Lecture 9
Advanced Topics in Protocol Synthesis

Ufuk Topcu

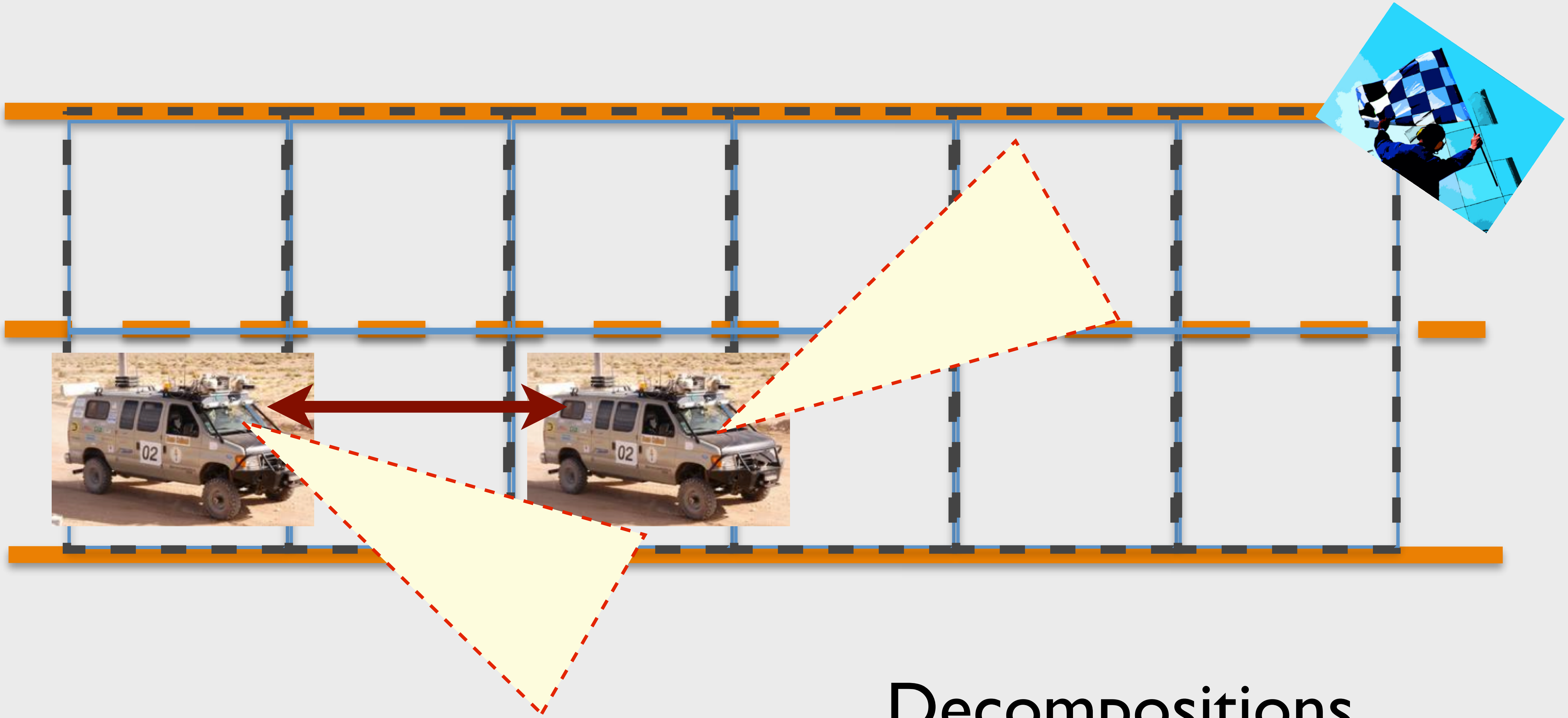
Nok Wongpiromsarn Richard M. Murray

EECI, 22 March 2013

Outline:

- Compositional synthesis of control protocols
- Synthesis of switching sequences
- Optimality in discrete synthesis

Decompositions in the state space

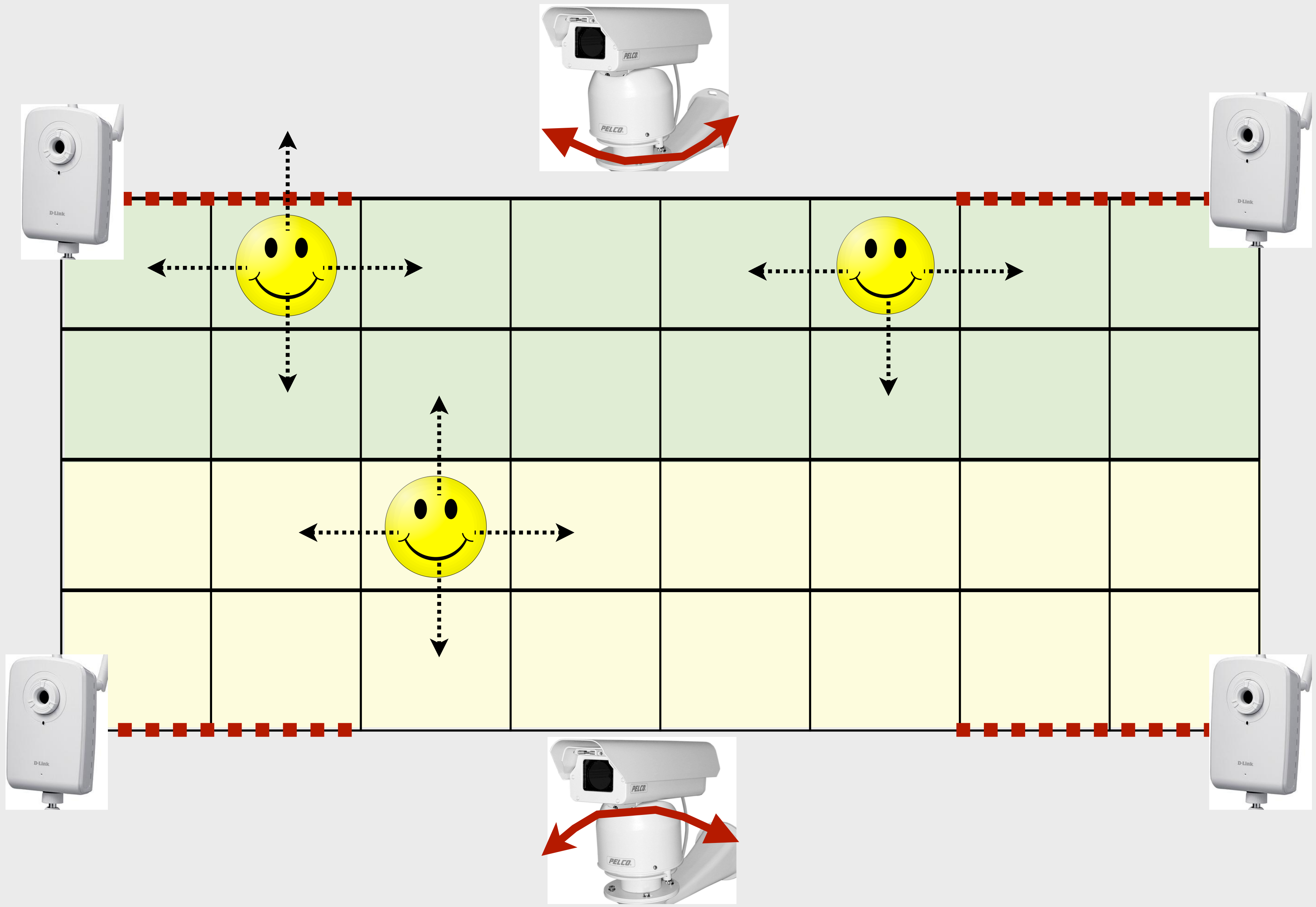


Decompositions induced by ...

receding horizon	goal
distributed synthesis	underlying network

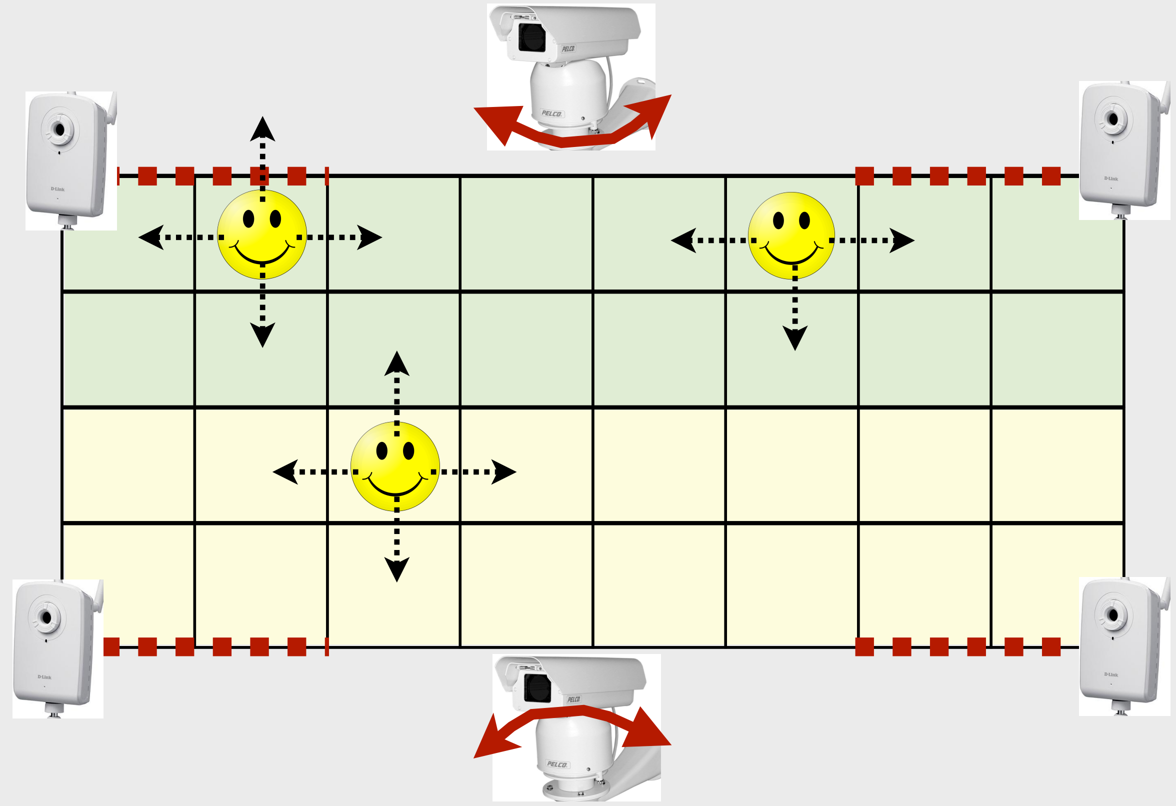
Smart camera networks

- 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

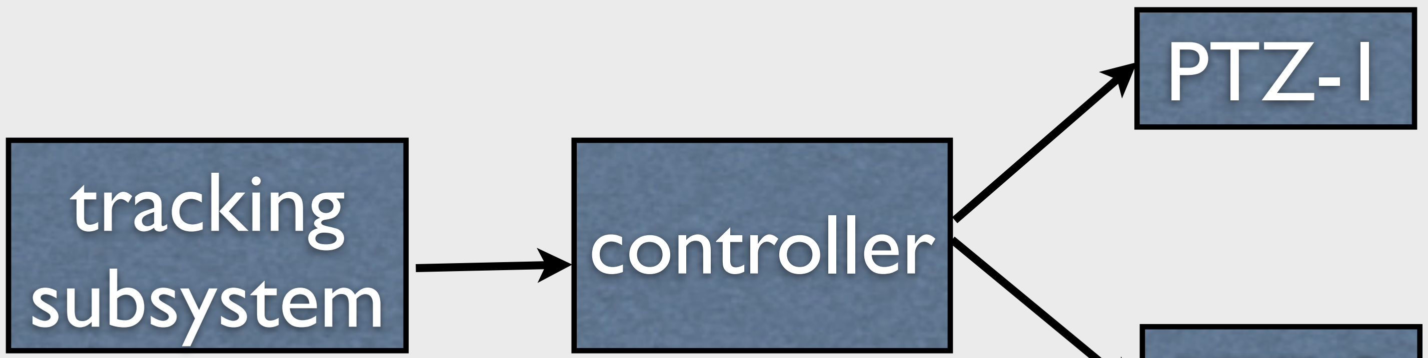


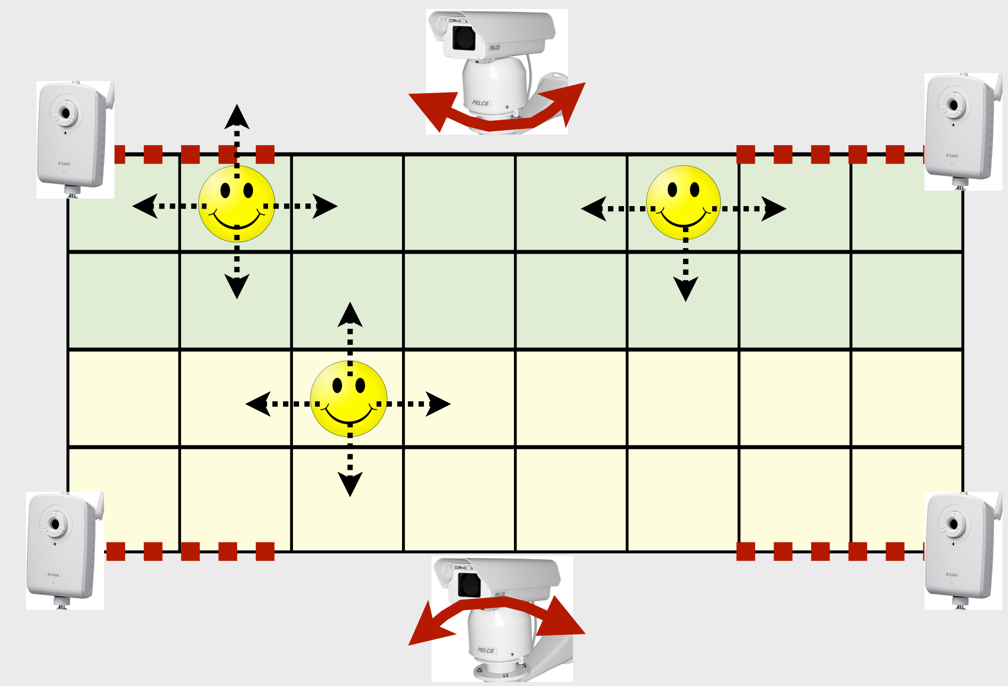
System:
- region of view of PTZs
- governed by finite state automata

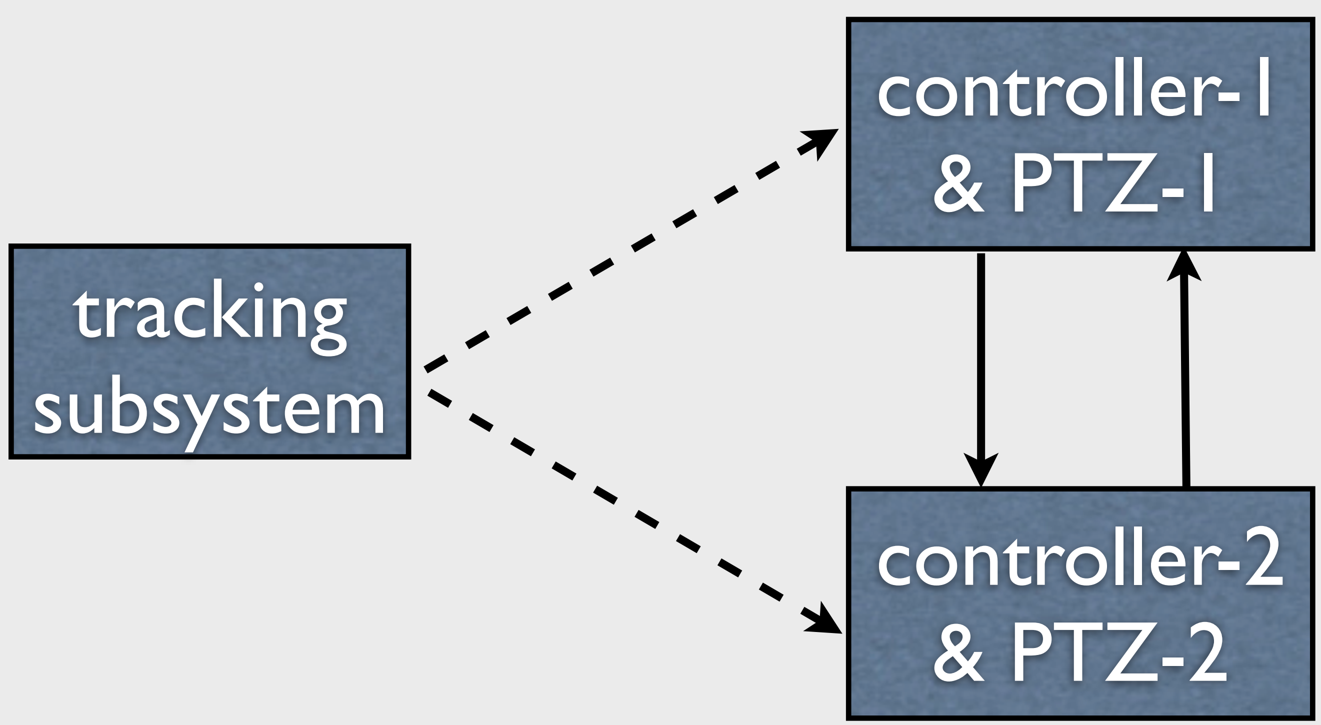
Additional requirement:
- Zoom-in the corner cells infinitely often.

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







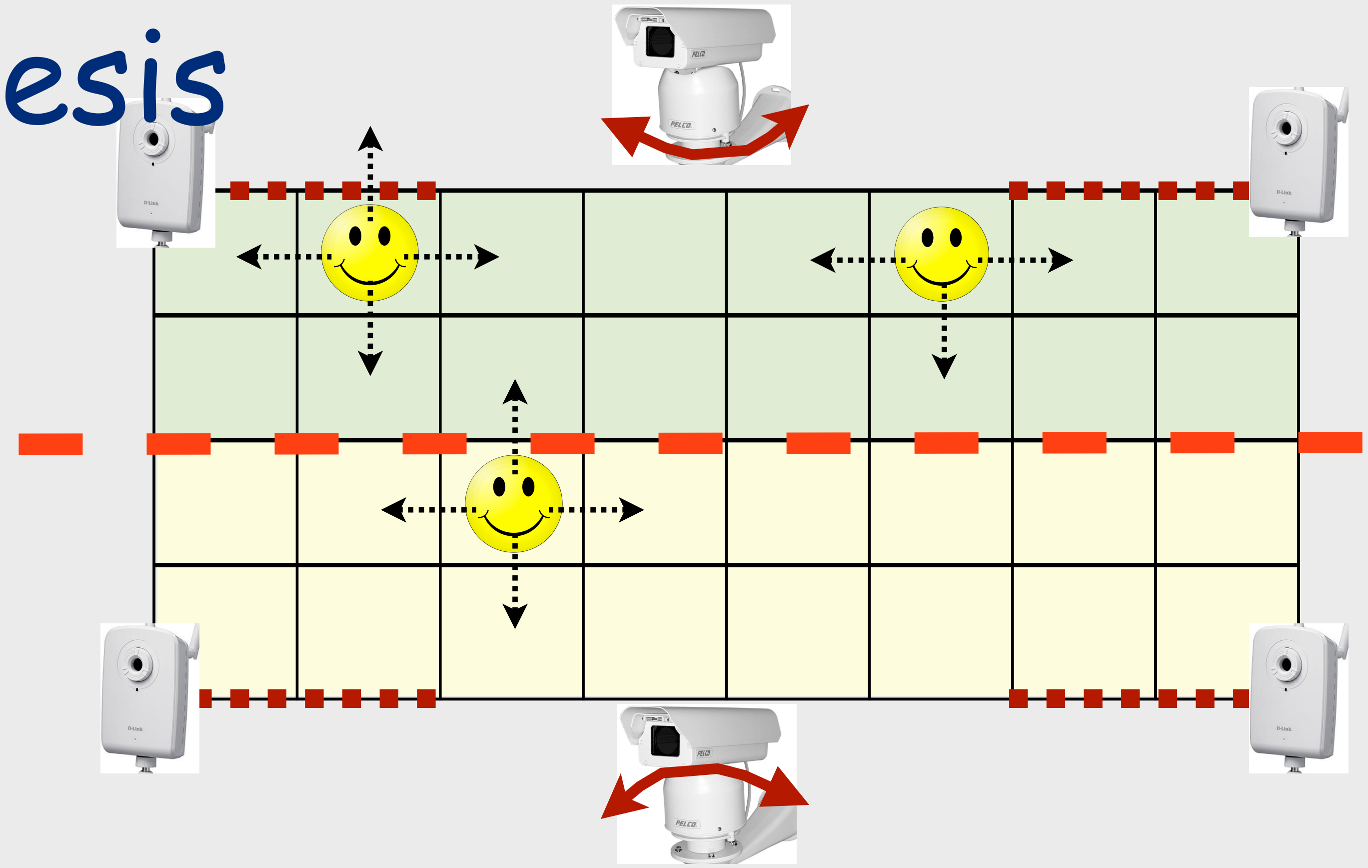
How to design control protocols that can be

- synthesized
- implemented

in a decentralized way?

What information exchange & interface models are needed?

Compositional Synthesis



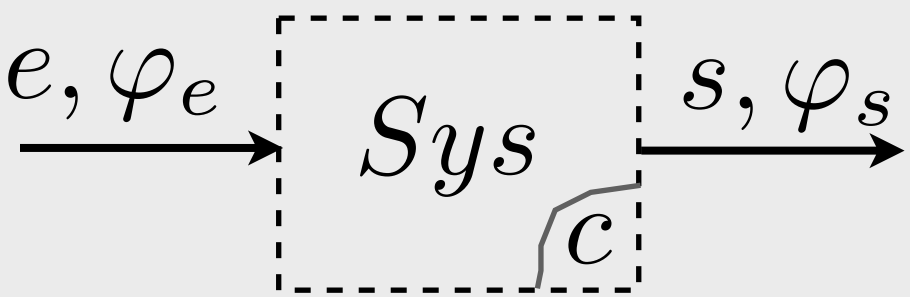
Goal: Find control protocols for PTZ-1 & PTZ-2 so that $\varphi_e \rightarrow \varphi_s$ holds.

Simple & not very useful composition:

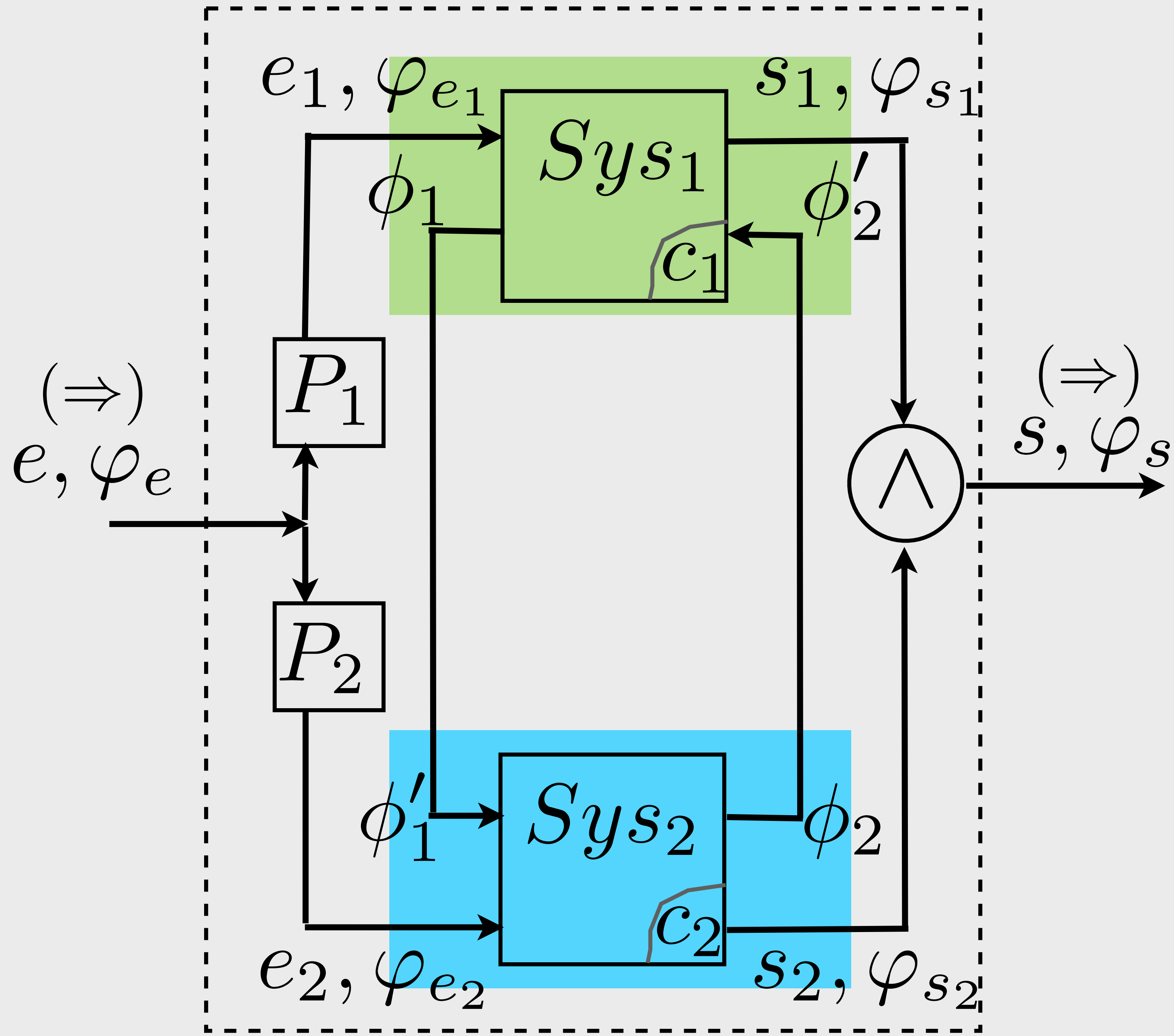
- Any execution of the env't, satisfying φ_e, also satisfies $\varphi_{e_1} \wedge \varphi_{e_2}$
- Any execution of the system, satisfying $\varphi_{s_1} \wedge \varphi_{s_2}$, also satisfies φ_s
- No common controlled variables in φ_{s_1} and φ_{s_2}
- There exist control protocols that realize $\varphi_{e_1}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

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

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 $\varphi_{e_1} \rightarrow \varphi_{s_1}$ & $\varphi_{e_2} \rightarrow \varphi_{s_2}$

Refined interfaces:

There exist control protocols that realize

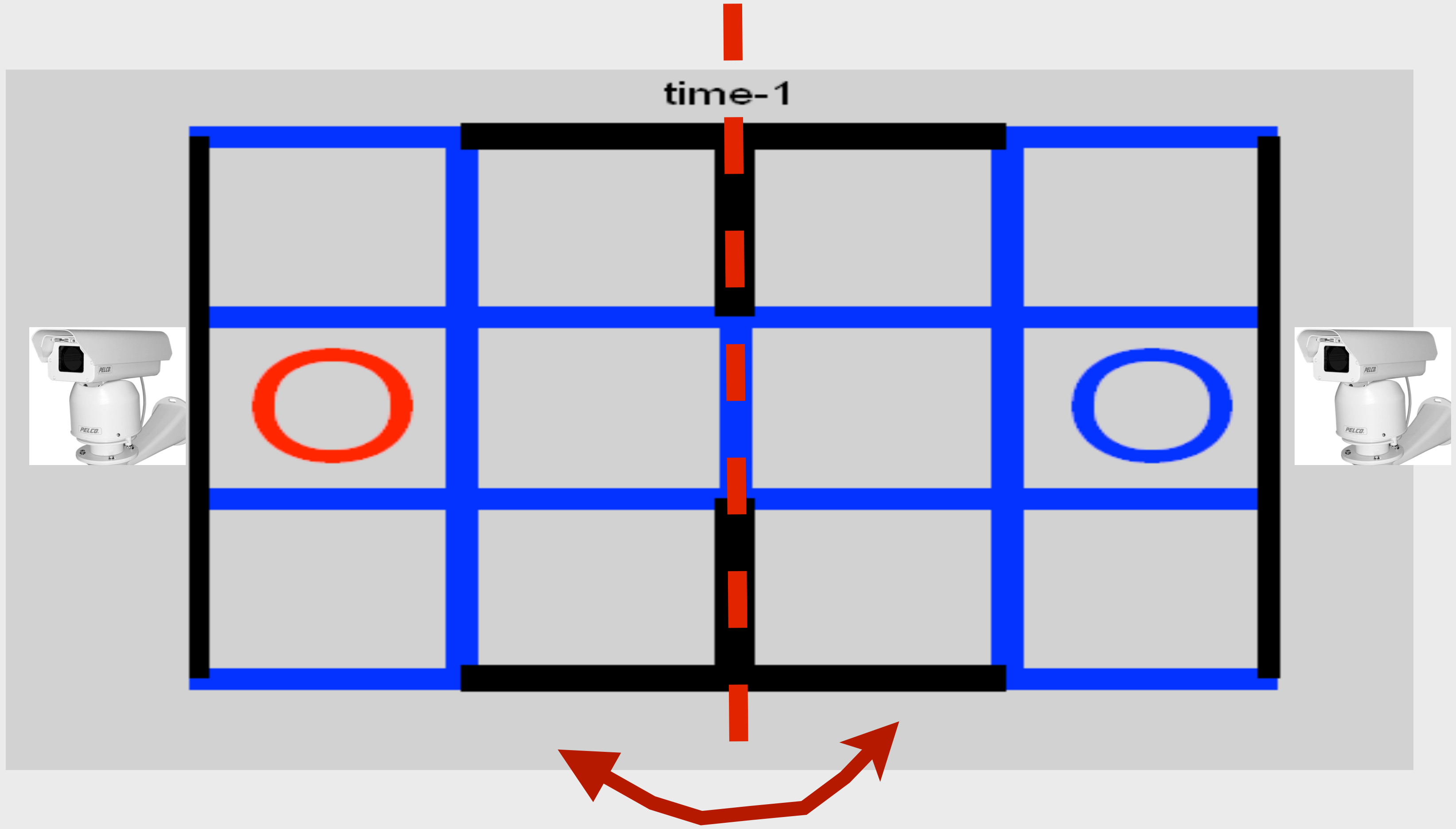
$(\phi_2 \wedge \varphi_{e_1}) \rightarrow (\varphi_{s_1} \wedge \phi_1)$ & $(\phi_1 \wedge \varphi_{e_2}) \rightarrow (\varphi_{s_2} \wedge \phi_2)$

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.

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

Case Study: Synthesis of Protocols for Electric Power Management

<u>Multiple criticality levels</u>:
- flight controllers
- active de-icing
- environmental control

increasing criticality

<u>Environment variables</u>:
- wind gust (w)
- outside temperature (T)

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

Source: http://www.e-envi2009.org/presentations/S3/Derouineau.pdf

For environment & control variables, use crude discretization over their respective ranges. For example,
$T \in \{\text{low}, \text{low-medium}, \text{medium-high}, \text{high}\}$
representing the range of $[-22^o F, 32^o F]$

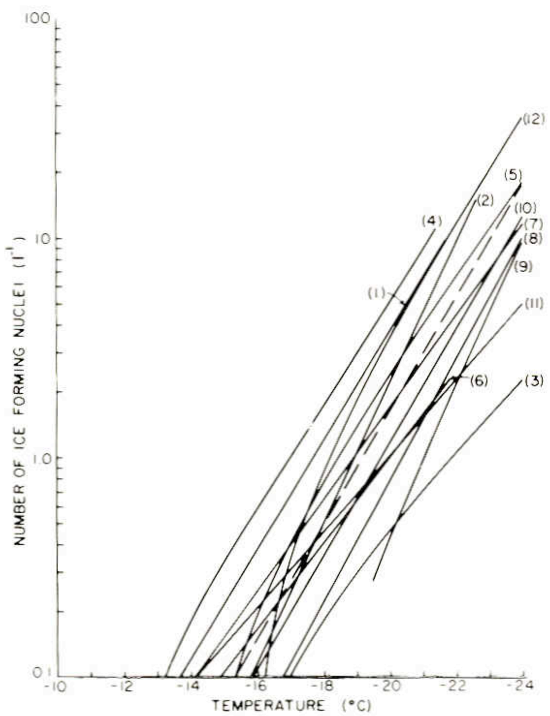
<u>Dependent (state) variables</u>:
- level of ice accumulation
- state-of-charge of the batteries
- cabin pressure level

Modeling & The Dependent Variables

Use models based on finite transitions systems from a combination of empirical data and first principles.

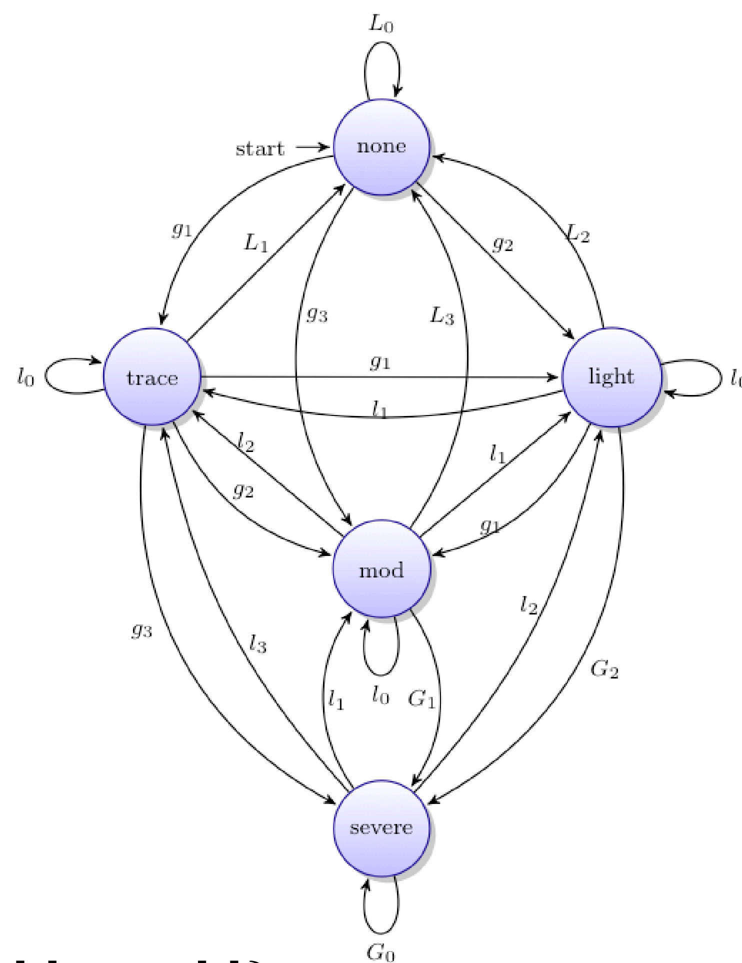
icing level	airspeed reduction	power increase to regain airspeed	climb-rate reduction	reduction in control authority
trace	< 10 knots	$< 10\%$	$< 10\%$	no effect
light	$10 - 19$ knots	$10 - 19\%$	$10 - 19\%$	no effect
moderate	$20 - 39$ knots	$20 - 39\%$	$\geq 20\%$	slow or overly sensitive response
severe	≥ 40 knots	unable	unable	limited or no response

likelihood of icing

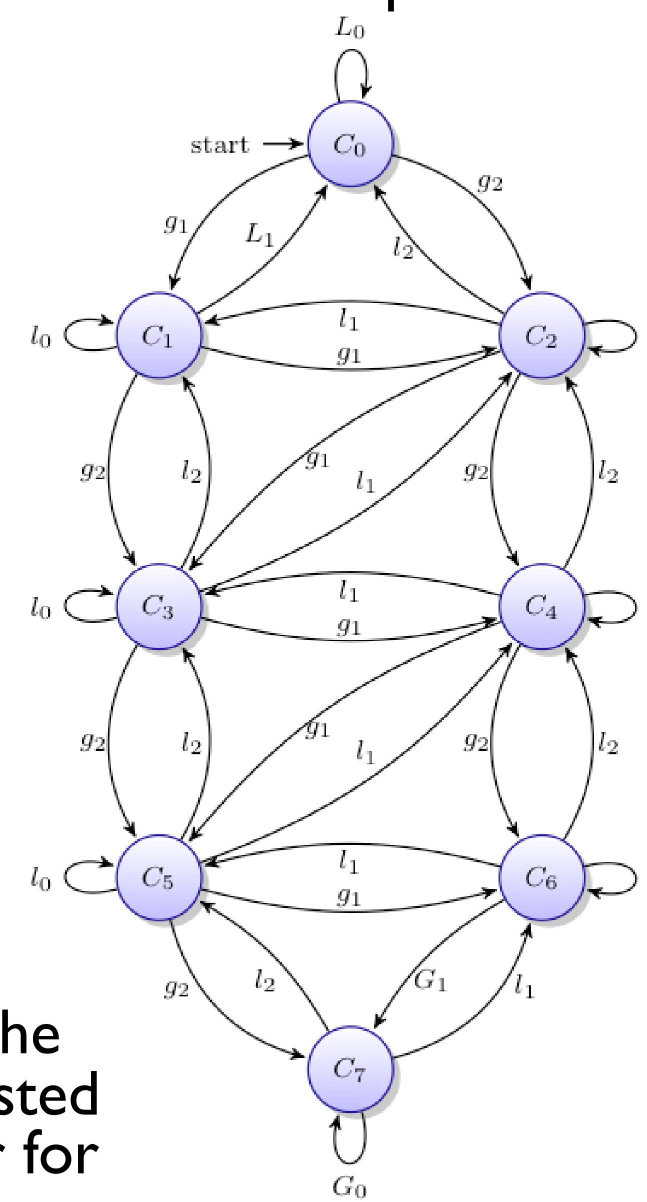


temperature

model of icing level



model of cabin pressure level



State-of-charge evolves with:

$$b[t+1] = \min\{B, b[t] + \bar{P} - p_f[t] - p_d[t] - p_e[t]\}$$

storage capacity

generation capacity

power supply to each functionality

Transitions model the gap between requested and supplied power for each functionality.

Sample Specifications

power requests from flight controller (f), deicing (d), and pressure control (e):

$$r_f \equiv r_f(h, a, w)$$
$$r_d \equiv r_d(T, h)$$
$$r_e \equiv r_e(T, h)$$

Resource constraint: $\square(p_f + p_d + p_e \leq \bar{P} + b)$

Prioritization: $\square(p_f \geq r_f)$

$\square(p_f = \text{high} \wedge p_d = \text{high} \Rightarrow p_e = \text{low})$

Safety: Altitude cannot change too much between to consecutive instants, e.g.,

$\square\,(h = \text{low} \Rightarrow (\circ h \neq \text{medium-high} \wedge \circ h \neq \text{high}))$

Ice accumulation limits allowable altitude change, e.g.,

$\square(a = \text{severe} \Rightarrow \circ h = h)$

Ice accumulation cannot be severe: $\square(a \neq \text{severe})$

Performance: Cabin pressure does not exceed the level at 8000 ft.

Always go back to the desirable altitude: $\square \diamond (h = \text{high})$

Assumptions: Wind gusts cannot be severe too many consecutive steps.

$\square(n_w \geq N_w \Rightarrow \circ(w \neq \text{severe})$

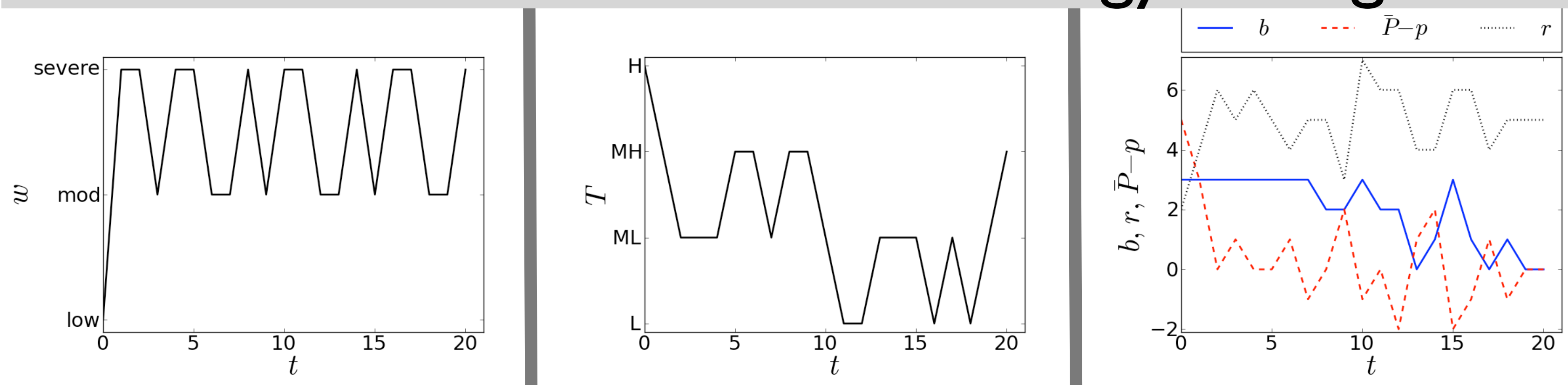
No abrupt change in outside temperature, e.g.,

$\square(T = \text{medium-low} \Rightarrow \circ T \neq \text{high})$

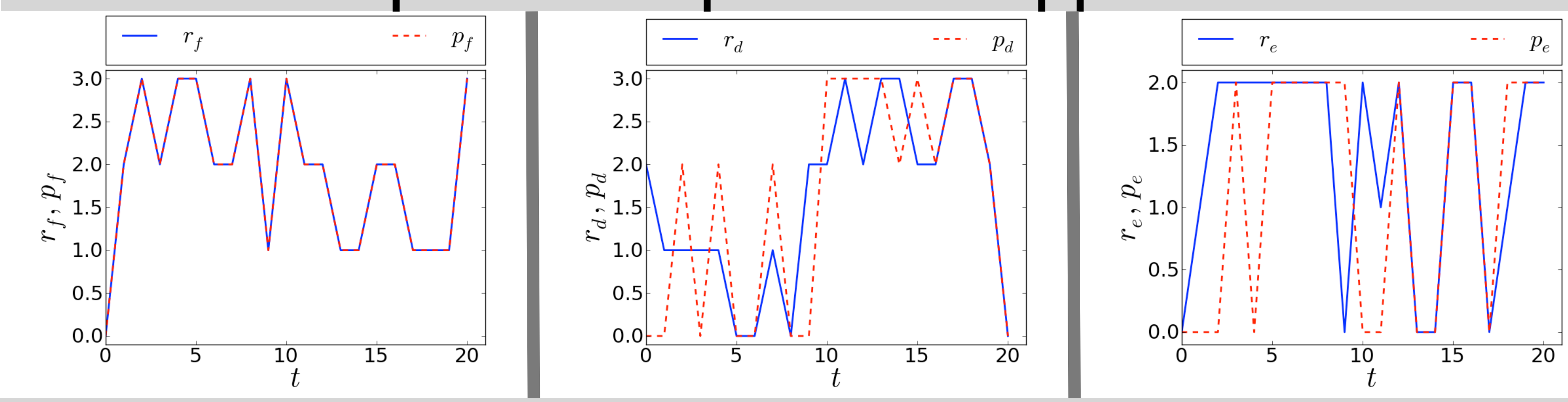
Notation may not be fully explained. Ask, if confused!!!

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

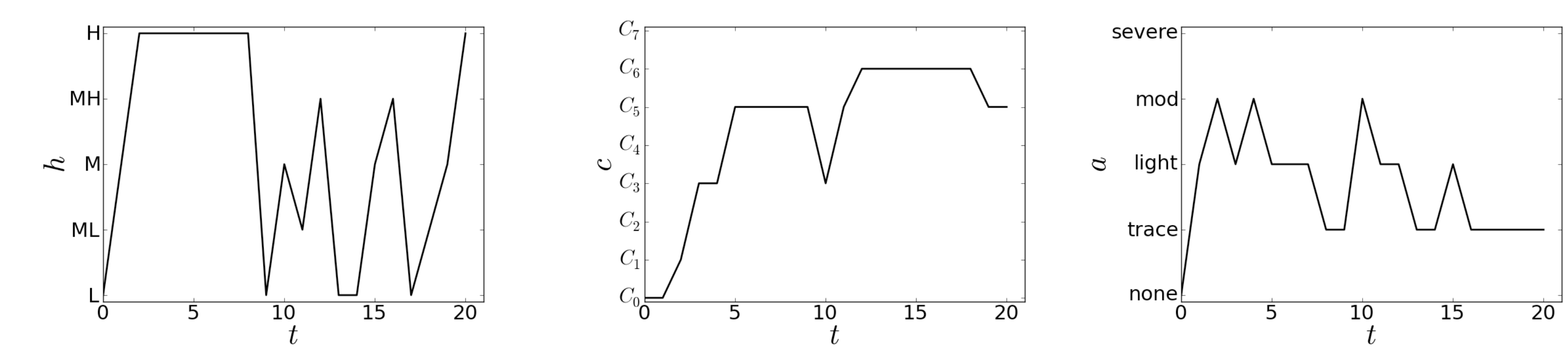
environment variables & energy storage



power requests & supplies



dependent variables



Formulate as a temporal logic, reactive planning problem

$$(\varphi_{environment} \wedge \varphi_{initial} \wedge \varphi_{criticality}) \rightarrow (\varphi_{performance} \wedge \varphi_{safety})$$

$N_w = 2, B = 3$

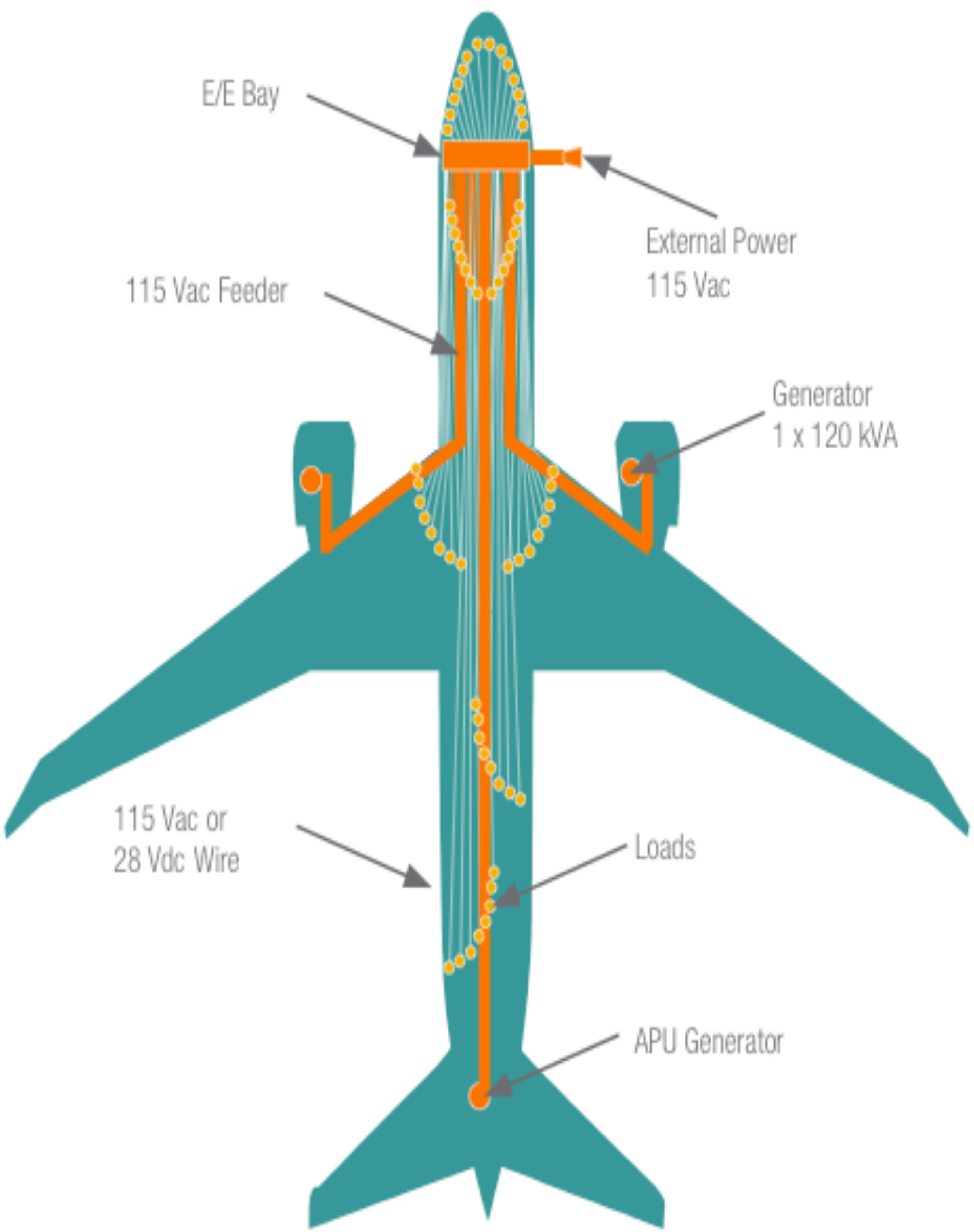
$\bar{P} = 5$

$r_f, r_d \in \{0, 1, 2, 3\}$

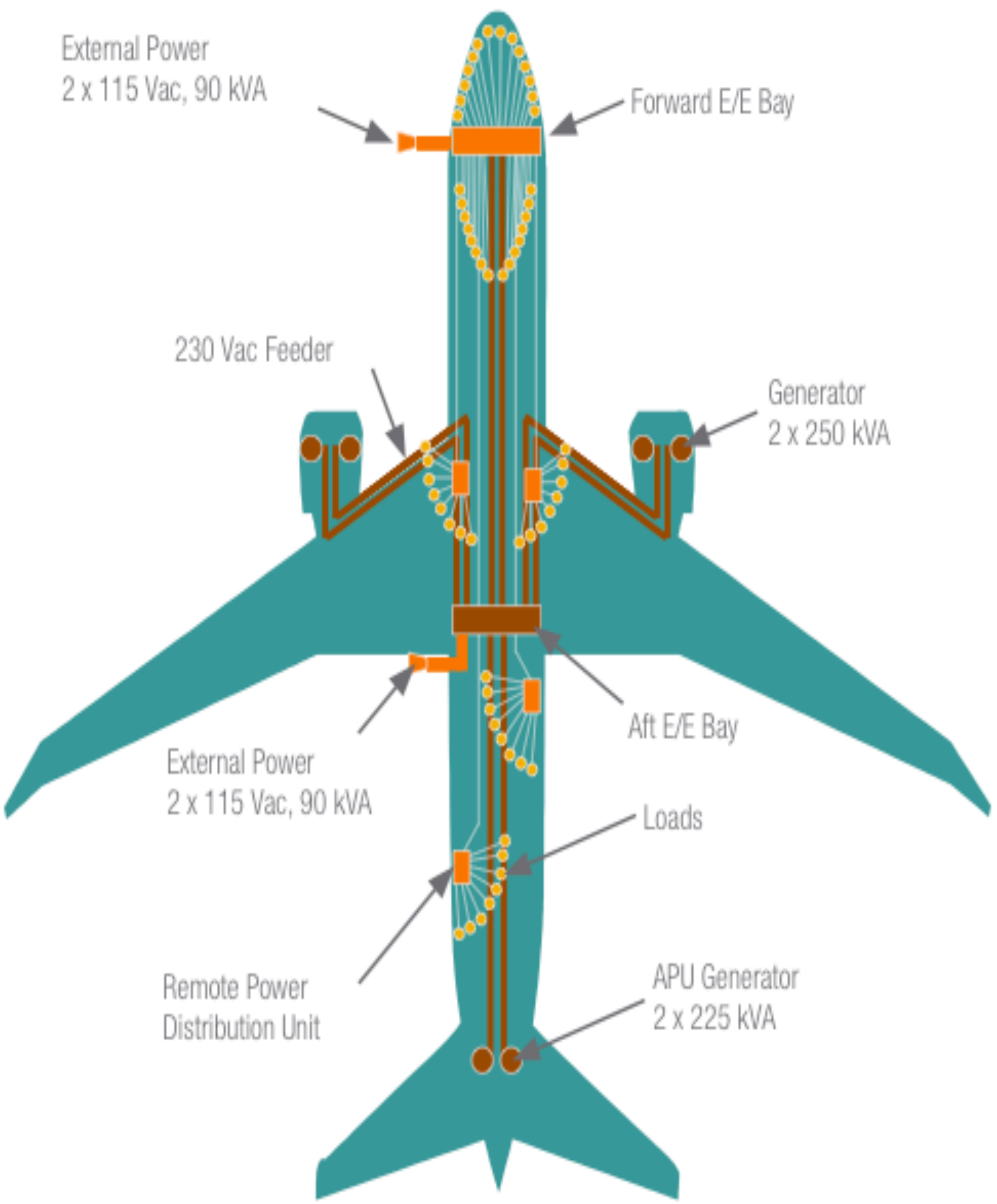
$r_e \in \{0, 1, 2\}$

Conventional vs. Boeing 787 Electric Power Network Structure

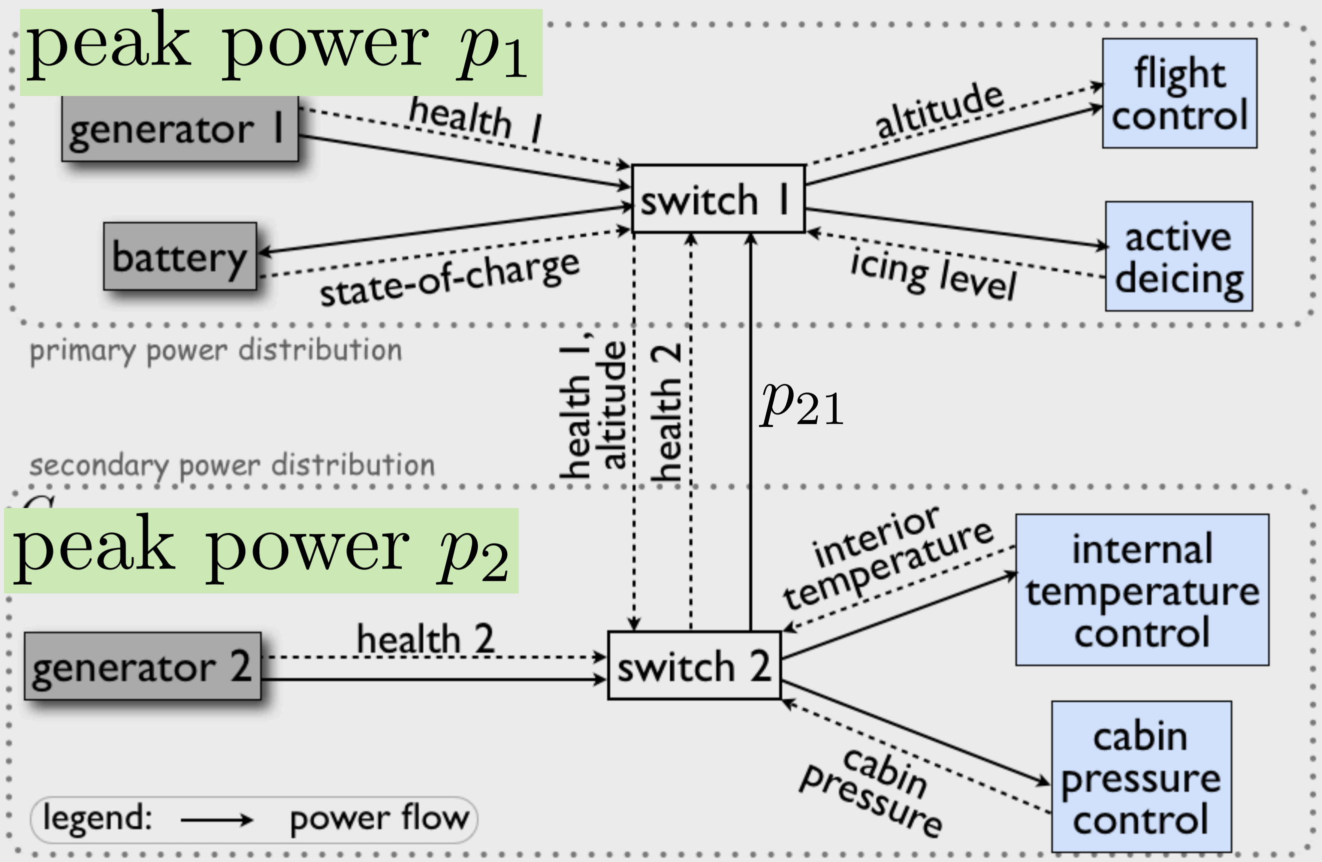
pre-787



787: distributed



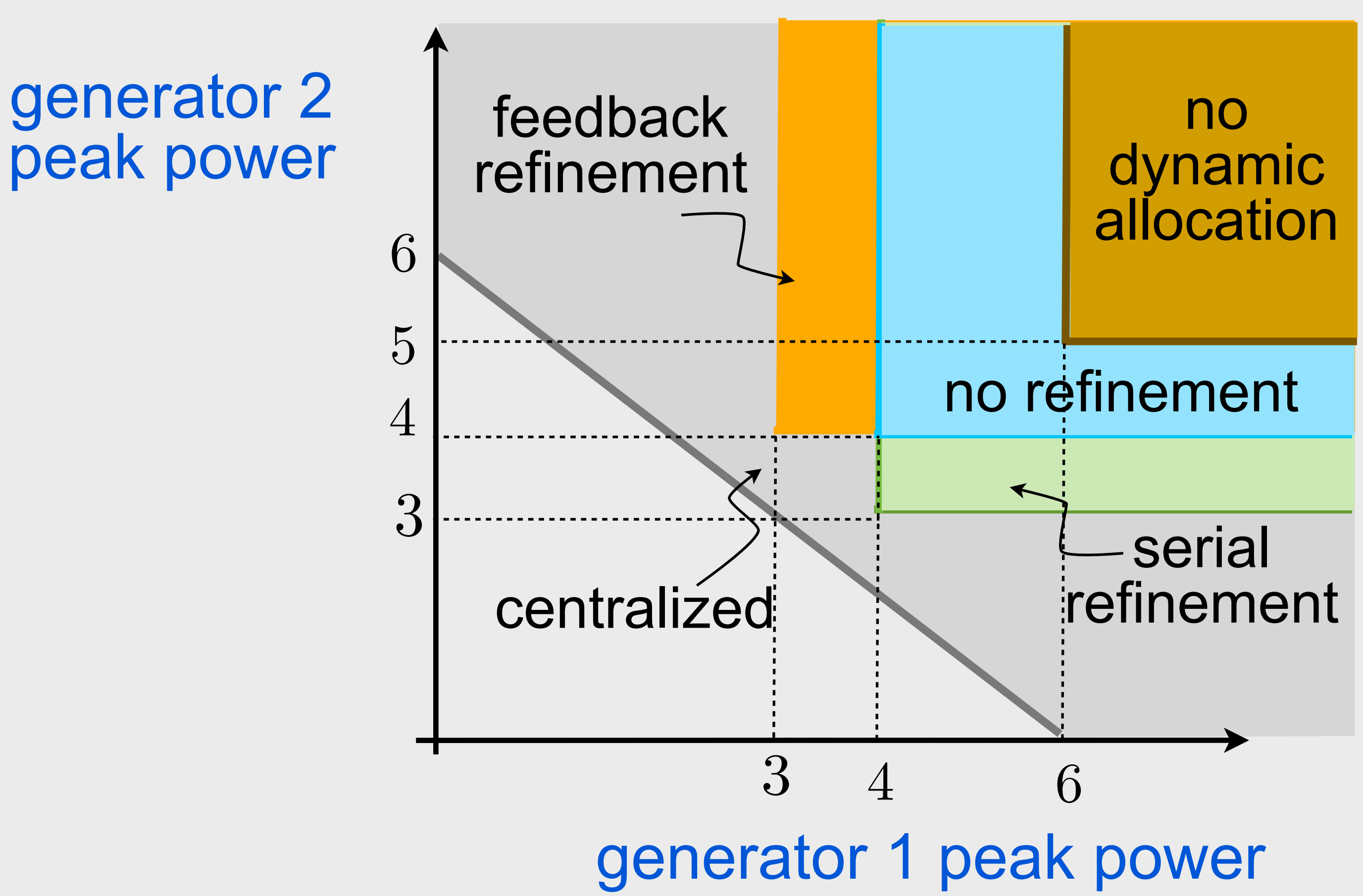
Distributed resource allocation



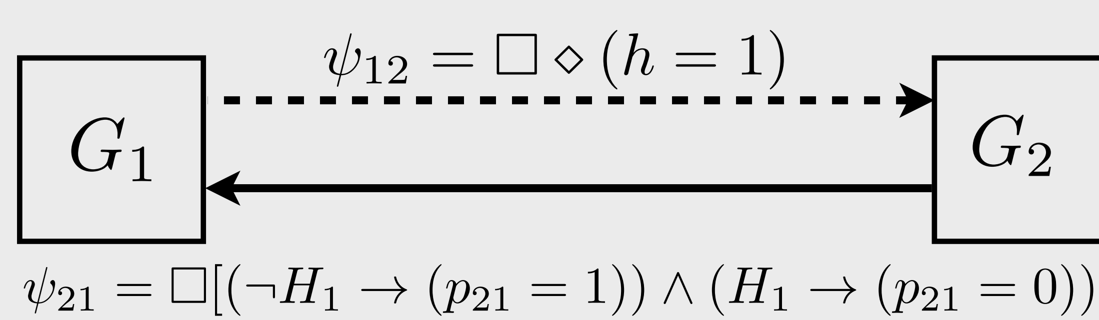
Controlled variables:
- Power supplies to each function
- Altitude

Environment variables:
- Wind gusts
- Outside temperature
- Generator health status

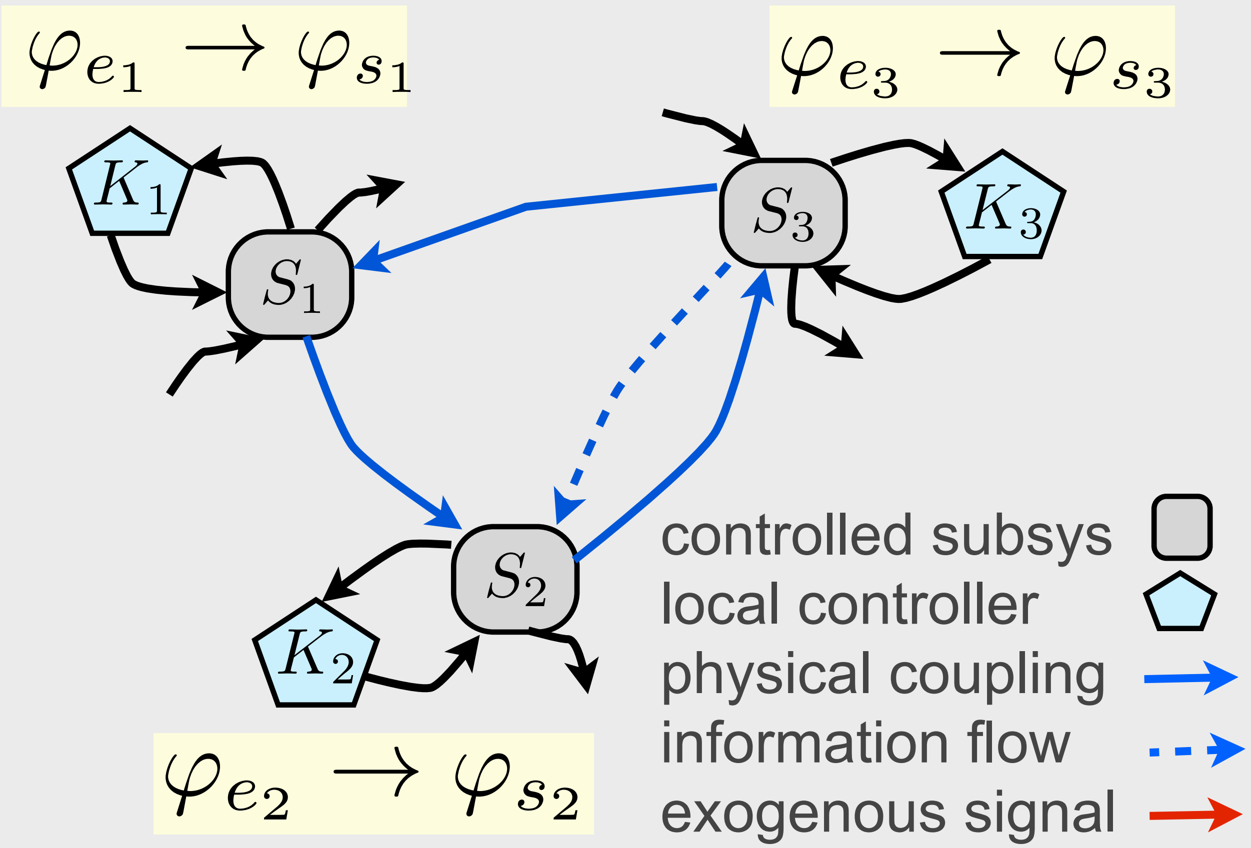
Dependent variables:
- Level of ice accumulation
- State-of-charge of the battery
- Cabin pressure & temperature



Interface refinements



Compositional Synthesis of Distributed Protocols



$$\underbrace{\varphi_e}_{\text{"weaker" environment assumptions}} \rightarrow \underbrace{\varphi_s}_{\text{"stronger" system requirements}}$$

Extra (mild) technical conditions: No common controlled variables & loops are well-posed.

Theorem: $\varphi_e \rightarrow \varphi_s$ is realizable if every $\varphi_{e_i} \rightarrow \varphi_{s_i}$ is realizable.

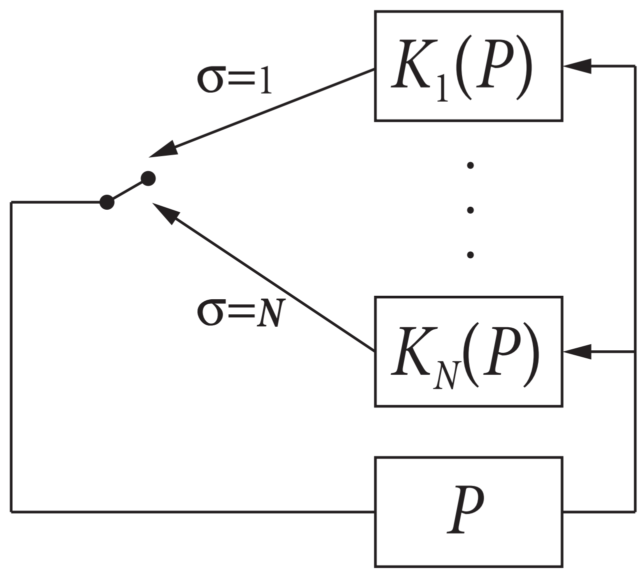
Contracts formalize the coupling and information exchange between subsystems.

Trade-offs:

conservatism vs. expressiveness of contracts vs. need for coordination & computational cost

Slide courtesy of Jun Liu

Motivation -- Switching control protocols



How to "automatically" design **switching control protocols** (high-level discrete decision making) that:

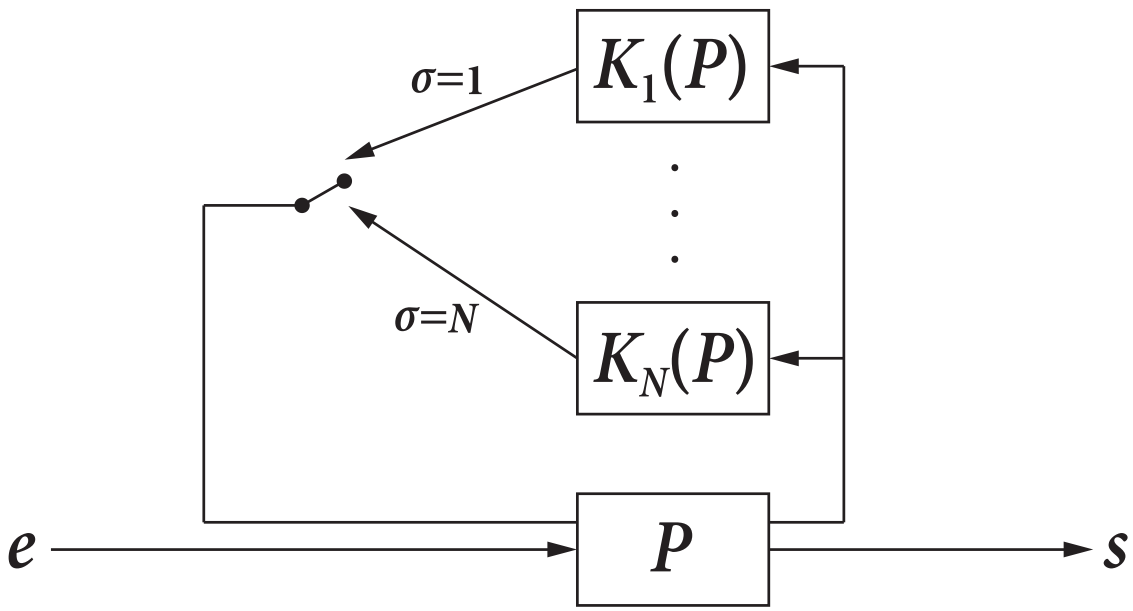
- regulate low-level, continuous dynamics;
- realize high-level specifications; and
- respond to external events in real-time,

with "**correctness guarantee**".

Why switching controllers?

- Situations where continuous, smooth feedback controllers are not sufficient/available.
- No need to redesign underlying dynamics/controllers.
- Handle mixture of discrete and continuous decision making.

Schematic description



$$\varphi \triangleq (\varphi_e \rightarrow \varphi_s)$$

Assumption   Guarantee

P	*plant*
K_i	*controllers*
σ	*switching protocol*
e	*environment signal*
s	*system behavior*
φ	*system specification*
φ_e	*environment assumption*
φ_s	*system guarantee*

Design σ such that P satisfies φ

System model

Continuous-time switched systems:

$$\dot{x} = f_p(x, d), \quad p \in \mathcal{P} \tag{1}$$

$\mathcal{P}$	a finite collection of modes
$\{f_p\}$	nonlinear vector fields
$x \in X$	system state
$d \in D$	exogenous disturbance

System specification

- **Linear temporal logic** (LTL) extends propositional logic with temporal operators:

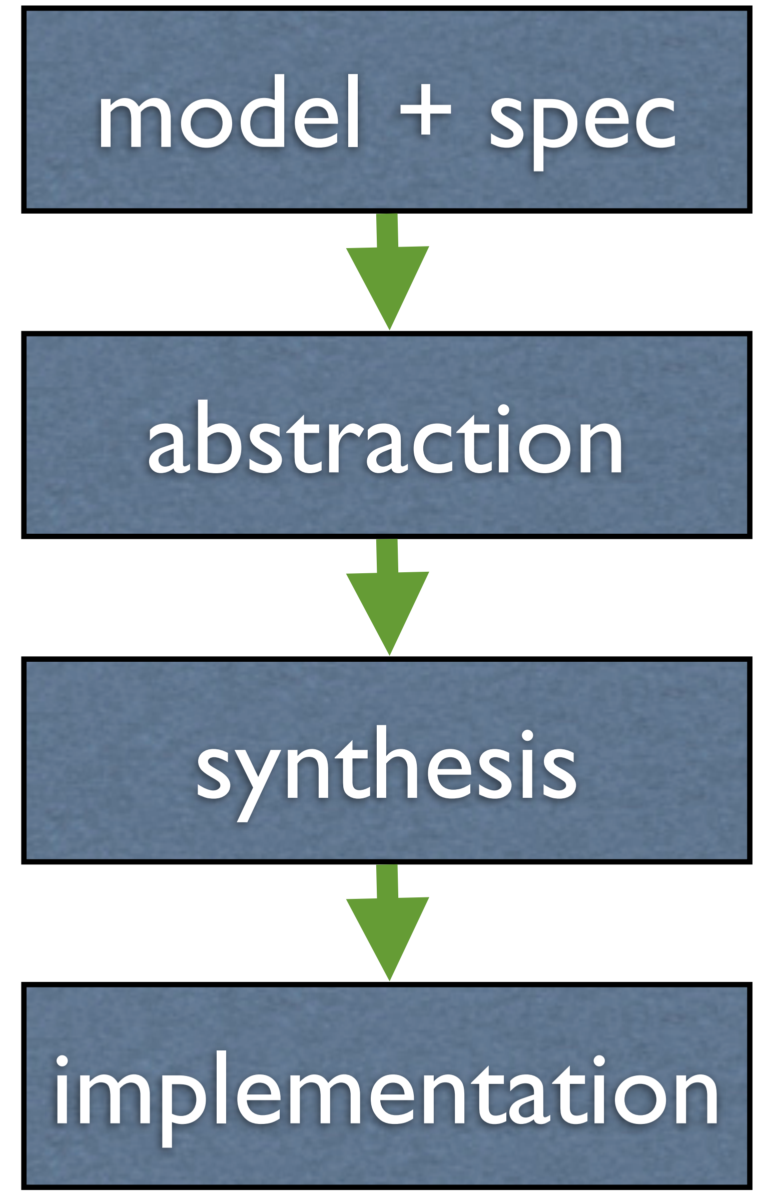
$\wedge$(and), $\vee$ (or), $\neg$ (not), $\rightarrow$ (imply)

$\Diamond$(eventually), $\Box$(always), $\bigcirc$(next), U (until) ...

- Allows to reason about infinite sequences of states
  - state: an evaluation of all variables (environment+system)
- LTL formulas can 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. Only ordering of events.
- LTL-*x*: we consider LTL without the $\bigcirc$(next) operator to specify system properties for continuous-time systems.

Overview of solution strategy

- Given
  $\dot{x} = f_p(x, d), \quad p \in \mathcal{P}$ and $\varphi \triangleq (\varphi_e \rightarrow \varphi_s)$
- Compute finite-state, proposition preserving approximations.
- Solve a discrete synthesis problem and obtain a discrete switching strategy σ.
- Implement the switching strategy σ *continuously* to ensure that all trajectories of $\dot{x} = f_\sigma(x, d)$ satisfies φ.



Questions:

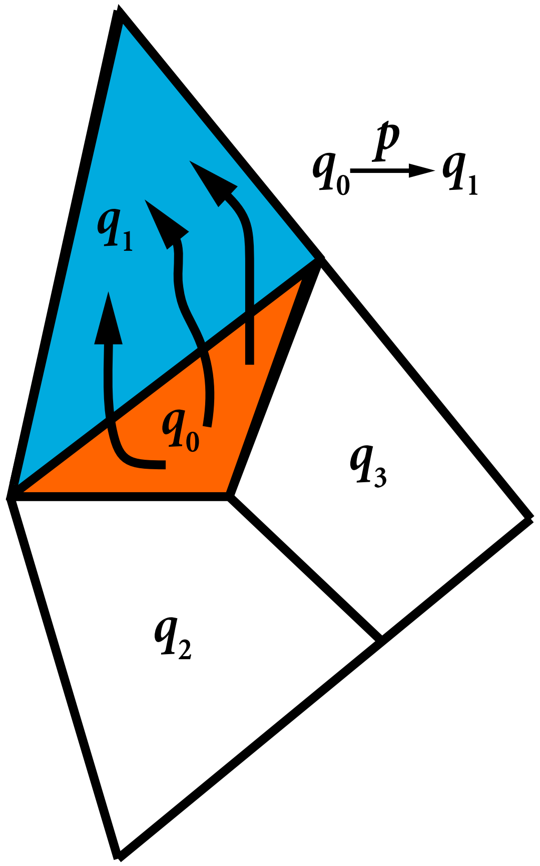
- What approximations are appropriate and how to compute them?
- What discrete synthesis problems to solve and how to solve them?

Slide courtesy of Jun Liu

Under- and over-approximations

Consider an abstraction map $T: X \to Q$, and a family of finite transition systems

$$\left\{\mathcal{T}_p := (Q, Q_0, \xrightarrow{p}) : p \in \mathcal{P}\right\}.$$



Under-approximation:

- Given $q, q' \in Q$ with $q \neq q'$, if $q \xrightarrow{p} q'$, then for all $x_0 \in T^{-1}(q)$, there exists a finite $\tau>0$, such that *all* trajectories of $\dot{x} = f_p(x,d)$ starting from x_0 satisfy
  $$\xi(\tau) \in T^{-1}(q') \quad \xi(t) \in T^{-1}(q) \cup T^{-1}(q'), \quad t \in [0,\tau].$$
- If $q \xrightarrow{p} q$, then $T^{-1}(q)$ is positively invariant w.r.t. $\dot{x} = f_p(x,d)$

In other words

- Every execution of $\mathcal{T}_p$ can be implemented by trajectories of $\dot{x} = f_p(x,d)$
- $\mathcal{T}_p$ is a deterministic *TS*.

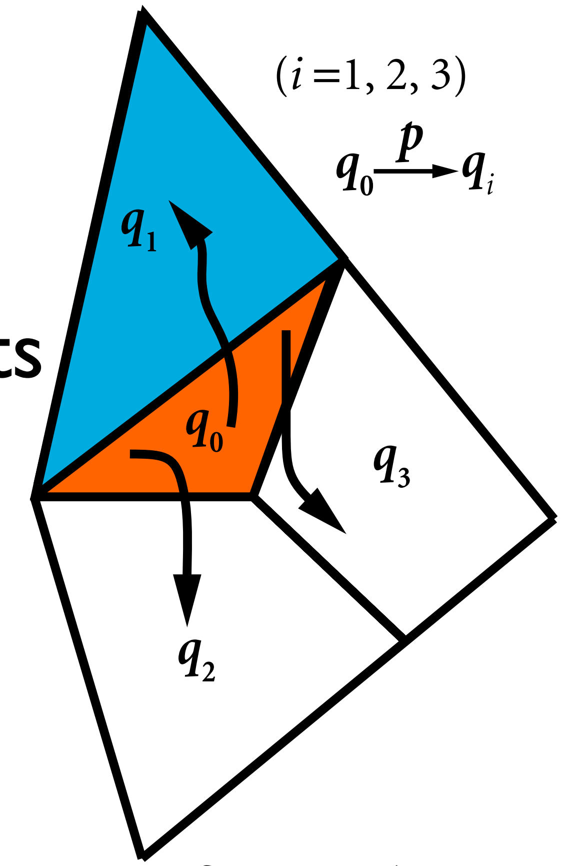
Slide courtesy of Jun Liu

Under- and over-approximations

Consider an abstraction map $T: X \to Q$, and a family of finite transition systems

$$\left\{\mathcal{T}_p := (Q, Q_0, \xrightarrow{p}) : p \in \mathcal{P}\right\}.$$

Over-approximation:



- Given $q, q' \in Q$ with $q \neq q'$, we have $q \xrightarrow{p} q'$, if there exists $x_0 \in T^{-1}(q)$, finite $\tau > 0$, and *some* trajectory of $\dot{x} = f_p(x, d)$ starting from x_0 satisfy

$$\xi(\tau) \in T^{-1}(q') \quad \xi(t) \in T^{-1}(q) \cup T^{-1}(q'), \quad t \in [0, \tau].$$

- Write $q \xrightarrow{p} q$, if $T^{-1}(q)$ contains a complete trajectory of $\dot{x} = f_p(x, d)$

In other words

- Every trajectory of $\dot{x} = f_p(x, d)$ is represented by an execution of $\mathcal{T}_p$
- $\mathcal{T}_p$ can be a nondeterministic *TS*.

Slide courtesy of Jun Liu

How to compute these approximations?

Abstraction Algorithm 1 (over-approximation)

Initialize $\mathcal{S} = \mathcal{Q}, \mathcal{S}_0 = \mathcal{Q}_0, \mathcal{A} = \mathcal{P}$ and $\rightarrow = \mathcal{Q} \times \mathcal{P} \times \mathcal{Q}$
for $i = 1 : |\mathcal{P}|$ **do**
  for $j = 1 : |\mathcal{Q}|$ **do**
    for $k = 1 : Neighbors(T^{-1}(q_j))$ **do**
      $F = commonFacet(T^{-1}(q_j), T^{-1}(q_k))$
      if $isBlocked(T^{-1}(q_j), F, p_i)$ **then** $\rightarrow = \rightarrow \setminus (q_j, p_i, q_k)$
      end if
    end for
    if $isTransient(q_j, p_i)$ **then** $\rightarrow = \rightarrow \setminus (q_j, p_i, q_j)$
    end if
  end for
end for
return $(\mathcal{S}, \mathcal{S}_0, \mathcal{A}, \rightarrow)$

Note:

- $T^{-1}(q)$: polytopes
- $isBlocked(T^{-1}(q), F, p)$: check if the facet F is blocked in mode p
- $isTransient(q, p)$: check if a cell q is transient in mode p
- $Neighbors(T^{-1}(q))$
- $commonFacet(T^{-1}(q), T^{-1}(q'))$

Can be built upon:

- Reachability analysis for affine systems (e.g. Habets et al (2006)), multi-affine systems (e.g. Bleta & Habets (2006)), and polynomial systems (e.g. Ben Sassi and Girard (2012)).

Slide courtesy of Jun Liu

Switching synthesis by model checking

- Given an **under-approximation** $\{\mathcal{T}_p\}$ and an LTL specification φ.
- Construct a product *TS* = $(\mathcal{Q}\times\mathcal{P},\mathcal{Q}_0\times\mathcal{P}_0,\to)$ by adding $(q, p)\to(q', p')$ if and only if $q \xrightarrow{p} q'$.
- Check if all executions of *TS* satisfy ¬φ.
- IF $TS \not\models \neg\varphi$, a counterexample is found, of the form

$$(q_0,p_0)\to(q_1,p_1)\to(q_2,p_2)\to(q_3,p_3)\to\cdots$$

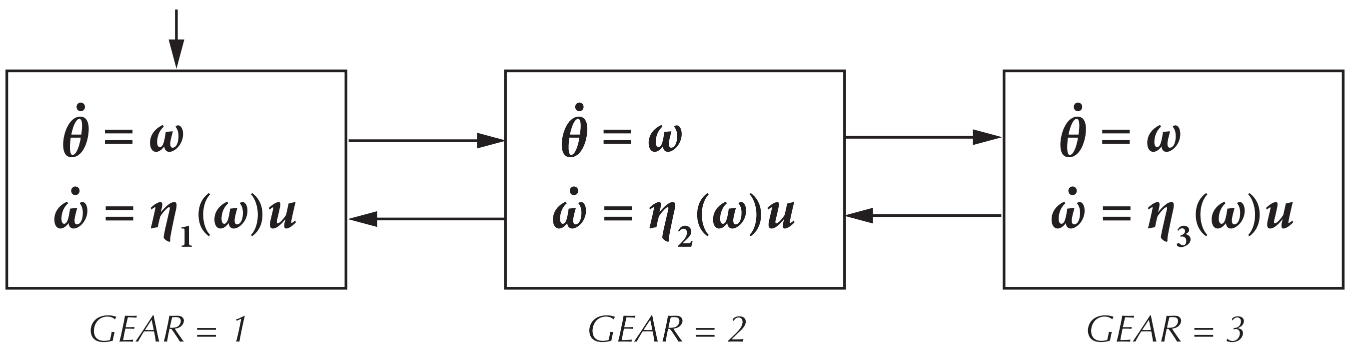
- The counterexample gives a "correct" switching strategy for (1).
- If $TS \models \neg\varphi$, inclusive.

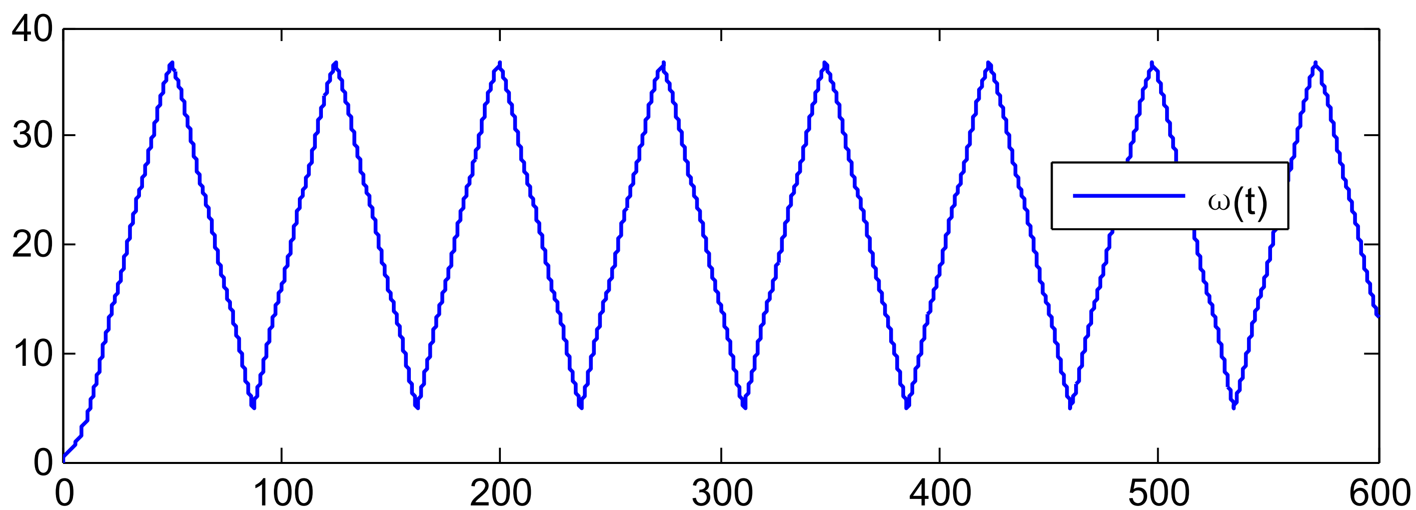
Note:

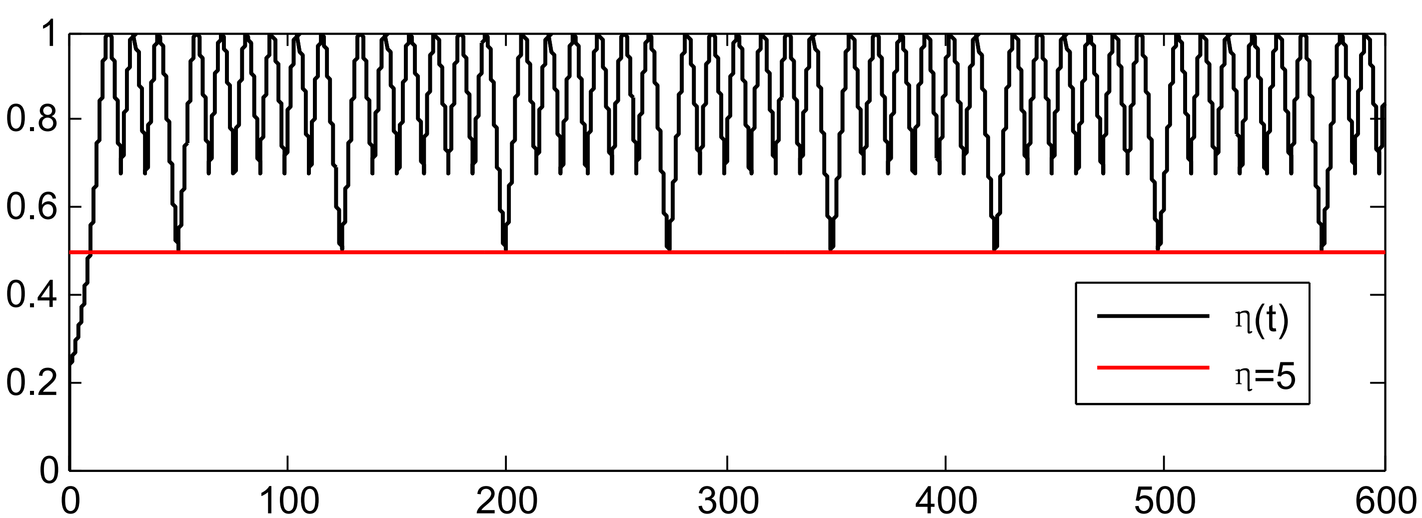
- The strategy obtained is an open-loop strategy.
- Rely on under-approximations being deterministic.

Example: synthesis by model checking

3-gear automatic transmission







Efficiency

$$\eta_i(\omega) = 0.99 \exp\left(-\frac{(\omega - a_i)^2}{64}\right) + 0.01$$

$$a_1 = 10,\ a_2 = 20,\ a_3 = 30$$

Specification

$$\square(\omega \geq 5 \rightarrow \eta \geq 0.5) \wedge (0 \leq \omega \leq 40)$$

States in the finite-state abstraction

$(q_i, gear, u)$

$\{q_i\}$ a partition of [0,40]
gear $\in$ {1, 2, 3}
$u \in$ {-1, 1}

Switching synthesis by two-player game

- Given an **over-approximation** $\{\mathcal{T}_p\}$ and an LTL specification φ.
- A state of the game is s=(e, q, p)∈ $\mathcal{E} \times \mathcal{Q} \times \mathcal{P}$, where

$\mathcal{E}$	environment states
$\mathcal{Q}$	plant states
$\mathcal{P}$	switching modes

- Specification:

real environment

allowable moves

GR(1)

$$\varphi = ((\underbrace{\varphi_q}_{\text{nondeterminism}} \wedge \varphi_e) \rightarrow \varphi_s),\ \ \varphi_\alpha := \boxed{\varphi^\alpha_{\text{init}} \wedge \bigwedge_{i\in I^\alpha_1} \Box\varphi^\alpha_{1,i} \wedge \bigwedge_{i\in I^\alpha_2} \Box\Diamond\varphi^\alpha_{2,i},}\ \alpha \in \{q, s, e\}$$

nondeterminism

initial conditions

goal

- Switching strategy:

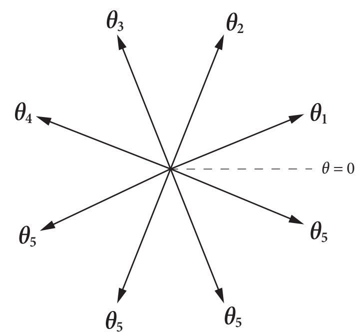
$$\boxed{(s_0 s_1 \cdots s_{t-1}, (q_t, e_t)) \mapsto p_t}$$

- Both q and e are treated as adversary.

Example: synthesis by game solving

2d robot motion planning:

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}\cos\theta & 0\\ \sin\theta & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}v\\ w\end{bmatrix}$$



$$\begin{bmatrix}\dot{x}\\ \dot{y}\end{bmatrix} = \begin{bmatrix}v_0\cos\theta_p\\ v_0\sin\theta_p\end{bmatrix}$$

(*x*, *y*)	position
v	linear velocity
w	angular velocity
mode	heading angles θ_i
PARK	environment signal

Specification

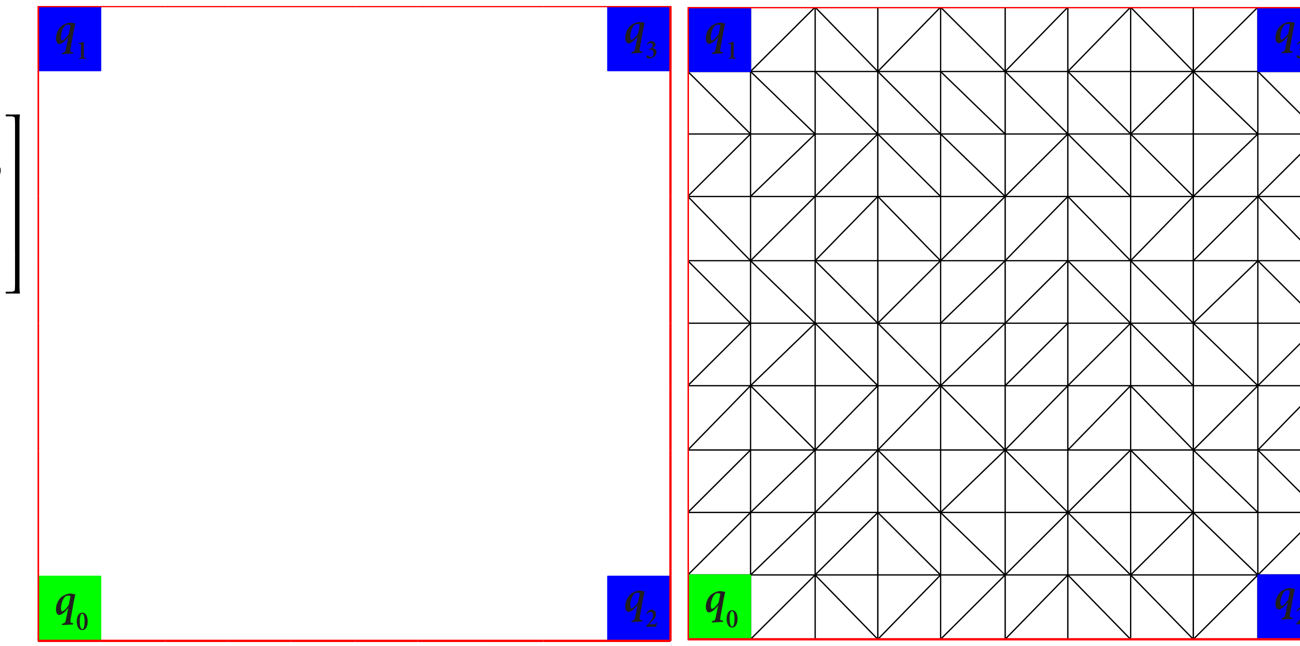
Design a switching sequence such that

P1 $\square\Diamond q_1 \wedge \square\Diamond q_2 \wedge \square\Diamond q_3$

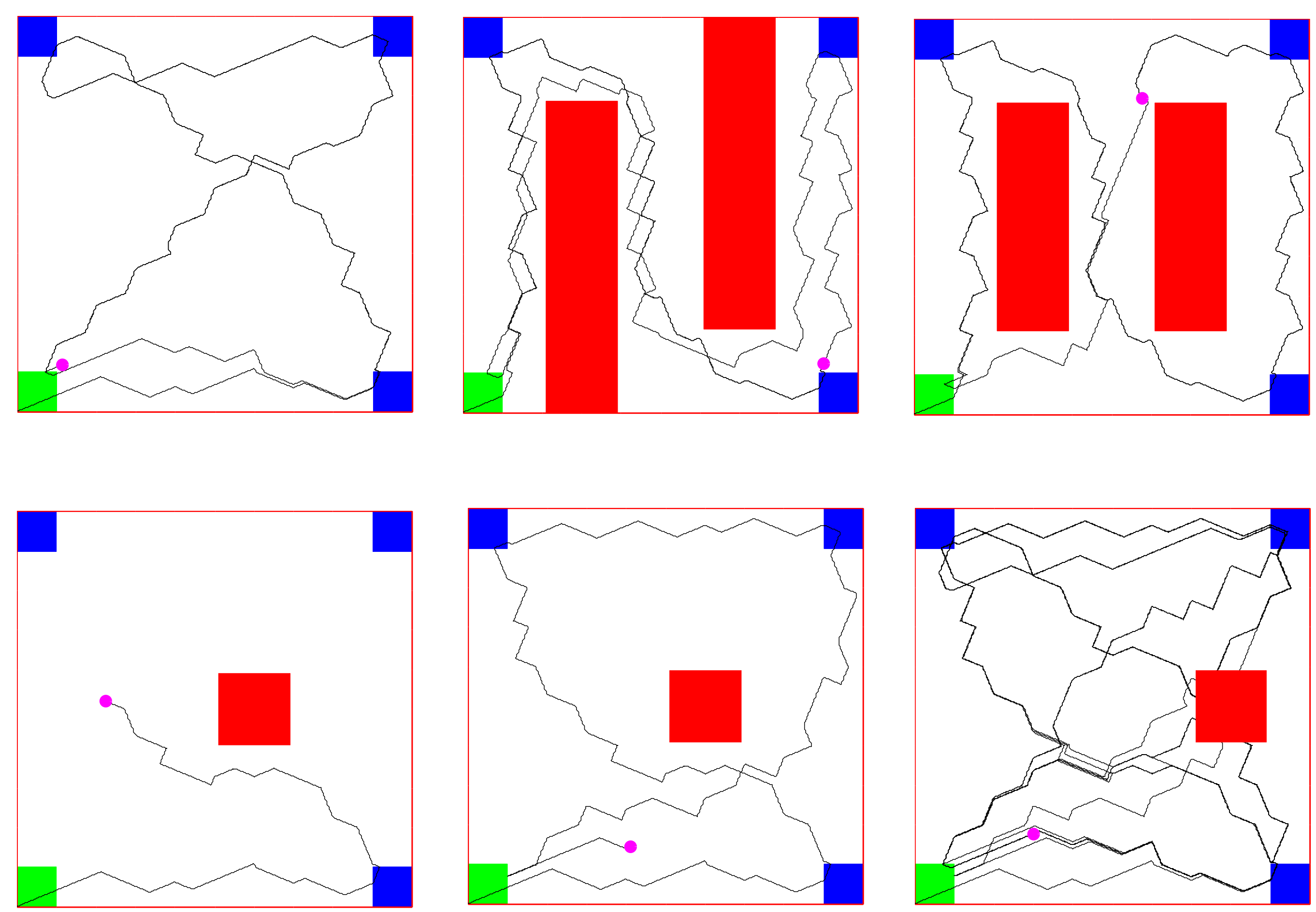
P2 $\square(\mathrm{PARK} \rightarrow \Diamond q_0)$

while avoiding static and moving obstacles.

Over-approximation



Simulation results



Motivation

Slide courtesy of Wolff.

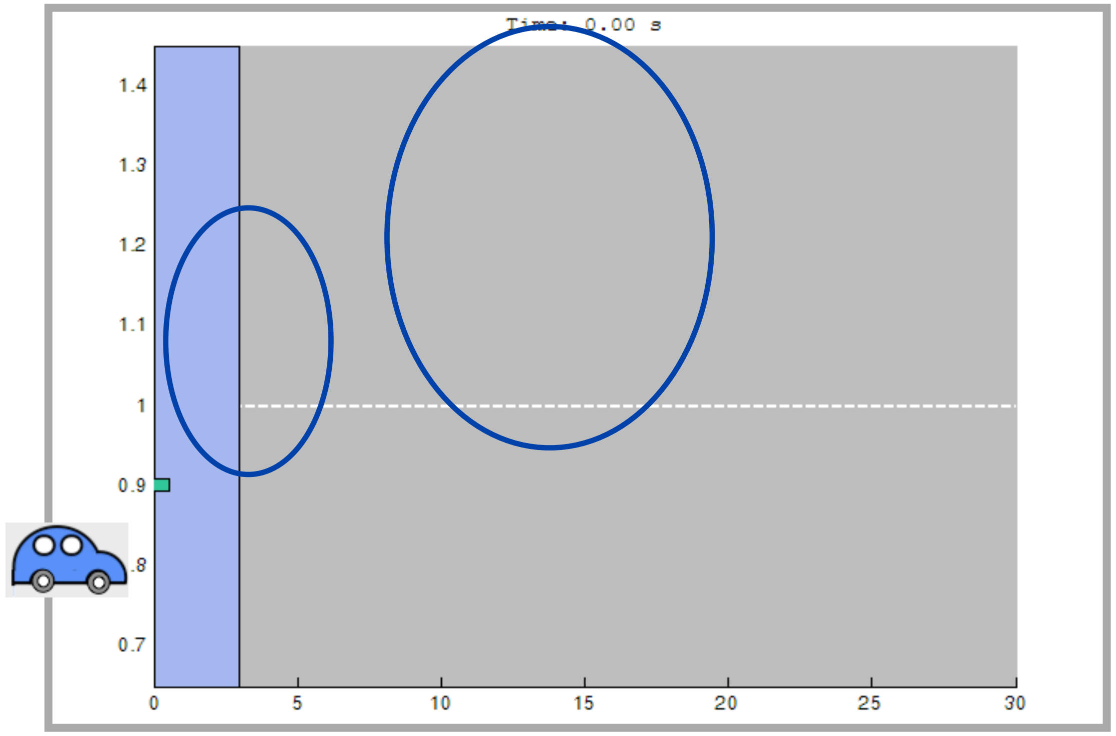
Goal:

- Optimal control for an autonomous system doing a complex task

Challenges:

- Reasoning about how system properties change over time
- Specifying properties like safety, response, priority, liveness, and persistence
- Optimizing the system trajectory to conserve fuel or minimize time

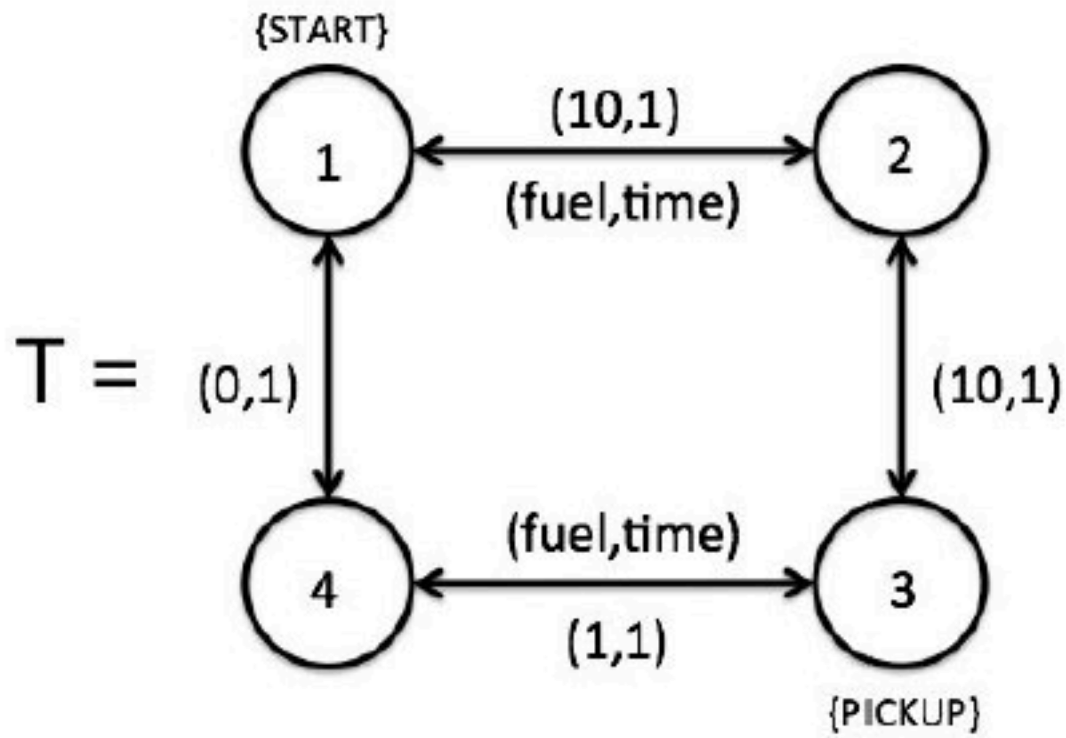
Example of a feasible but not necessarily "optimal":



Problem Description

Wolff, T., Murray, 2012

Simple example:



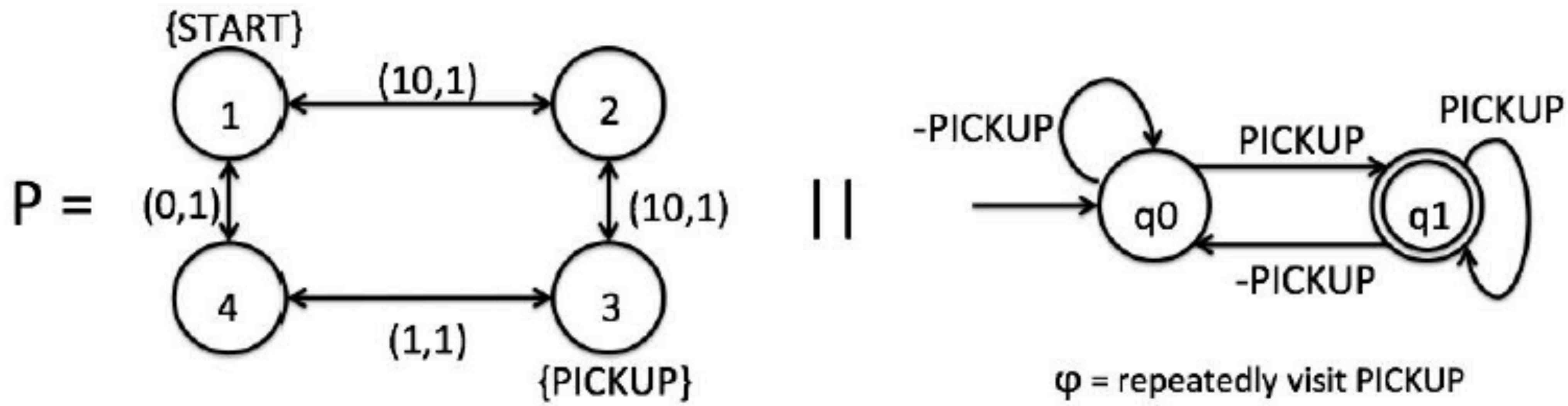
- Task: repeatedly visit PICKUP

Given:

- System model: transition system $\mathcal{T}$ with costs and weights
- Task specification: linear temporal logic (LTL) formula φ

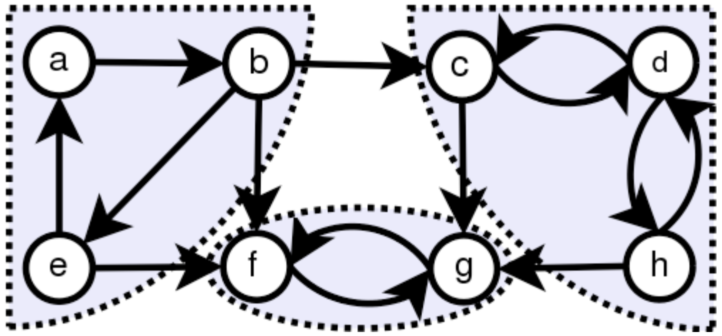
Solution overview (for finite-memory strategies)

1. Construct the product automaton $\mathcal{P} = \mathcal{T} \times \mathcal{A}_{\varphi}$



Computation of an optimal, accepting cycle

[for w(e) = 1 for each transition e]



- Can search in each strongly connected component G = (V,E) of the product automaton with at least one accepting cycle
- For each accepting vertex (state) s ∈ V, define $F_k(v)$ as the minimum cost of length k from s to v
- $F_k(v)$ can be computed by dynamic programming:

$$F_k(v) = \min_{(u,v)\in E} F_{k-1}(u) + c(u,v)$$

- Complexity: O(|V||E|).
- $F_k(s)$: Minimum cost of cycle of length k through the accepting state s
- Complexity in general: $O(n_a(|V||E|+|V|^2\log|V|)$ with n_a accepting states

Example—autonomous driving

Task: repeatedly visit a, b, and c and avoid obstacles x

LTL spec: $\varphi = \square \diamond a \wedge \square \diamond b \wedge \square \diamond c \wedge \square \neg x$

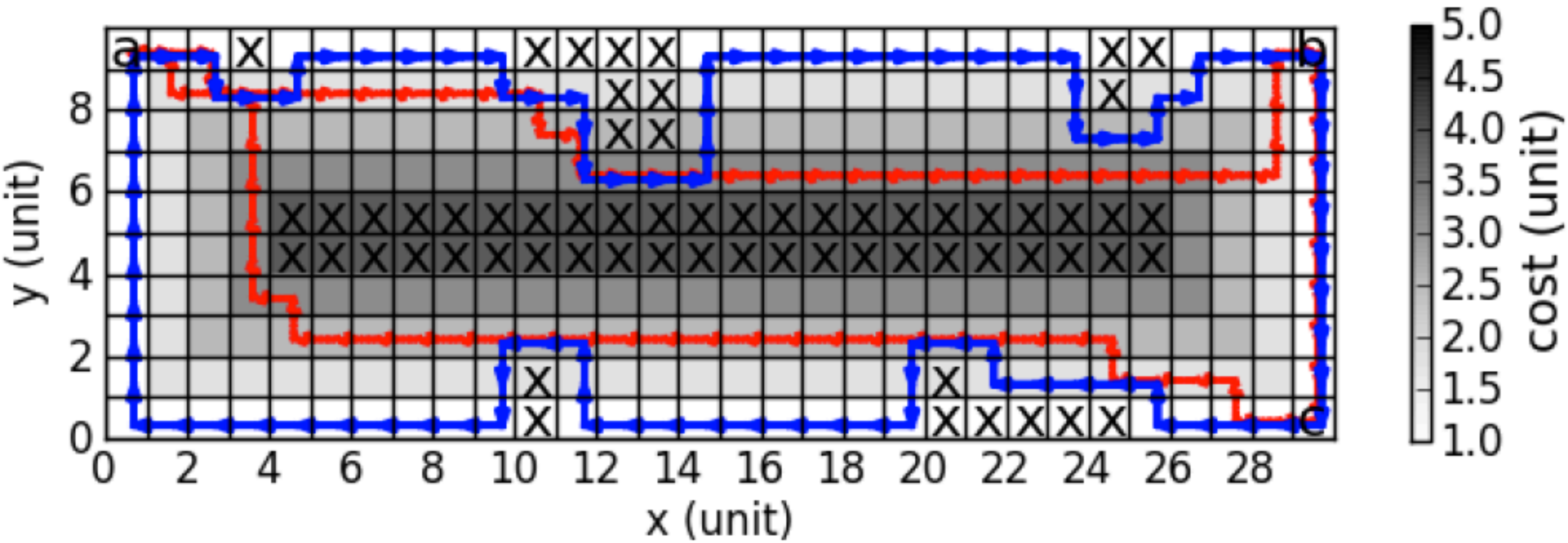


Figure: Driving task, with optimal run (blue) and feasible run (red).

Cost: $J_{\mathrm{opt}} = 49$ and $J_{\mathrm{feas}} = 71$ (units)

CPU time: $t_{\mathrm{opt}} = 2.5$ and $t_{\mathrm{feas}} = 0.68$ (sec)