Caltech

Computer Lab 2

TuLiP: A Software Toolbox for Temporal Logic Planning

Richard M. Murray
Caltech

Nok Wongpiromsarn
UT Austin/Iowa State

EECI-IGSC, 11 Mar 2020

Outline

- Overview of TuLiP
- Computer Lab

Problem Description

Problem: Given a plant model and an LTL specification φ, design a controller to ensure that any execution of the system satisfies φ

- The evolution of the system is described by differential/difference equations

$$\begin{aligned} s(t+1) &= As(t) + Bu(t) + Ed(t)) \\ u(t) &\in U \\ d(t) &\in D \end{aligned}$$

  where $s \in \mathbb{R}^n, U \subseteq \mathbb{R}^m, D \subseteq \mathbb{R}^p$

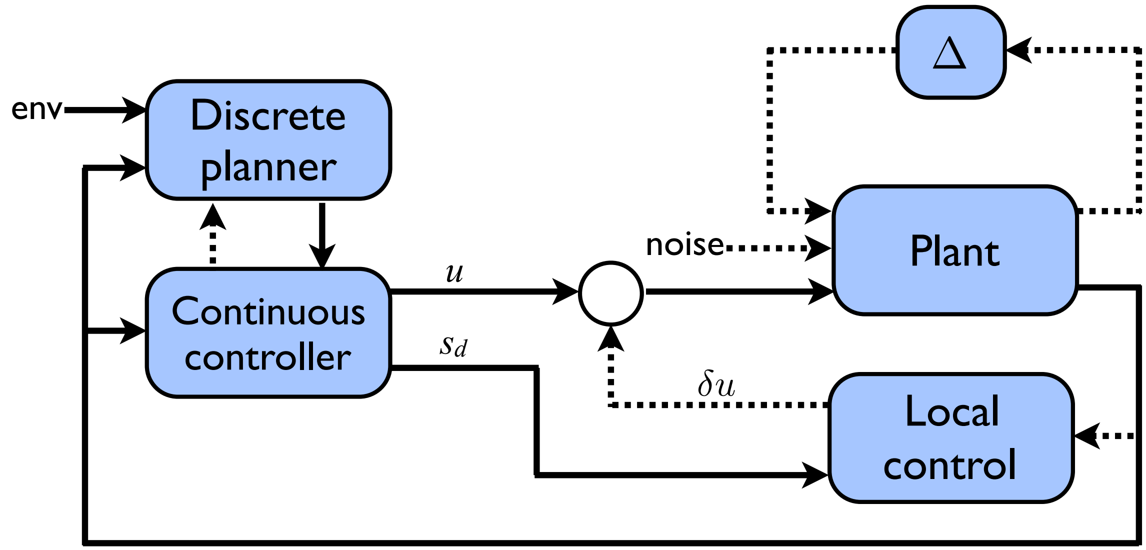
- φ must be satisfied regardless of the environment in which the system operates
- Assume that φ is of the form

$$\varphi = \Big(\underbrace{\psi^e_{init}}_{\substack{\text{assumptions on} \\ \text{initial condition}}} \wedge \underbrace{\Box\psi^e_s \wedge \bigwedge_{i \in I_f} \Box\Diamond\psi^e_{f,i}}_{\substack{\text{assumptions on} \\ \text{environment}}} \Big) \implies \underbrace{\Big(\psi^s_{init} \wedge \Box\psi^s_s \wedge \bigwedge_{i \in I_g} \Box\Diamond\psi^s_{g,i} \Big)}_{\substack{\text{desired} \\ \text{behavior}}}$$

Embedded Control Software Synthesis

Key elements to specify the problem

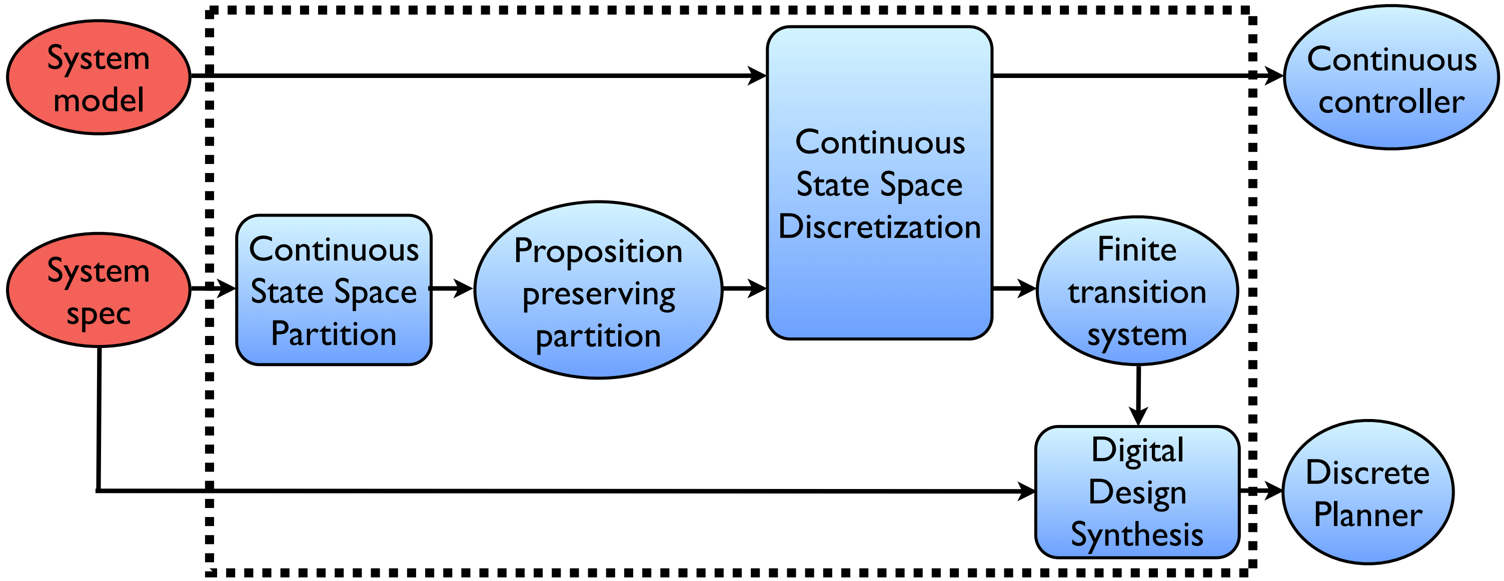
- discrete system state
- continuous system state
- (discrete) environment state
- specification



Hierarchical Approach

- Discrete planner computes the next cell to go to in order to satisfy φ
  - The synthesis algorithm considers all the possible behaviors of the environment
  - **Issue**: state explosion
- Continuous controller *simulates* the plan
  - Constrained optimal control problem
  - Continuous execution preserves the correctness of the plan

Main Steps



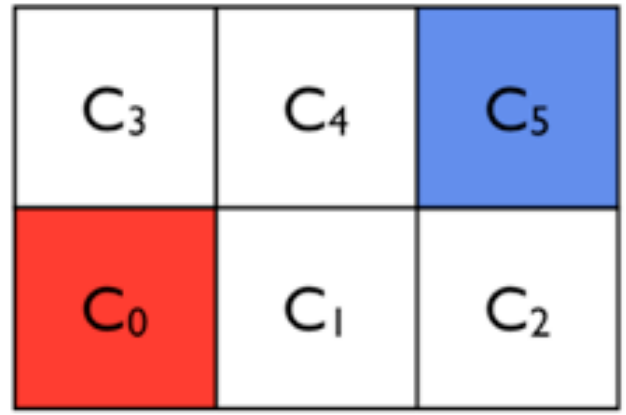
- Generate a proposition preserving partition of the continuous state space
  - cont_partition = tlp.abstract.**prop2part**(cont_state_space, cont_props)
- Discretize the continuous state space based on the evolution of the continuous state
  - disc_dynamics = tlp.abstract.**discretize**(cont_partition, sys_dyn, N=8, …)
- Digital design synthesis
  - specs = tlp.spec.**GRSpec**(env_vars, sys_vars, env_init, sys_init, env_safe, …)
  - ctrl = tlp.synth.**synthesize**(specs, sys=disc_dynamics.ts, ignore_sys_init=True)
- Simulate (not yet implemented; code manually for now)
  - tlp.abstract.simulate(sys, ctrl)

Example #1: robot_simple_discrete.py

System Model: Robot can move to the cells that share a face with the current cell

Desired Properties

- Visit the blue cell infinitely often
- Eventually go to the red cell when a PARK signal is received



Assumption

- Infinitely often, PARK signal is not received

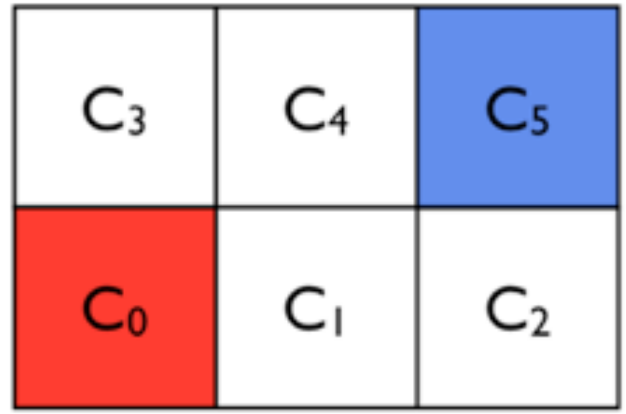
$$\varphi = \Box\Diamond(\neg park) \implies (\Box\Diamond(s \in C_5) \wedge \Box(park \implies \Diamond(s \in C_0)))$$

This spec is not a GR[1] formula

- Introduce an auxiliary variable $X0reach$ that starts with True
- $\Box(\bigcirc X0reach = (s \in C_0 \vee (X0reach \wedge \neg park)))$
- $\Box\Diamond X0reach$

Example #2: robot_simple_continuous.py

Dynamics $\dot{x} = u_x, \dot{y} = u_y \text{ where } u_x, u_y \in [-1, 1]$



Desired Properties

- Visit the blue cell infinitely often
- Eventually go to the red cell when a PARK signal is received

Assumption

- Infinitely often, PARK signal is not received

$$\varphi = \Box\Diamond(\neg park) \implies (\Box\Diamond(s \in C_5) \wedge \Box(park \implies \Diamond(s \in C_0)))$$

This spec is not a GR[1] formula

- Introduce an auxiliary variable *X0reach* that starts with True
- $\Box(\bigcirc X0reach = ((s \in C_0 \vee X0reach) \wedge \neg park))$
- $\Box\Diamond X0reach$

Computer Exercise 1

C_6	C_7	C_8
C_3	C_4	C_5
C_0	C_1	C_2

Synthesize a reactive planner with the following specifications

System variables: X0,...,X8 -- Xi = 1 if robot in Ci, Xi = 0 otherwise.
Environment variables: obs ∈ {1,4,7}, park ∈ {0,1}

Desired Properties

- Visit the blue cell (C_8) infinitely often
- Eventually go to the green cell (C_0) after a PARK signal is received
- Avoid an obstacle (red cell) which can be one of the C_1, C_4, C_7 cells and can move arbitrarily

[]<>X8

X0reach ∧
[]<>X0reach ∧
[](next(X0reach)=((X0∨X0reach)∧ !park))

[](obs=0 → !X1)∧ [](obs=1 → !X4)∧
[](obs=2 → !X7)

Assumption

- Infinitely often, PARK signal is not received
- The obstacle always moves to an adjacent cell

[]<>!park

[](obs=0 → next(obs)=1) ∧
[](obs=1 → (next(obs)=0 ∨ next(obs)=2))∧
[](obs=2 → next(obs)=1)

Constraints (or discrete dynamics)

- The robot can only move to an adjacent cell, i.e., a cell that shares an edge with the current cell

Computer Exercise 2

Synthesize intersection logic for the car with the following specification:

Desired Properties

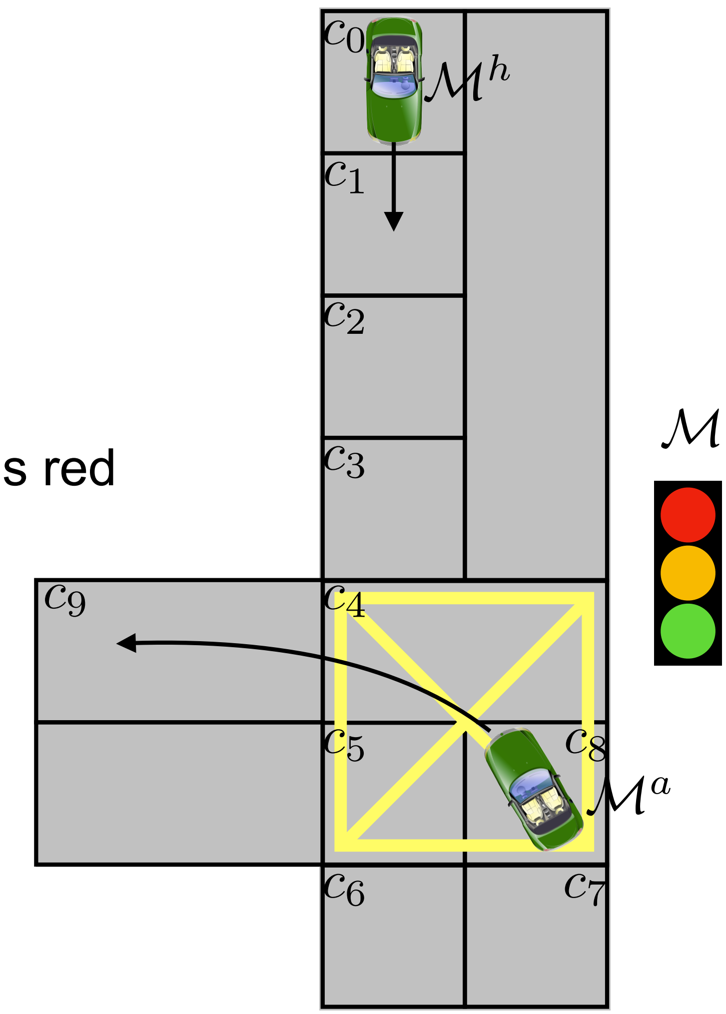
- Vehicle *a* should eventually go to C9
- Vehicle *a* does not collide with vehicle *h*
- Vehicle *h* is not in the intersection when the light is red

Assumptions

- find a set of "non-trivial" assumptions that render the problem realizable
- allow human vehicle to start in any location

Constraints

- Vehicles can only move to cells representing their possible travel lanes



Validation Possibilities

Add continuous-time dynamics to intersection problem

Synthesize decision-making logic for road network

Automated valet parking