Metric Temporal Logic Falsification and Path Planning for Robotic Systems

Bardh Hoxha and Georgios Fainekos

Arizona State University

CPS Lab

Sponsors:

CENTER FOR EMBEDDED SYSTEMS an NSF Industry/University Cooperative Research Center

Problem Formulation

Model: A Cyber-Physical System Σ which can be viewed as a mapping from a set of initial conditions X_0 and input signals $U \subseteq U^R$ to output signals Y^R. Here R is a bounded time domain equipped with a metric d_R, U is a compact set of possible input values at each point in time(input space) and Y is the set of output values(output space)

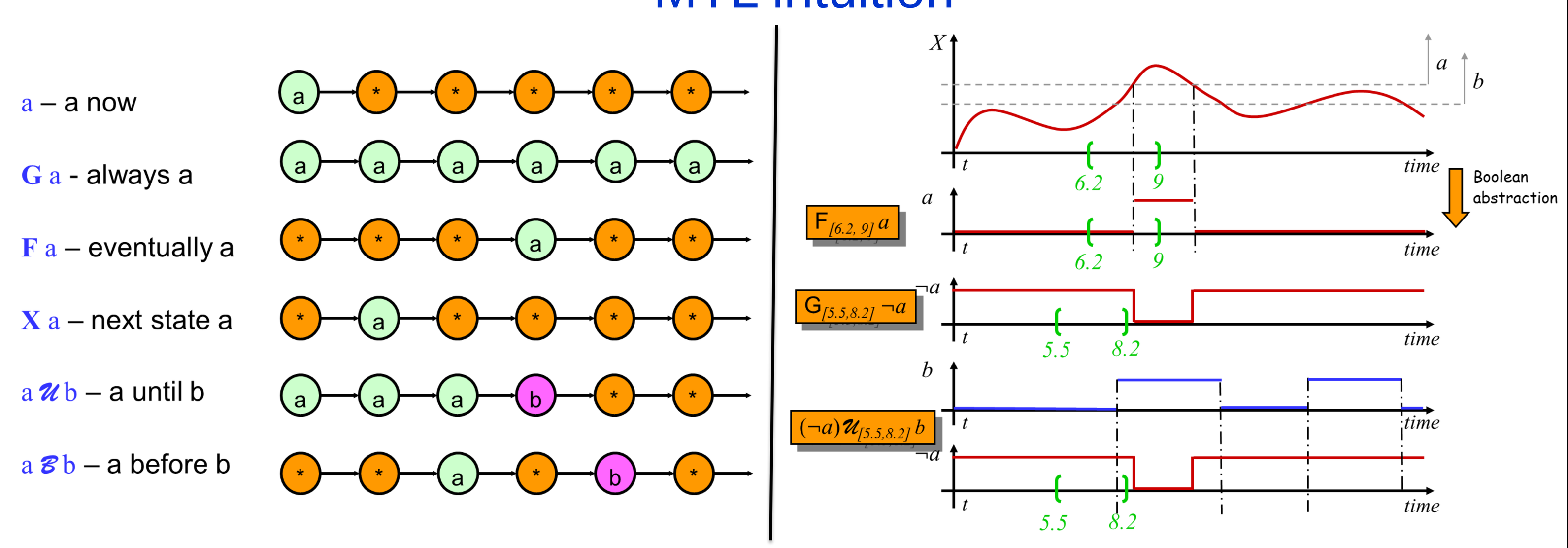
Specification: A metric temporal logic (MTL) formula ϕ that captures the systems' desired behavior.

Problem: For an MTL specification ϕ, the MTL falsification problem consists of finding an output signal y of the system Σ starting from some valid initial state $x_0 \in X_0$ under an input signal $u \in U$ such that y does not satisfy specification ϕ.

MTL specifications

The formulas are built from a finite number of atomic propositions Π which label areas of interest in the environment such as *rooms* or *obstacles*. The propositional formulas are formed using the traditional operators of *conjunction* (∧), *disjunction* (∨), *negation* (¬), *implication* (⇒), and *equivalence* (⇔). MTL formulas are obtained from the standard propositional logic by adding temporal operators such as *eventually* (◊), *always* (□), and *until* (U). MTL also allows timing constraints.

MTL intuition



Discrete-time Robust Semantics for MTL

timed trace μ = (σ,τ)

$$[\![c]\!]_D(\mu,i) := c$$
$$[\![p]\!]_D(\mu,i) := \mathbf{Dist}_d(\sigma(i), \mathcal{O}(p))$$
$$[\![\neg\phi_1]\!]_D(\mu,i) := -[\![\phi_1]\!]_D(\mu,i)$$
$$[\![\phi_1 \vee \phi_2]\!]_D(\mu,i) := [\![\phi_1]\!]_D(\mu,i) \sqcup [\![\phi_2]\!]_D(\mu,i)$$
$$[\![\phi_1 \mathcal{U}_\mathcal{I} \phi_2]\!]_D(\mu,i) := \bigsqcup_{j \in \tau^{-1}(\tau(i)+_R \mathcal{I})} \left([\![\phi_2]\!]_D(\mu,j) \sqcap \bigsqcap_{i \le k < j} [\![\phi_1]\!]_D(\mu,k) \right)$$

Algorithm I
- Based on formula re-writing
- Suitable for runtime monitoring algorithms
- Details on:

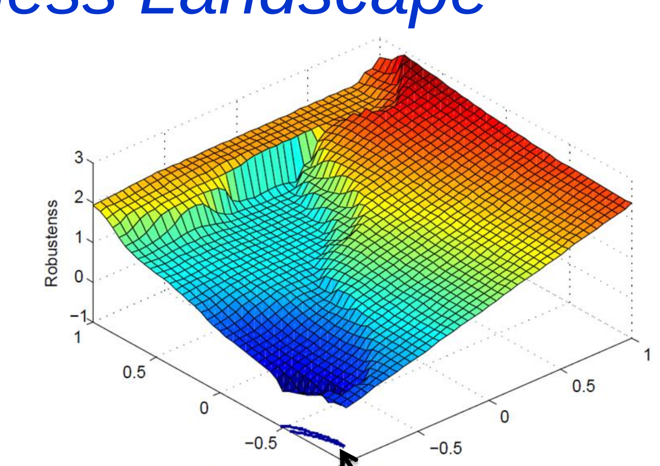
Fainekos and Pappas. "Robustness of temporal logic specifications for continuous-time signals." *Theoretical Computer Science* 410.42 (2009): 4262-4291.

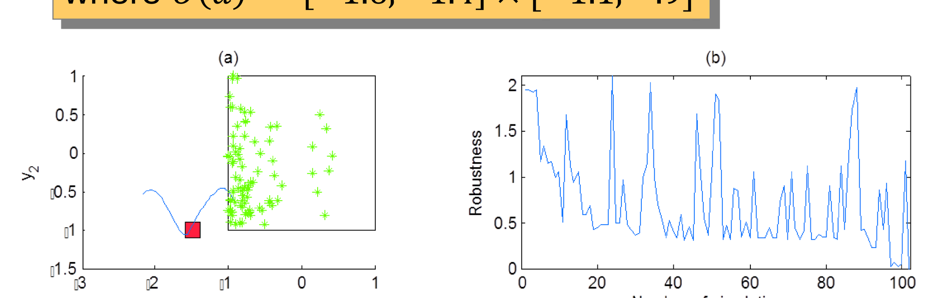
Algorithm II
- Based on dynamic programming
- Suitable for offline testing
- MTL formulas: O(|φ| |τ| c), where c = max $_{0\le j\le|\tau|,\ I\in T(\varphi)}$ |[j, max J(j, I)]|
- Details on:

Fainekos et al. "Verification of automotive control applications using s-taliro." *American Control Conference (ACC), 2012*. IEEE, 2012.

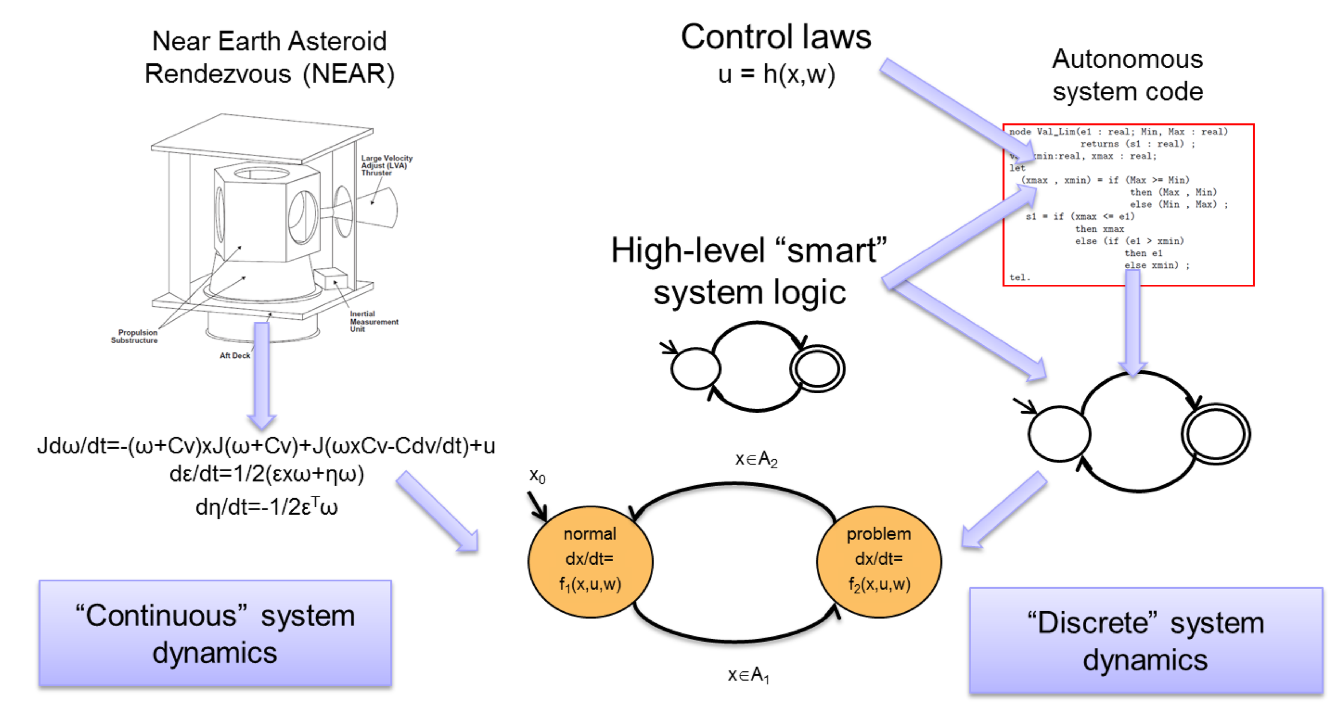
Example of Robustness Landscape

System:
$$\frac{dx}{dt} = x - y + 0.1t$$
$$\frac{dx}{dt} = y\cos(2\pi y) - x\sin(2\pi x) + 0.1t$$
Initial conditions:
$$[-1,1] \times [-1,1]$$
Specification:
$$G_{[0,2]} \neg a$$
where $O(a) = [-1.6,-1.4] \times [-1.1,-.9]$





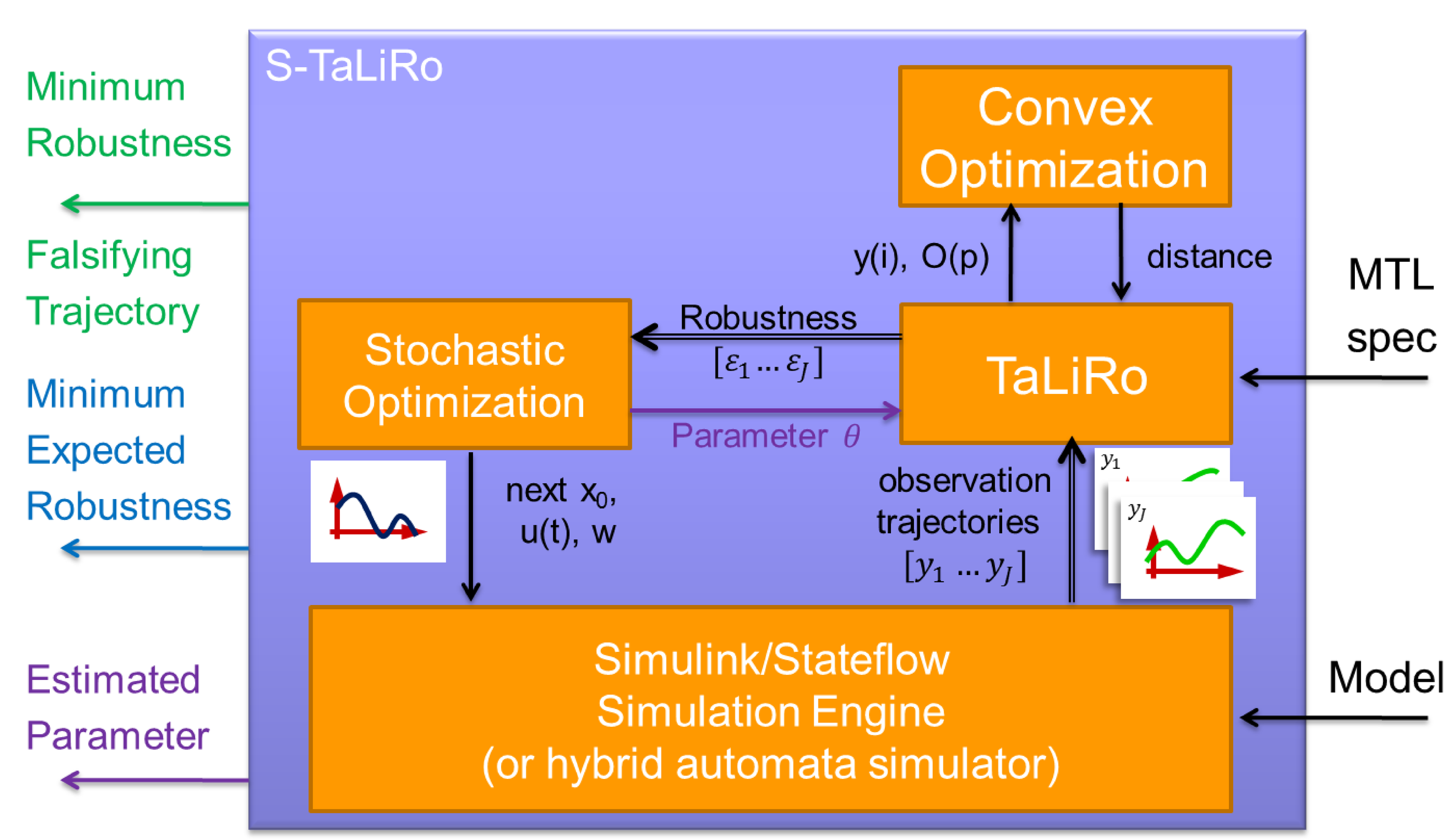
Why is the problem challenging?



Recent examples of Automotive Recalls due to CPS Errors (2011-2012)

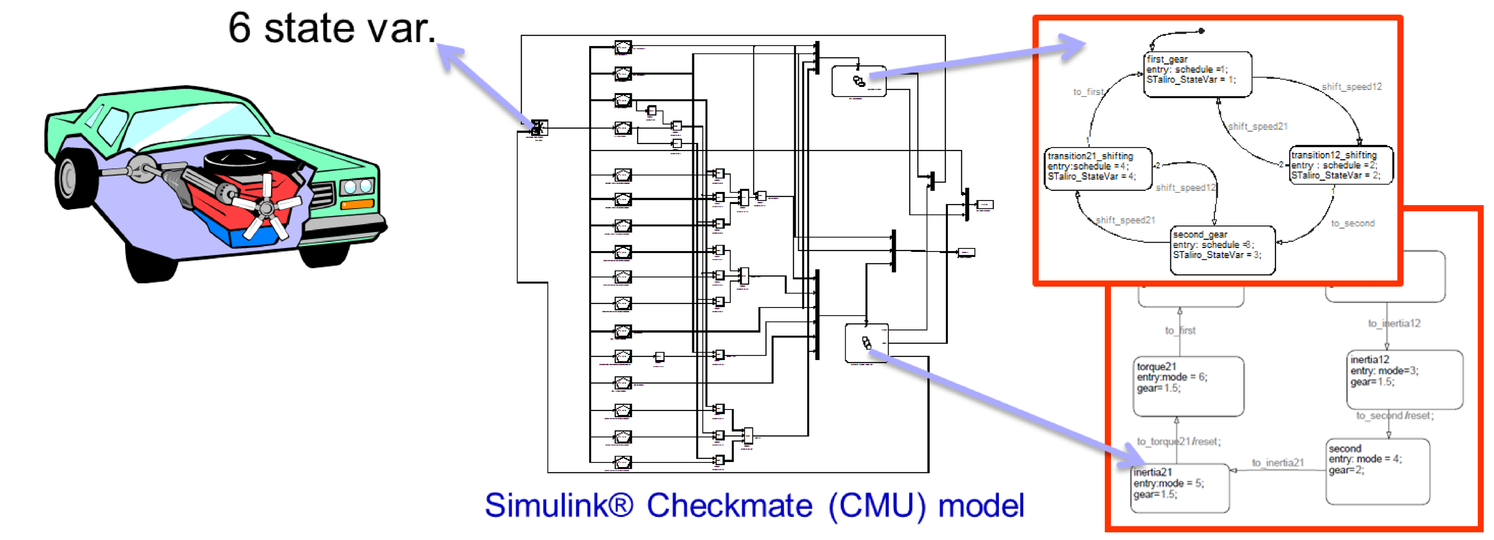
- No downshifting from 5th to 4th under certain operating conditions
- Rough idling or stalling due to complicated adaptive ECU
- Electric motor to rotate in the direction opposite to that selected by the transmission
- Cruise control does not disengage unless turning off the ignition

Solution: S-TaLiRo Toolbox



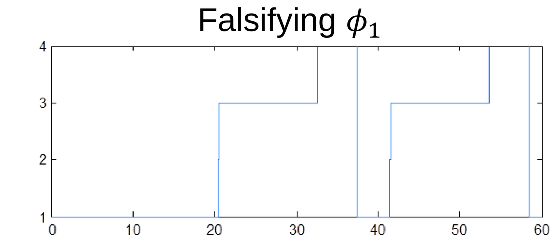
Three main features:
- Finding Falsifying Trajectories for Deterministic Cyber Physical Systems
- Minimum Expected Robustness for Stochastic Cyber Physical Systems
- Parameter Estimation of MTL formulas for Cyber Physical Systems

Powertrain Problem



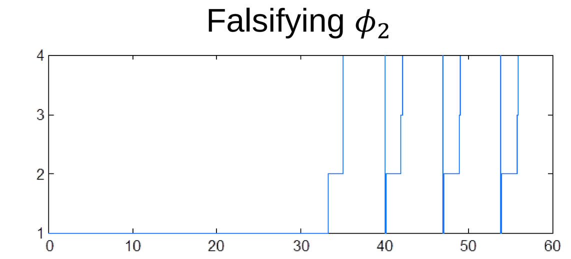
Formalizing Specification ϕ_1:
Find values for the initial parameters such that starting from 0 speed, the gear transitions from second to first to second.
$\phi_1 = \neg F(\text{gear}_2 \wedge F(\text{gear}_1 \wedge F\text{gear}_2))$

Falsifying ϕ_1



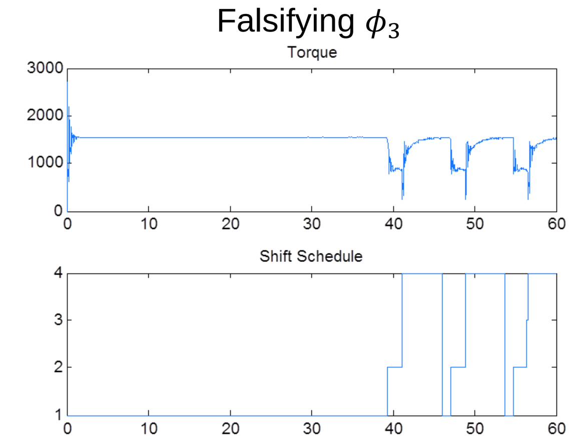
Formalizing Specification ϕ_2:
A "useful" property is to find constrain the gear change from second to first to second not happen within 2.5 sec.
$\phi_2 = G((\neg\text{gear}_1 \wedge X\ \text{gear}_1) \rightarrow G_{[0,2.5]}\ \neg\text{gear}_2)$

Falsifying ϕ_2

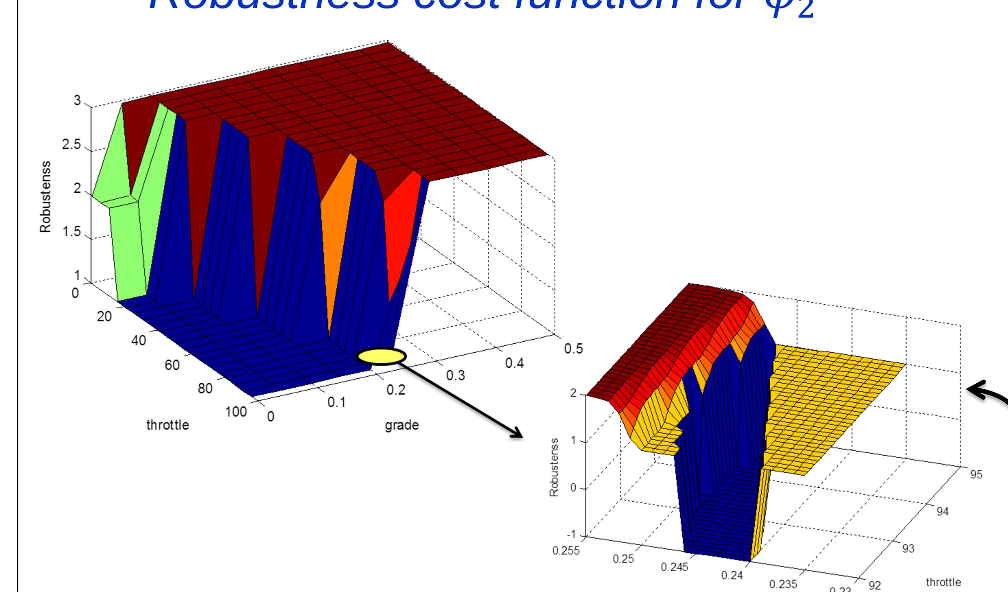


Formalizing Specification ϕ_3:
Verify that the jitter is within acceptable limits.
$\phi_3 = G(\text{gear}_{21} \rightarrow b)$

Falsifying ϕ_3



Robustness cost function for ϕ_2



Quadrotor Path Planning

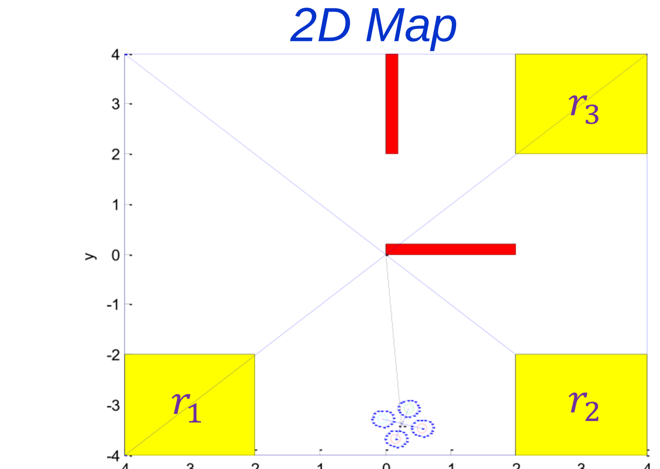
Model: Quadrotor Model from the Robotics Toolbox[1]

2D and 3D Path Planning

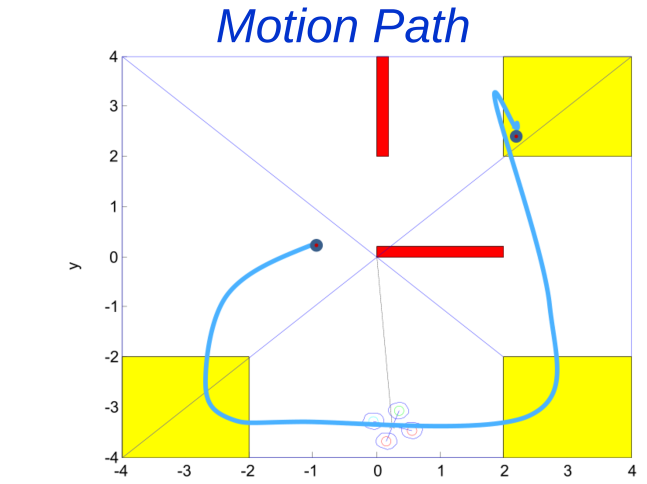
Specification for 2D Path Planning: Visit the yellow marked regions r_1, r_2, r_3 while avoiding the red marked regions r_5, r_6

$$\phi_{2D} = F_{[0,10]}(r_1 \wedge r_2 \wedge r_3) \wedge G\neg(r_5 \vee r_6)$$

2D Map



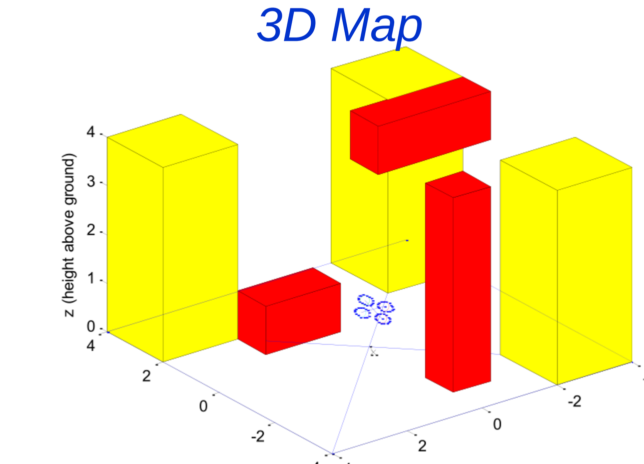
Motion Path



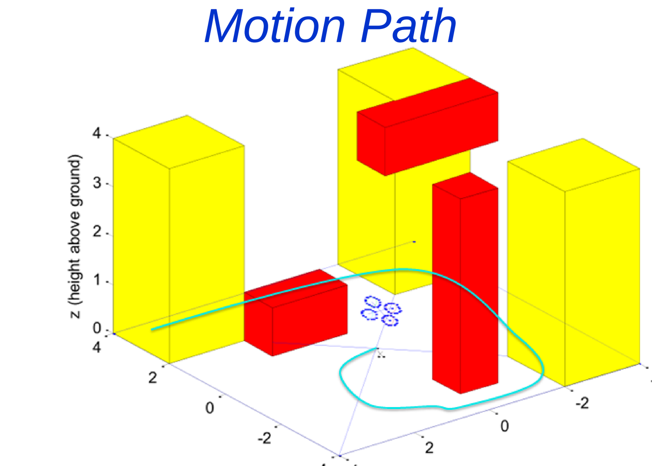
Specification for 3D Path Planning: Visit the yellow marked regions r_1, r_2, r_3 while avoiding the red marked regions r_5, r_6

$$\phi_{3D} = F_{[0,10]}(r_1 \wedge r_2 \wedge r_3) \wedge G\neg(r_5 \vee r_6 \vee r_7)$$

3D Map



Motion Path



[1] - P.I. Corke, "Robotics, Vision & Control", Springer 2011, ISBN 978-3-642-20143-1