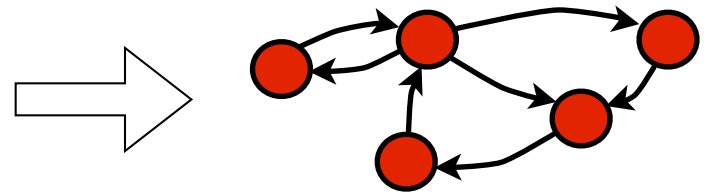
$$\Box\ (\text{request} \rightarrow \Diamond \text{response})$$

Discrete Abstractions for (Hybrid) Dynamical Systems



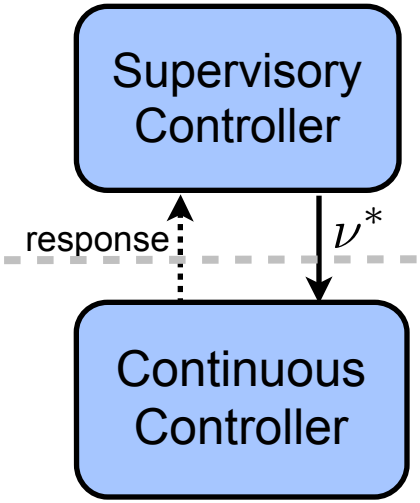
Continuous models to discrete abstractions

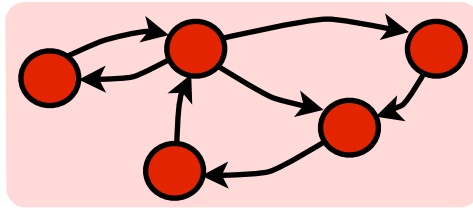
$$\dot{x} = f_\alpha(x, u)$$
$$g_\alpha(x, u, z) \leq 0$$

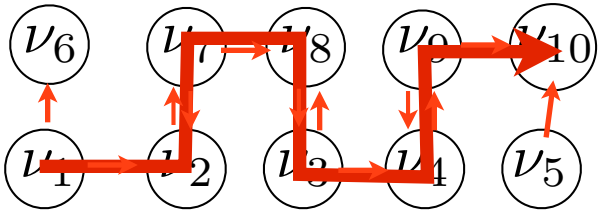


Formal tools available to create abstractions

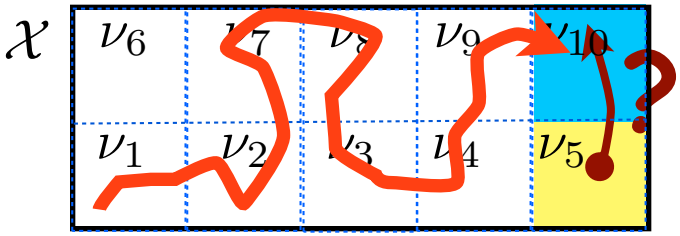
- Use reachability analysis (trajectory generation) to compute regions, transitions
- Account for disturbances, uncertainty, failures (using, for example, MPT toolbox)



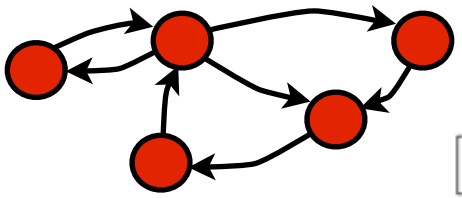




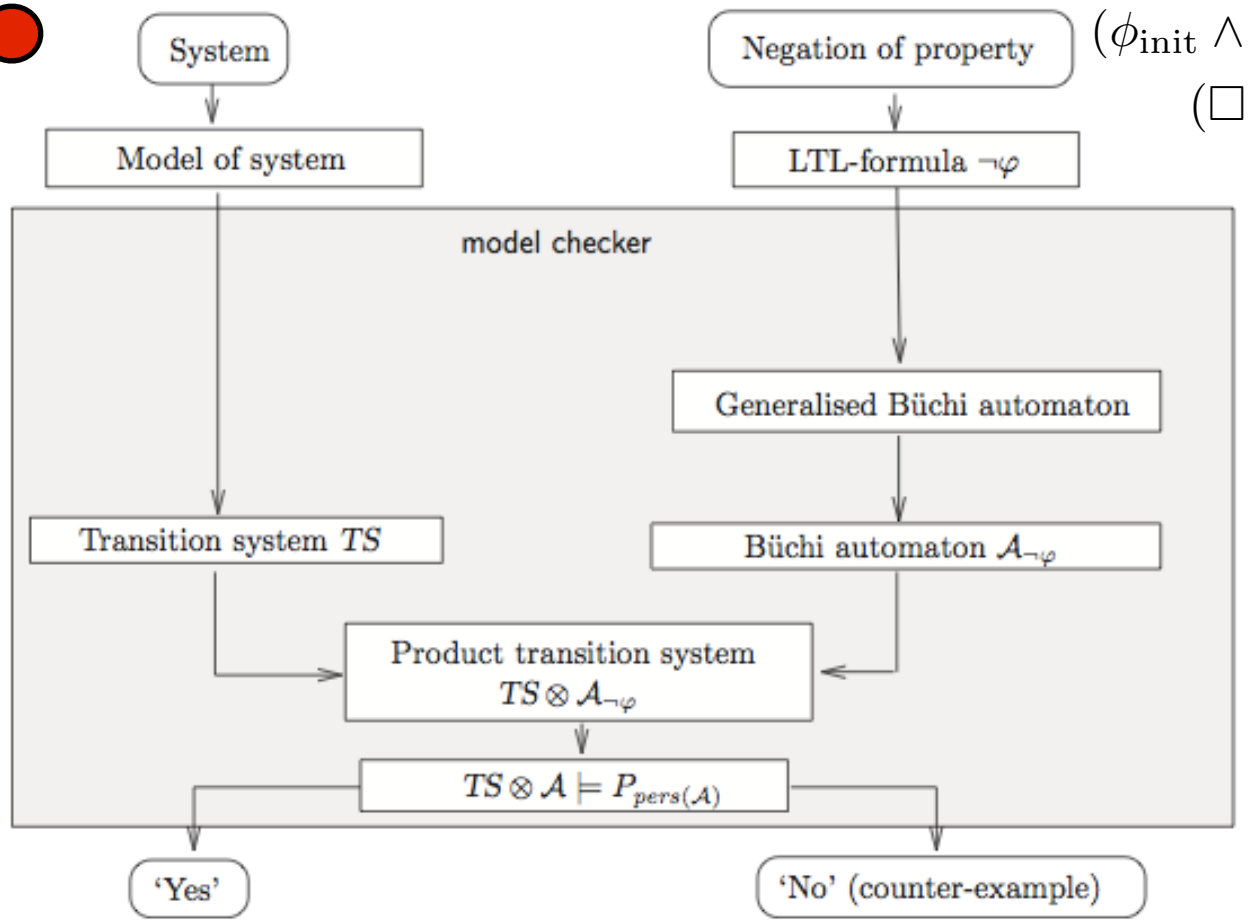
$$\min J = \int_0^T L_\alpha(x, u)\, dt + V(x(T))$$



Model Checking: Design and Verify



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

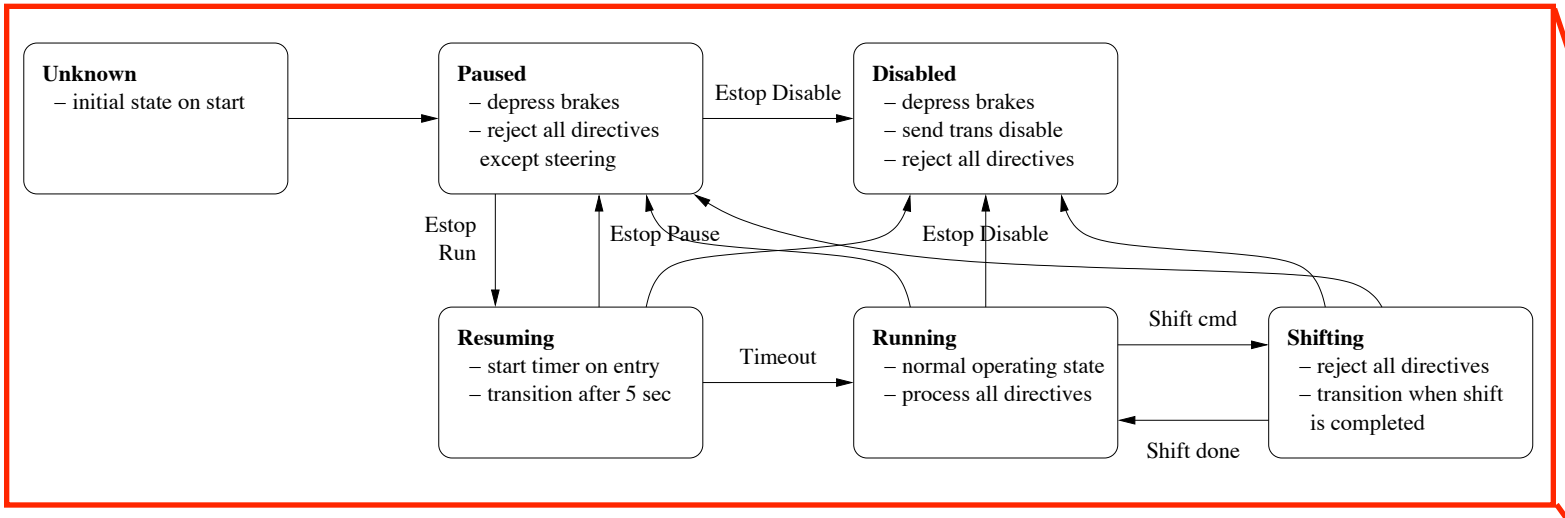


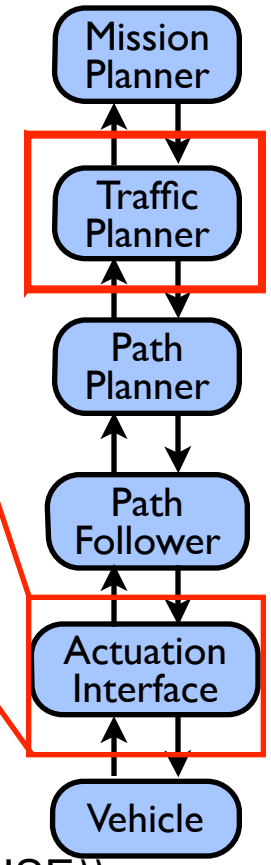
Baier and Katoen, *Principles of Model Checking*, 2007

Approach: enumeration of all possible execution sequences (!)

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

Example: Verification of Safety Logic





Function: respond to control commands + DARPA pause/emergency stop

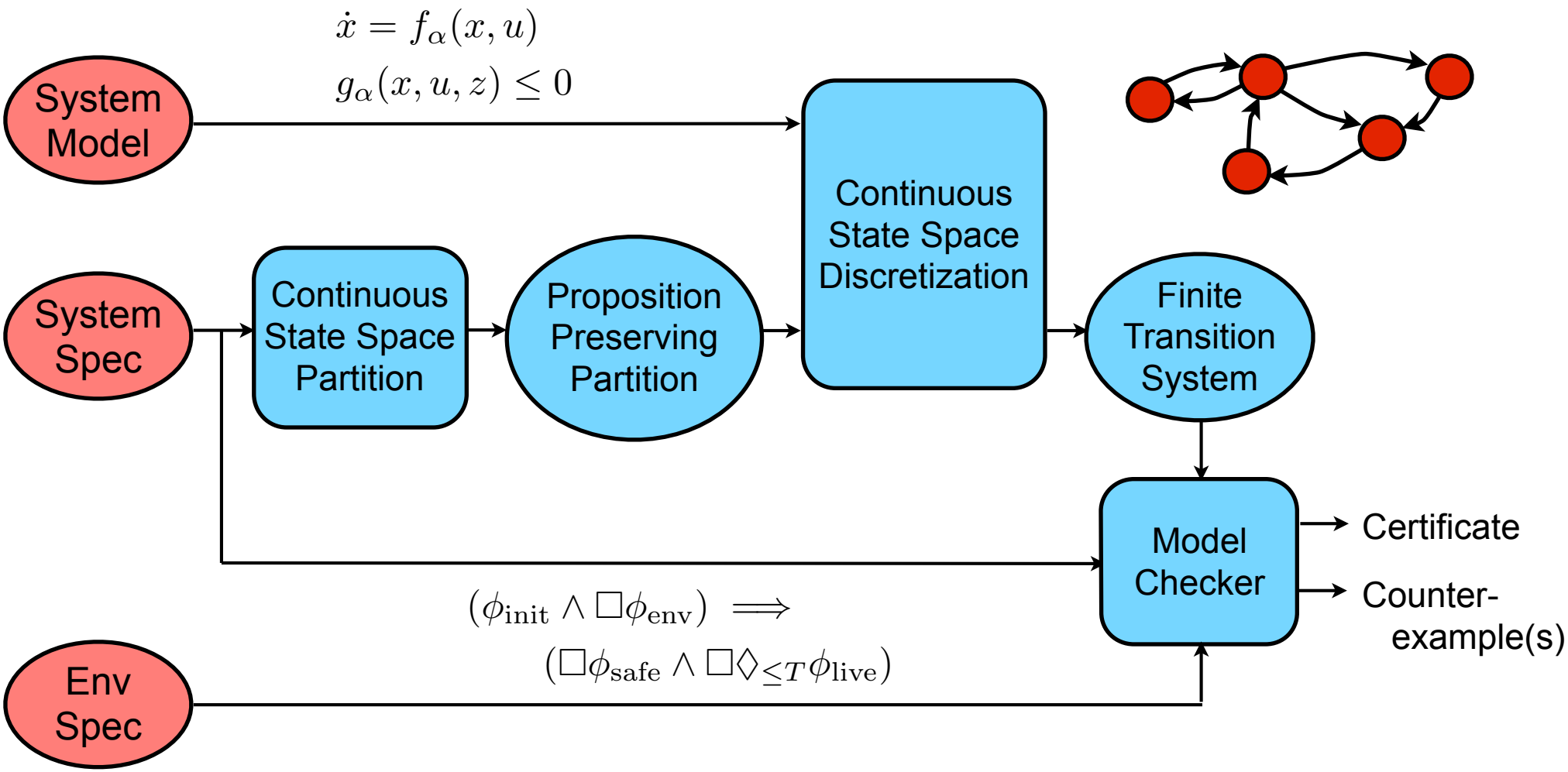
Verify the following properties

- $\Box((estop = \text{DISABLE}) \Rightarrow \Diamond\Box(state = \text{DISABLED} \wedge acc = -1))$
- $\Box((estop = \text{PAUSE}) \Rightarrow \Diamond(state = \text{PAUSED} \vee estop = \text{DISABLE}))$
- $\Box((estop = \text{RUN}) \Rightarrow \Diamond(state = \text{RUNNING} \vee state = \text{RESUMING}))$
- $\Box((state = \text{RESUMING}) \Rightarrow \Diamond(state = \text{RUNNING} \vee estop = \text{DISABLE} \vee estop = \text{PAUSE}))$
- $\Box((\text{state} \in \{\text{DISABLE, PAUSED, RESUMING, SHIFTING}\} \Rightarrow \text{acc} = -1)$

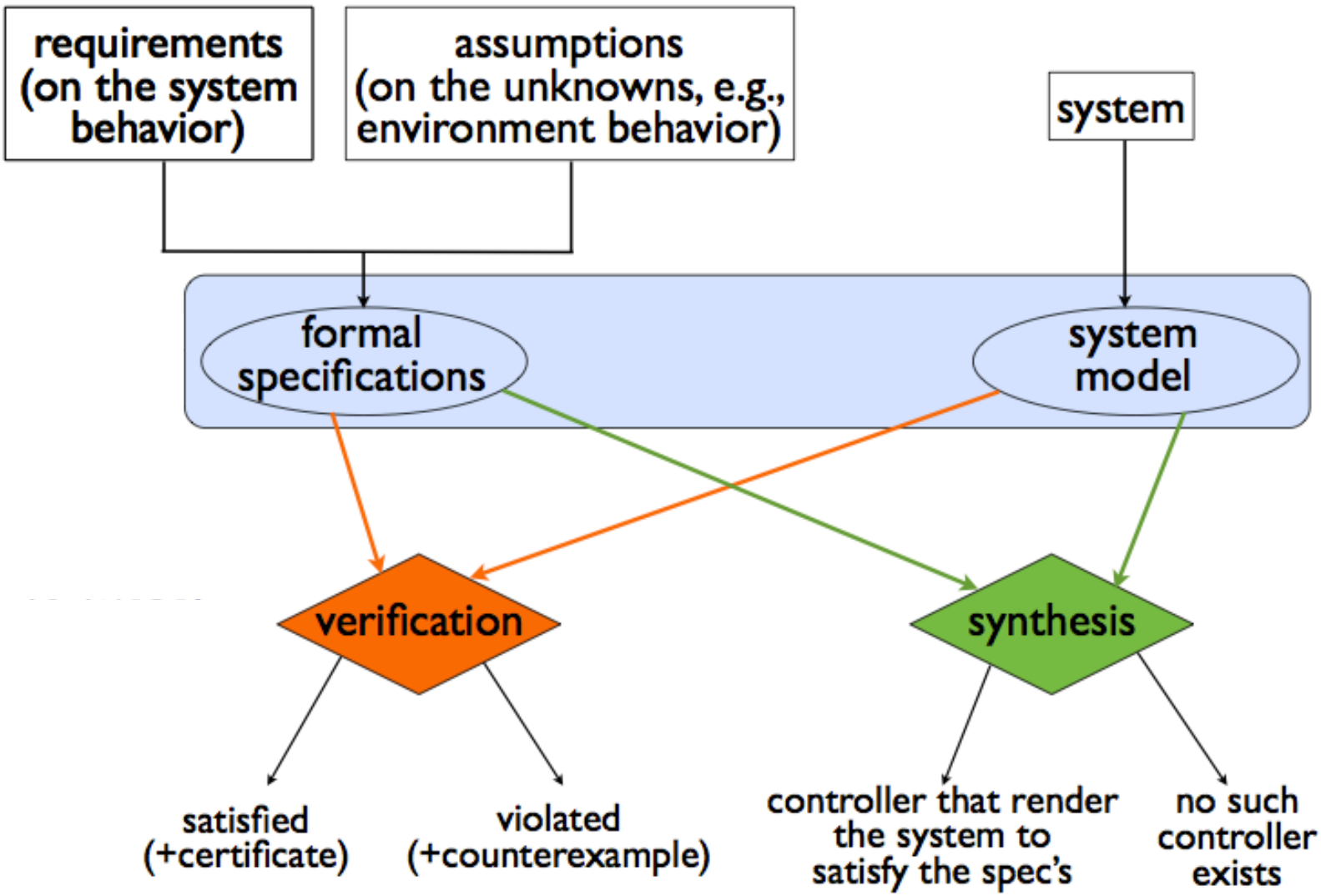
Verification using temporal logic (Lamport's TLC, TLA+)

- Model follower, Actuation Interface, DARPA, accModule, transModule in TLC
- Shared variables: *state, estop, acc, acc_command, trans, trans_command*

Model Checking for *Hybrid* Systems



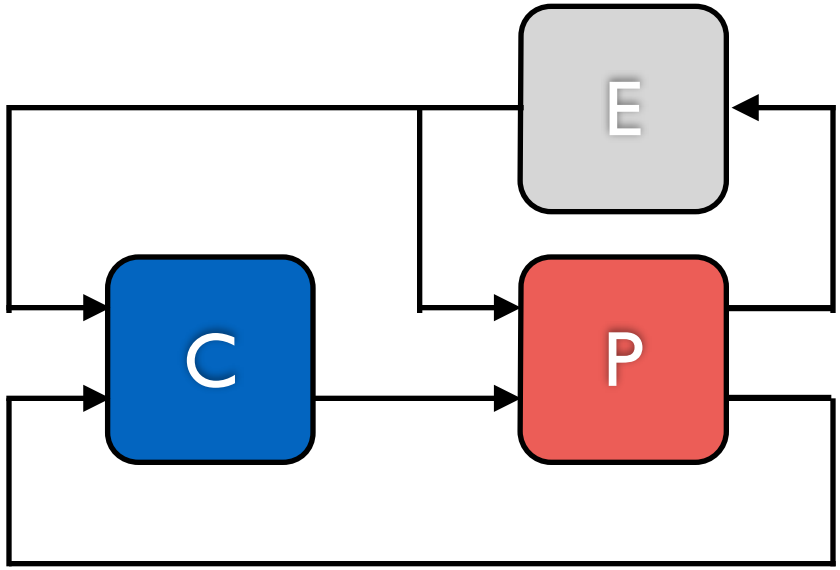
Formal Methods for System Verification & *Synthesis*



"Correct-by-Construction" Controller Synthesis

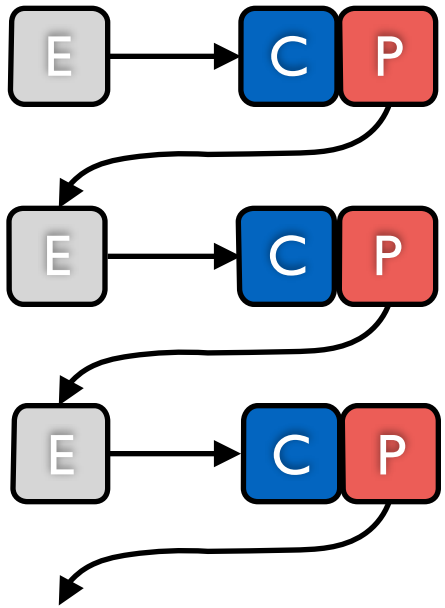
Reactive Protocol Synthesis

- 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 (!)



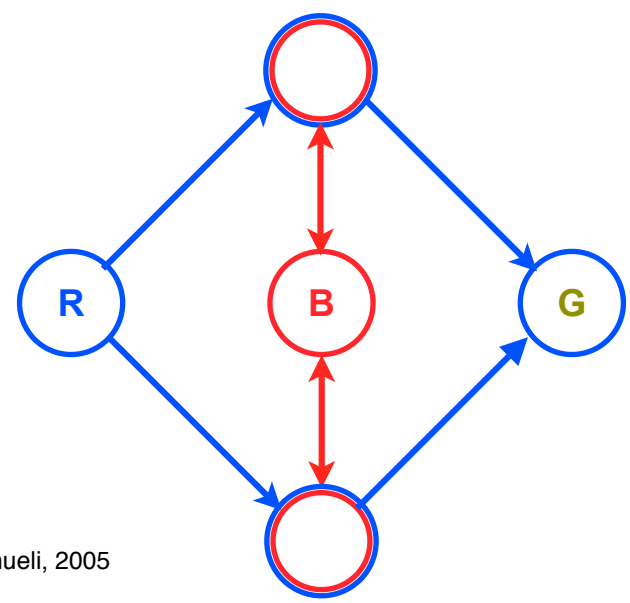
$$(\phi^{\mathrm{e}}_{\mathrm{init}} \wedge \Box\phi^{\mathrm{e}}_{\mathrm{safe}} \wedge \Box\Diamond\phi^{\mathrm{e}}_{\mathrm{prog}}) \rightarrow (\phi^{\mathrm{s}}_{\mathrm{init}} \wedge \Box\phi^{\mathrm{s}}_{\mathrm{safe}} \wedge \Box\Diamond\phi^{\mathrm{s}}_{\mathrm{prog}})$$

Environment assumption System guarantee

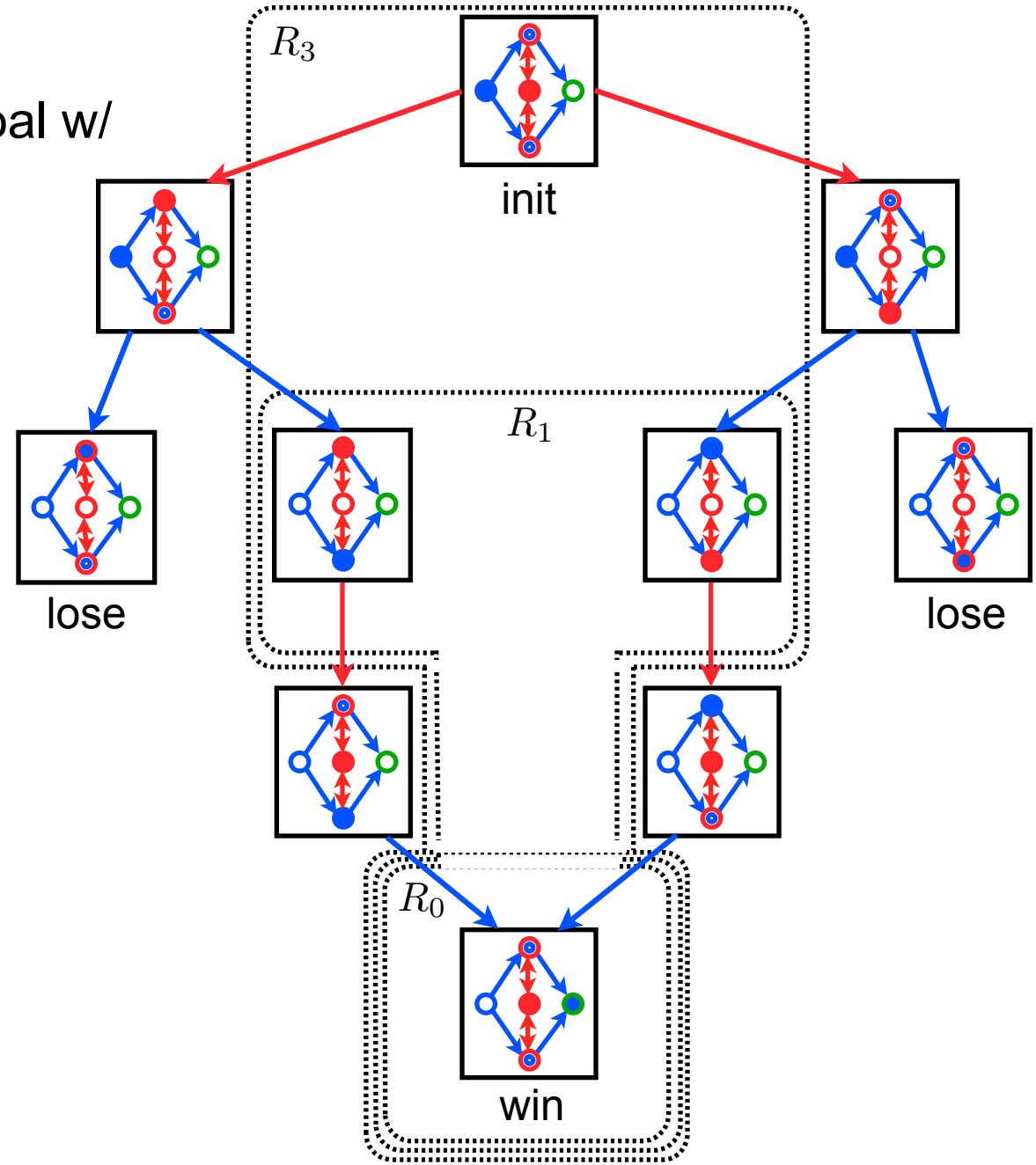
Example: Runner Blocker System

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 sequence of winning sets



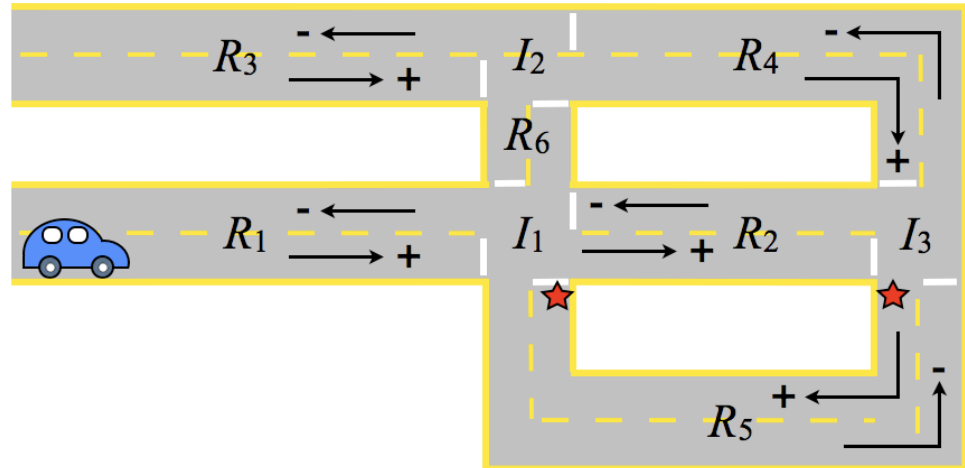
A. Pnueli, 2005



Example: Autonomous Navigation in Urban Environment

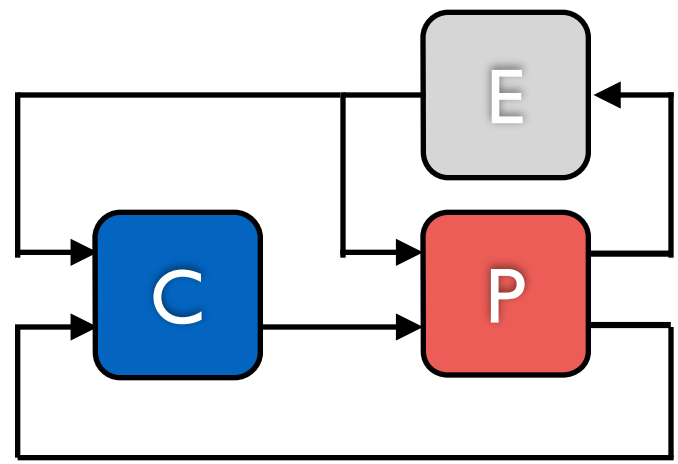
Traffic rules

- 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



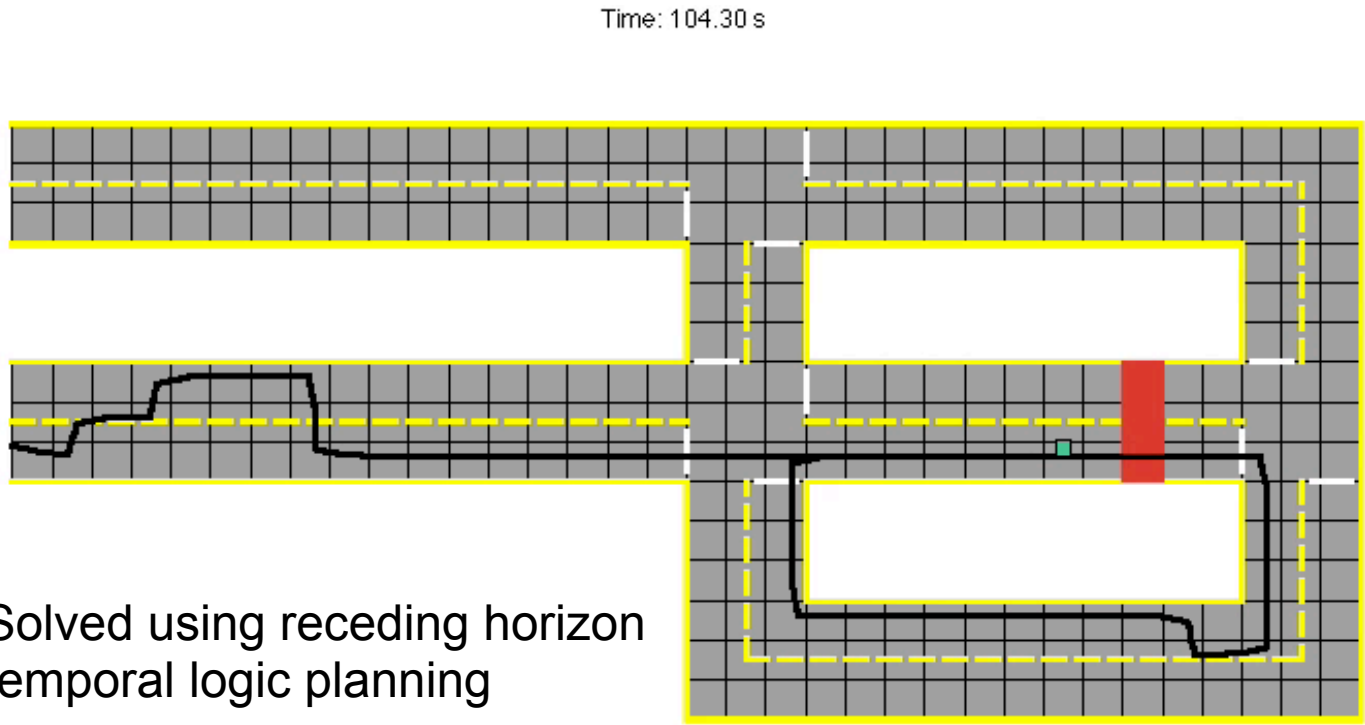
Assumptions

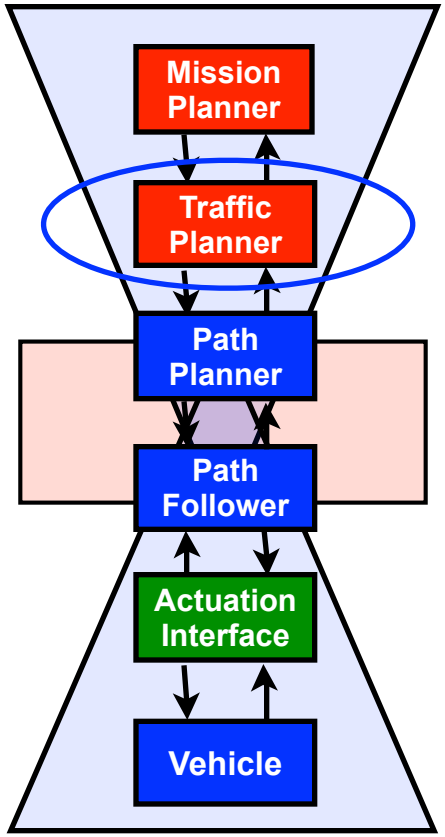
- Obstacle may not block a road
- Obstacle is detected before vehicle gets too close
- Limited sensing range
- Obstacle does not disappear when the vehicle is in its vicinity
- Obstacles may not span more than a certain number of consecutive cells in the middle of the road
- Each intersection is clear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



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

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

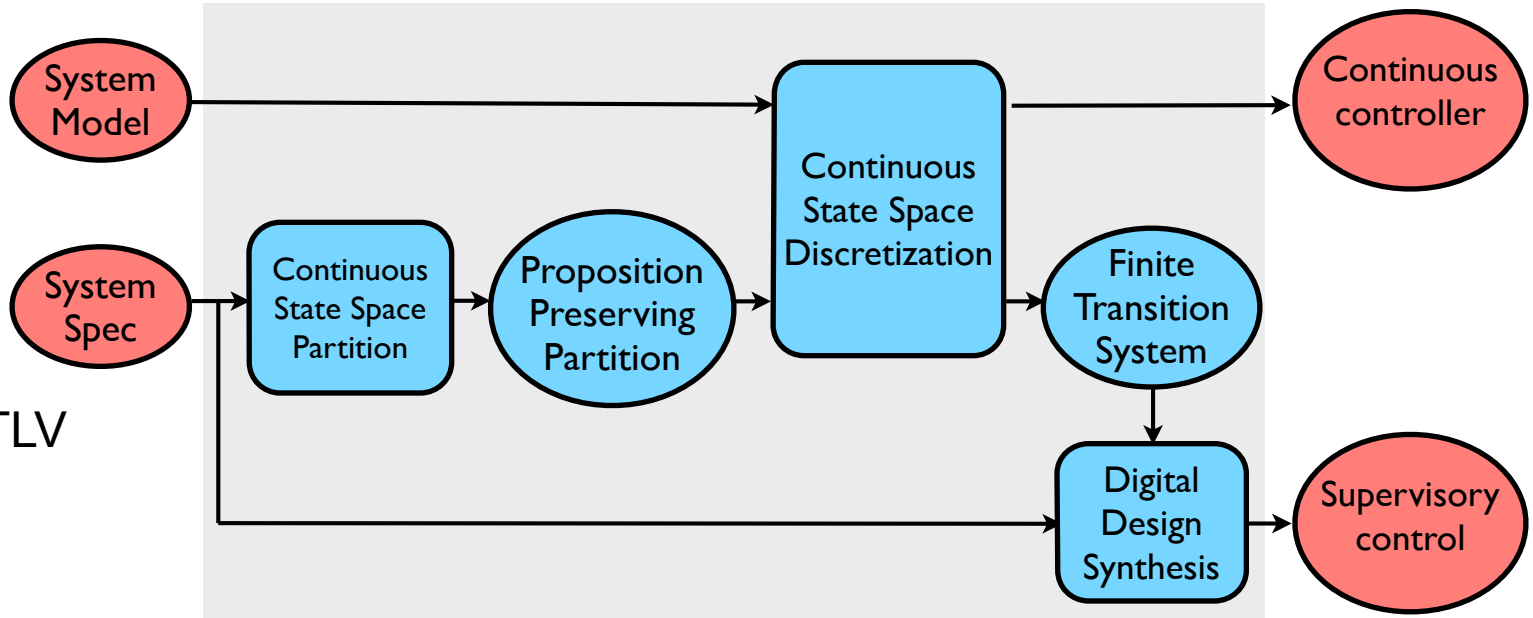
- Trajectory planner sees blockage and fails to find strategy satisfying specification
- Trajectory planner reports failure to goal generator
- Goal generator re-computes a (high level) path to the goal state

Temporal Logic Planning (TuLiP) toolbox

http://tulip-control.org

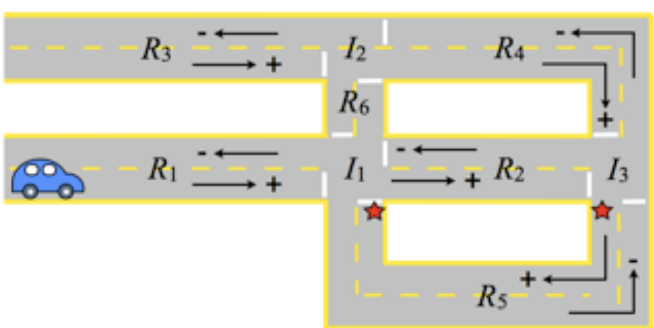
Python Toolbox

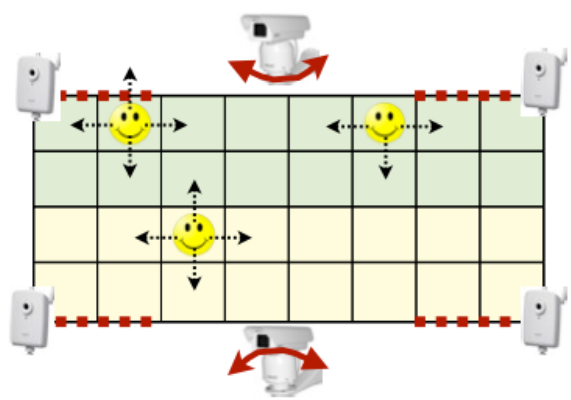
- GR(1), LTL specs
- Nonlin dynamics
- Supports discretization via MPT
- Control protocol designed using JTLV
- Receding horizon compatible

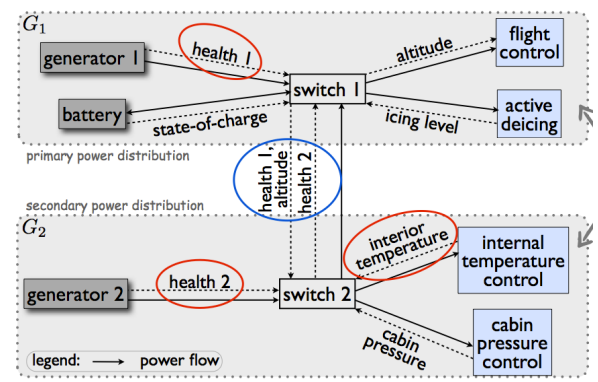


Applications of TuLiP

- 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







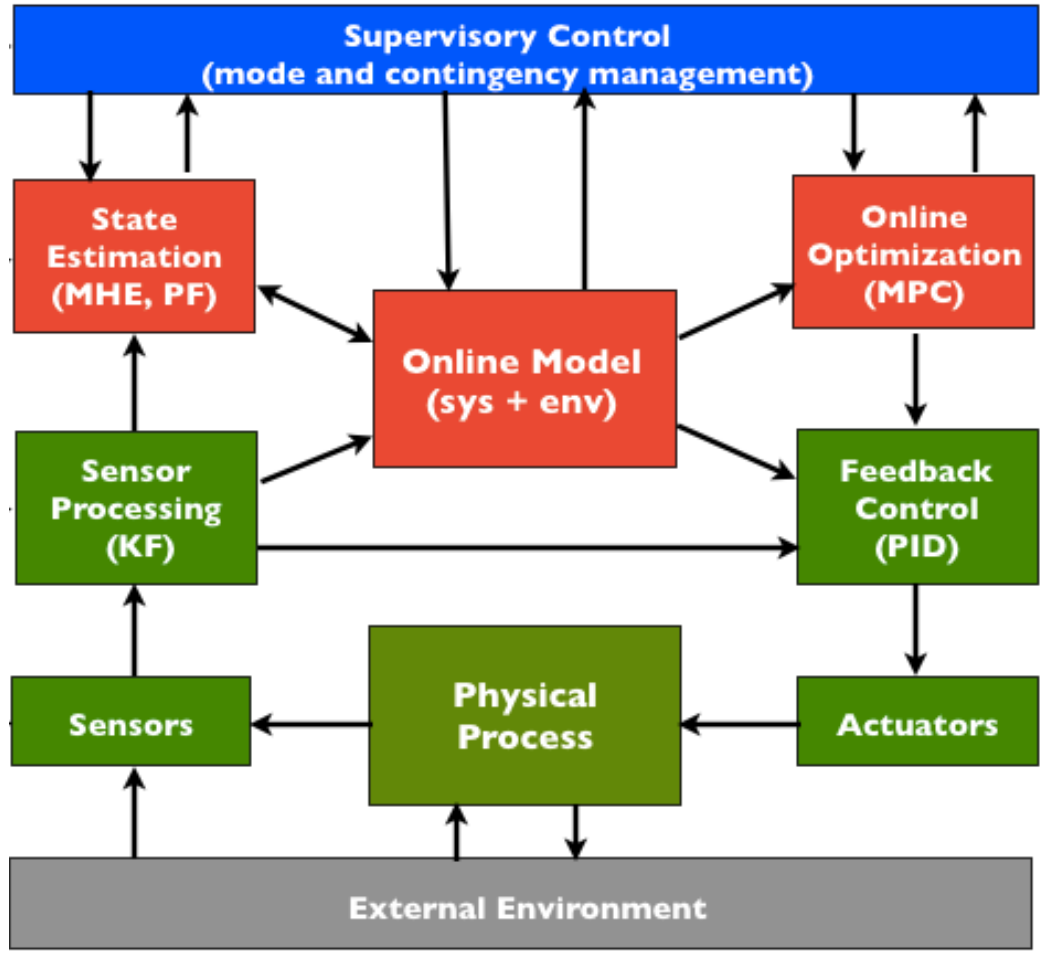
Summary and Future Research

Design of abstraction layers + interfaces

Synthesis of reactive protocols for STL specs

Uncertainty and robustness (for discrete representations)

Many other directions: scaling, incremental, probabilistic, performance metrics, …



Special thanks:

- Nok Wongpiromsarn
- Ufuk Topcu