Synthesis of Embedded Control Software

Ufuk Topcu
Caltech, Control and Dynamical Systems

Papers, slides, notes, software tools at
www.cds.caltech.edu/~UTopcu

CMACS, CMU, Fall 2010

Synthesis of Embedded Control Software

Joint work with
N. Wongpiromsarn, N. Ozay, and R. Murray
(MIT, Singapore) (Caltech) (Caltech)

Outline

- Setup
- Receding horizon temporal logic synthesis
- Vehicle management systems
- Distributed synthesis

How to automatically design control protocols, that...

Handle mixture of discrete and continuous decision-making

Account for both high-level specs and low-level dynamics

Ensure proper response to external events in real-time,

... with "correctness certificates"?

How to "automatically" design control protocols that...

- Handle mixture of discrete and continuous decision-making
- Account for both high-level specs and low-level dynamics
- Ensure proper response to external events in real-time

Autonomous driving

Vehicle management

federated ⟶ IMA

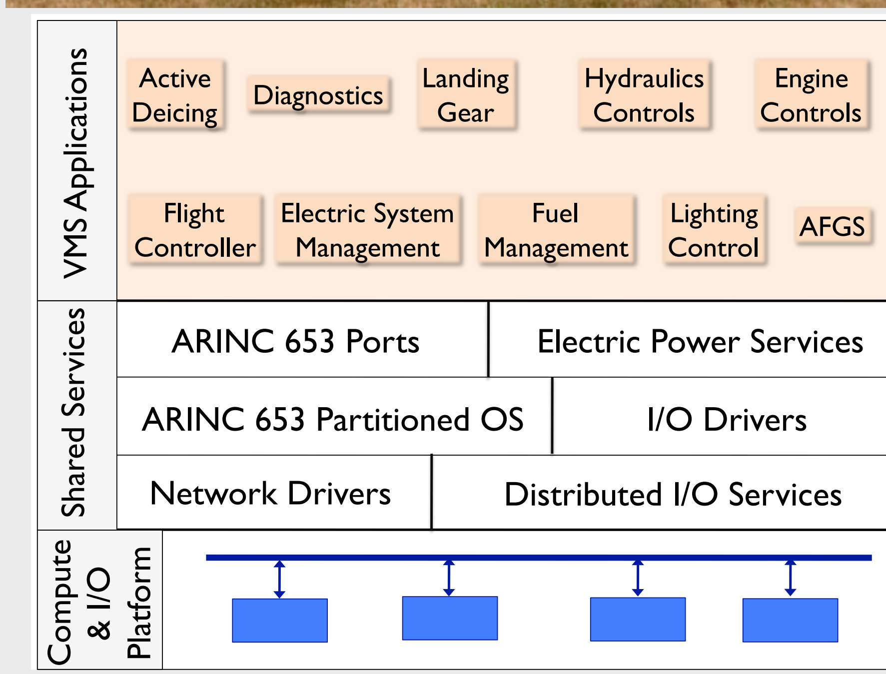
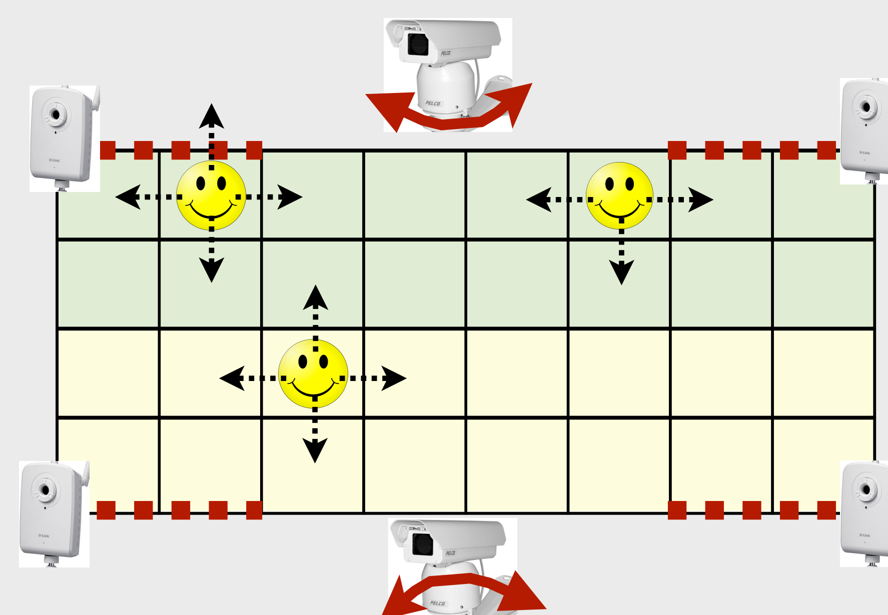
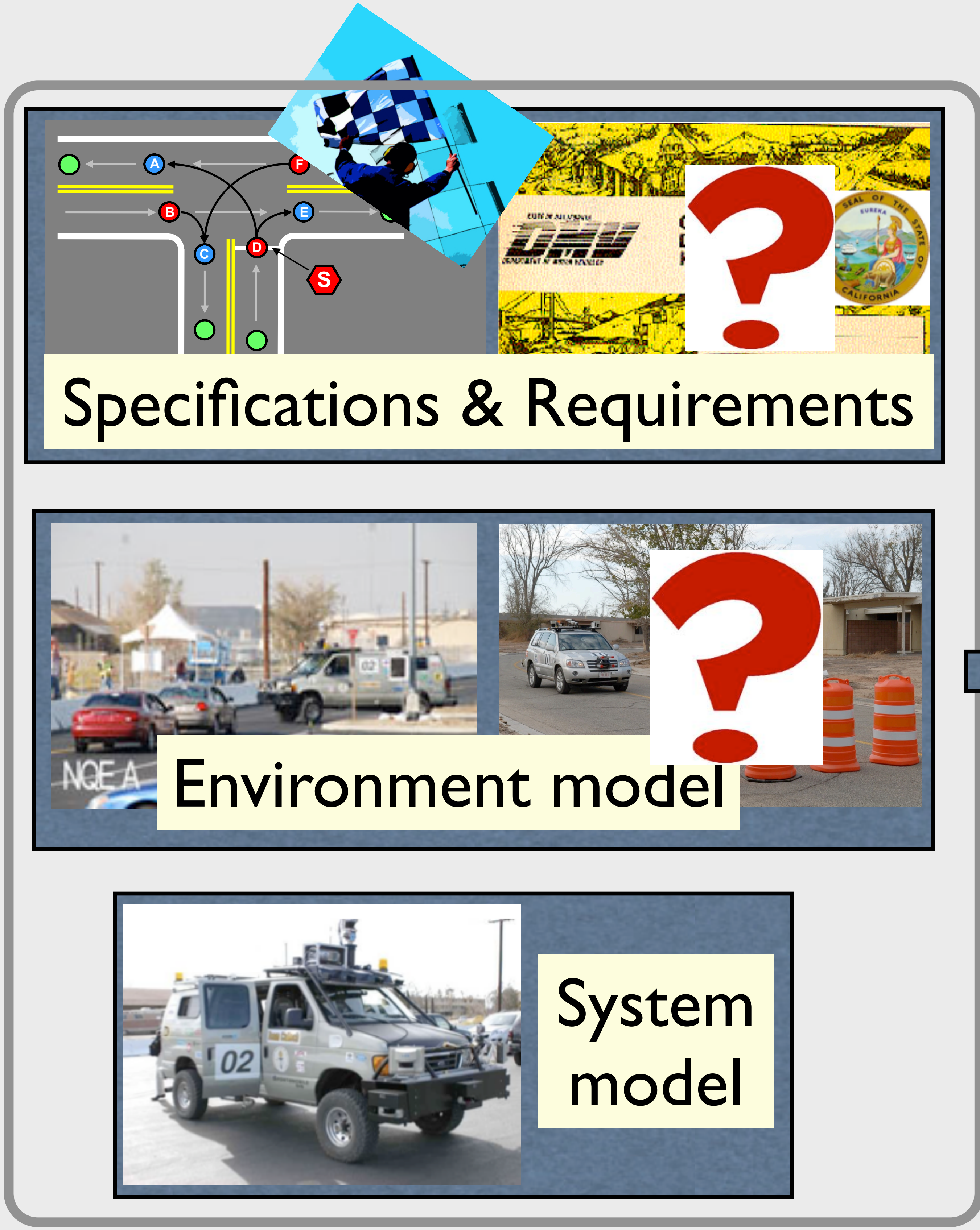


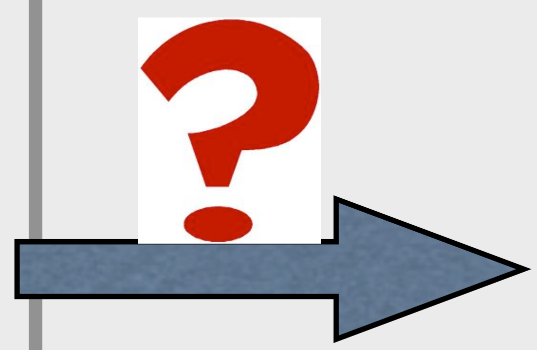
Figure – regenerated from a similar figure by W. P. Kinahan, Sikorsky Aircraft

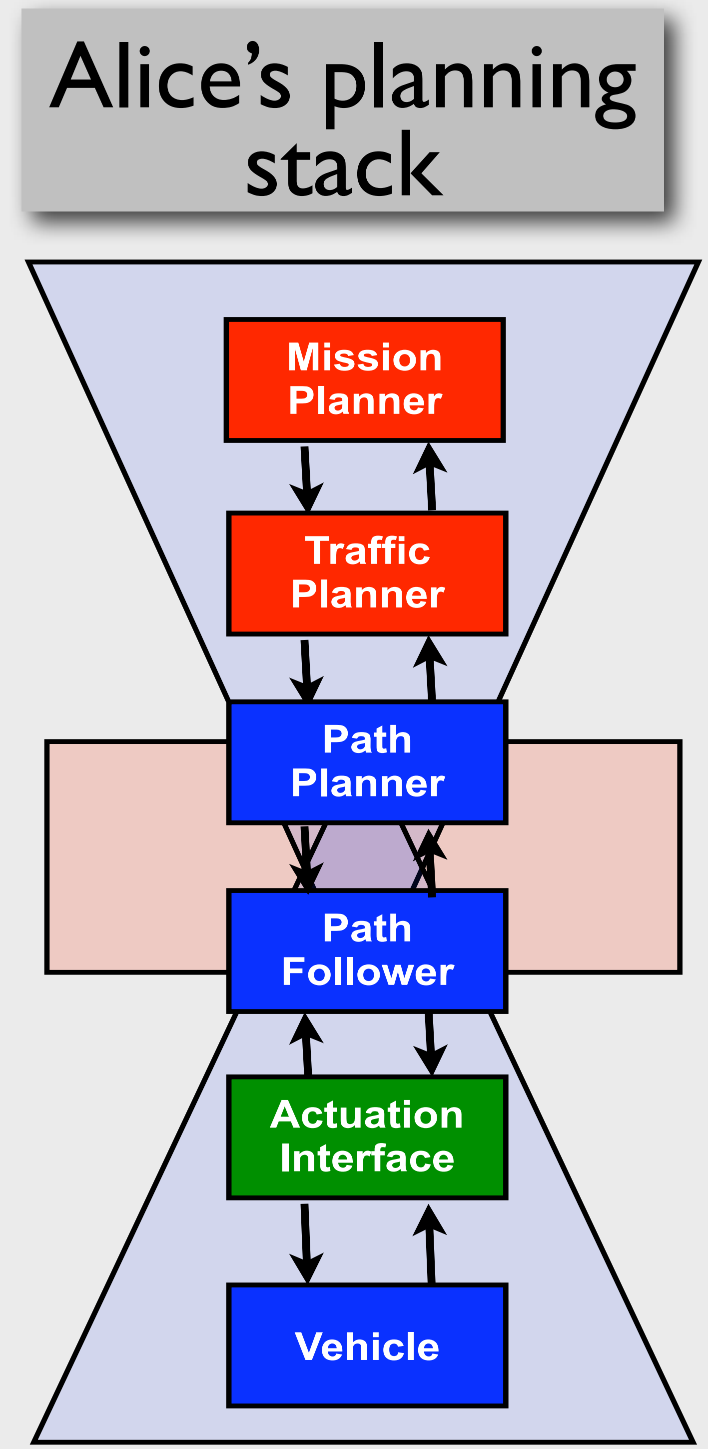
Active surveillance



Inputs & Outputs







Specifying behavior with linear temporal logic (LTL)

Extends propositional logic with temporal operators

$\wedge$ (and), $\vee$ (or), $\rightarrow$ (implies), ! (not), **+** $\diamond$ (eventually), $\square$ (always), $\mathcal{U}$ (until).

- Allows to reason about infinite sequences of states
  - -state: snapshot of values of all variables (environment+system)
- Specifications (formulas) describe sets of allowable behavior
  - -safety specs: what actions are allowed
  - -fairness: when an action can be taken (e.g., infinitely often)
- No strict notion of time. Just ordering of events.

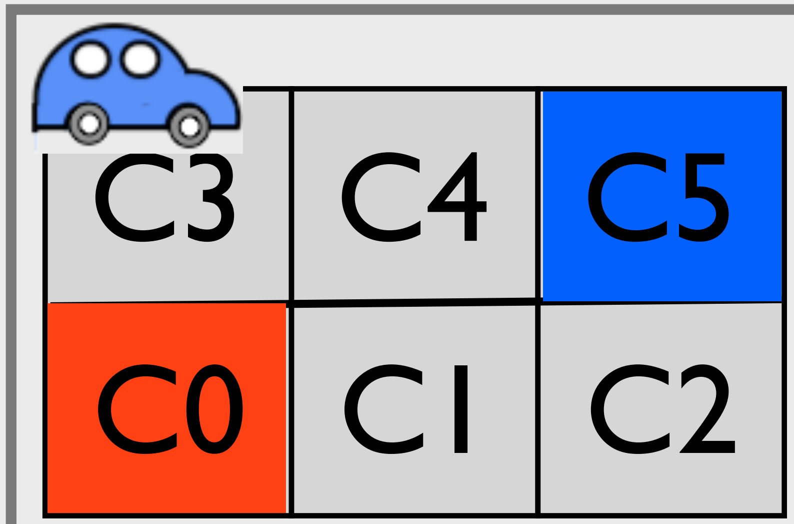
Compose to specify interesting behavior

$p \rightarrow \diamond q \equiv p$ implies eventually q (~ response)

$\square \diamond p \equiv$ always eventually p (~ progress)

$\diamond \square p \equiv$ eventually always p (~ stability)

$p \rightarrow q \mathcal{U} r \equiv p$ implies q until r (~ precedence)



Desired properties:

- Visit C5 infinitely often.
- Whenever a park signal is received go to C0.

Environment assumption:

- Park signal is not received infinitely often.

$$\square \diamond (\,!\text{park}\,) \rightarrow \{ \square \diamond (s \in C5) \;\wedge\; \square (\,\text{park} \rightarrow \diamond (s \in C0)\,)\, \}$$

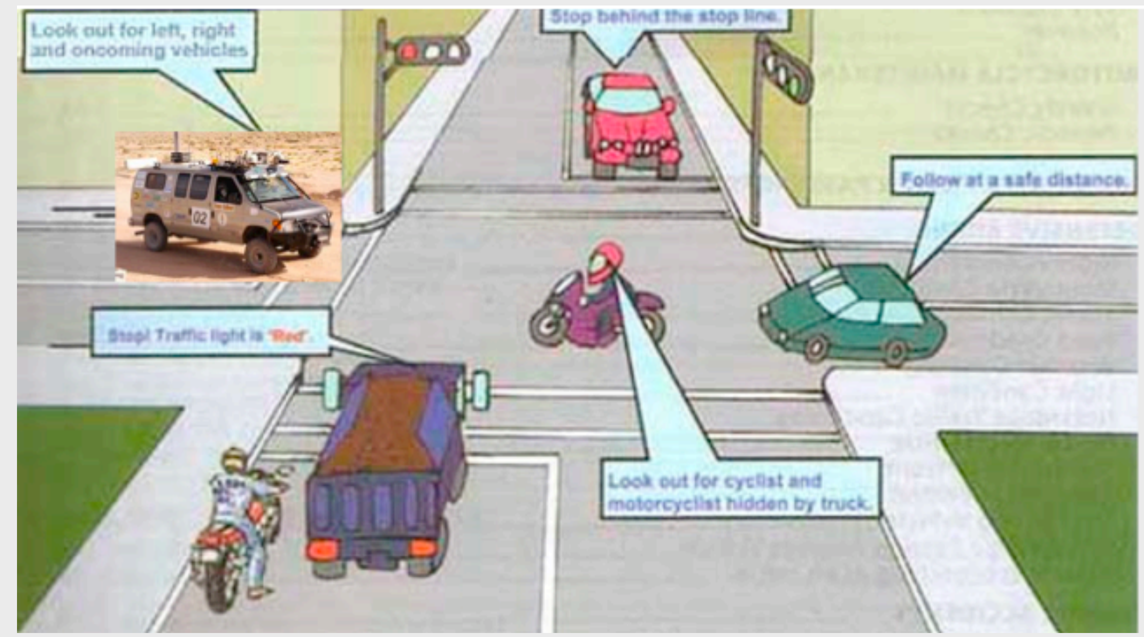
Sample Specifications

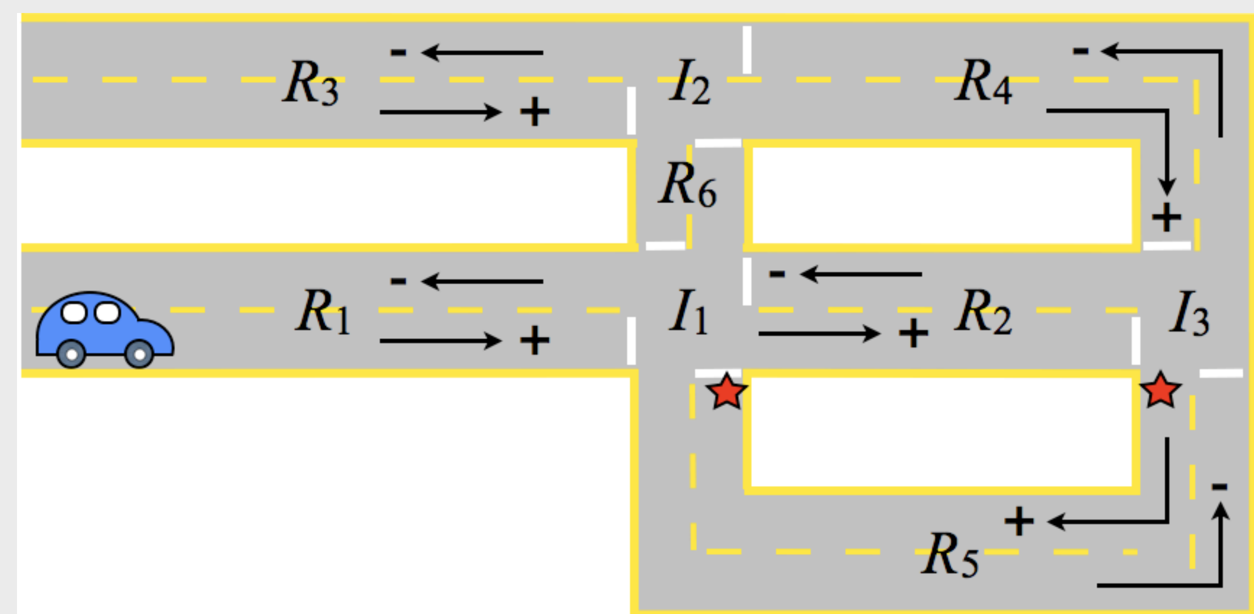
Traffic rules:

- No collision
- Stay in travel lane unless blocked
- Go through an intersection only when it is clear

Environment Assumptions:

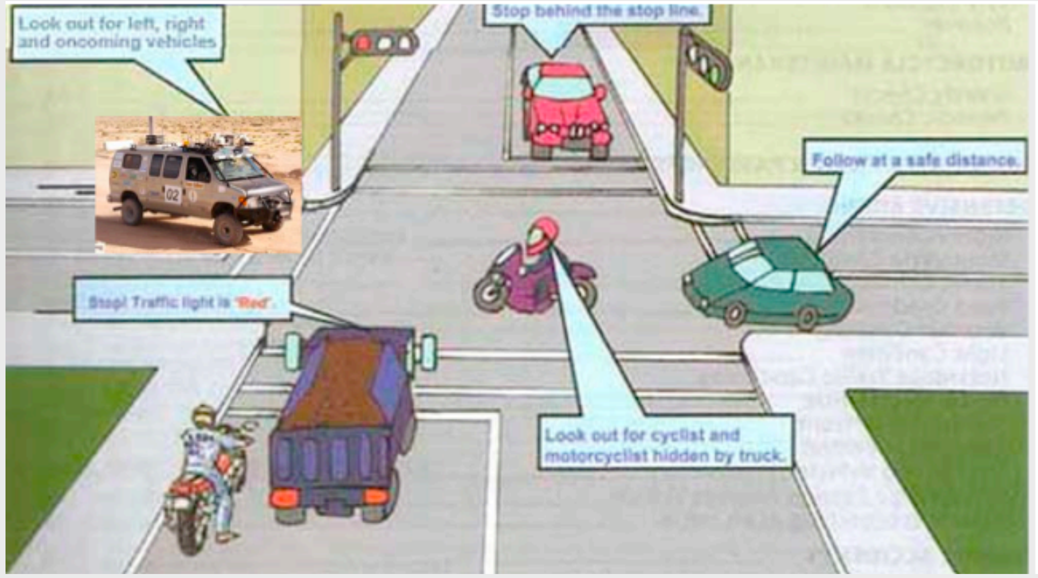
- No road blockage
- Limited sensing range
- Detect obstacles before too late
- Obstacles close to the car do not disappear
- Each intersection is clear infinitely often
- Vicinity of ★'s is obstacle-free infinitely often





Goals: Go through ★'s infinitely often

Temporal Logic Planning

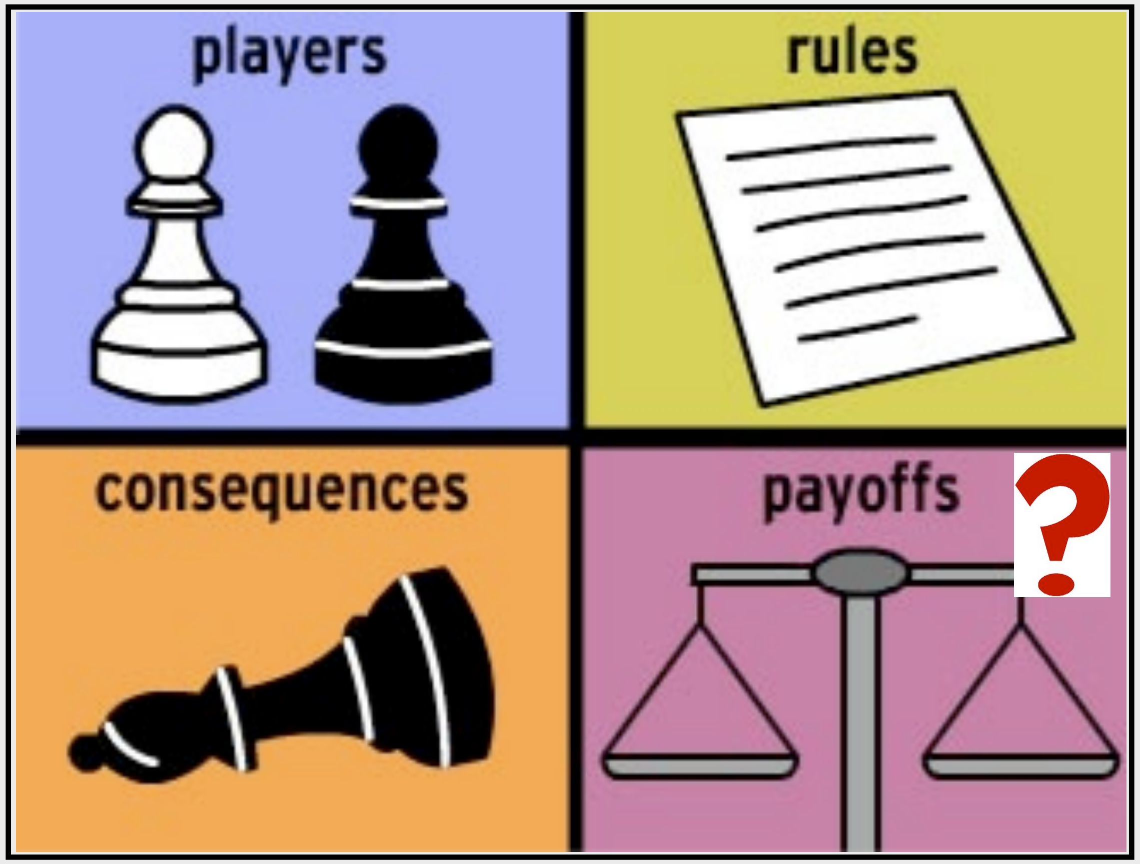


Construct a
control protocol
such that the system satisfies

$$\varphi_{init} \wedge \varphi_{env} \rightarrow \varphi_{safety} \wedge \varphi_{goal}$$

Game interpretation:

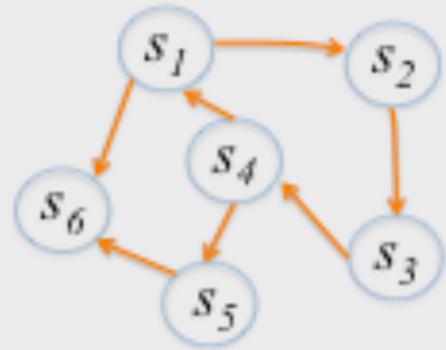
A game between
system & environment



Discrete Synthesis

- states = (system, environment)
- all executions satisfy the spec's

Finite Transition System



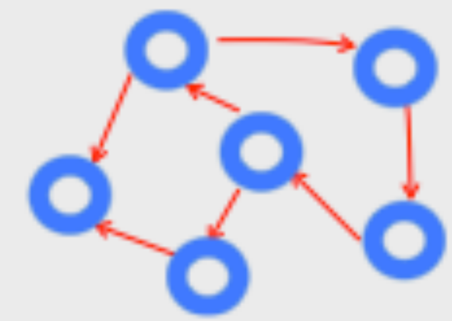
Specifications

$$\varphi_{init} \wedge \varphi_{env} \rightarrow \varphi_{safety} \wedge \varphi_{goal}$$

Discrete Synthesis Tool

Piterman, Pnueli, Sa'ar

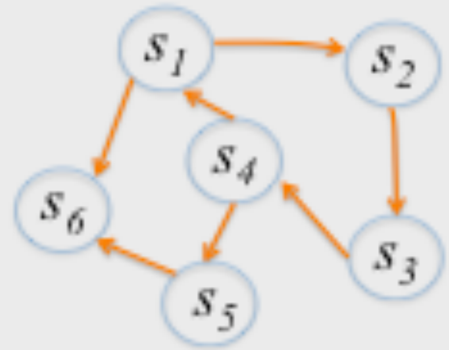
Discrete Planner



Discrete Synthesis

- states = (system, environment)
- all executions satisfy the spec's

Finite Transition System



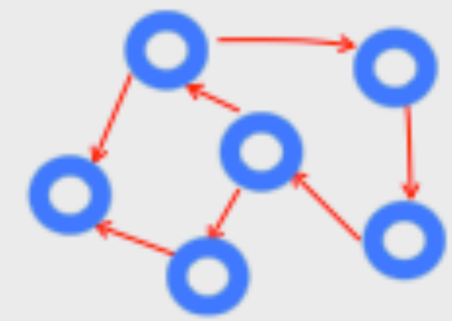
Specifications

$$\varphi_{init} \wedge \varphi_{env} \rightarrow \varphi_{safety} \wedge \varphi_{goal}$$

Discrete Synthesis Tool

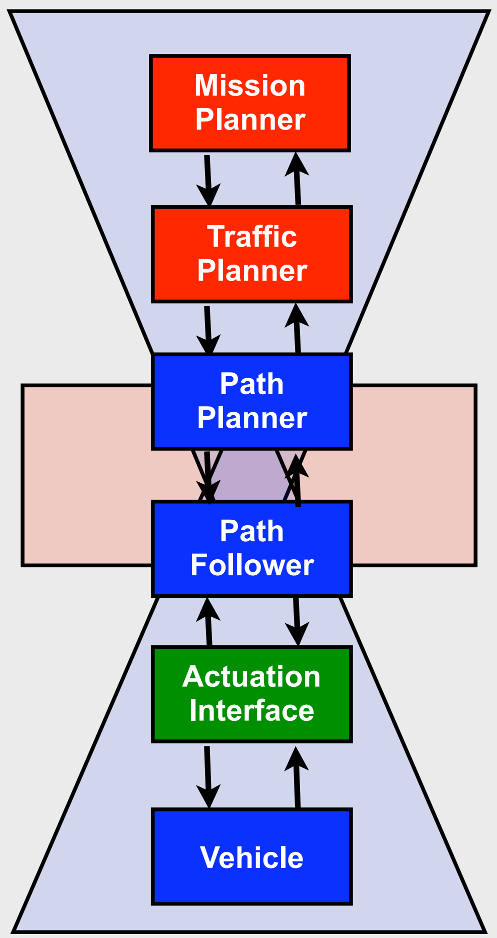
Piterman, Pnueli, Sa'ar

Discrete Planner



Most systems of interest feature interaction between

- physical components
- computing, communication,...



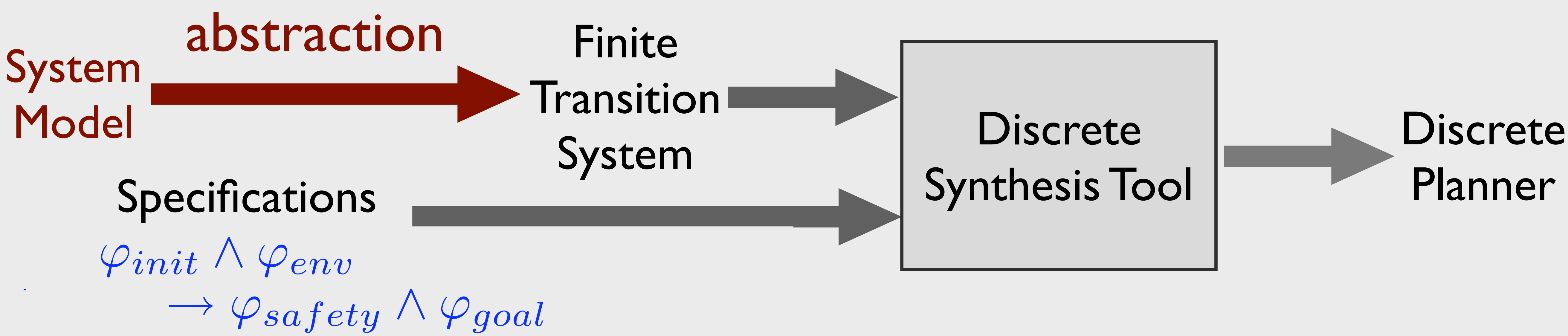
Incorporating Continuous Dynamics

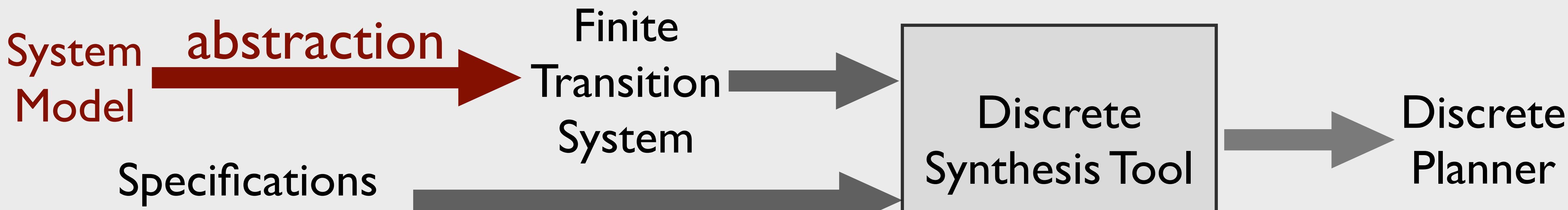
System model:

$$\xi(t+1) = f(\xi(t), w(t), u(t))$$

- bounded control authority $u \in \mathcal{U}$
- external disturbances $w \in \mathcal{W}$

+ modeling uncertainties



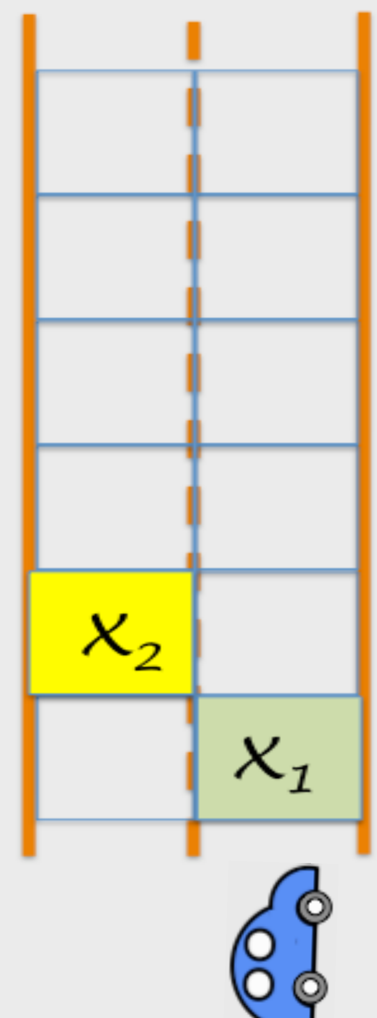


$$\varphi_{init} \wedge \varphi_{env} \rightarrow \varphi_{safety} \wedge \varphi_{goal}$$

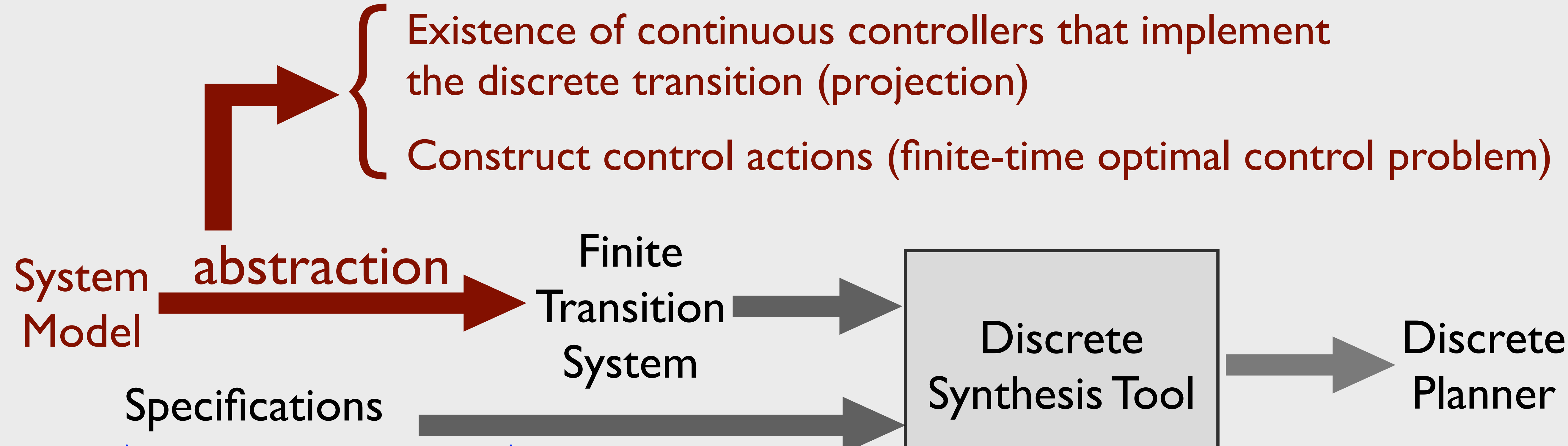
Starting with a proposition preserving partition:

Control-oriented tools to account for ...

- Finite-time reachability to determine discrete transitions



WTM@CDC09 & WTM@AAAI, SS,10

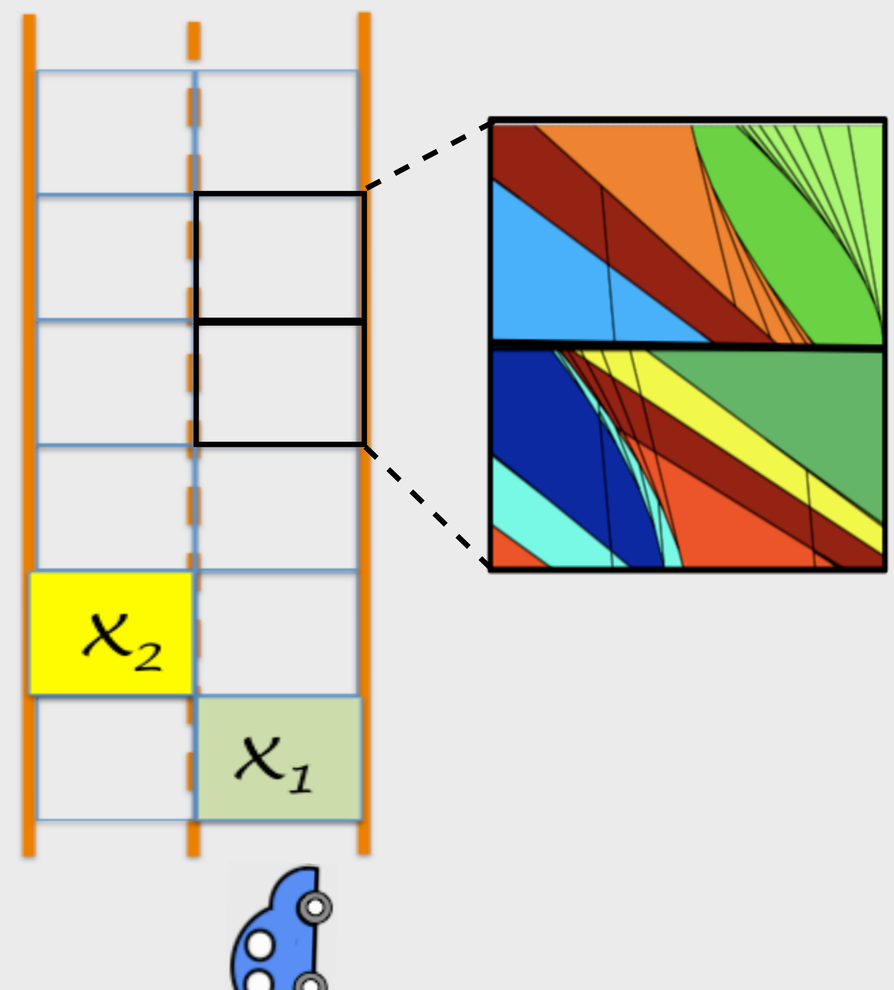


$\varphi_{init} \wedge \varphi_{env} \rightarrow \varphi_{safety} \wedge \varphi_{goal}$

Starting with a
proposition preserving partition:

Control-oriented tools to account for ...

- Finite-time reachability to determine discrete transitions
- Refine the partition to increase the number of valid discrete transitions



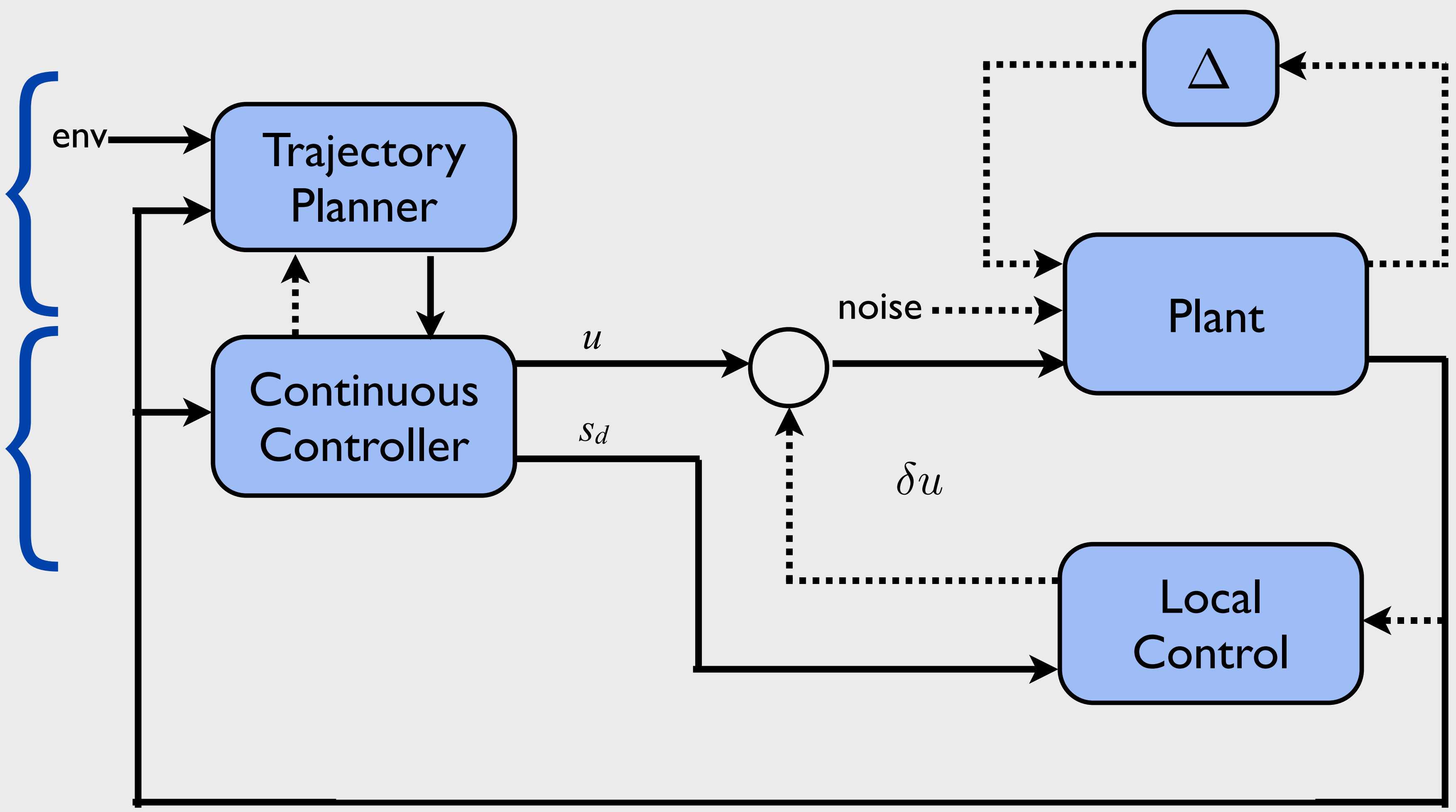
WTM@CDC09 & WTM@AAAI, SS,10

Hierarchical Control Architecture

Discrete planner ensures that the spec is satisfied

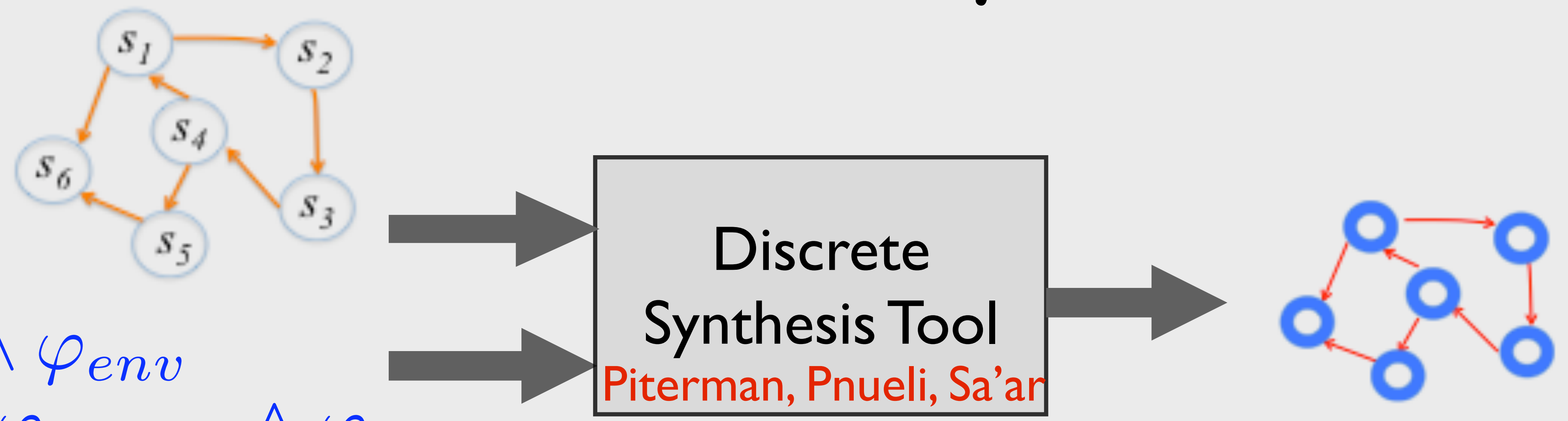
+

Continuous controller *implements* the discrete plan (handles low-level dynamics & constraints)



When put together, guaranteed to work "correctly."

More on the Discrete Synthesis Tool...



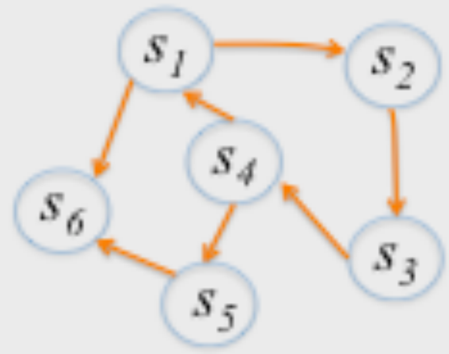
$$\varphi_{init} \wedge \varphi_{env} \rightarrow \varphi_{safety} \wedge \varphi_{goal}$$

- General LTL synthesis is hard
- An expressive subclass (GR(1) games) takes "polynomial" effort

$$\bigwedge_{i=1}^{m} \Box \Diamond p_i^e \rightarrow \bigwedge_{j=1}^{n} \Box \Diamond q_j^s$$

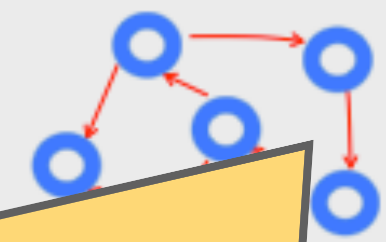
- Based on fixpoint computations & BDDs
- Implemented in JTLV

More on the Discrete Synthesis Tool...



$\varphi_{init} \wedge \varphi_{env}$
$\rightarrow \varphi_{safety} \wedge \varphi_{goal}$

Discrete Synthesis Tool
Piterman, Pnueli, Sa'ar

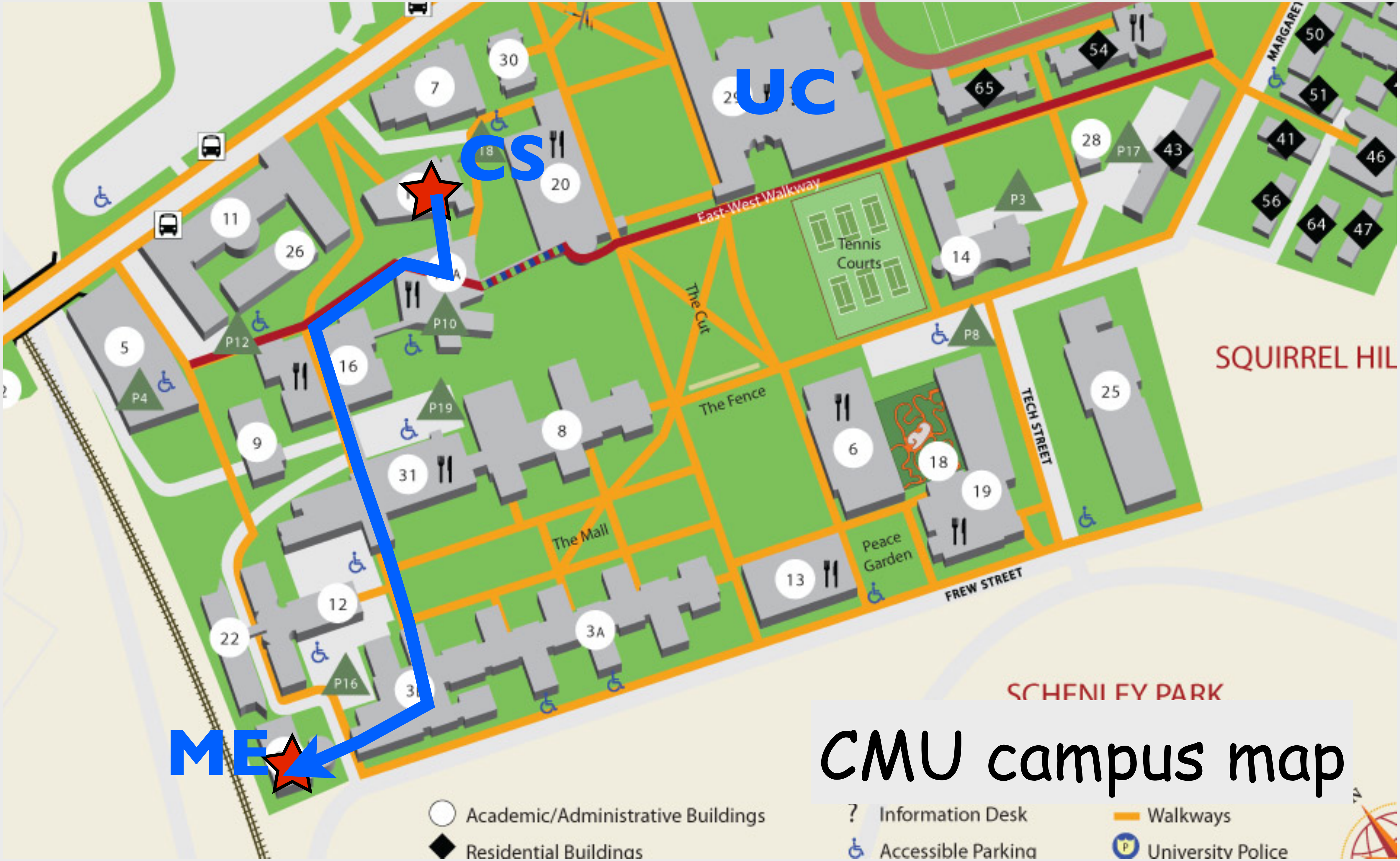


A limiting factor:
synthesis procedure considers all possible environment behaviors

...class (GR(1) games) takes ...omial" effort

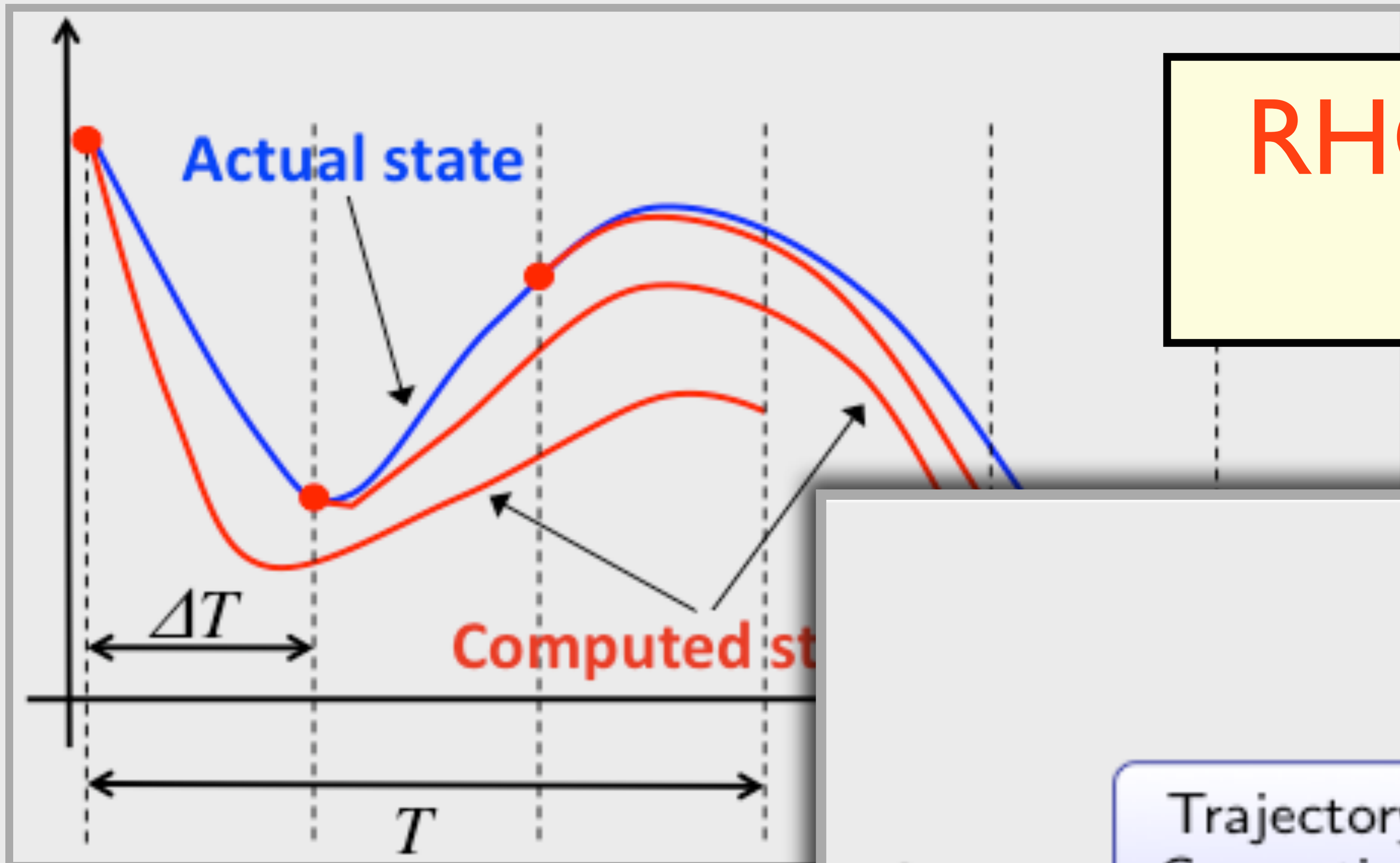
$$\bigwedge_{i=1}^{m} \Box \Diamond p_i^e \rightarrow \bigwedge_{j=1}^{n} \Box \Diamond q_j^s$$

- Based on fixpoint computations & BDDs
- Implemented in JTLV

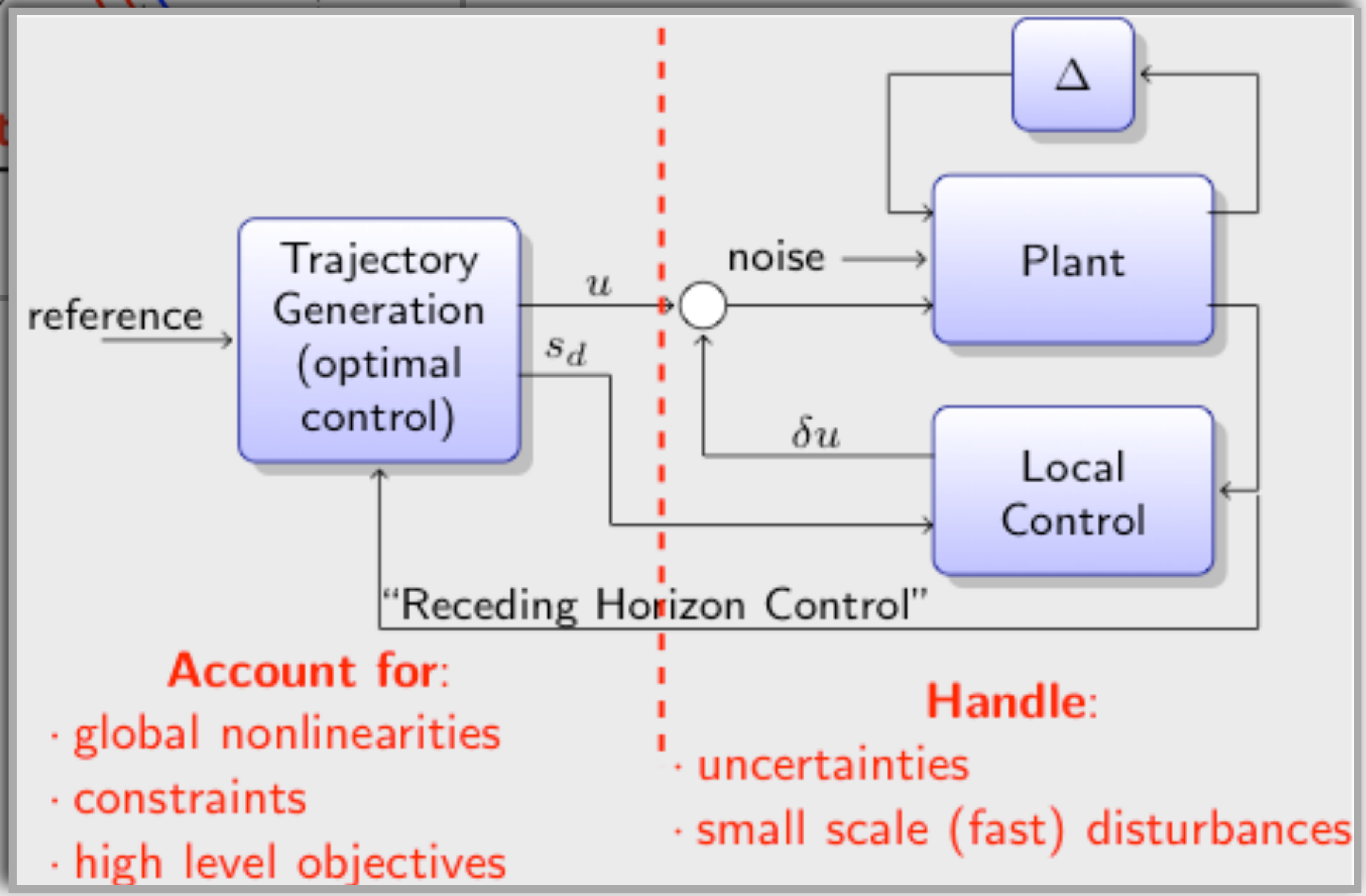


CMU campus map

Recall: Receding Horizon Control (RHC)



RHC can destabilize if not done properly!



Receding horizon, temporal logic planning?

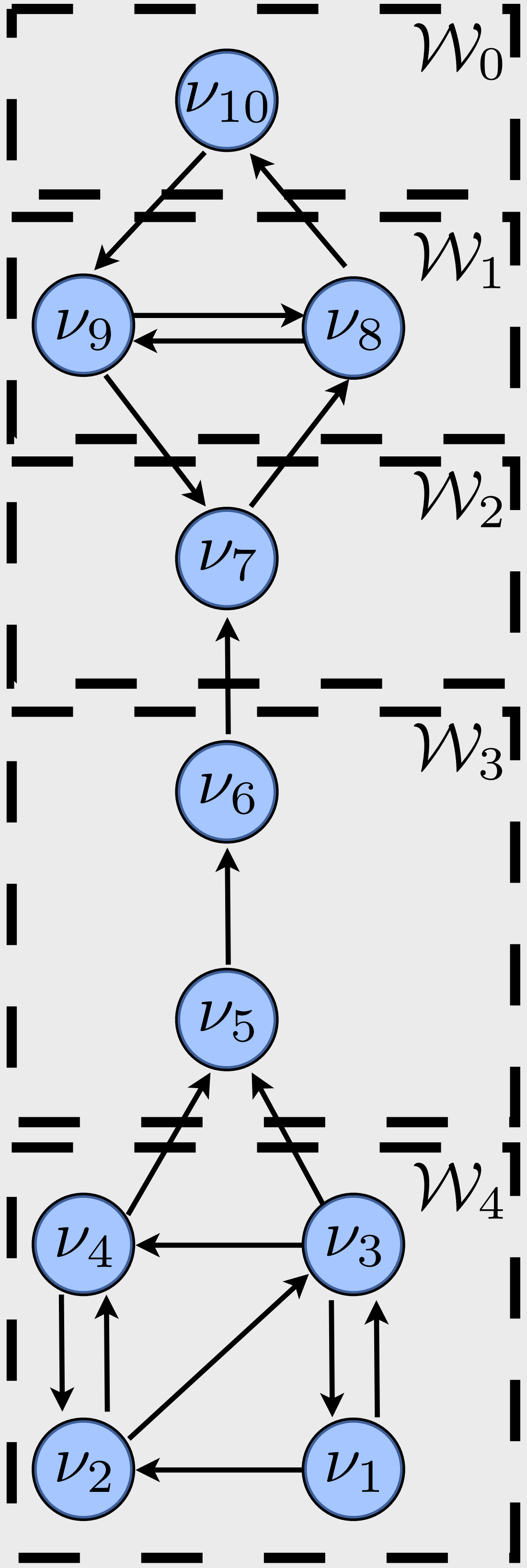
Receding Horizon for LTL Synthesis

$$(\varphi_{init} \wedge \varphi_{env}) \rightarrow (\varphi_{safety} \wedge \varphi_{goal})$$

- $(\{\mathcal{W}_j\}, \preceq_{\varphi_g})$ partial order covering system states
- A mapping $\mathcal{F}$ such that

  $\mathcal{F}(\mathcal{W}_j) \prec_{\varphi_g} \mathcal{W}_j$ for $j \neq 0$ & $\mathcal{F}(\mathcal{W}_0) = \mathcal{W}_0$
- Φ, a propositional formula such that
- For each j, there exists a short-horizon controller that realizes

$$\big((\xi \in \mathcal{W}_j) \wedge \Phi \wedge \varphi_{env}^j\big) \rightarrow \Big(\varphi_{safety}^j \wedge \Box \diamond (\xi \in \mathcal{F}(\mathcal{W}_j)) \wedge \Box\Phi\Big)$$

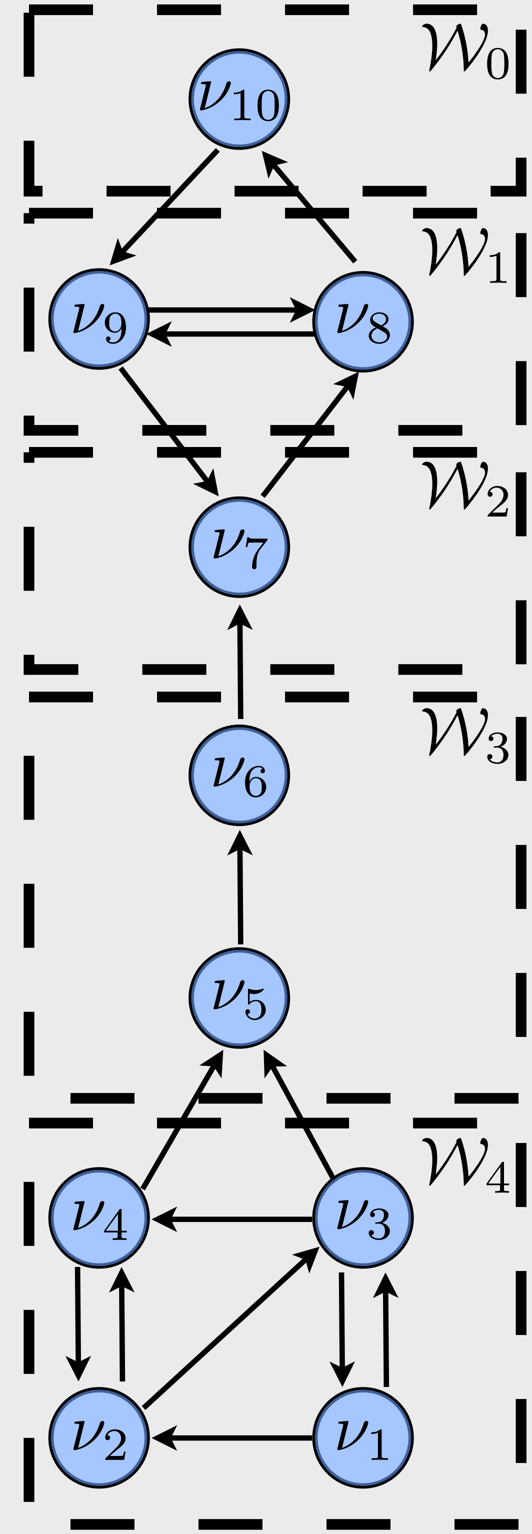


<u>Theorem</u>: When the system state is in $\mathcal{W}_j$, implement the corresponding short-horizon controller. Then, the "global" spec's hold.

WTM@HSCC10
WTM@ITAC(s)

What is Φ?

- Receding horizon invariant, a propositional formula
- Used to exclude the initial states that render synthesis infeasible, e.g.,
  - States from which a collision is unavoidable
- Given partial order and $\mathcal{F}$, computation of the invariant can be automated.

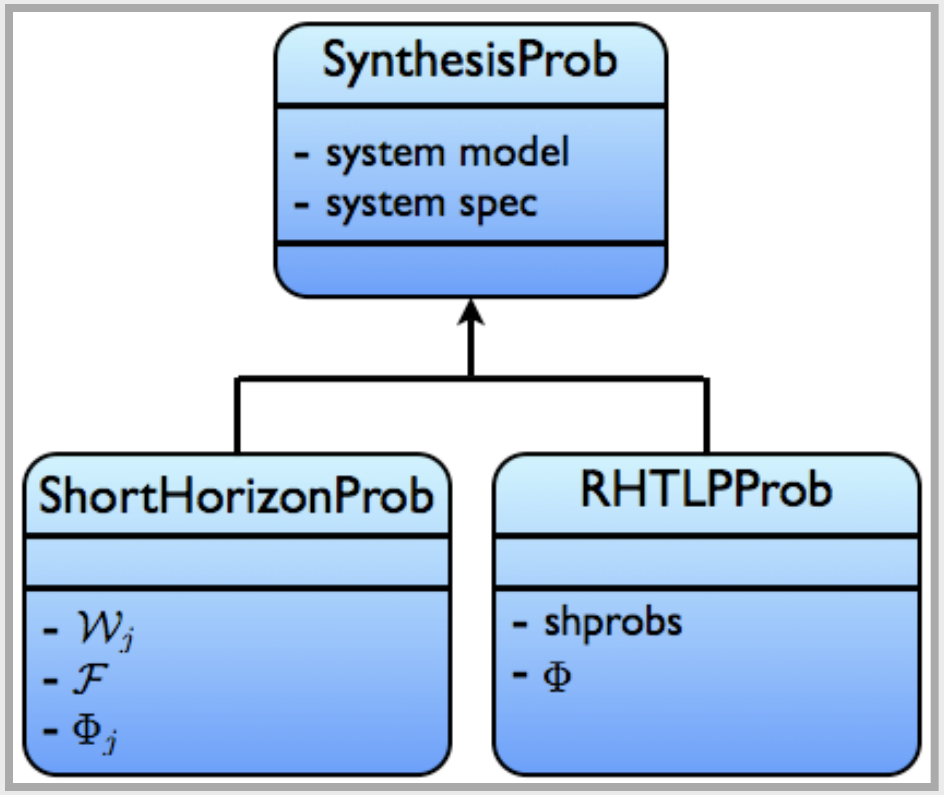


> • Check realizability
> • If realizable, done.
> • If not,
>   - collect violating initiation conditions
>   - negate and put in Φ
> • Repeat.

TuLiP automates...

- Proposition preserving partitioning
- Abstraction
- Given partial order, compute an invariant (if exists)
- Verify that all conditions for applying the receding horizon strategy are satisfied
- Create short-horizon problems and implement the receding horizon strategy
- Interface to the synthesis tool
- Compute counter-examples
- Simulate the resulting strategy

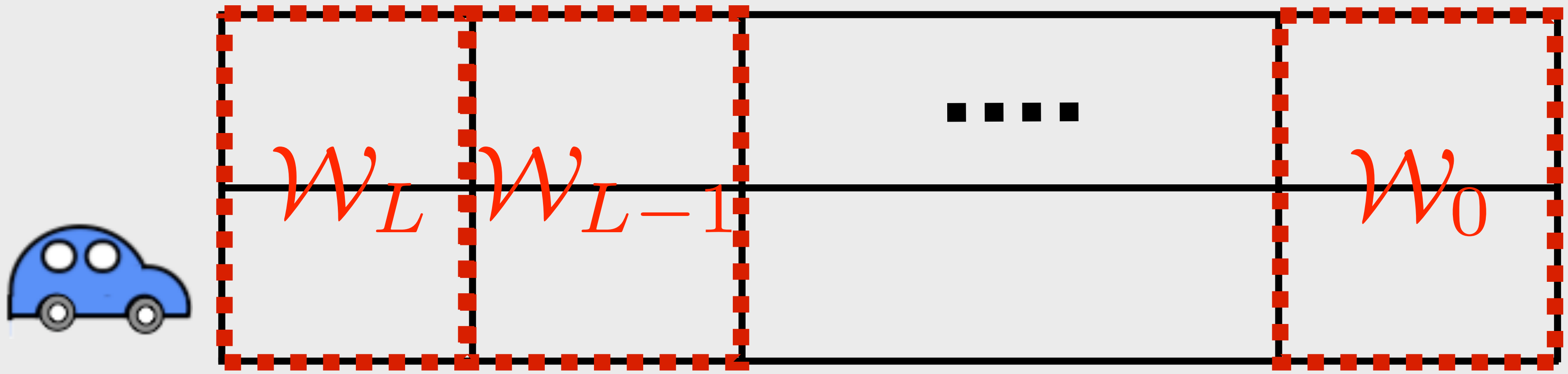
TuLiP
A software toolbox for receding horizon Temporal Logic Planning
www.cds.caltech.edu/tulip



How to come up with partial order?

- Problem-dependent
- Currently requires user guidance

<u>Simple example</u>

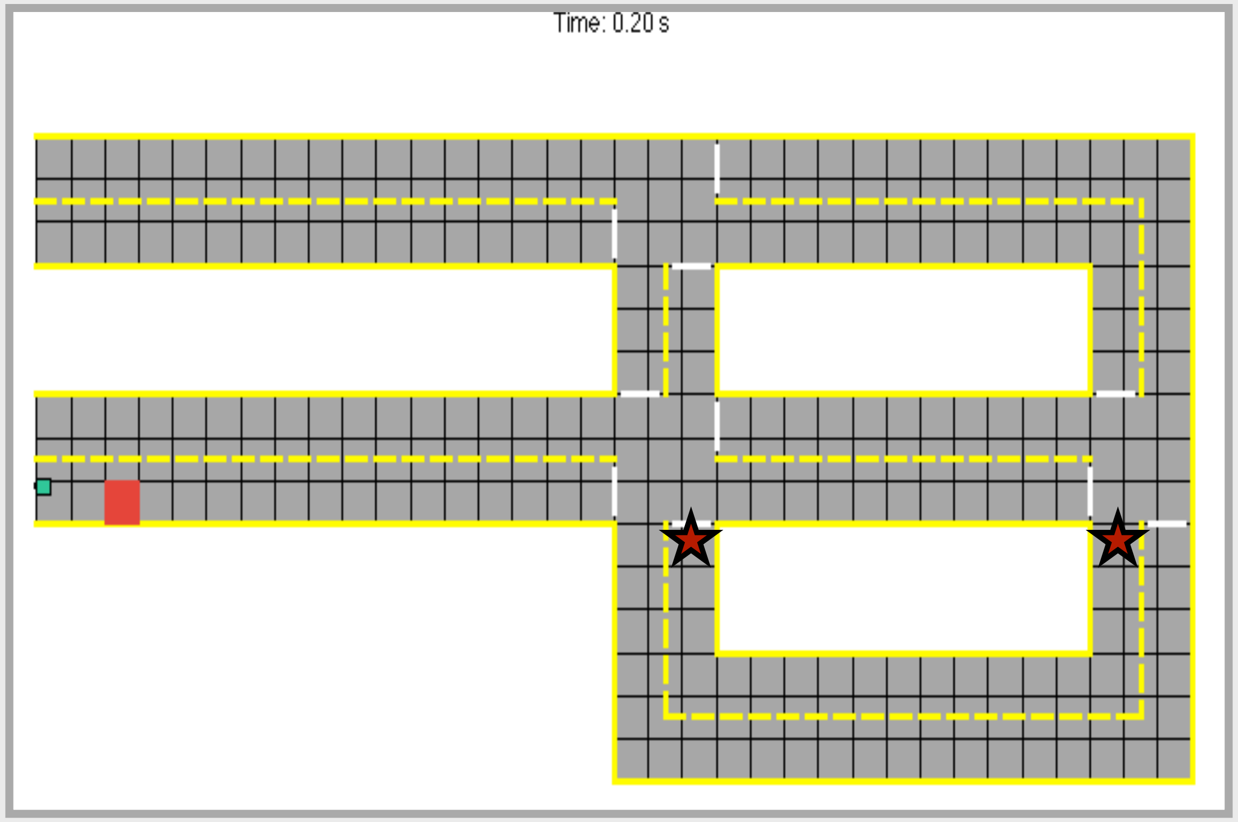


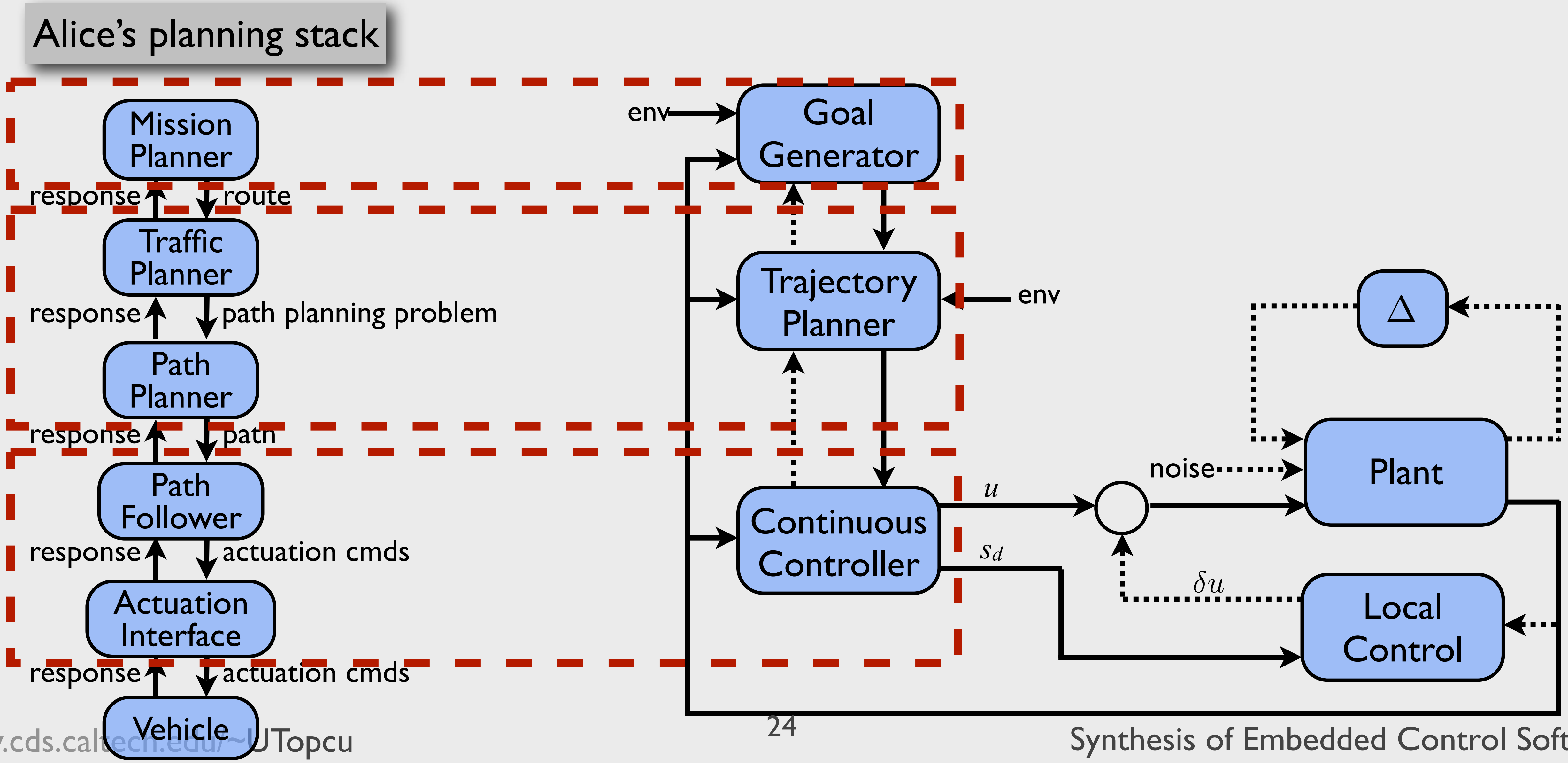
$$\mathcal{W}_0 \prec \ldots \prec \mathcal{W}_{L-1} \prec \mathcal{W}_L$$

$$\mathcal{F}(\mathcal{W}_j) = \mathcal{W}_{j-2}, \quad j \geq 2$$
$$\mathcal{F}(\mathcal{W}_j) = \mathcal{W}_0, \quad j < 2$$

How to come up with partial order?

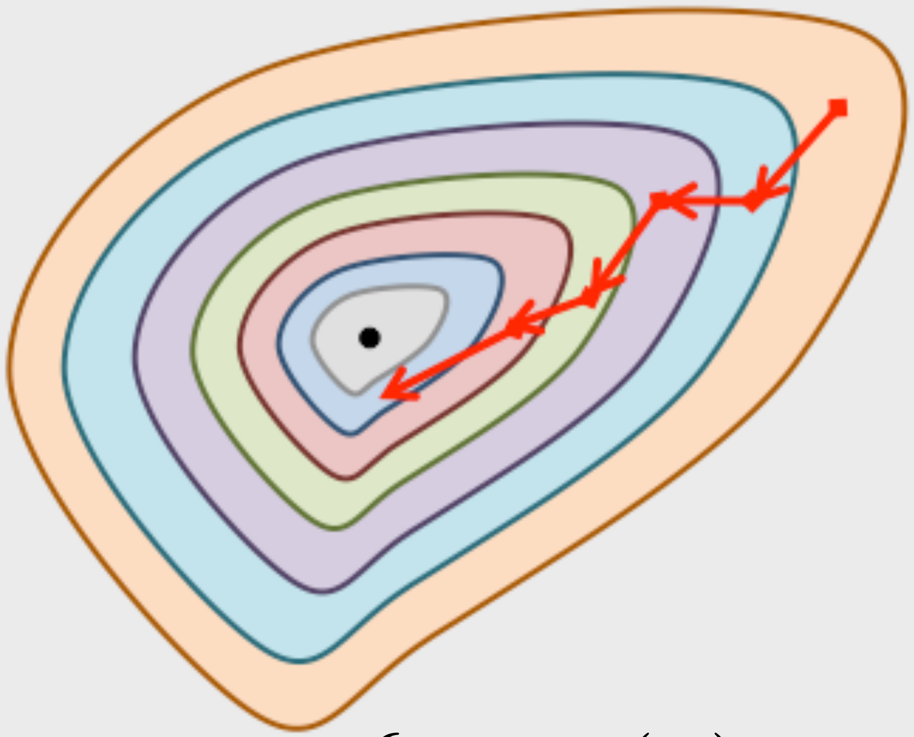
In some problems, it naturally pops up.



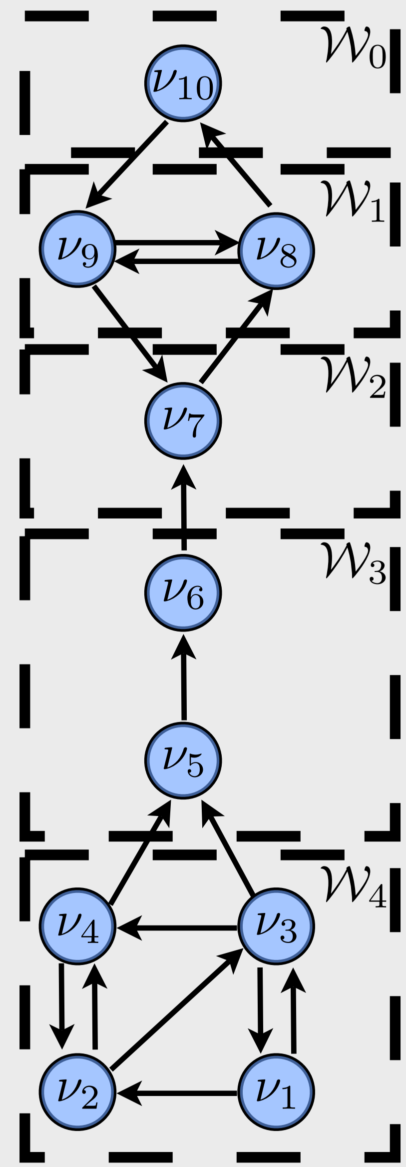


Contraction Constraints ~ Partial Order

Receding horizon control



Receding horizon temporal logic planning



Level sets $\{x : V(x) \leq \alpha_i\}$ induce an <u>order</u> on $\mathbb{R}^n$, e.g., V : control Lyapunov function.

Norms, level-sets, etc. on continuous spaces do not generalize; but, (partial) orders do!

Outline

- Setup
- Receding horizon temporal logic synthesis
- Vehicle management systems
- Distributed synthesis

Vehicle Management Systems

Manages a number of avionics functionalities and their power/computation/communication resources.

Reacts to the changes in the "environment" in real time.

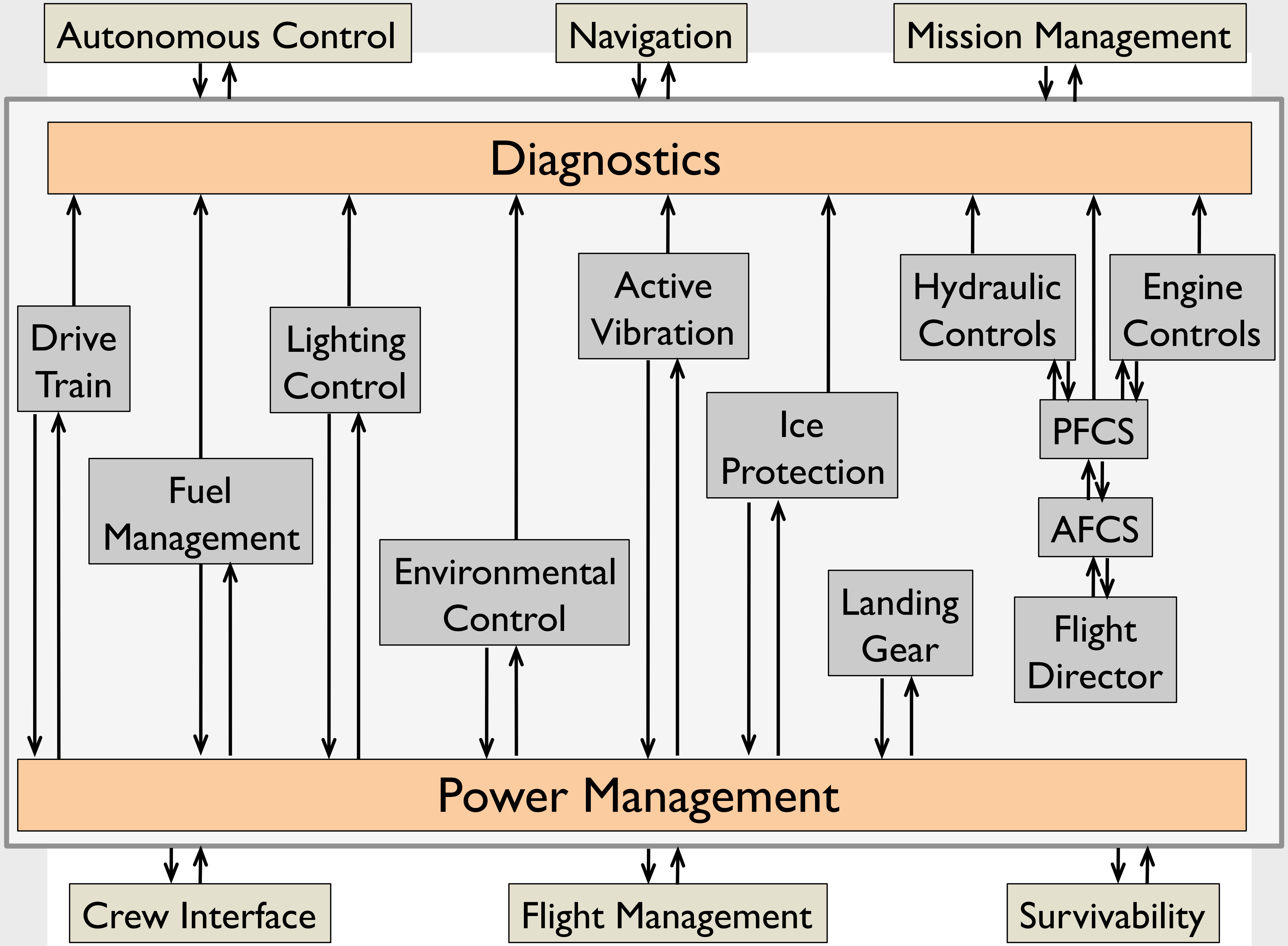
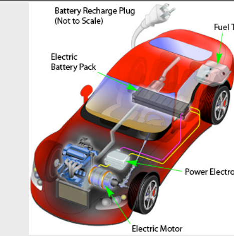
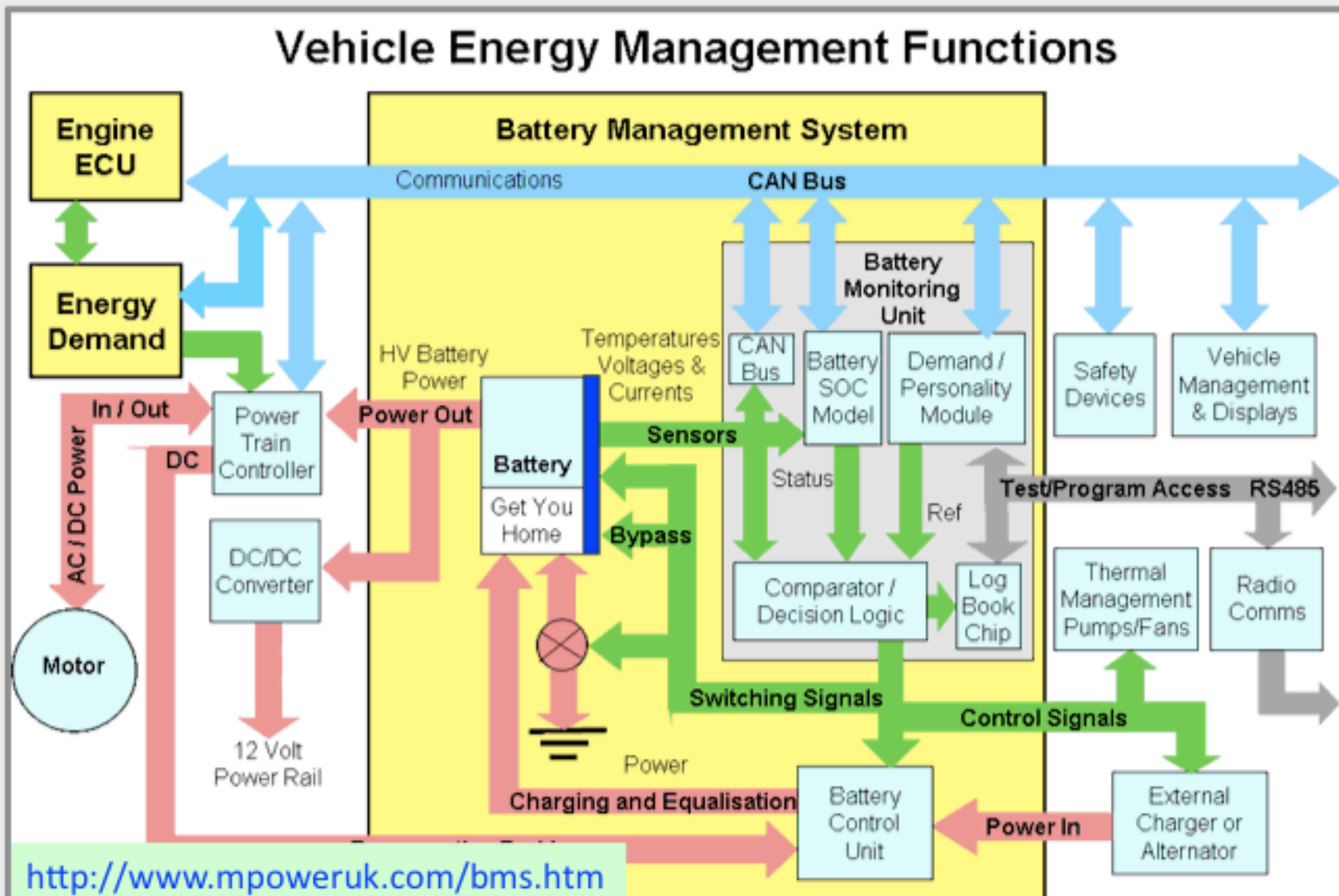


Figure – recreated from a similar figure by W. P. Kinahan, Sikorsky

Vehicle Management Systems



Toyota's Vehicle Dynamics Integrated Management System integrates active safety, comfort, and entertainment functionalities. (pressroom.toyota.com)

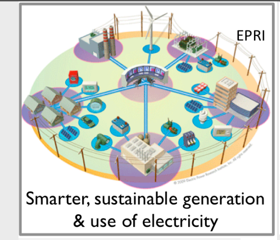


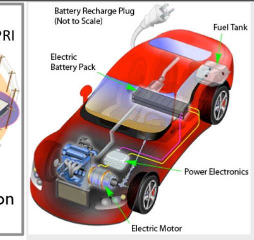
Comfort
+
Active safety

V2I communication

~~Vehicle~~ Management Systems

Energy

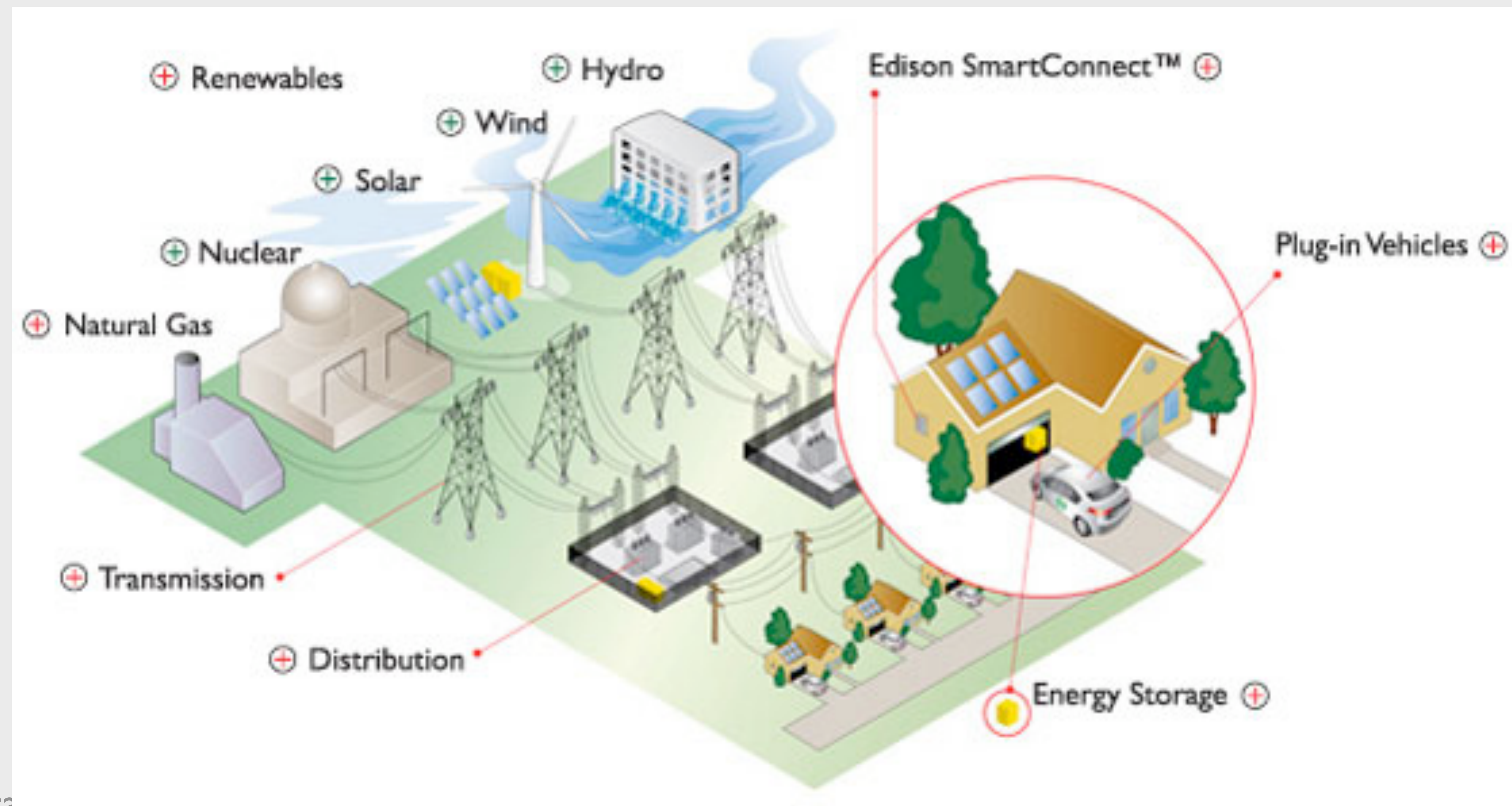




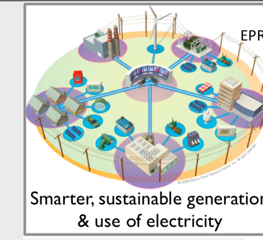
The landscape is changing:

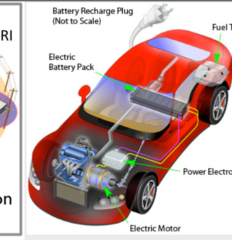
(at multiple levels: devices, buildings, vehicles, power grid)

AMI, energy-smart appliances, electric vehicles, demand response, distributed generation & storage, inverters, AVVC



~~Vehicle~~ Management Systems
Energy





Common driver:

Energy/resource optimization

Common enabler:

System-level management using information systems

Common issues:

- Heterogeneity
  - subsystems
  - requirements
  - (safety) criticality
- Uncertainties of multiple, overlapping scales
- Highly distributed architectures
- Verification of safety & performance
- Managing complexity

Federated ⟶ Integrated Modular

Driver: Support for a number of trends, e.g., more-electric, autonomy,...

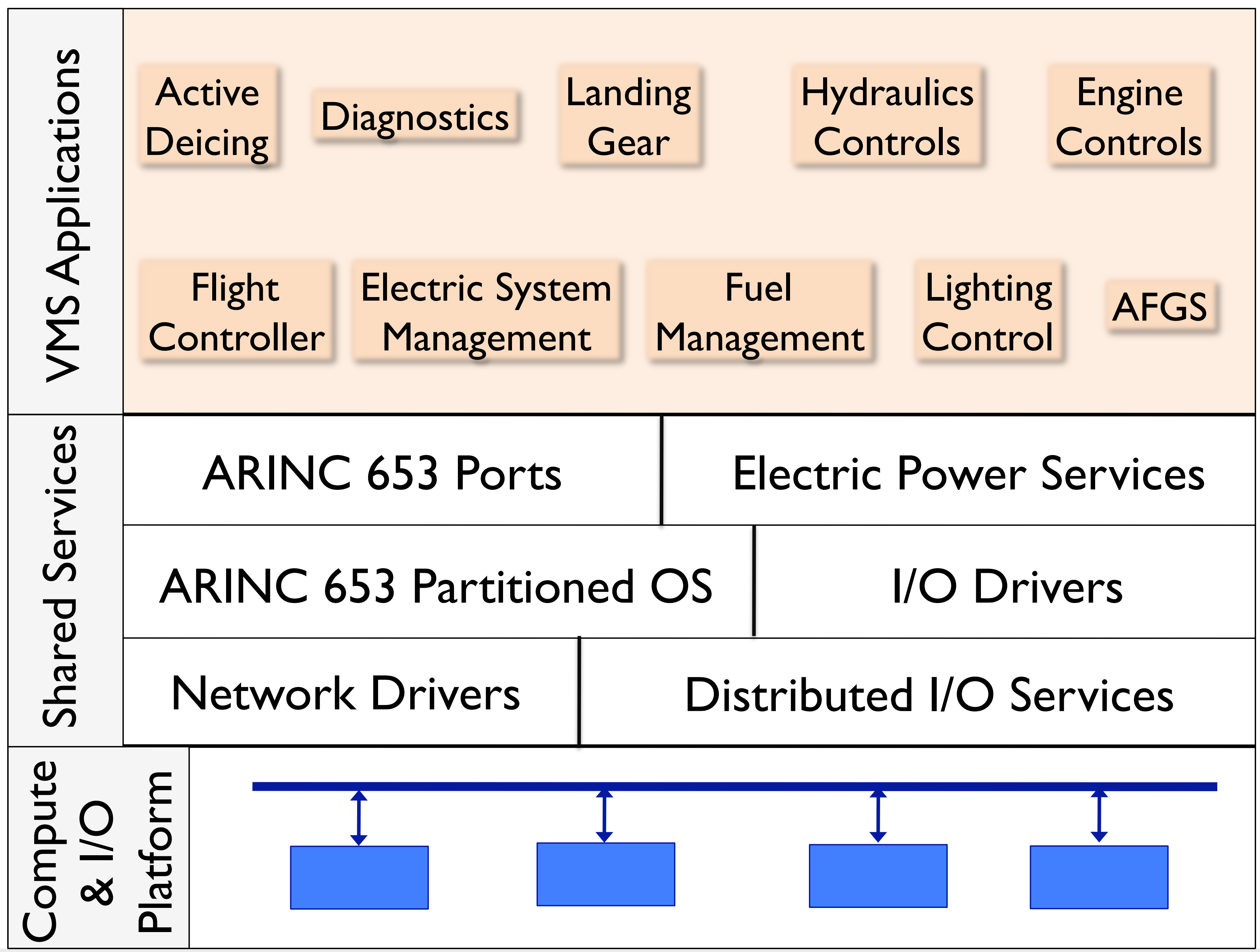


Figure – regenerated from a similar figure by W. P. Kinahan, Sikorsky Aircraft

Federated ⟶ Integrated Modular

Driver: Support for a number of trends, e.g., more-electric, autonomy,...

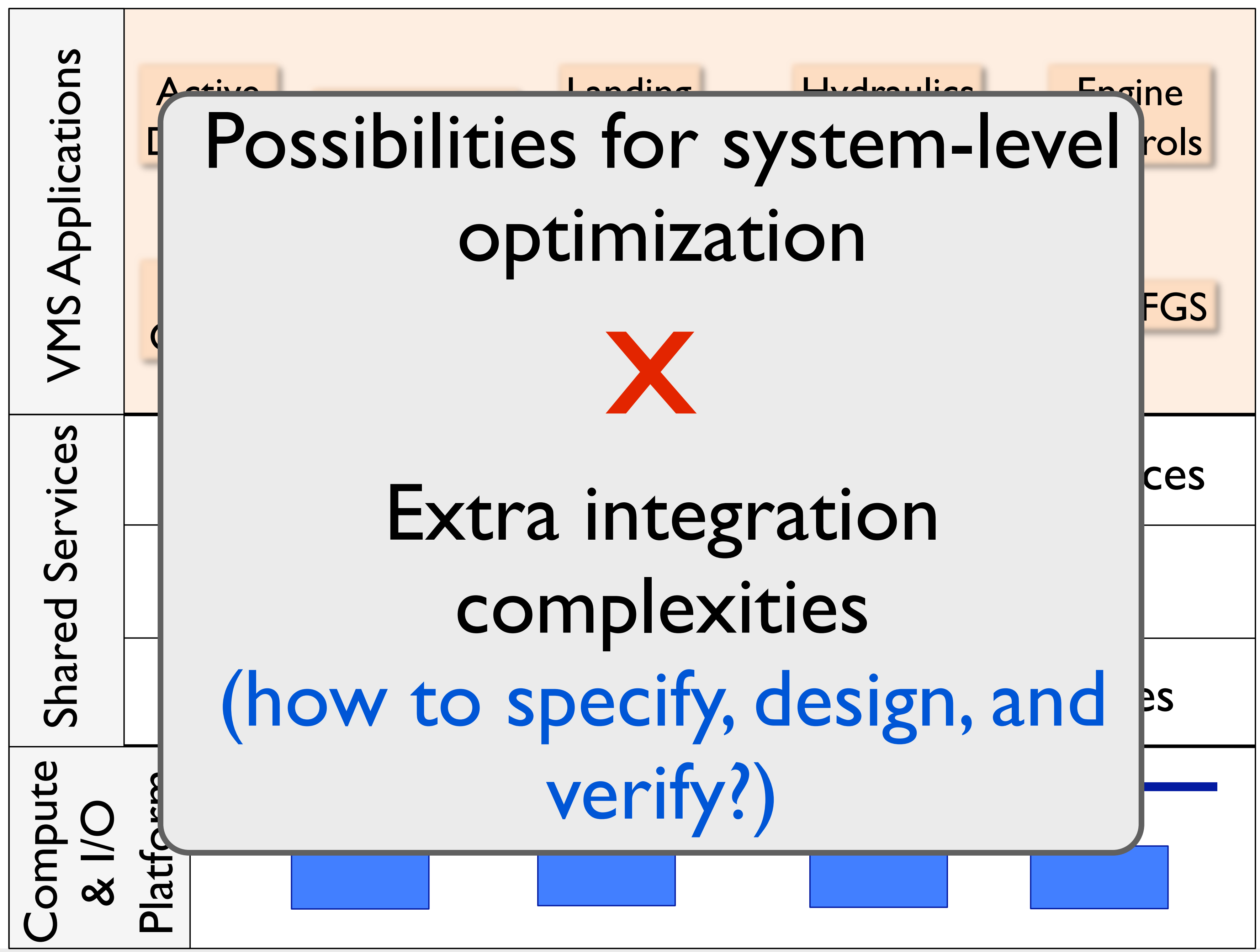


Figure – regenerated from a similar figure by W. P. Kinahan, Sikorsky Aircraft

Case Study: Control Protocols for VMS

Power management between
- flight controllers
- active de-icing
- environmental control

increasing flight criticality

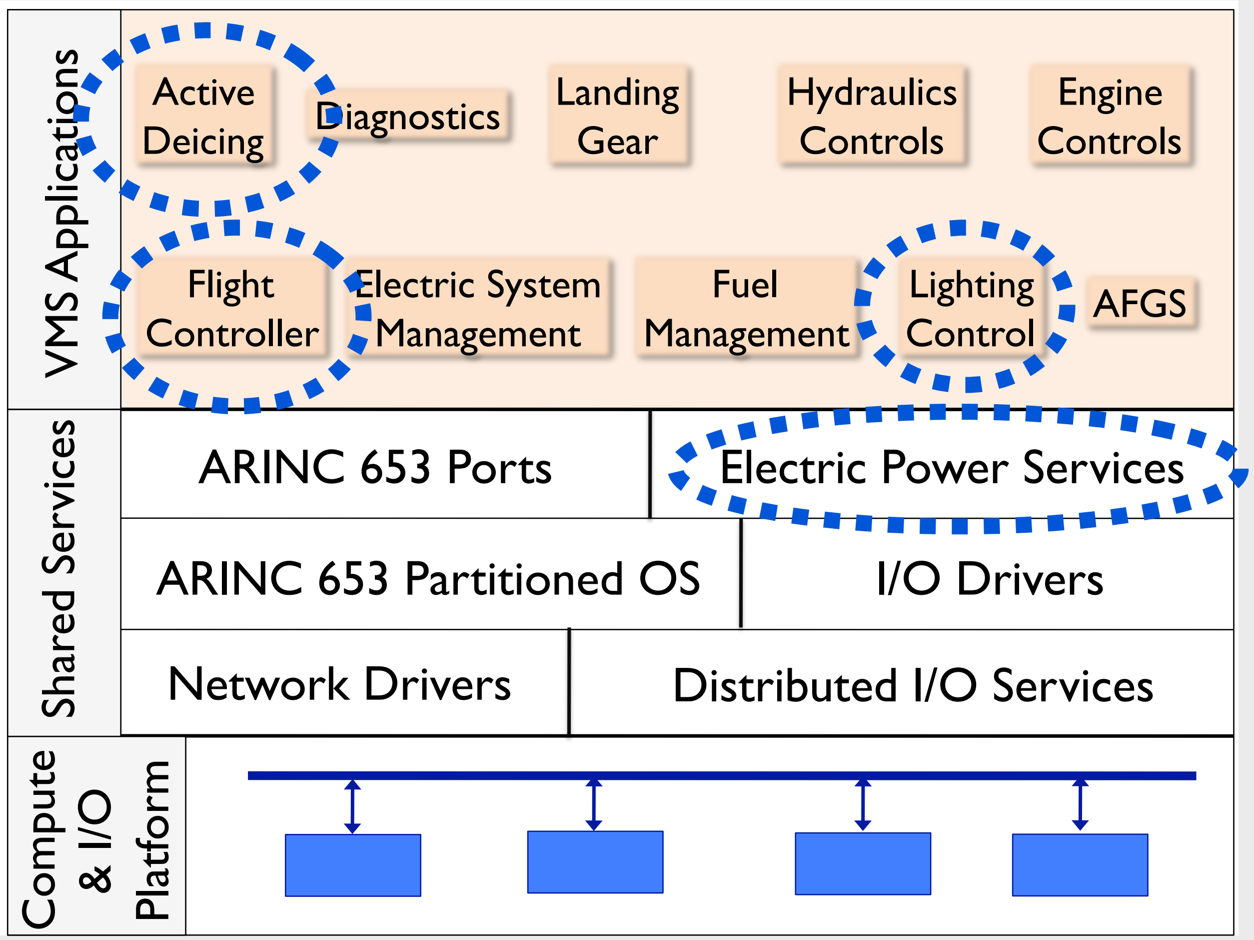


Figure – regenerated from a similar figure by W. P. Kinahan, Sikorsky Aircraft

<u>Environment variables</u>:
wind gust & outside temperature

<u>Controlled variables</u>:
altitude, power supply to different components

<u>Dependent (state) variables</u>:
ice accumulation, energy storage, cabin pressurization

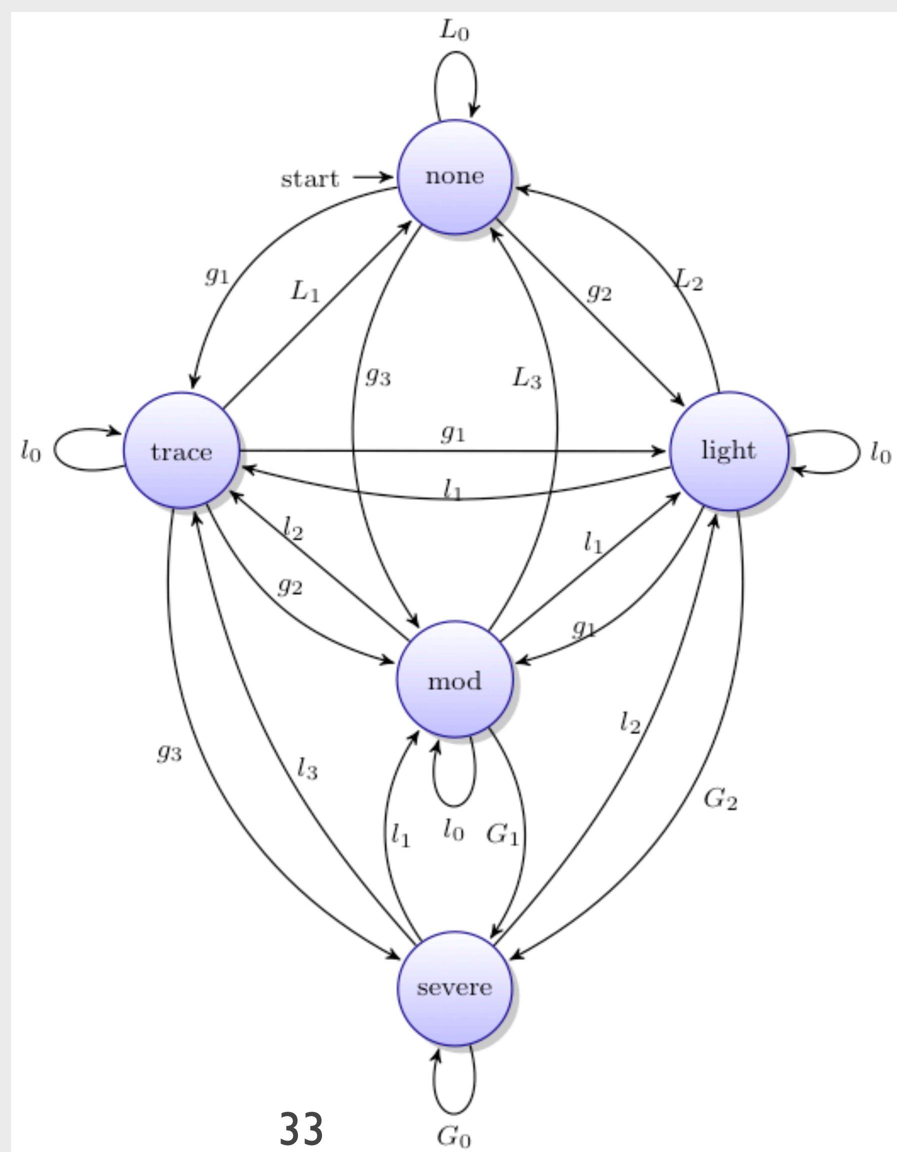
Specifications

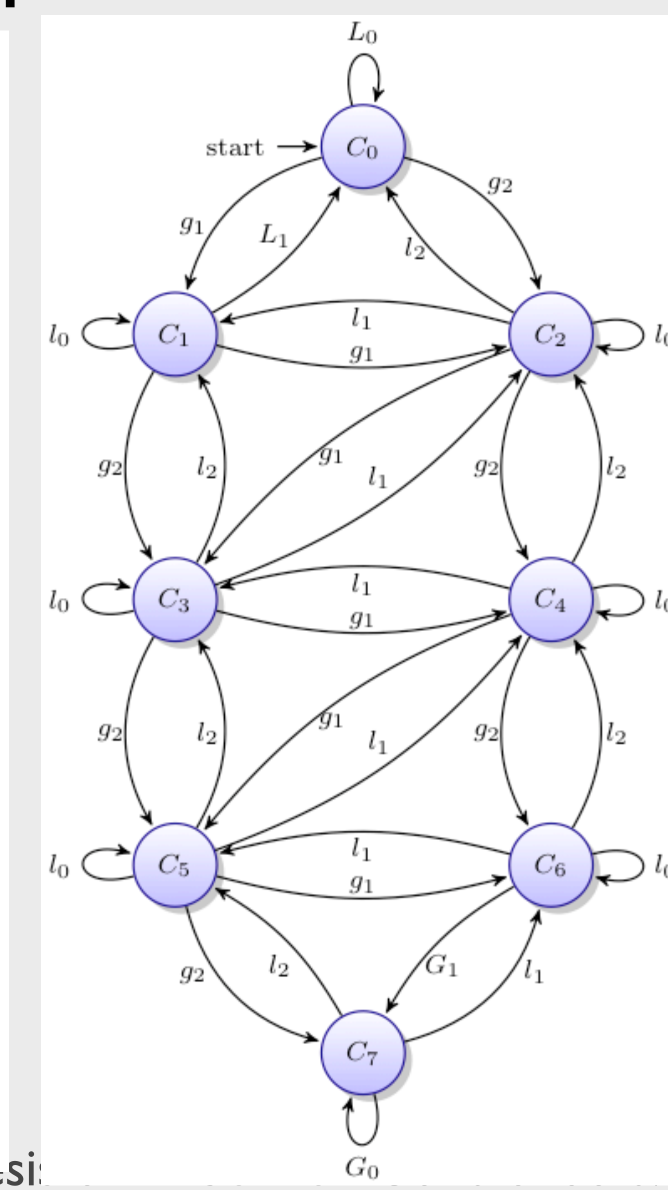
- Limited resources (electric power)
- Safety: prioritization based on flight-criticality & constraint on altitude change and ice accumulation
- Performance: maintain cabin pressure & altitude in desirable ranges
- Environment sssumptions on wind gust & temperature

System model

Finite state automata for the evolution of

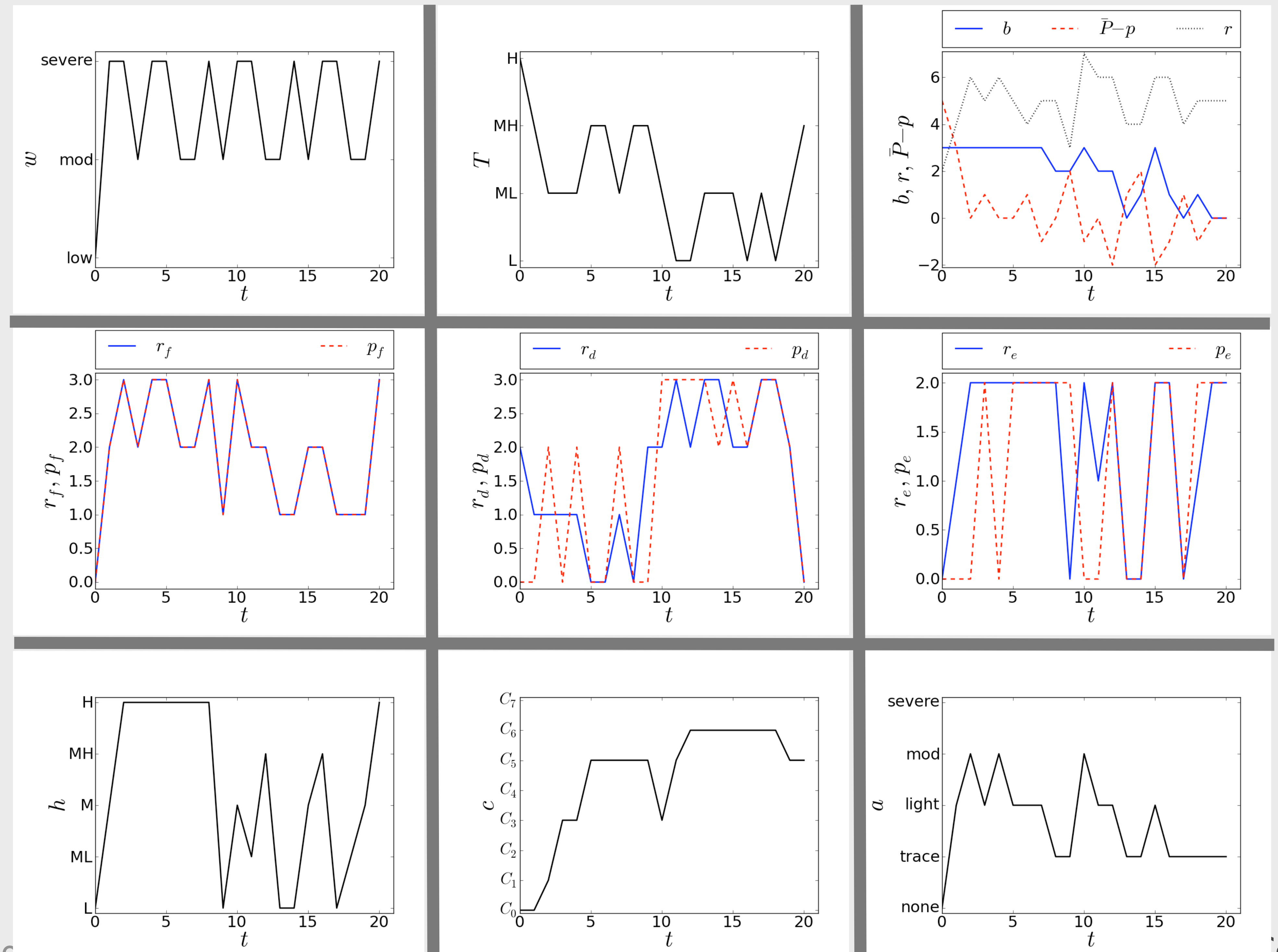
- ice accumulation
- cabin pressure
- energy storage





Preliminary results:

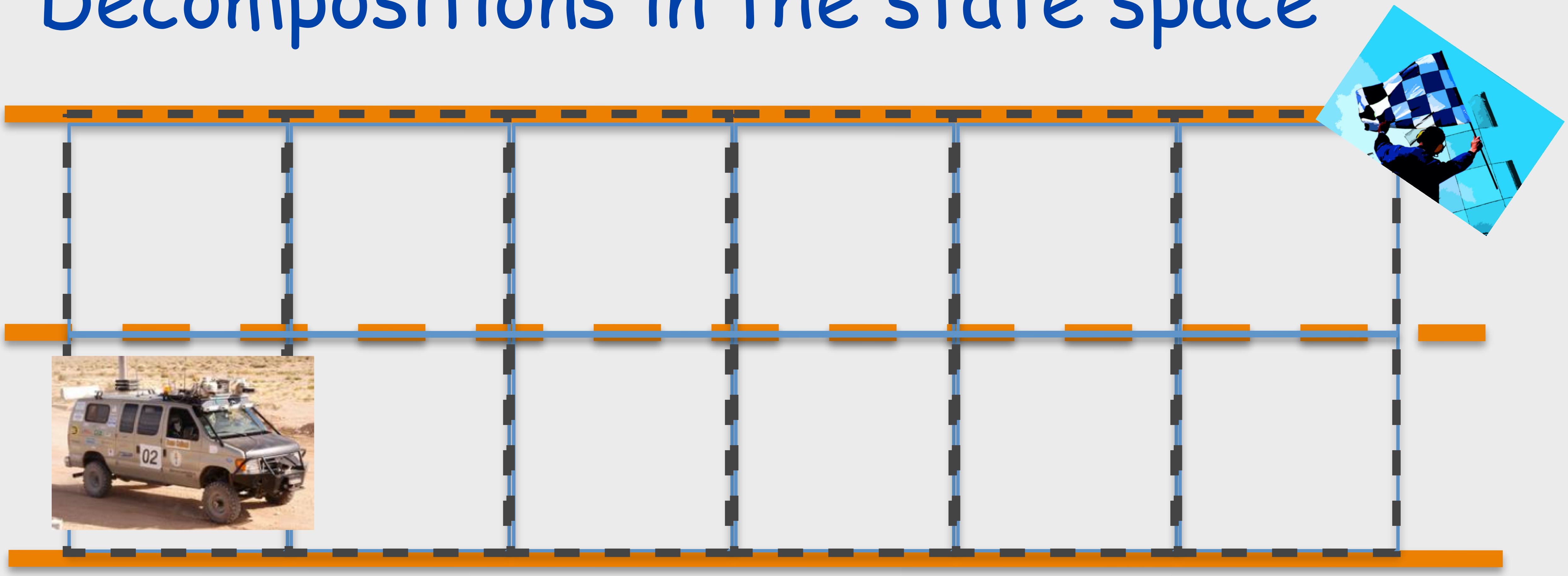
Dynamic power allocation allows reductions in peak power (i.e., generator weight) requirements.



A sample of open issues

- Optimality vs. feasibility
- Hard time constraints
- Design-for-verification
- Incremental synthesis/verification
- Scalability by exploiting the underlying structure

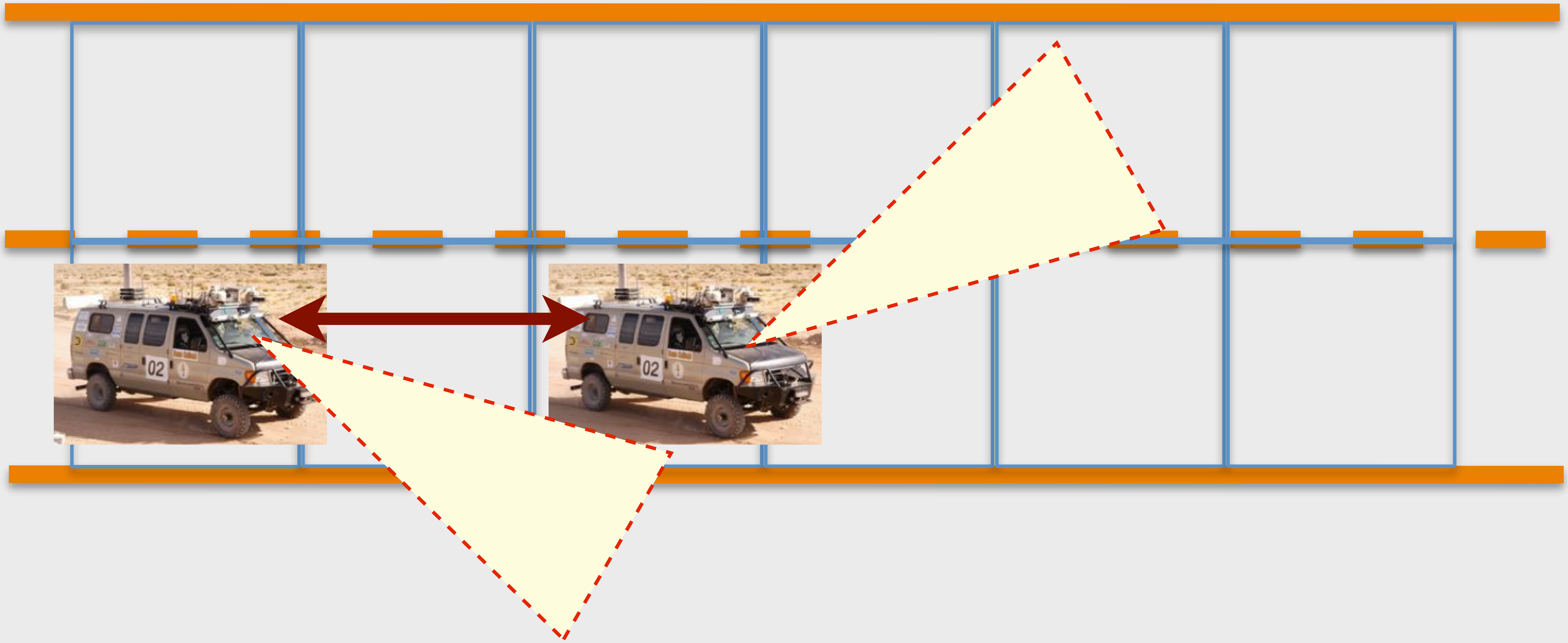
Decompositions in the state space



Decompositions induced by ...

receding horizon	goal
distributed synthesis	underlying network

Decompositions in the state space

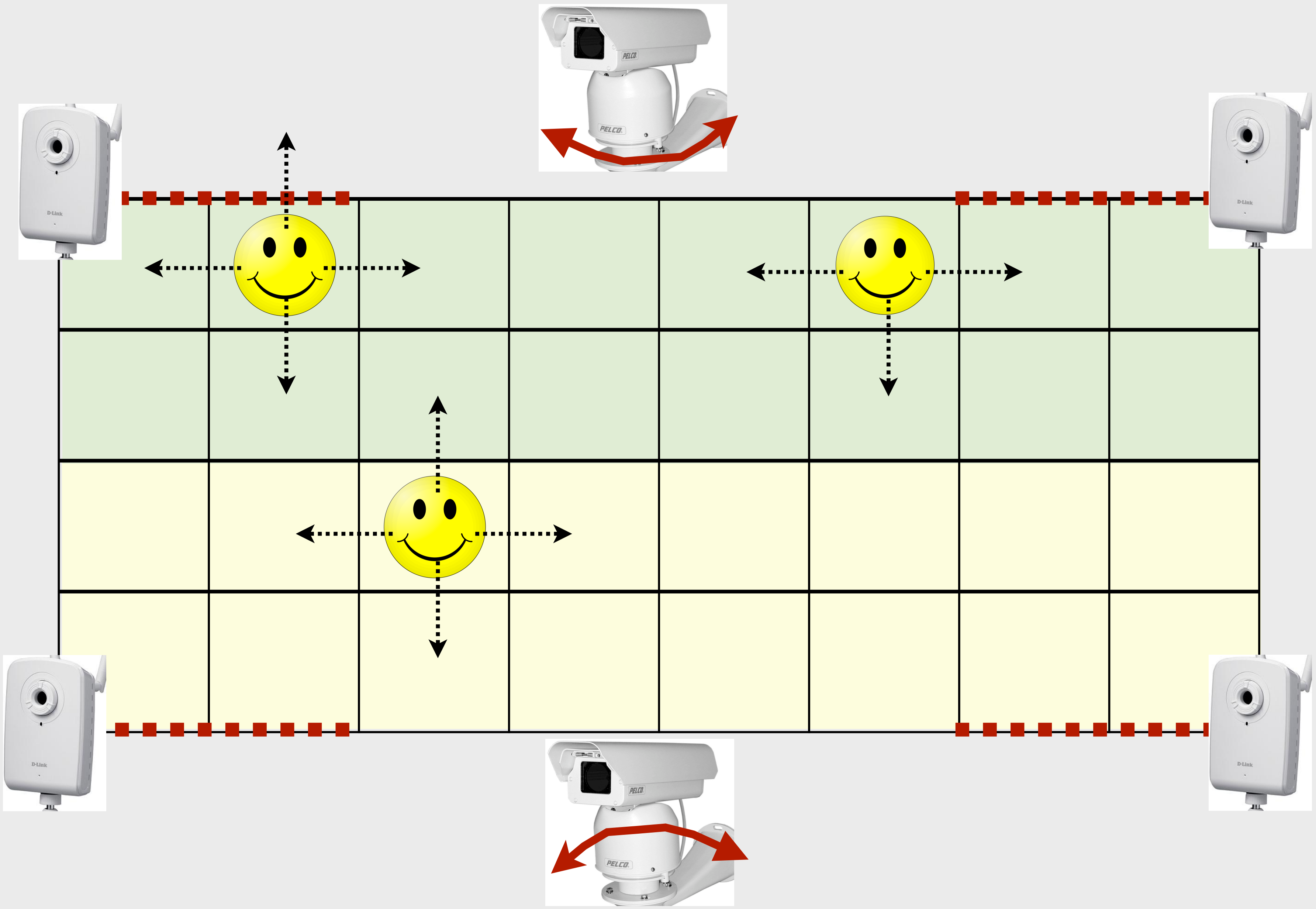


Decompositions induced by ...

receding horizon	goal
distributed synthesis	underlying network

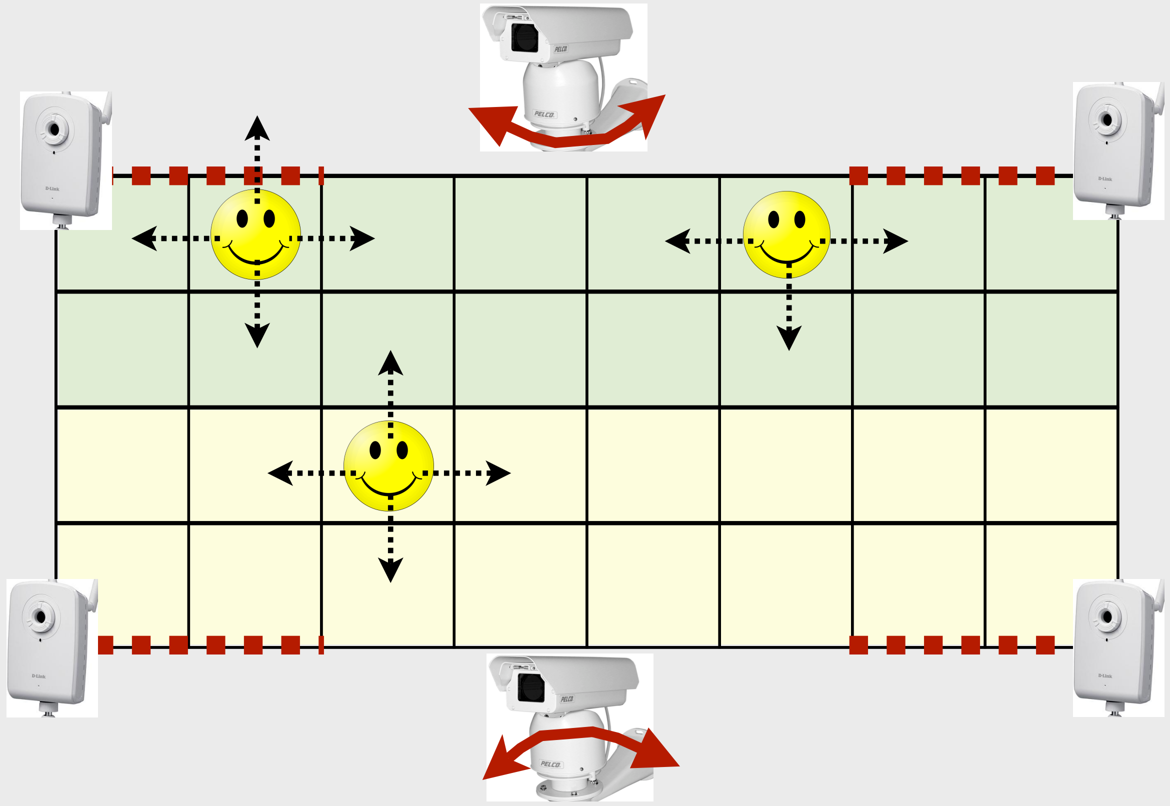
Smart camera networks

- static cameras for tracking targets
- pan-tilt-zoom (PTZ) for active recognition



Goal: synthesize control protocols for PTZ to ensure that one high resolution image of each target is captured at least once

Synthesis of protocols for active surveillance



System:

- region of view of PTZs
- governed by finite state automata

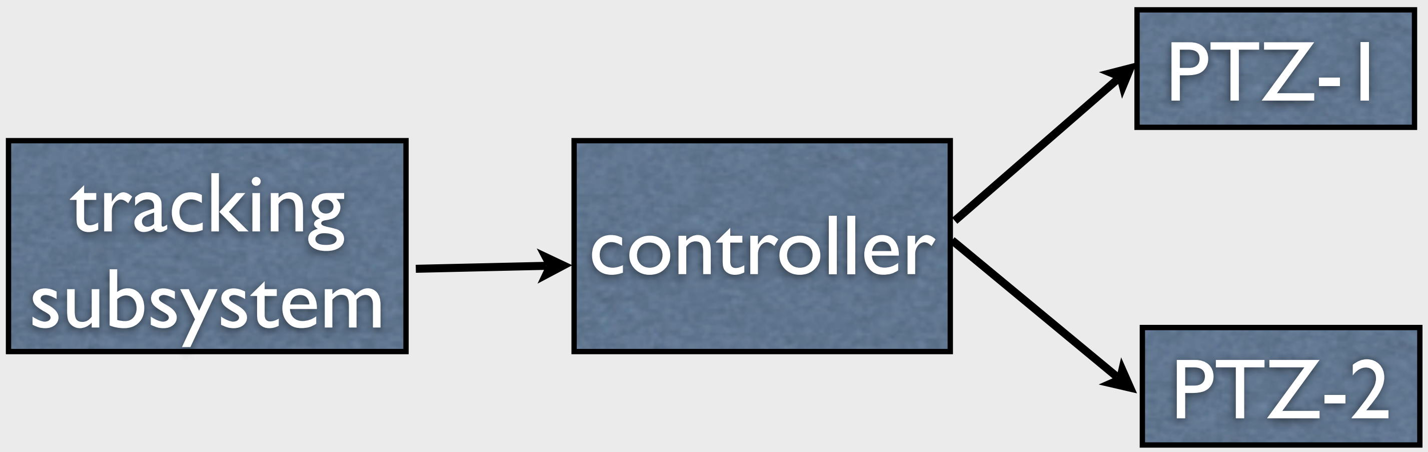
Additional requirement:

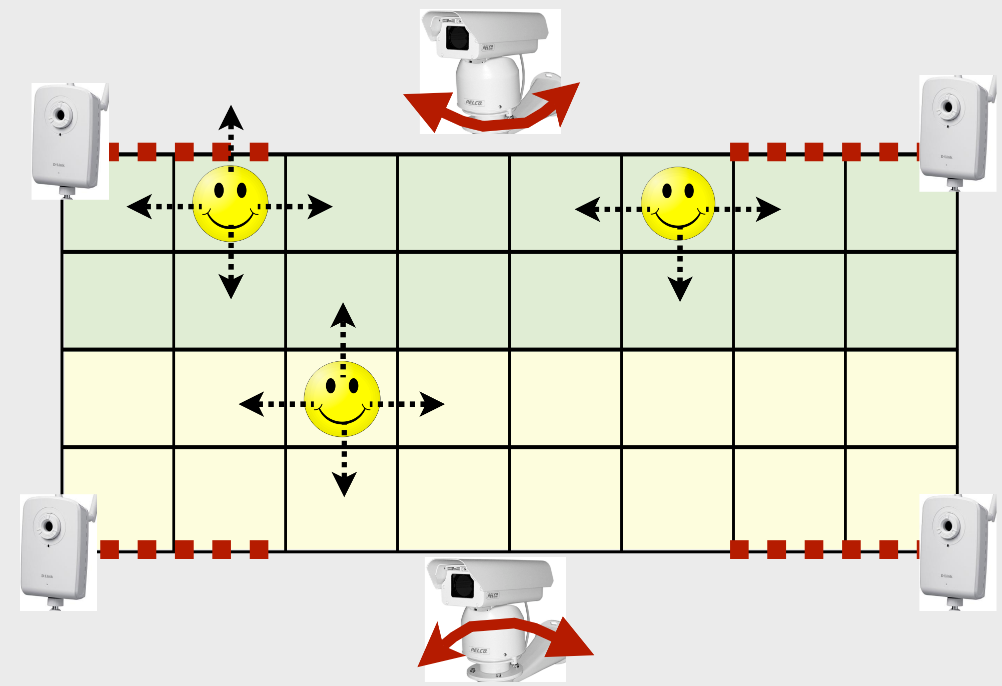
- Zoom-in the corner cells infinitely often.

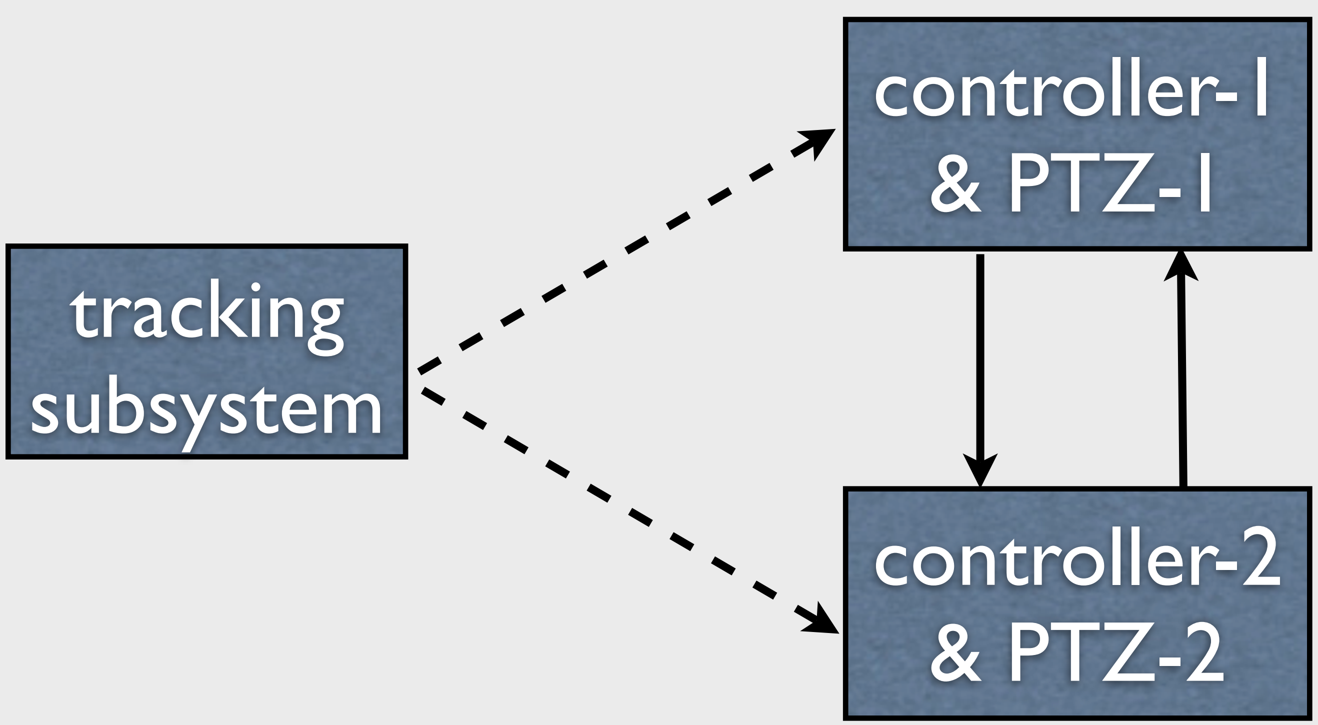
Environment specifications:

- At most N targets at a time.
- Every target remains at least T time steps and eventually leaves.
- Can only enter/exit through doors.
- Can only move to neighbors.

Centralized vs. decentralized control architecture





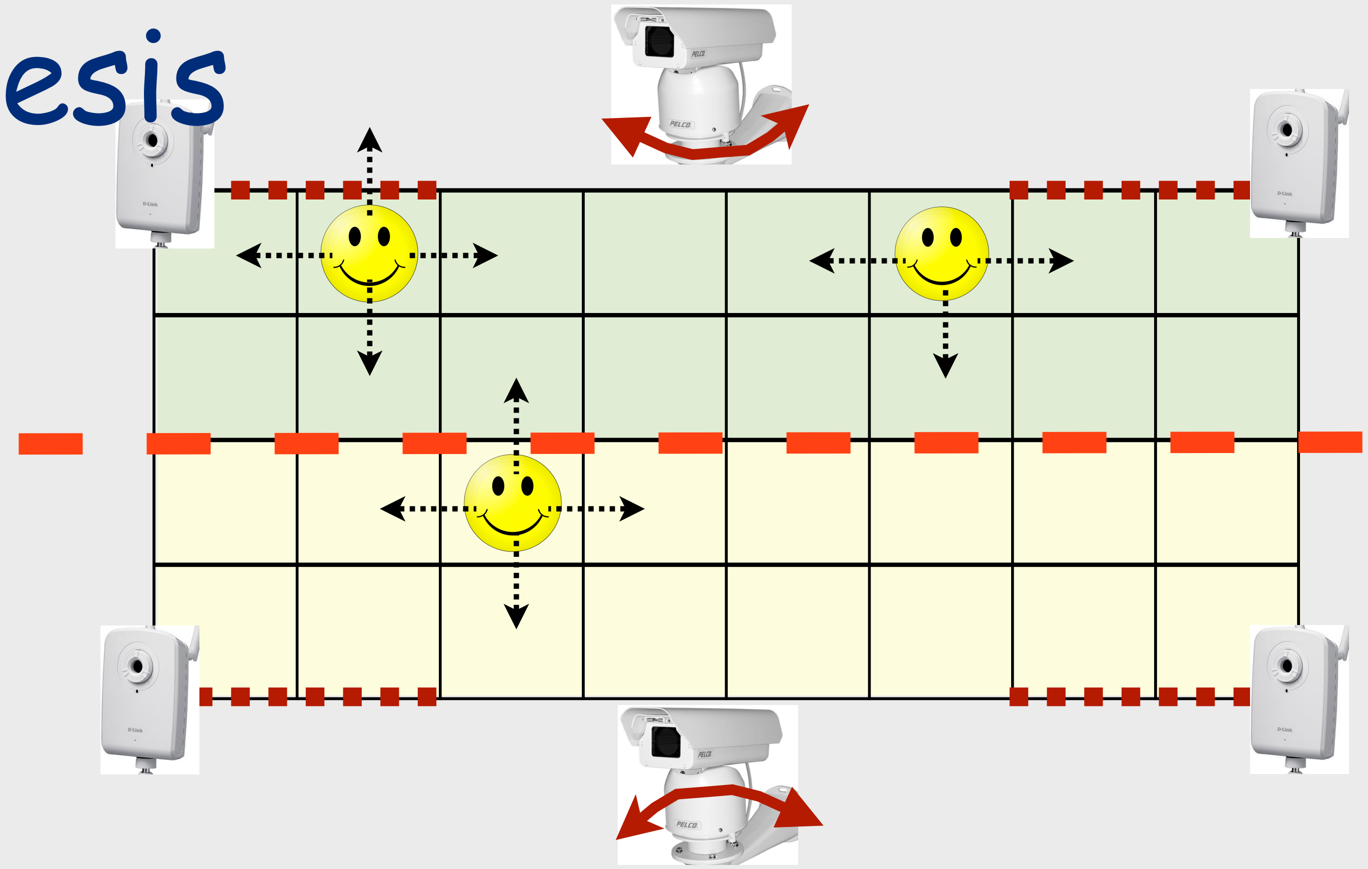


How to design control protocols that can be
- synthesized
- implemented

in a decentralized way?

What information exchange & interface models are needed?

Compositional Synthesis



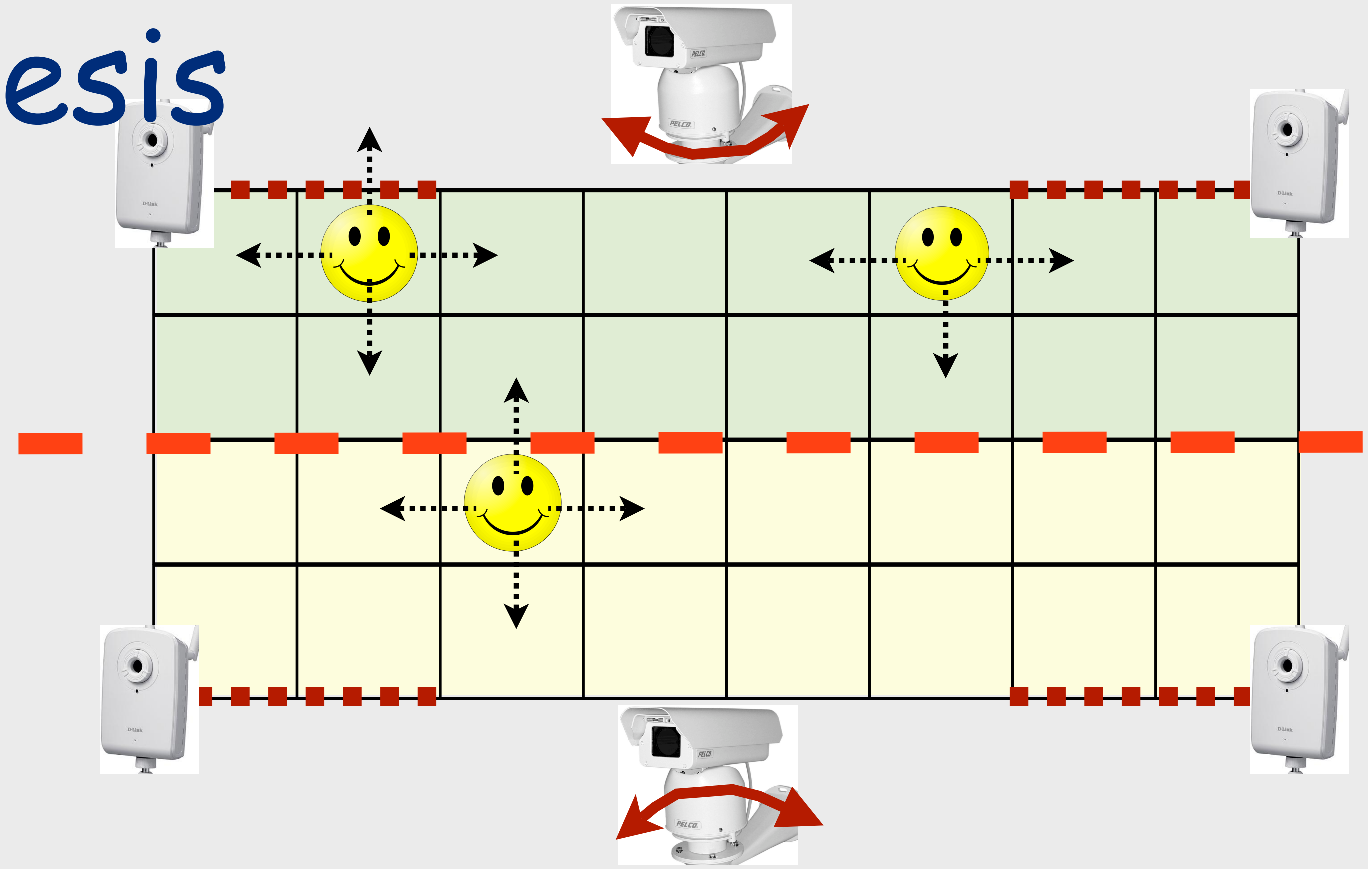
Goal: Find control protocols for PTZ-1 & PTZ-2 so that $\varphi_e \rightarrow \varphi_s$ holds.

Simple & not very useful composition:

- Any execution of the env't, satisfying φ_e, also satisfies $\varphi_{e_1} \wedge \varphi_{e_2}$
- Any execution of the system, satisfying $\varphi_{s_1} \wedge \varphi_{s_2}$, also satisfies φ_s
- No common controlled variables in φ_{s_1} and φ_{s_2}

Compositional Synthesis

<u>Goal</u>: Find control protocols for PTZ-1 & PTZ-2 so that $\varphi_e \rightarrow \varphi_s$ holds.

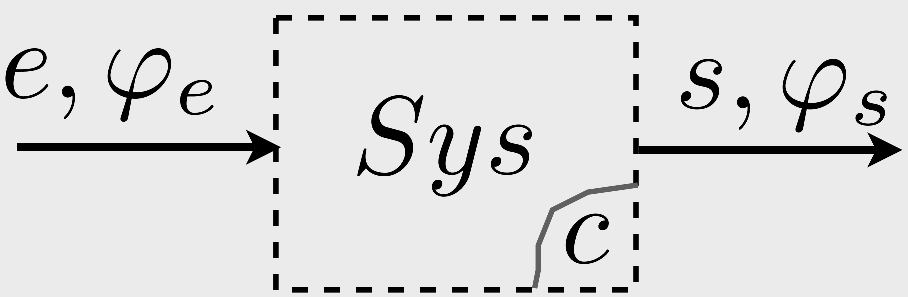


Simple & not very useful composition:

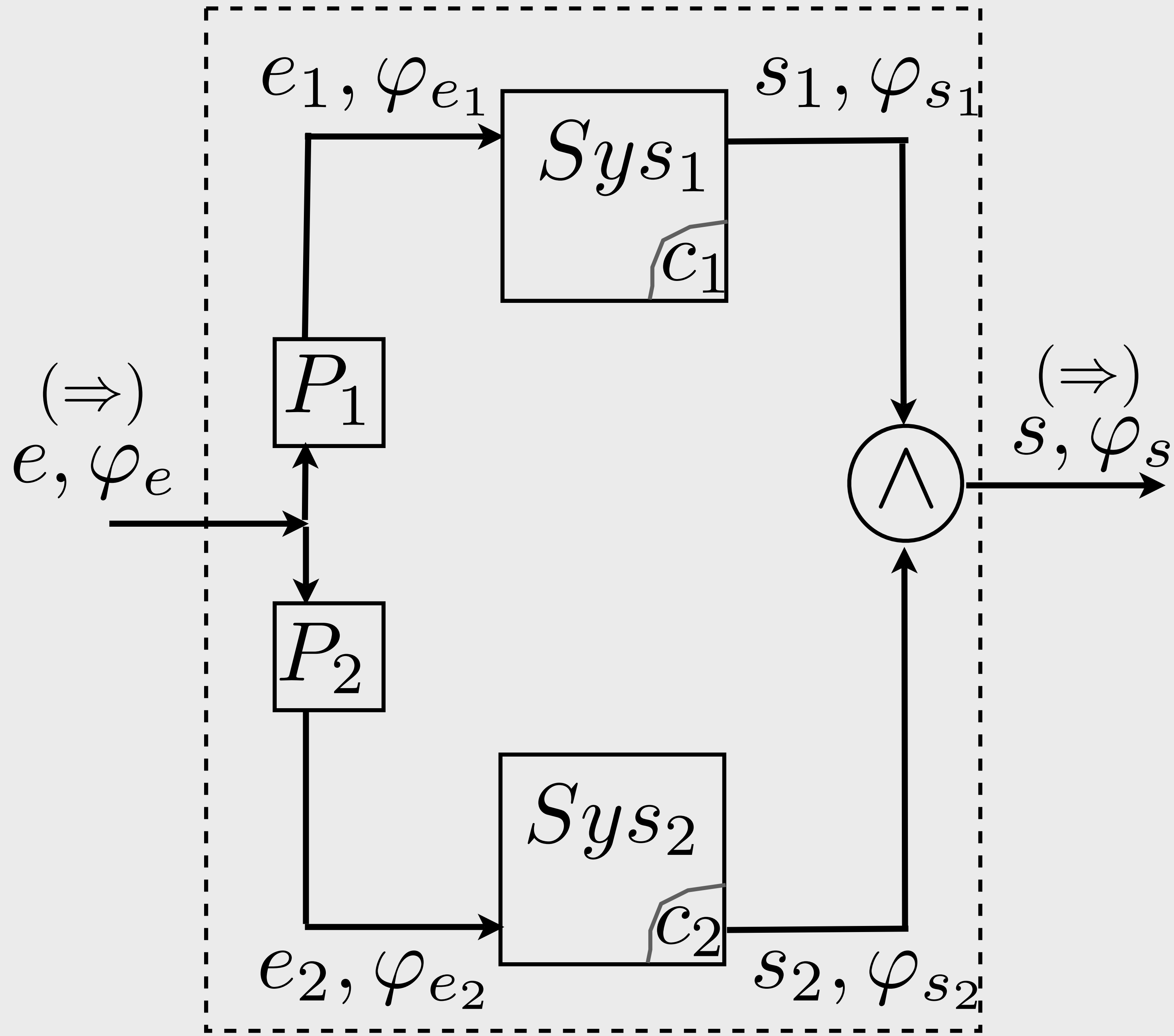
- Any execution of the env't, satisfying φ_e, also satisfies $\varphi_{e_1} \wedge \varphi_{e_2}$
- Any execution of the system, satisfying $\varphi_{s_1} \wedge \varphi_{s_2}$, also satisfies φ_s
- No common controlled variables in φ_{s_1} and φ_{s_2}
- There exist control protocols that realize $\varphi_{e_1} \rightarrow \varphi_{s_1}$ & $\varphi_{e_2} \rightarrow \varphi_{s_2}$

➡ $\varphi_e \rightarrow \varphi_s$ is realized.

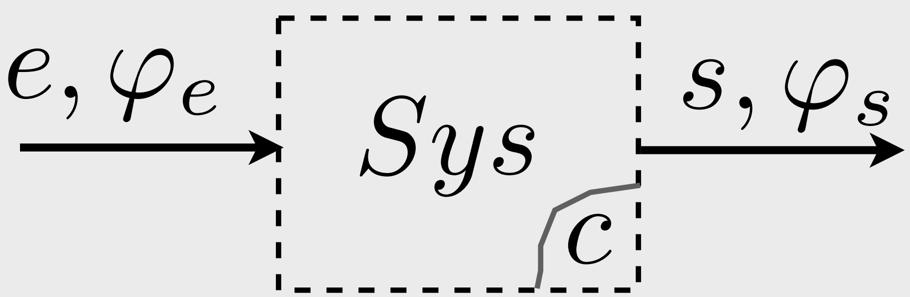
Central



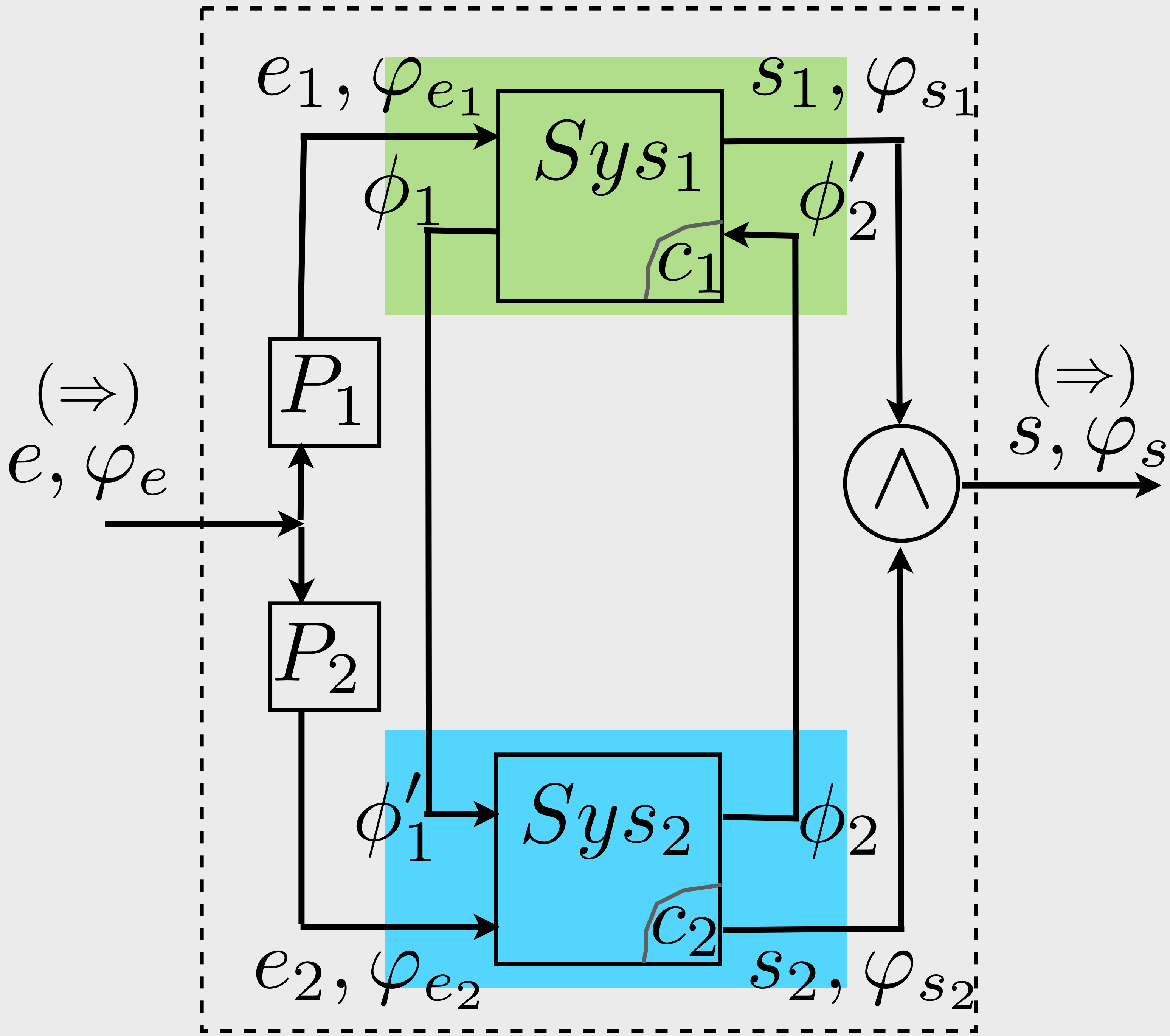
Compositional



Central



Compositional



(Refined) Compositional Synthesis

As before:

Any execution of the env't, satisfying φ_e, also satisfies $\varphi_{e_1} \wedge \varphi_{e_2}$

Any execution of the system, satisfying $\varphi_{s_1} \wedge \varphi_{s_2}$, also satisfies φ_s

No common controlled variables in φ_{s_1} and φ_{s_2}

There exist control protocols that realize $\varphi_{e_1} \to \varphi_{s_1}$ & $\varphi_{e_2} \to \varphi_{s_2}$

➡ $\varphi_e \to \varphi_s$ is realized.

(Refined) Compositional Synthesis

As before:

Any execution of the env't, satisfying φ_e, also satisfies $\varphi_{e_1} \wedge \varphi_{e_2}$

Any execution of the system, satisfying $\varphi_{s_1} \wedge \varphi_{s_2}$, also satisfies φ_s

No common controlled variables in φ_{s_1} and φ_{s_2}

There exist control protocols that realize $\varphi_{e_1} \to \varphi_{s_1}$ & $\varphi_{e_2} \to \varphi_{s_2}$

➡ $\varphi_e \to \varphi_s$ is realized.

(Refined) Compositional Synthesis

As before:

Any execution of the env't, satisfying φ_e, also satisfies $\varphi_{e_1} \wedge \varphi_{e_2}$

Any execution of the system, satisfying $\varphi_{s_1} \wedge \varphi_{s_2}$, also satisfies φ_s

No common controlled variables in φ_{s_1} and φ_{s_2}

Refined interfaces:

There exist control protocols that realize

$(\phi'_2 \wedge \varphi_{e_1}) \rightarrow (\varphi_{s_1} \wedge \phi_1)$ & $(\phi'_1 \wedge \varphi_{e_2}) \rightarrow (\varphi_{s_2} \wedge \phi_2)$

➡ $\varphi_e \rightarrow \varphi_s$ is realized.

(Refined) Compositional Synthesis

As before:

Any execution of the env't, satisfying φ_e, also satisfies $\varphi_{e_1} \wedge \varphi_{e_2}$

Any execution of the system, satisfying $\varphi_{s_1} \wedge \varphi_{s_2}$, also satisfies φ_s

No common controlled variables in φ_{s_1} and φ_{s_2}

Refined interfaces:

There exist control protocols that realize

$(\phi_2' \wedge \varphi_{e_1}) \rightarrow (\varphi_{s_1} \wedge \phi_1)$ & $(\phi_1' \wedge \varphi_{e_2}) \rightarrow (\varphi_{s_2} \wedge \phi_2)$

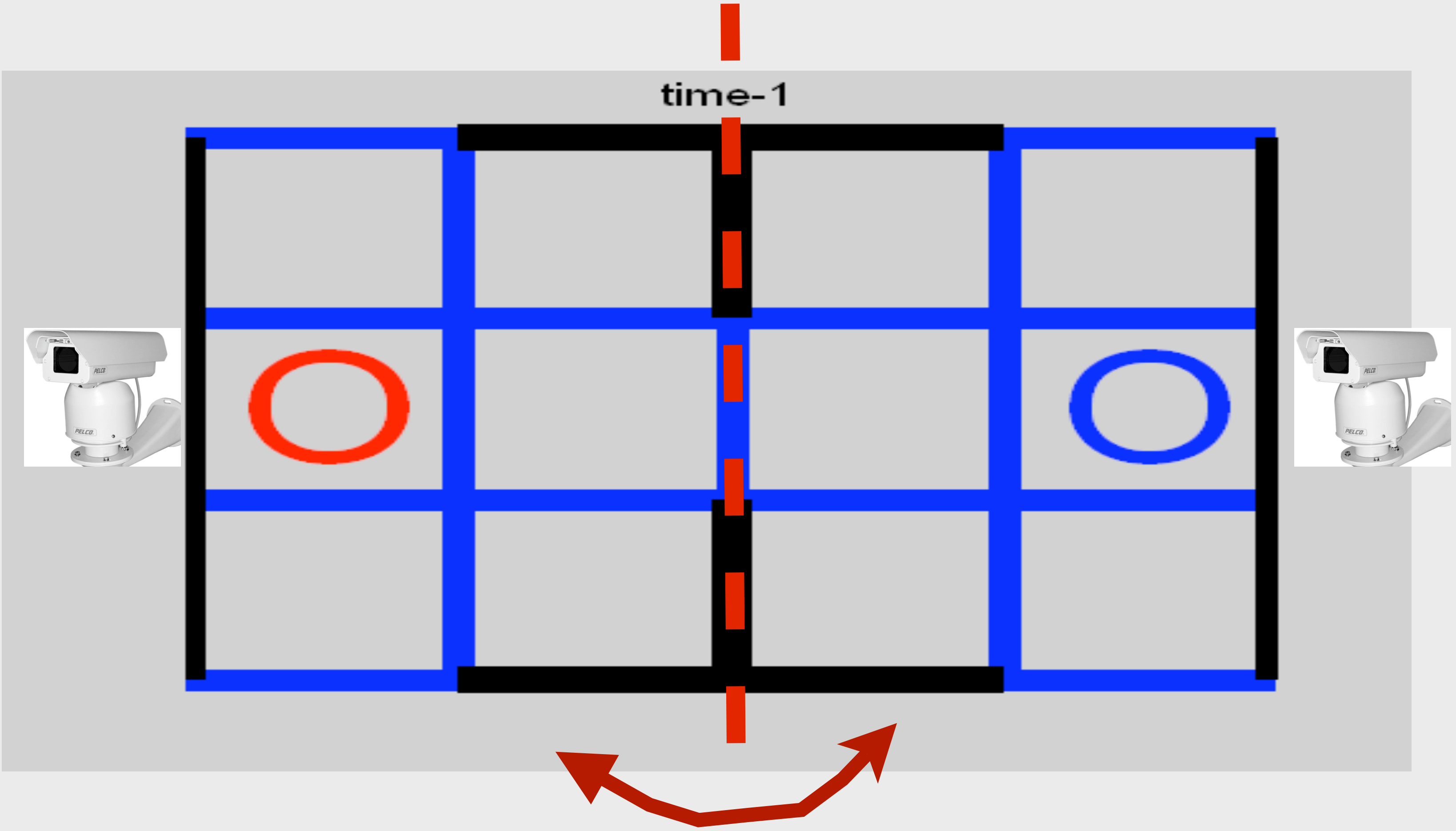
For soundness and to avoid circularity:

$$\Box (\phi_i \rightarrow \circ \phi_i') \quad \text{for } i = 1, 2$$

➡ $\varphi_e \rightarrow \varphi_s$ is realized.

OTWM@ICCPS11(s)

Application to a (very simple) smart camera network



IsZoomed & StepsInZone

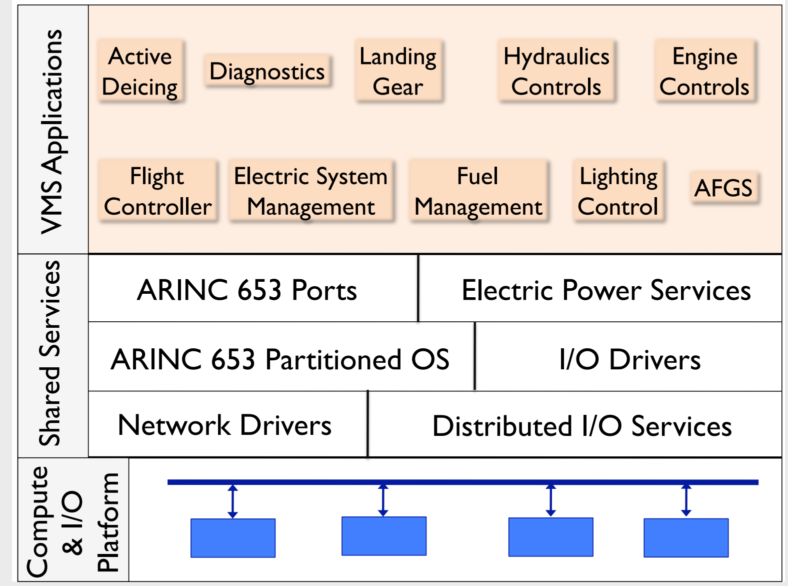
ϕ_1 and ϕ_1'
limit the number of unzoomed targets entering zone 2 from zone 1

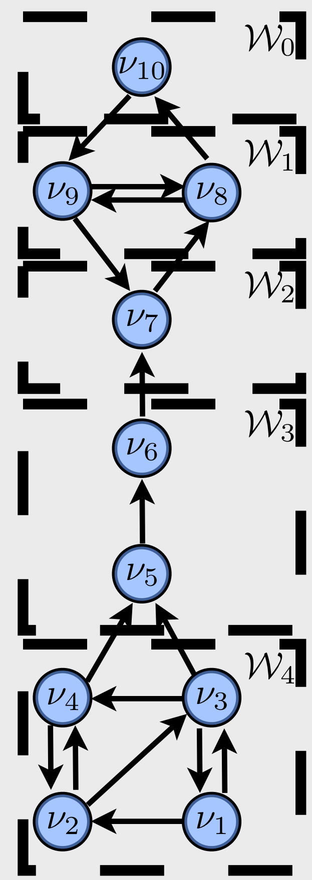
Summary

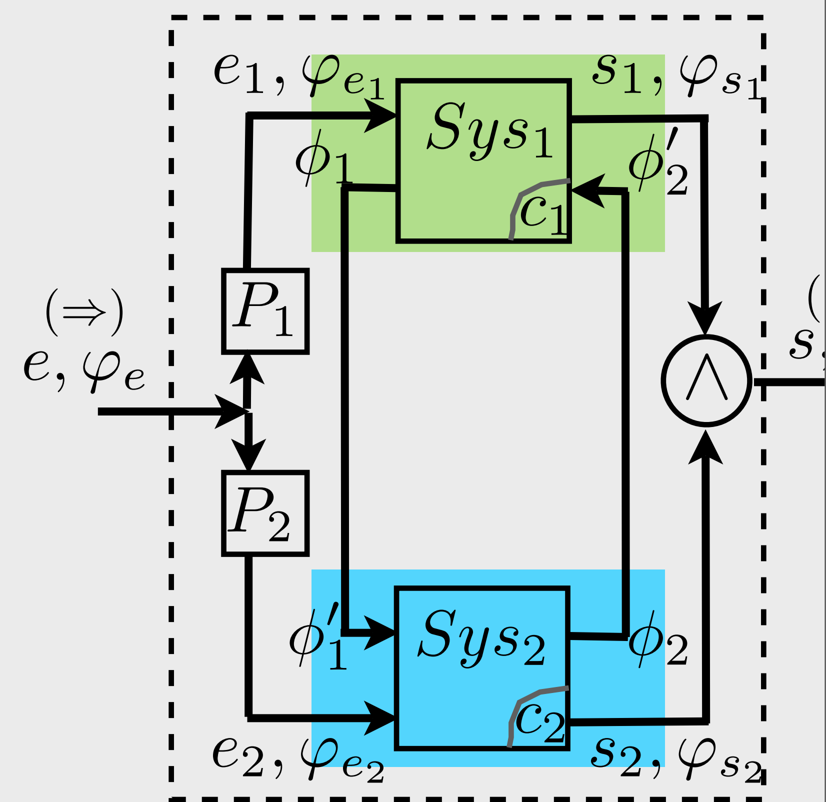
- Receding horizon temporal logic synthesis
- Distributed synthesis
- Applications
  - Vehicle management systems
  - Autonomous driving
  - Active surveillance

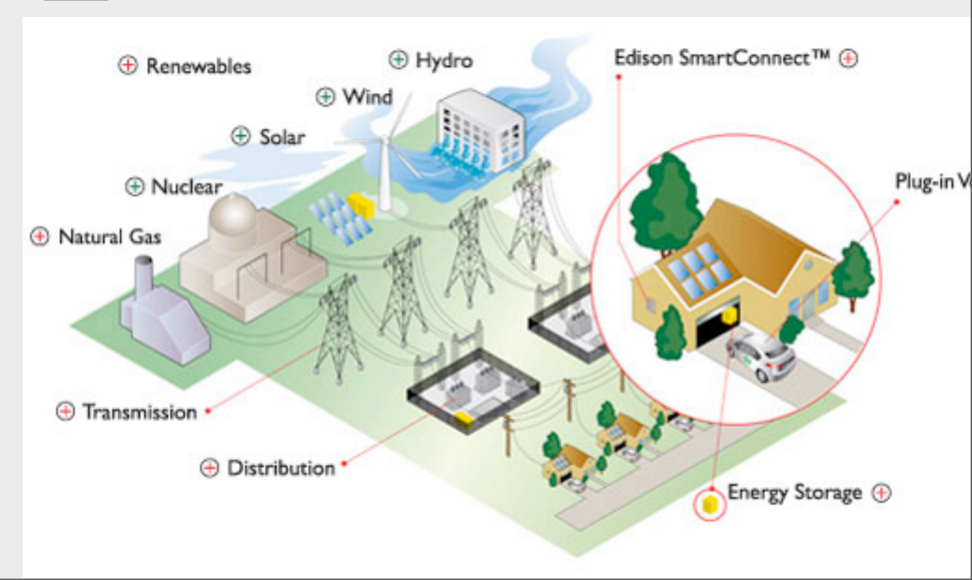
A sample of open issues

- Optimality vs. feasibility
- Hard time constraints
- Incremental synthesis/verification
- Fidelity of models/abstractions
- Exploiting the underlying structure









All references

WTM@CDC09
WTM@AAAI, SS,10
WTM@HSCC10
WTM@ITAC(s)
WTOXM@HSCC11(s)
WTM- Infotech@Aerospace, 2011

available at

www.cds.caltech.edu/~UTopcu