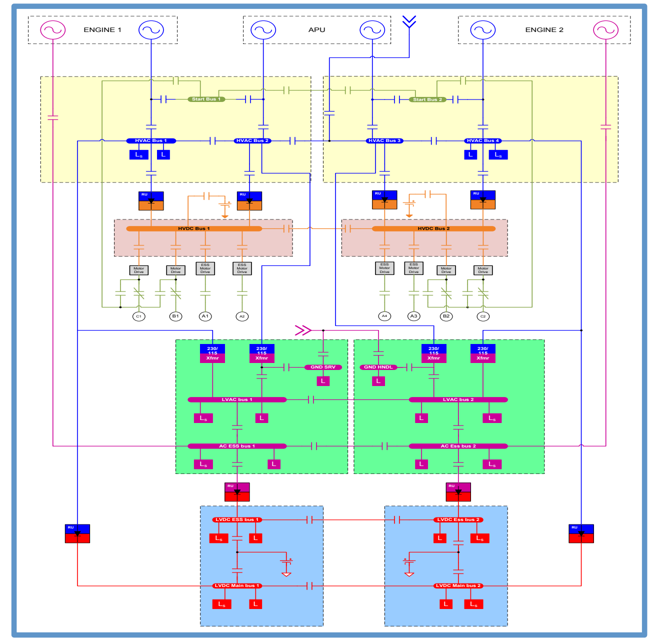
Switching Protocols for Formal Composition of Low-level Dynamics in Cyber-Physical Systems

Necmiye Ozay
Control and Dynamical Systems, Caltech

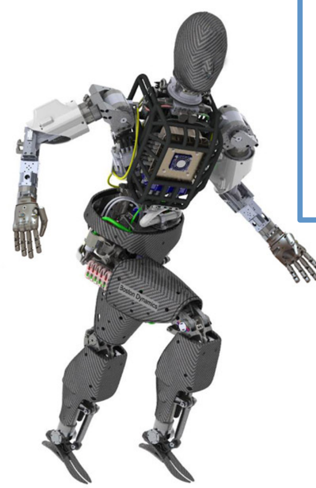
LCCC Workshop on Formal Verification of Embedded Control Systems
18 April, 2013

Joint work with: Jun Liu (Sheffield), Ufuk Topcu (UPenn), Pavithra Prabhakar (IMDEA), Richard M. Murray (Caltech), and iCyPhy team.

Motivation and Applications

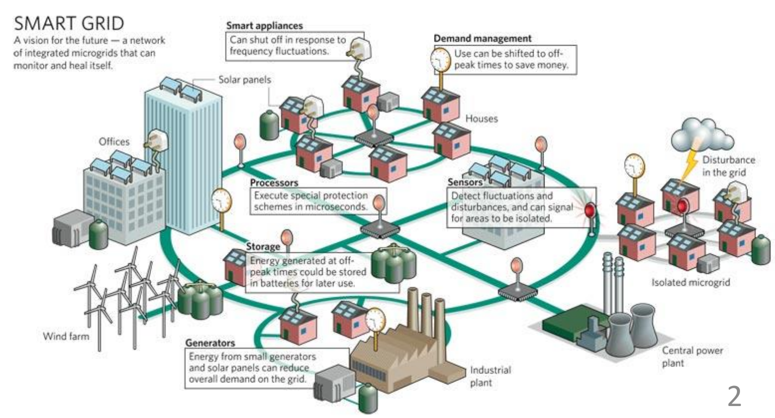


- Large-scale, complex, distributed sensing, actuation and control systems:
  - **Smart grid, Smart buildings, Aircraft systems, Automotive, Robotics, Manufacturing & Automation, Security & Surveillance**



- Designing controllers for complex heterogeneous sensing and control systems is challenging!

Scalable tools for modular control design and verification (theory and software) are lagging!!!



An Industry Scale Problem: Aircraft Electric Power Systems

WHAT ARE THE CONTROL/LOGIC SYNTHESIS PROBLEMS?

Generation: Continuous controller to regulate the output voltage around a nominal value.

Distribution: Logic to reroute the power according to flight phases or fault conditions (Ufuk's talk yesterday).

Load management: Logic to shed unimportant loads when failures in generation.

Fault detection: Logic to detect faults based on sensor measurements.

Cockpit interaction: Logic to coordinate controllers to accommodate pilot requests.

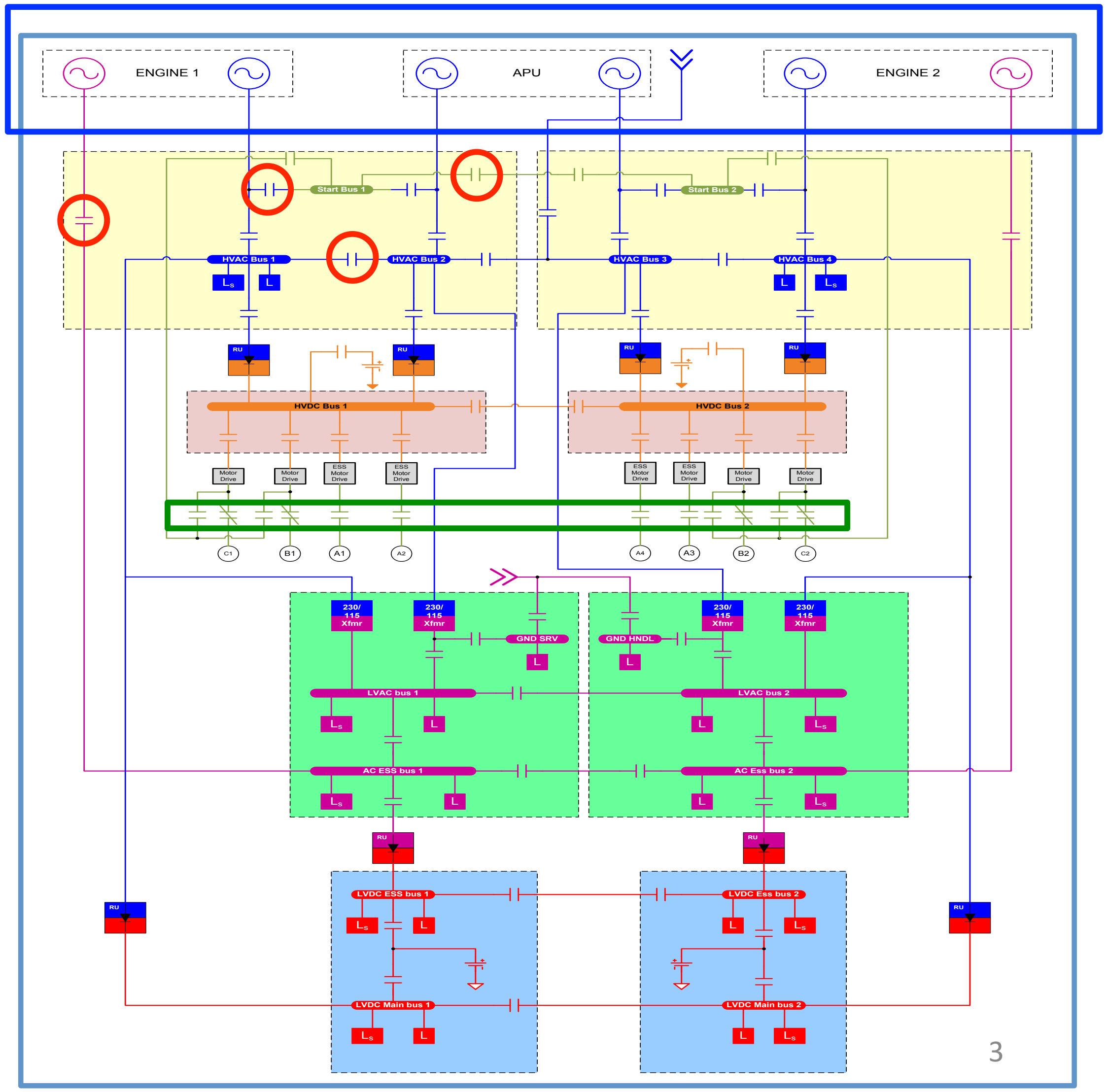
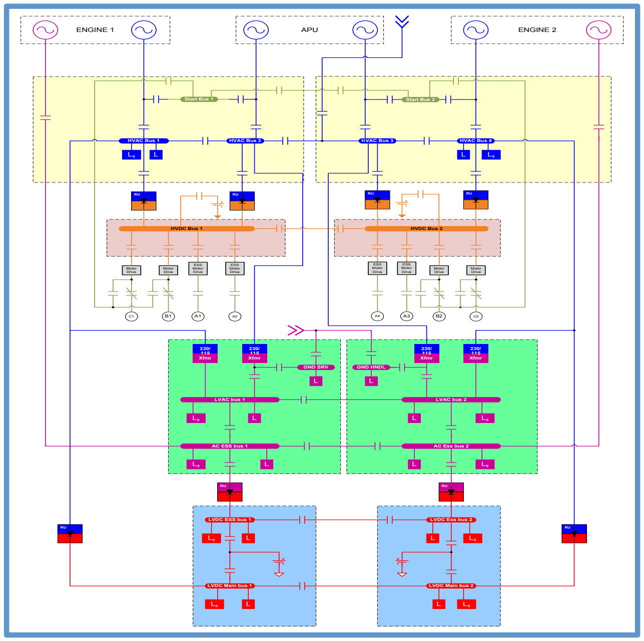
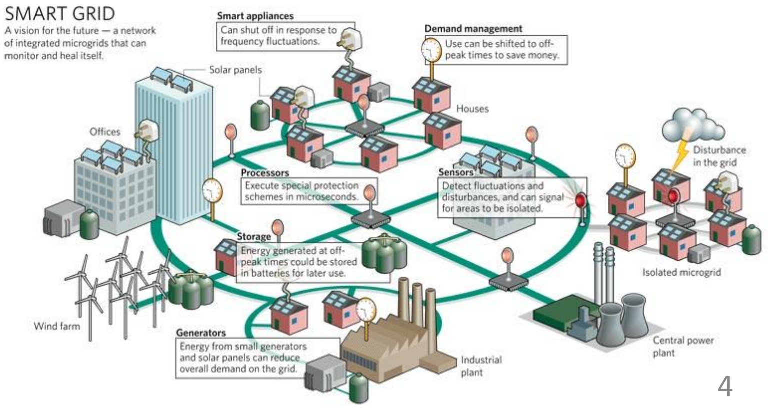


Figure courtesy of Rich Poisson, UTAS. Adapted from Honeywell Patent US 7,439,634 B2

Motivation

- Current control design process for cyber-physical systems:
  - Given some spec (plain English) use **art of design** (engineering intuition, experience) and extensive testing/fine-tuning to come up with a single solution
  - little or no formal guarantees on correctness
  - no formal insight as to internal mechanisms





Better alternative: model-based approach, formal methods for specification, modular design, correct-by-construction embedded controller synthesis

Synthesis of Control Protocols

Given

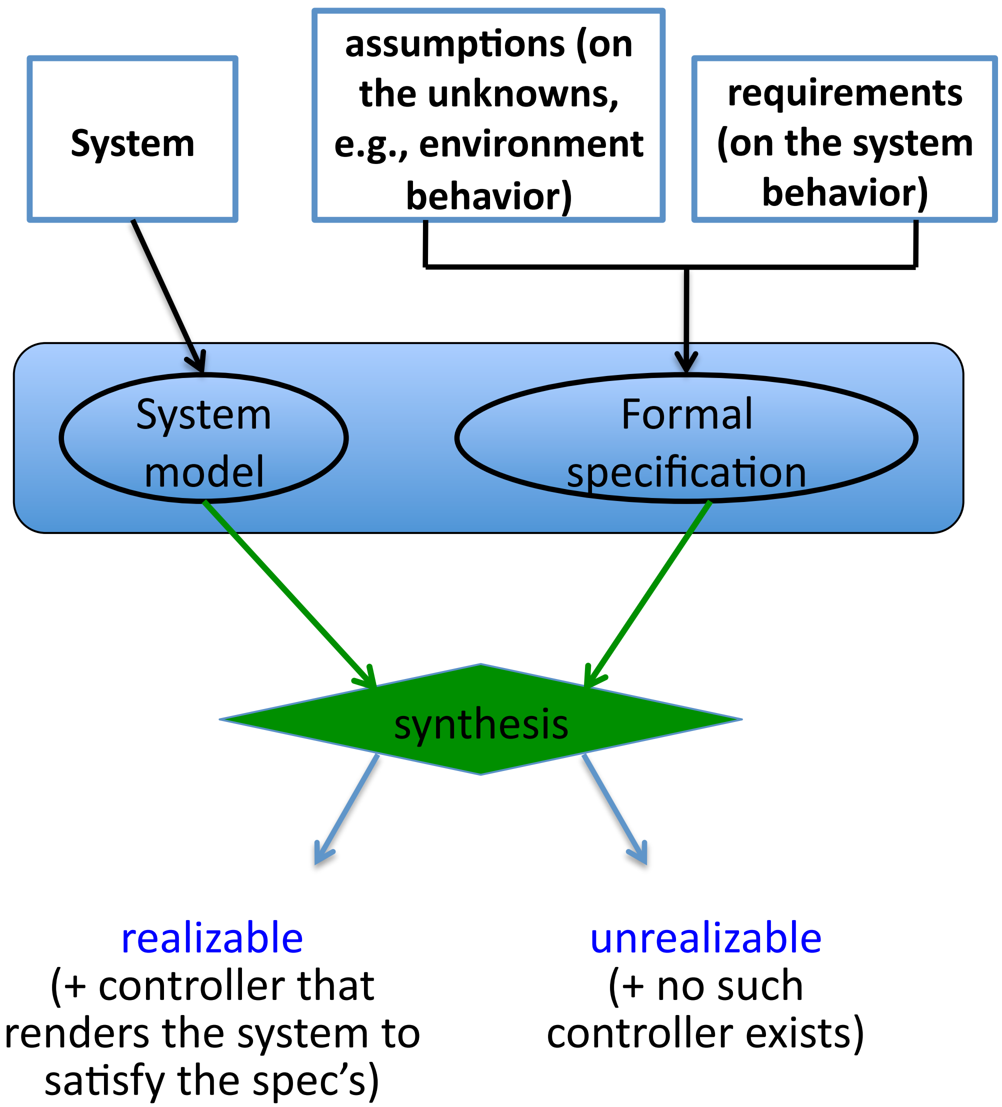
- models for the system and its environment
- specifications for the desired behavior

how to automatically design control protocols that

- manage the behavior of the system
- respond to changes in
  - internal system state
  - external environment

with

- ``correctness” guarantees?



Synthesis of Control Protocols

Given

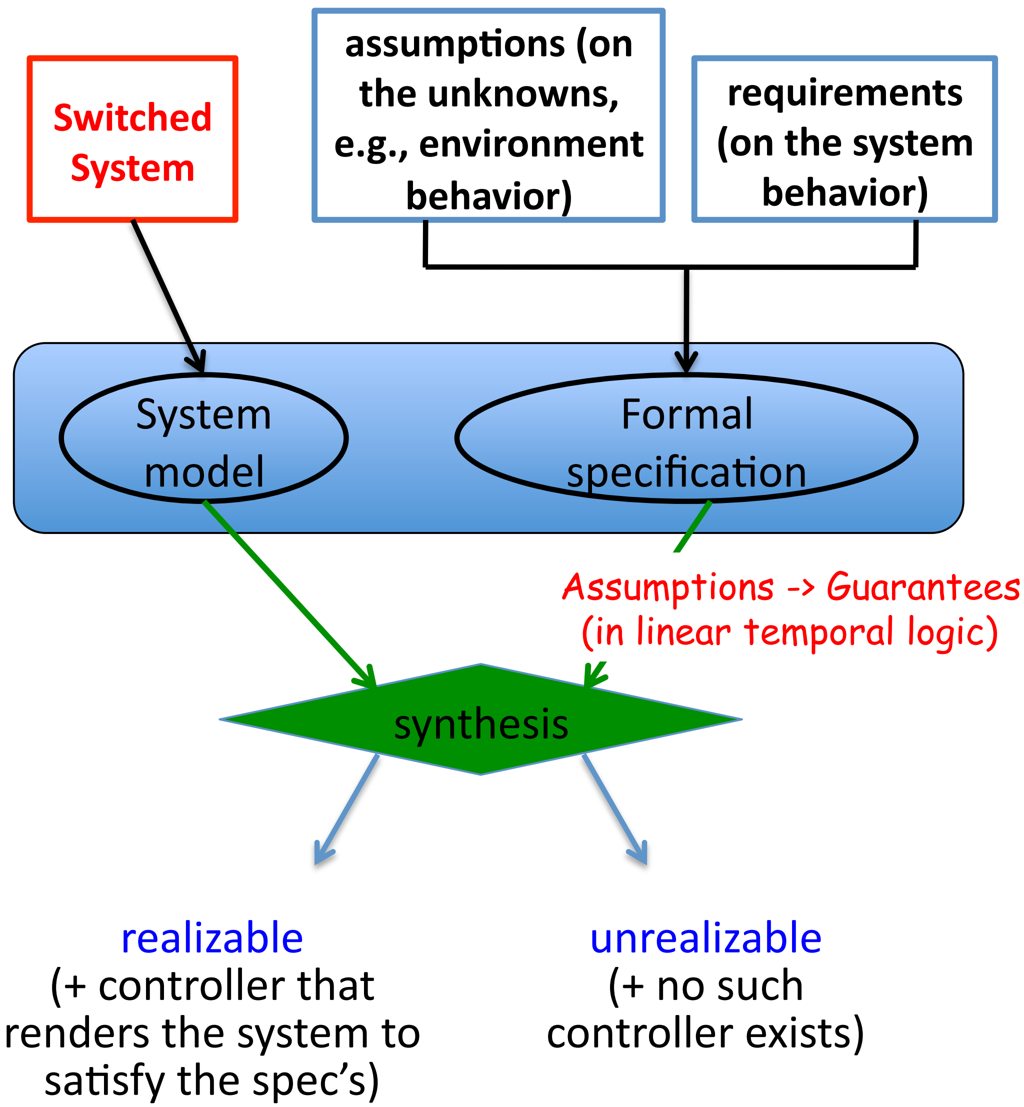
- models for the system and its environment
- specifications for the desired behavior

how to automatically design control protocols that

- manage the behavior of the system
- respond to changes in
  - internal system state
  - external environment

with

- ``correctness” guarantees?



An Industry Scale Problem: Aircraft Electric Power Systems

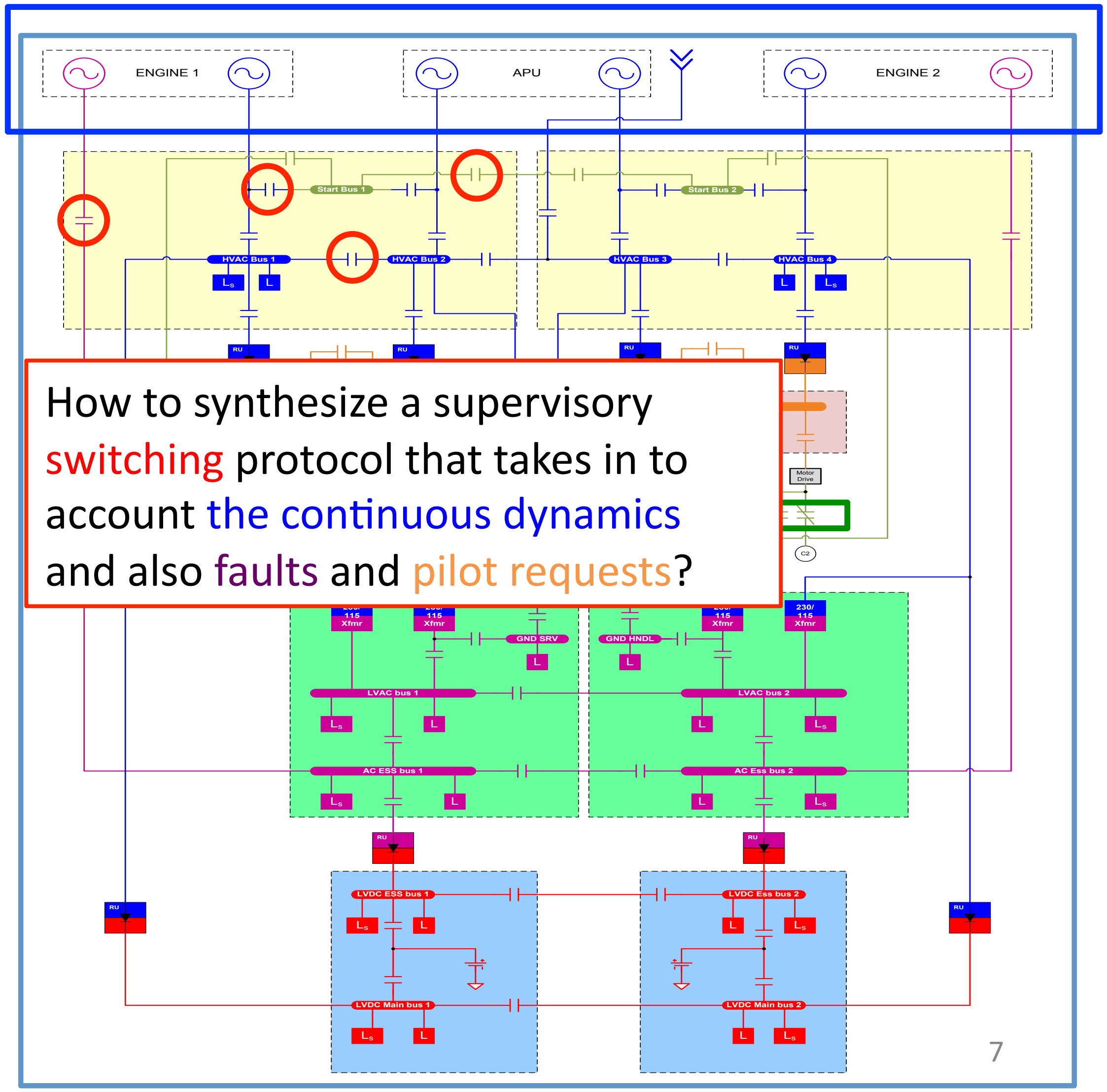
WHAT ARE THE CONTROL SYNTHESIS PROBLEMS?

Generation: Continuous controller to regulate the output voltage around a nominal value.

Distribution: Logic to reroute the power according to flight phases or fault conditions.

Fault detection: Logic to detect faults based on sensor measurements.

Cockpit interaction: Logic to coordinate controllers to accommodate pilot requests.



How to synthesize a supervisory switching protocol that takes in to account the continuous dynamics and also faults and pilot requests?

Figure courtesy of Rich Poisson, UTAS. Adapted from Honeywell Patent US 7,439,634 B2

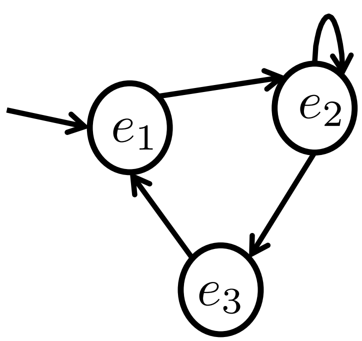
Systems of Interest

Switched systems:

$$\dot{x}(t) = f_{\sigma(t)}(x(t), d(t))$$
$$x(t) \in X \subseteq \mathbb{R}^n, \ \forall t$$
$$x(0) \in X_0 \subseteq X$$
$$d(t) \in D \subseteq \mathbb{R}^n, \ \forall t$$
$$\sigma(t) \in \{1, 2, \ldots, N\}, \ \forall t$$

Environment:

$$e(t) \in \{e_1, e_2, \ldots, e_N\}; \forall t$$



Continuous-time discrete-valued signal (*with finite variability*)

WHY SWITCHED SYSTEMS?

- Naturally arise from
  - modular design principles (motion primitives; a set of pre-designed feedback controllers, each with different performance criteria) or
  - physical components (different configurations of a system due to physical switches or valves)
- *Good fit* for discrete (logic-based) tools in hand. Also, not easy to deal with using standard cont. control tools.

Problem Definition

Switched systems:

$$\dot{x}(t) = f_{\sigma(t)}(x(t), d(t))$$
$$x(t) \in X \subseteq \mathbb{R}^n, \ \forall t$$
$$x(0) \in X_0 \subseteq X$$
$$d(t) \in D \subseteq \mathbb{R}^n, \ \forall t$$
$$\sigma(t) \in \{1, 2, \ldots, N\}, \ \forall t$$

Problem Definition: Given a switched system,

$$\mathcal{S} = (X, X_0, , \{f_a\}_{a \in A}, \Pi, h)$$

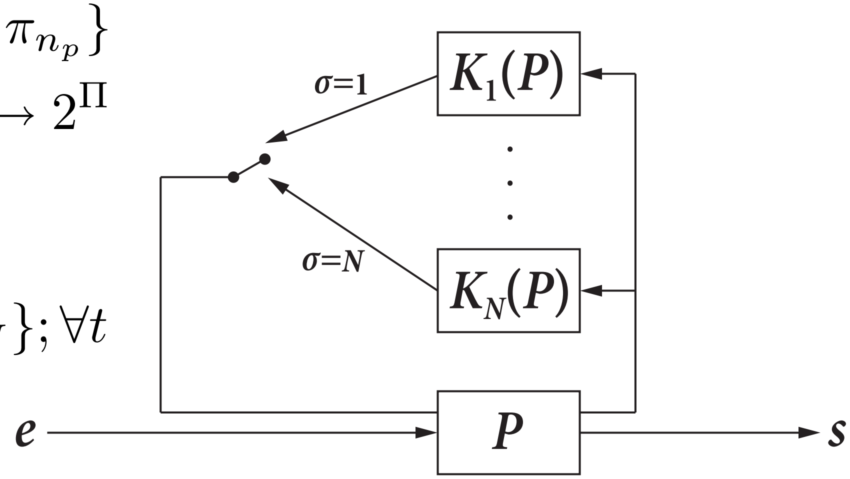
an environment description and some LTL specification (Φ), design a mode signal σ(x(t),e(t)) such that the trajectories of the system satisfies the spec for all initial conditions x(0) in a given set and for all disturbances d.

Propositions & observations:

$$\Pi \doteq \{\pi_{init}, \pi_1, \ldots, \pi_{n_p}\}$$
$$h : X \rightarrow 2^{\Pi}$$

Environment:

$$e(t) \in \{e_1, e_2, \ldots, e_N\}; \forall t$$



Problem Definition

Switched systems:

$$\dot{x}(t) = f_{\sigma(t)}(x(t), d(t))$$
$$x(t) \in X \subseteq \mathbb{R}^n, \ \forall t$$
$$x(0) \in X_0 \subseteq X$$
$$d(t) \in D \subseteq \mathbb{R}^n, \ \forall t$$
$$\sigma(t) \in \{1, 2, \ldots, N\}, \ \forall t$$

Problem Definition: Given a switched system,

$$\mathcal{S} = (X, X_0, , \{f_a\}_{a \in A}, \Pi, h)$$

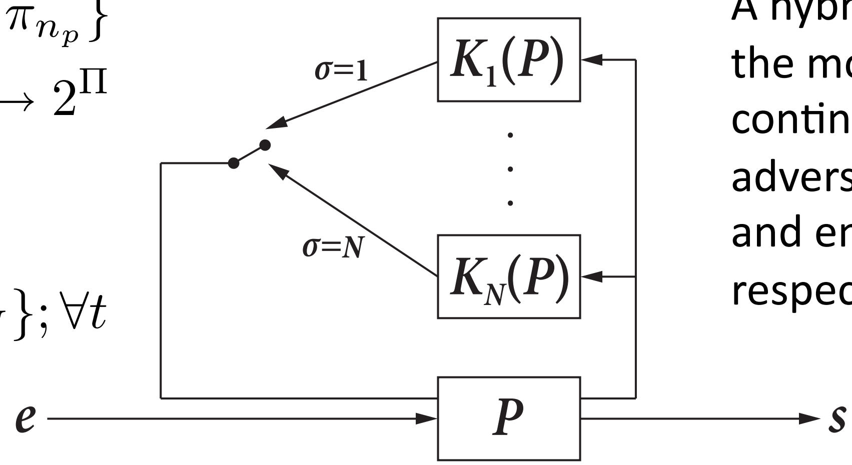
an environment description and some LTL specification (Φ), design a mode signal σ(x(t),e(t)) such that the trajectories of the system satisfies the spec for all initial conditions x(0) in a given set and for all disturbances d.

Propositions & observations:

$$\Pi \doteq \{\pi_{init}, \pi_1, \ldots, \pi_{n_p}\}$$
$$h : X \rightarrow 2^{\Pi}$$

Environment:

$$e(t) \in \{e_1, e_2, \ldots, e_N\}; \forall t$$



A hybrid game between the mode signal and continuous and discrete adversaries (disturbance and environment, respectively).

Problem Definition

Switched systems:

$$\dot{x}(t) = f_{\sigma(t)}(x(t), d(t))$$
$$x(t) \in X \subseteq \mathbb{R}^n, \ \forall t$$
$$x(0) \in X_0 \subseteq X$$
$$d(t) \in D \subseteq \mathbb{R}^n, \ \forall t$$
$$\sigma(t) \in \{1, 2, \ldots, N\}, \ \forall t$$

Problem Definition: Given a switched system,

$$\mathcal{S} = (X, X_0, , \{f_a\}_{a \in A}, \Pi, h)$$

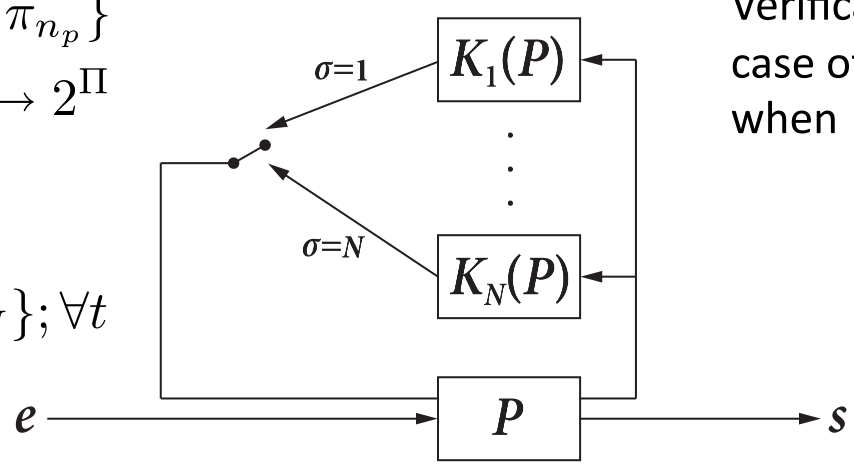
an environment description and some LTL specification (Φ), design a mode signal σ(x(t),e(t)) such that the trajectories of the system satisfies the spec for all initial conditions x(0) in a given set and for all disturbances d.

Propositions & observations:

$$\Pi \doteq \{\pi_{init}, \pi_1, \ldots, \pi_{n_p}\}$$
$$h : X \rightarrow 2^{\Pi}$$

Environment:

$$e(t) \in \{e_1, e_2, \ldots, e_N\}; \forall t$$



Verification is a special case of this problem when |A|=1.

Related Work

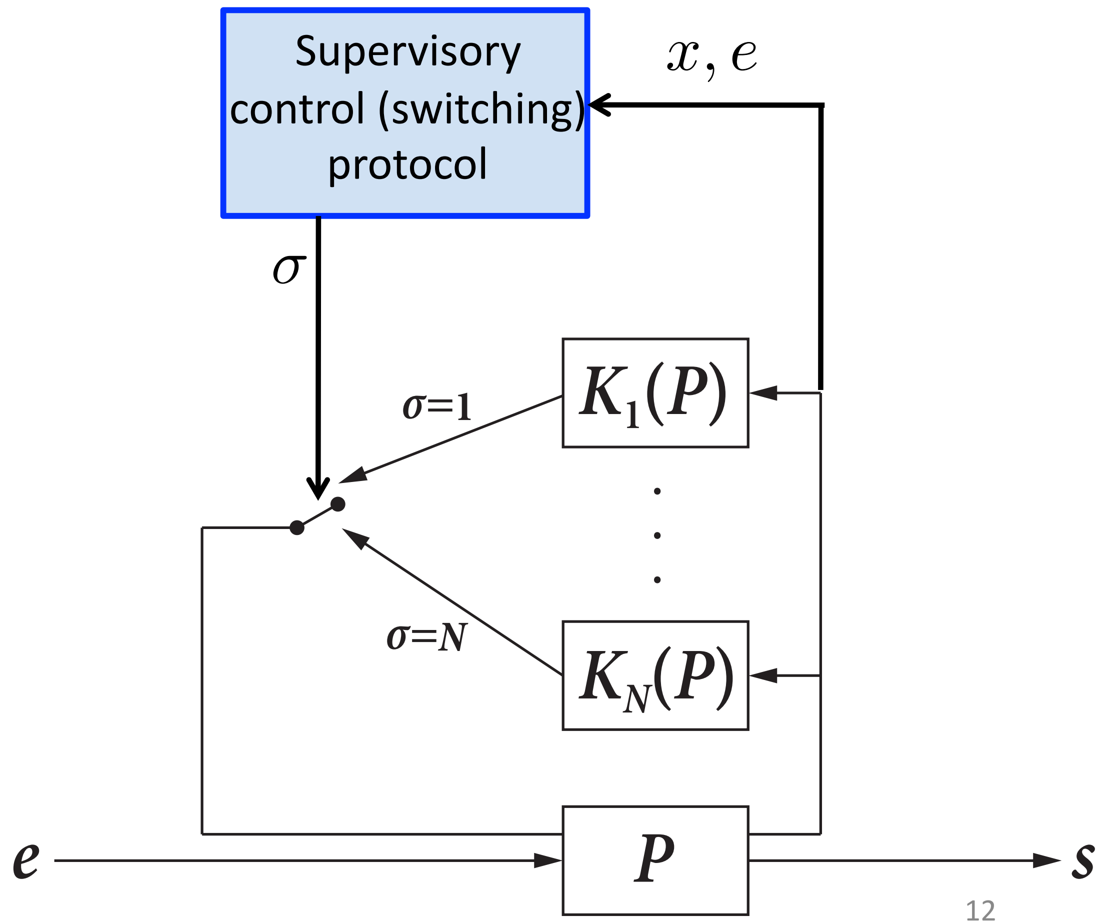
Many references in literature for mostly special cases (different specs, open/close systems, rectangular/linear/nonlinear dynamics, disturbances, etc.)…

Direct methods:

- Asarin et al. 2000, Ding and Tomlin 2010, Moor and Davoren 2001, Lygeros et al. 2000, Henzinger et al. 1999, Alur et al. 2012…

Abstraction based methods:

- Camara et al. 2011, Yordonav et al. 2012, Gol et al. 2012, …



Overview of Solution Strategy

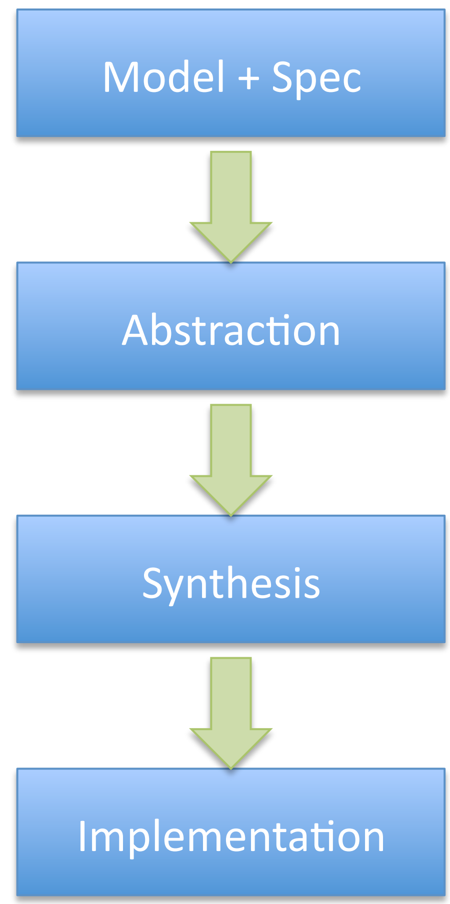
- Given

$$\dot{x} = f_\sigma(x, d), \ \sigma \in \mathcal{A}, \ \text{and} \ \varphi = \varphi_e \rightarrow \varphi_s$$

- Compute finite-state proposition preserving approximations

- Solve a discrete synthesis problem and obtain a discrete switching strategy σ

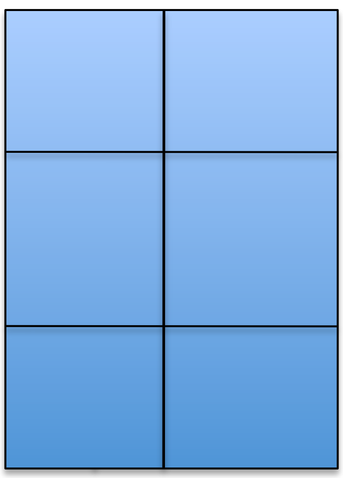
- Implement the switching strategy σ continuously to ensure that the all trajectories of the system satisfy φ



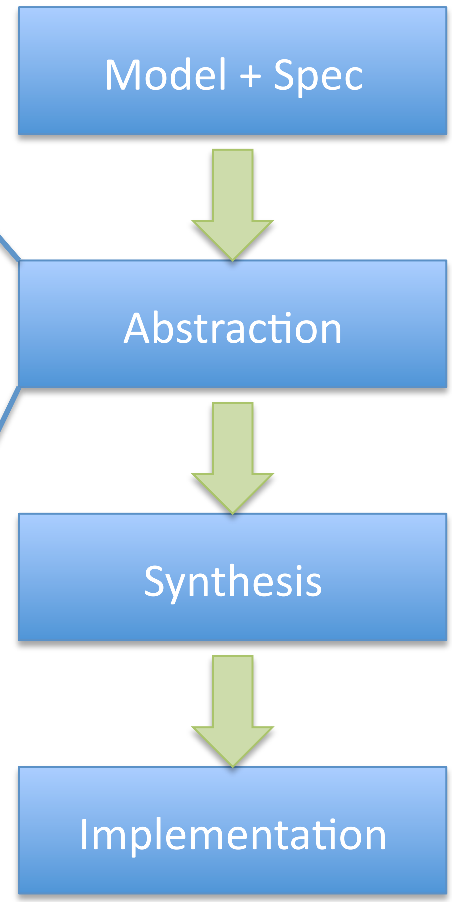
For now, assume no environment -> it will be easy to incorporate

Abstraction

Find a finite transition system that approximates the continuous dynamics



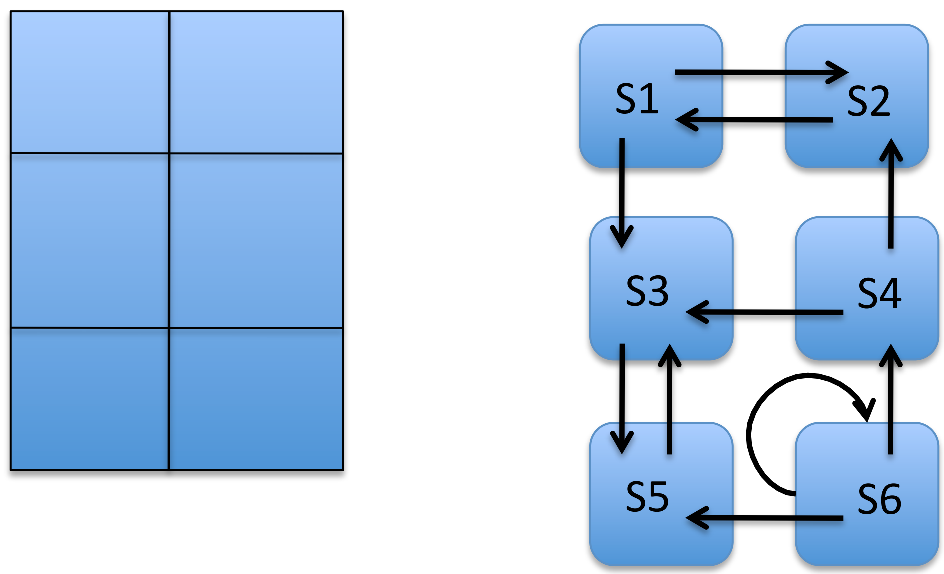
$T = (Q, Q_0, \mathcal{A}, \rightarrow_T, \Pi, L)$
Q, finite set of states
$Q_0 \subseteq Q$, set of initial states
$\mathcal{A}$, finite set of actions
$\rightarrow_T \subseteq Q \times \mathcal{A} \times Q$, transition relation
Π, finite set of propositions
$L : Q \rightarrow 2^{\Pi}$, labeling function

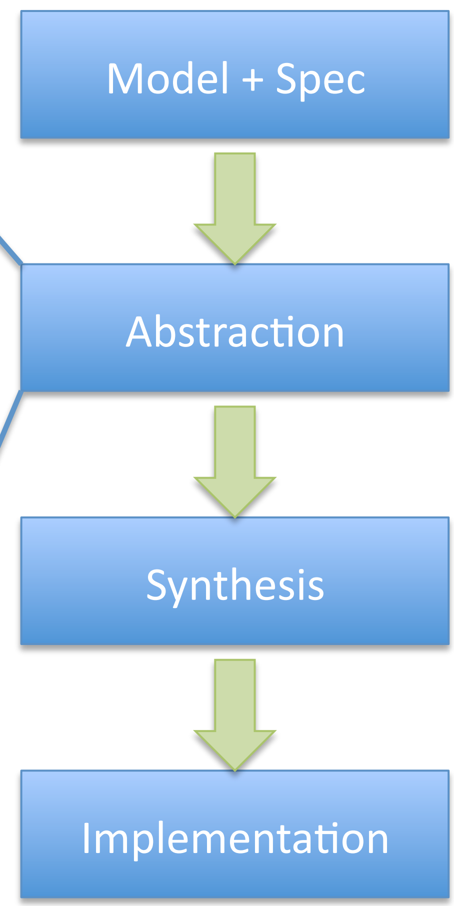


Abstraction

Find a finite transition system that approximates the continuous dynamics

For each mode:

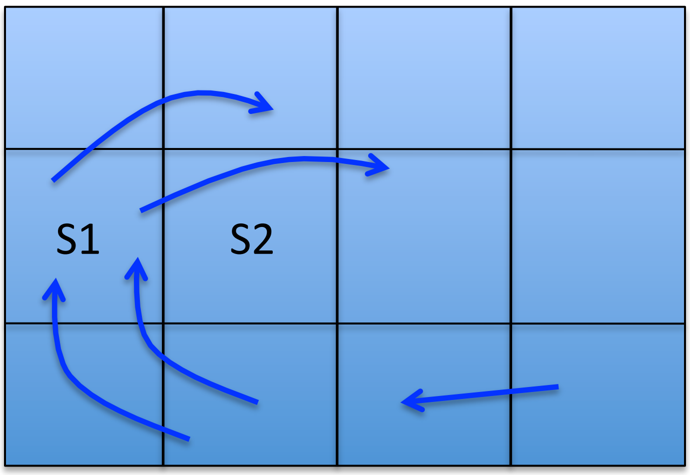


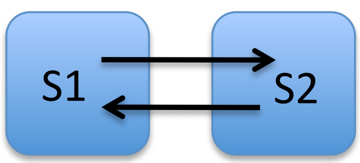


A nondeterministic transition system T that "simulates" the original system

What is missing in transition systems and simulations?

Flow on one mode:





A spurious cycle!
T cannot make any progress!

- Simulation -> matching the transitions (short-term behavior)
- Cannot capture long-term behaviors (e.g. liveness)
- If we want to synthesize controllers for LTL (beyond safety properties), a natural extension is to augment the transition system with liveness properties

$T = (Q, Q_0, \mathcal{A}, \rightarrow_T, \Pi, L)$
Q, finite set of states
$Q_0 \subseteq Q$, set of initial states
$\mathcal{A}$, finite set of actions
$\rightarrow_T \subseteq Q \times \mathcal{A} \times Q$, transition relation
Π, finite set of propositions
$L : Q \rightarrow 2^{\Pi}$, labeling function

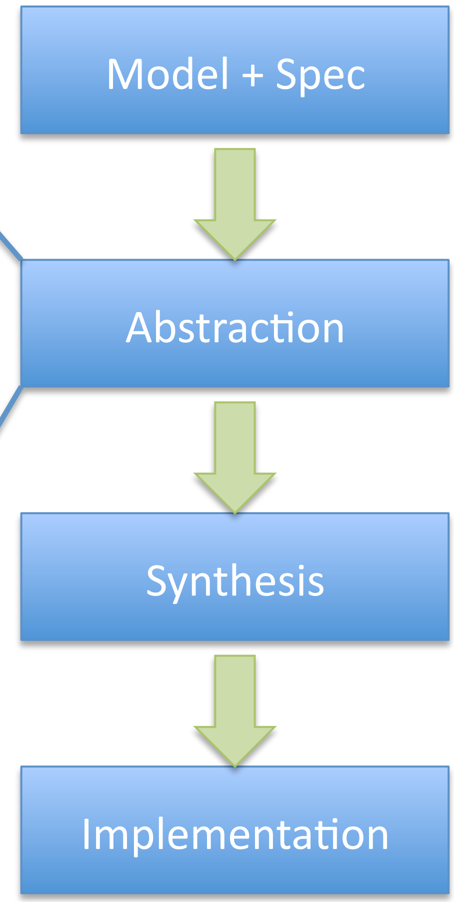
Abstraction

Main idea:

Augmented finite transition systems

$$T_{aug} = (Q, Q_0, \mathcal{A}, \rightarrow_T, \Pi, L, \mathcal{G})$$

Q, finite set of states
$Q_0 \subseteq Q$, set of initial states
$\mathcal{A}$, finite set of actions
$\rightarrow_T \subseteq Q \times \mathcal{A} \times Q$, transition relation
Π, finite set of propositions
$L : Q \rightarrow 2^{\Pi}$, labeling function
$\mathcal{G} : \mathcal{A} \rightarrow 2^{2^{Q}}$, progress group map



Simple extension of just transition systems by Kesten and Pnueli 2000.
- augment the transition systems with justice/weak fairness assumptions

Augmented Finite Transition Systems

Main tool:

Augmented finite transition Systems (FTS):

$$T_{aug} = (Q, Q_0, \mathcal{A}, \to_T, \Pi, L, \mathcal{G})$$

Q, finite set of states

$Q_0 \subseteq Q$, set of initial states

$\mathcal{A}$, finite set of actions

$\to_T \subseteq Q \times \mathcal{A} \times Q$, transition relation

Π, finite set of propositions

$L : Q \to 2^{\Pi}$, labeling function

$\mathcal{G} : \mathcal{A} \to 2^{2^Q}$, progress group map

What is progress group map?

Given $a \in \mathcal{A}$, if a set $G \subseteq Q$ is such that $G \in \mathcal{G}(a)$, then the system cannot remain within G indefinitely using just a. Or, in LTL, $\mathcal{G}$ imposes:

$$\varphi_g \doteq \bigwedge_{a \in \mathcal{A}} \bigwedge_{G \in \mathcal{G}(a)} \neg \diamond \Box((\vee_{q \in G} q) \wedge a)$$

We can define a simulation-like preorder between augmented finite transition systems:

DEFINITION: $\hat{T}_{aug} \underset{\text{A.S.}}{\succeq} T_{aug}$, if there exists a function $\beta : Q \to \hat{Q}$ that defines simulation and for all actions, for all $\hat{G} \in \hat{\mathcal{G}}(a)$, there exists $G \in \mathcal{G}(a)$ such that for all $\hat{q} \in \hat{G}$, we have $\beta^{-1}(\hat{q}) \subset G$.

Augmented Finite Transition Systems

We can define a simulation-like preorder between augmented finite transition systems:

DEFINITION: $\hat{T}_{aug} \underset{\text{A.S.}}{\succeq} T_{aug}$, if there exists a function $\beta : Q \rightarrow \hat{Q}$ that defines simulation and for all actions, for all $\hat{G} \in \hat{\mathcal{G}}(a)$, there exists $G \in \mathcal{G}(a)$ such that for all $\hat{q} \in \hat{G}$, we have $\beta^{-1}(\hat{q}) \subset G$.

- $\hat{T}_{aug}$ has more behaviors (due to adversarial uncertainty)
- T_{aug} has more achievable behaviors (enforced by control)

Augmented Finite Transition Systems

Main tool:

Augmented finite transition Systems (FTS):

$$T_{aug} = (Q, Q_0, \mathcal{A}, \rightarrow_T, \Pi, L, \mathcal{G})$$

Q, finite set of states

$Q_0 \subseteq Q$, set of initial states

$\mathcal{A}$, finite set of actions

$\rightarrow_T \subseteq Q \times \mathcal{A} \times Q$, transition relation

Π, finite set of propositions

$L : Q \rightarrow 2^{\Pi}$, labeling function

$\mathcal{G} : \mathcal{A} \rightarrow 2^{2^Q}$, progress group map

What is progress group map?

Given $a \in \mathcal{A}$, if a set $G \subseteq Q$ is such that $G \in \mathcal{G}(a)$, then the system cannot remain within G indefinitely using just a. Or, in LTL, $\mathcal{G}$ imposes:

$$\varphi_g \doteq \bigwedge_{a \in \mathcal{A}} \bigwedge_{G \in \mathcal{G}(a)} \neg \Diamond \Box ((\vee_{q \in G} q) \wedge a)$$

Progress group map can capture long-term behaviors of underlying dynamics:

If a set K of the state space is *transient* (i.e., does not contain any invariant sets), then all the discrete states corresponding to a concrete subset of K form a progress group map.

Computation of Abstract Models

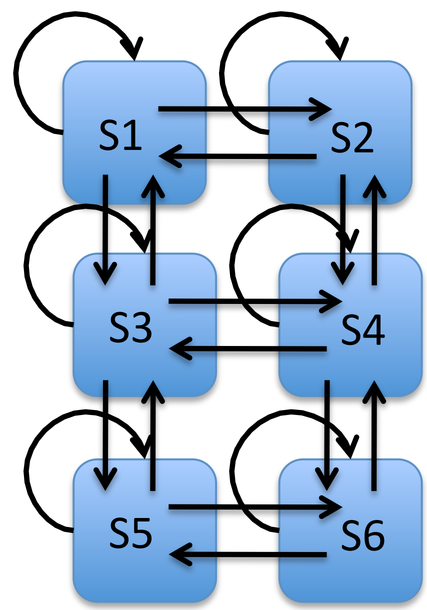
Algorithm 1 Abstraction Procedure

Input: switched system $\mathcal{S} = (X, X_0, \mathcal{A}, \{f_a\}_{a \in \mathcal{A}}, \Pi, h)$, proposition preserving partition $P = \{\mathcal{P}_i\}_{i=1}^{N}$

Output: augmented finite transition system $\mathcal{T} = (Q, Q_0, \mathcal{A}, \rightarrow_{\mathcal{T}}, \Pi, L, \mathcal{G})$ such that $\mathcal{T} \underset{\text{O.A}}{\succeq} \mathcal{S}$

Let α be the proposition preserving map
Set $Q = \{1, \ldots, N+1\}$, $Q_0 = \{i : \mathcal{P}_i \subseteq X_0\}$, $L = h \circ \alpha^{-1}$
Initialize $\rightarrow_{\mathcal{T}} = (Q \setminus \{N+1\}) \times \mathcal{A} \times Q$
for $a \in \mathcal{A}$ **do**
  $\mathcal{G}(a) = \emptyset$
  $\rightarrow_{\mathcal{T}} = \rightarrow_{\mathcal{T}} \cup \{(N+1, a, N+1)\}$
  for $i \in \{1, \ldots, N\}$ **do**
    for $j = \{1, \ldots, N+1\} \setminus \{i\}$ **do**
      if *isBlocked*$(\alpha^{-1}(i), \alpha^{-1}(j), f_a)$ **then**
        $\rightarrow_{\mathcal{T}} = \rightarrow_{\mathcal{T}} \setminus \{(i, a, j)\}$
    if *isTransient*$(\alpha^{-1}(i), f_a)$ **then**
      $\mathcal{G}(a) = \mathcal{G}(a) \cup \{\{i\}\}$
return $\mathcal{T} = (Q, Q_0, \mathcal{A}, \rightarrow_{\mathcal{T}}, \Pi, L, \mathcal{G})$

IDEA: start with a "complete" graph : transitions to all neighbors



Computation of Abstract Models

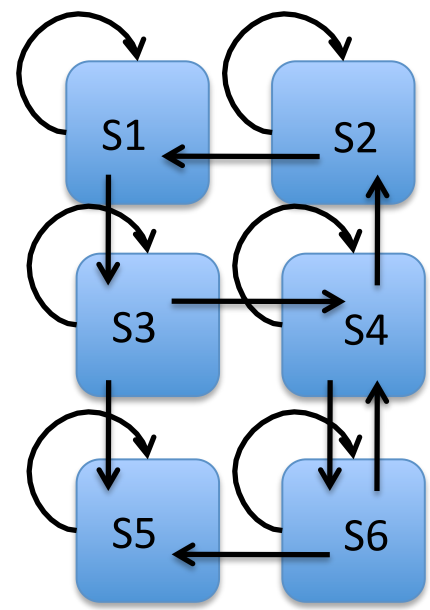
Algorithm 1 Abstraction Procedure

Input: switched system $\mathcal{S} = (X, X_0, \mathcal{A}, \{f_a\}_{a \in \mathcal{A}}, \Pi, h)$, proposition preserving partition $P = \{\mathcal{P}_i\}_{i=1}^{N}$

Output: augmented finite transition system $\mathcal{T} = (Q, Q_0, \mathcal{A}, \rightarrow_{\mathcal{T}}, \Pi, L, \mathcal{G})$ such that $\mathcal{T} \underset{\text{O.A}}{\succeq} \mathcal{S}$

Let α be the proposition preserving map
Set $Q = \{1, \ldots, N+1\}$, $Q_0 = \{i : \mathcal{P}_i \subseteq X_0\}$, $L = h \circ \alpha^{-1}$
Initialize $\rightarrow_{\mathcal{T}} = (Q \setminus \{N+1\}) \times \mathcal{A} \times Q$
for $a \in \mathcal{A}$ **do**
  $\mathcal{G}(a) = \emptyset$
  $\rightarrow_{\mathcal{T}} = \rightarrow_{\mathcal{T}} \cup \{(N+1, a, N+1)\}$
  for $i \in \{1, \ldots, N\}$ **do**
    for $j = \{1, \ldots, N+1\} \setminus \{i\}$ **do**
      if *isBlocked*$(\alpha^{-1}(i), \alpha^{-1}(j), f_a)$ **then**
        $\rightarrow_{\mathcal{T}} = \rightarrow_{\mathcal{T}} \setminus \{(i, a, j)\}$
    if *isTransient*$(\alpha^{-1}(i), f_a)$ **then**
      $\mathcal{G}(a) = \mathcal{G}(a) \cup \{\{i\}\}$
return $\mathcal{T} = (Q, Q_0, \mathcal{A}, \rightarrow_{\mathcal{T}}, \Pi, L, \mathcal{G})$

isBlocked: Yes, if we can verify that there exists no flow from one cell to the other.



Computation of Abstract Models

Algorithm 1 Abstraction Procedure

Input: switched system $\mathcal{S} = (X, X_0, \mathcal{A}, \{f_a\}_{a \in \mathcal{A}}, \Pi, h)$, proposition preserving partition $P = \{\mathcal{P}_i\}_{i=1}^{N}$

Output: augmented finite transition system $\mathcal{T} = (Q, Q_0, \mathcal{A}, \rightarrow_{\mathcal{T}}, \Pi, L, \mathcal{G})$ such that $\mathcal{T} \underset{\mathrm{O.A}}{\succeq} \mathcal{S}$

Let α be the proposition preserving map

Set $Q = \{1, \ldots, N+1\}$, $Q_0 = \{i : \mathcal{P}_i \subseteq X_0\}$, $L = h \circ \alpha^{-1}$

Initialize $\rightarrow_{\mathcal{T}} = (Q \setminus \{N+1\}) \times \mathcal{A} \times Q$

for $a \in \mathcal{A}$ **do**

  $\mathcal{G}(a) = \emptyset$

  $\rightarrow_{\mathcal{T}} = \rightarrow_{\mathcal{T}} \cup \{(N+1, a, N+1)\}$

  for $i \in \{1, \ldots, N\}$ **do**

    for $j = \{1, \ldots, N+1\} \setminus \{i\}$ **do**

      if $isBlocked(\alpha^{-1}(i), \alpha^{-1}(j), f_a)$ **then**

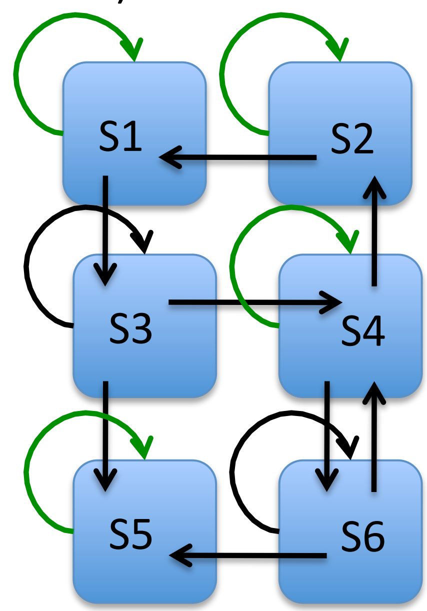
        $\rightarrow_{\mathcal{T}} = \rightarrow_{\mathcal{T}} \setminus \{(i, a, j)\}$

    if $isTransient(\alpha^{-1}(i), f_a)$ **then**

      $\mathcal{G}(a) = \mathcal{G}(a) \cup \{\{i\}\}$

return $\mathcal{T} = (Q, Q_0, \mathcal{A}, \rightarrow_{\mathcal{T}}, \Pi, L, \mathcal{G})$

isTransient: Yes, if we can verify that a cell contains no invariant sets (i.e., all trajectories eventually leave).



Computation of Abstract Models

Algorithm 1 Abstraction Procedure

Input: switched system $\mathcal{S} = (X, X_0, \mathcal{A}, \{f_a\}_{a \in \mathcal{A}}, \Pi, h)$, proposition preserving partition $P = \{\mathcal{P}_i\}_{i=1}^{N}$

Output: augmented finite transition system $\mathcal{T} = (Q, Q_0, \mathcal{A}, \to_\mathcal{T}, \Pi, L, \mathcal{G})$ such that $\mathcal{T} \underset{\text{O.A}}{\succeq} \mathcal{S}$

Let α be the proposition preserving map

Set $Q = \{$

Initialize

for $a \in$

$\mathcal{G}(a) =$

$\to_\mathcal{T} =$

for $i \in$

for

if *isBlocked*$(\alpha^{-1}(i), \alpha^{-1}(j), f_a)$ **then**

$\to_\mathcal{T} = \to_\mathcal{T} \setminus \{(i, a, j)\}$

if *isTransient*$(\alpha^{-1}(i), f_a)$ **then**

$\mathcal{G}(a) = \mathcal{G}(a) \cup \{\{i\}\}$

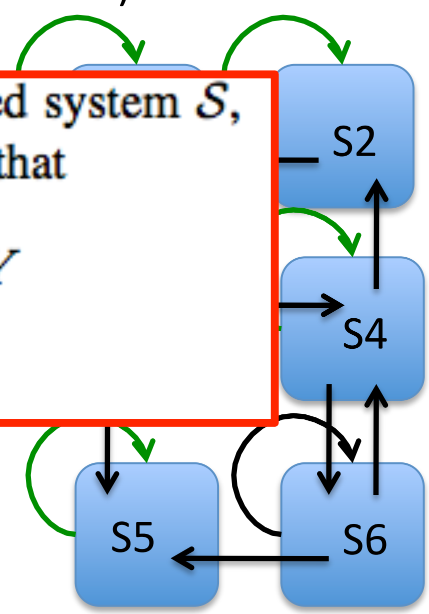
return $\mathcal{T} = (Q, Q_0, \mathcal{A}, \to_\mathcal{T}, \Pi, L, \mathcal{G})$

A set Y is transient on a mode a of a switched system $\mathcal{S}$, if there exists a $\mathcal{C}^1$ function $B : \mathbb{R}^n \to \mathbb{R}$ such that

$$\dot{B}(\xi) = \frac{\partial B(\xi)}{\partial \xi} f_a(\xi) \leq -\varepsilon, \quad \forall \xi \in Y$$

for some $\varepsilon > 0$.

isTransient: Yes, if we can verify that a cell contains no invariant sets (i.e., all trajectories eventually leave).



Computation of Abstract Models

Algorithm 1 Abstraction Procedure

Input: switched system $\mathcal{S} = (X, X_0, \mathcal{A}, \{f_a\}_{a \in \mathcal{A}}, \Pi, h)$, proposition preserving partition $P = \{\mathcal{P}_i\}_{i=1}^{N}$

Output: augmented finite transition system $\mathcal{T} = (Q, Q_0, \mathcal{A}, \to_\mathcal{T}, \Pi, L, \mathcal{G})$ such that $\mathcal{T} \underset{\mathrm{O.A}}{\succeq} \mathcal{S}$

Let α be the proposition preserving map
Set $Q = \{1, \ldots, N+1\}$, $Q_0 = \{i : \mathcal{P}_i \subseteq X_0\}$, $L = h \circ \alpha^{-1}$
Initialize $\to_\mathcal{T} = (Q \setminus \{N+1\}) \times \mathcal{A} \times Q$
for $a \in \mathcal{A}$ **do**
  $\mathcal{G}(a) = \emptyset$
  $\to_\mathcal{T} = \to_\mathcal{T} \cup \{(N+1, a, N+1)\}$
  for $i \in \{1, \ldots, N\}$ **do**
    for $j = \{1, \ldots, N+1\} \setminus \{i\}$ **do**
      if *isBlocked*$(\alpha^{-1}(i), \alpha^{-1}(j), f_a)$ **then**
        $\to_\mathcal{T} = \to_\mathcal{T} \setminus \{(i, a, j)\}$
    if *isTransient*$(\alpha^{-1}(i), f_a)$ **then**
      $\mathcal{G}(a) = \mathcal{G}(a) \cup \{\{i\}\}$
return $\mathcal{T} = (Q, Q_0, \mathcal{A}, \to_\mathcal{T}, \Pi, L, \mathcal{G})$

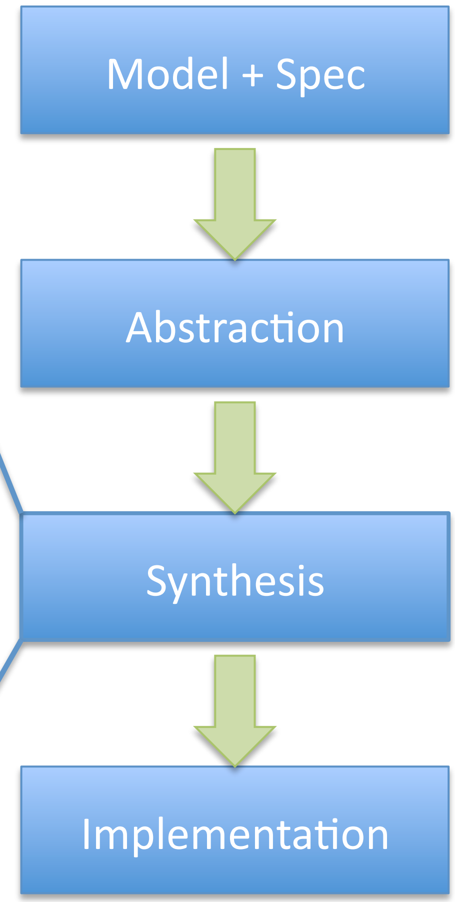
- isBlocked and isTransient can be efficiently computed for linear dynamics
- Computable via polynomial algebra and quantifier elimination for polynomial dynamics
- “Efficiently” computable for polynomial dynamics by using convex relaxations and semidefinite programming

Synthesis

Two player discrete game between the abstract states of the low level dynamic modes and the switching controller (Pnueli, Ramadge & Wonham).

Output:

- If realizable -> control automaton
- If not
  - partial controller and suggestion for refinement
  - impossibility certificate

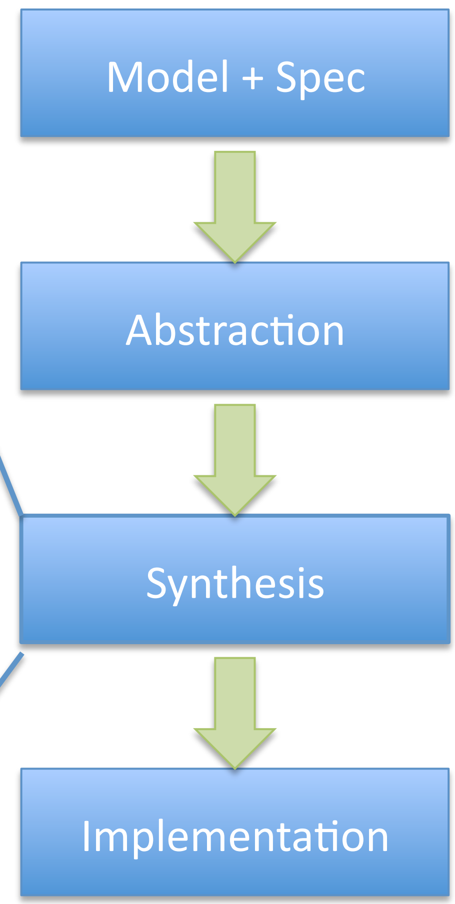


Synthesis (runtime reactiveness)

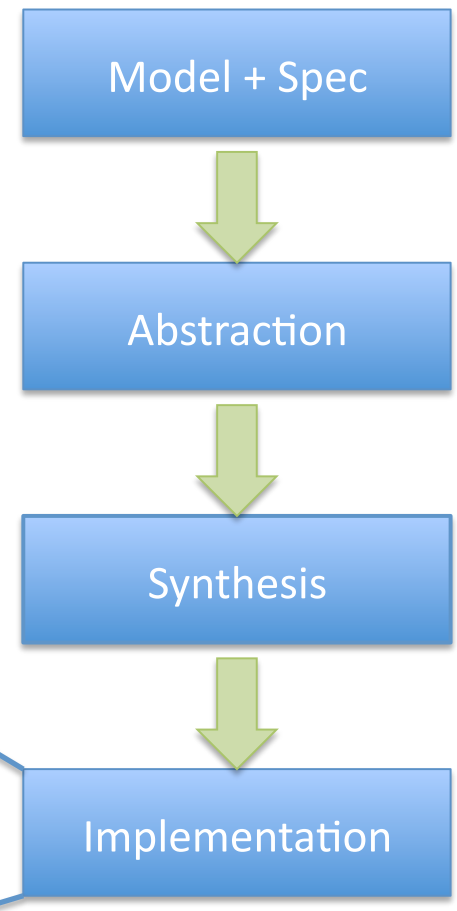
Two player discrete game between the abstract states of the abstracted nondeterministic dynamics and external environment (just define an asynchronous products of T and E) and the switching controller

Output:

- If realizable -> control automaton
- If not
  - partial controller and suggestion for refinement
  - impossibility certificate



Implementation

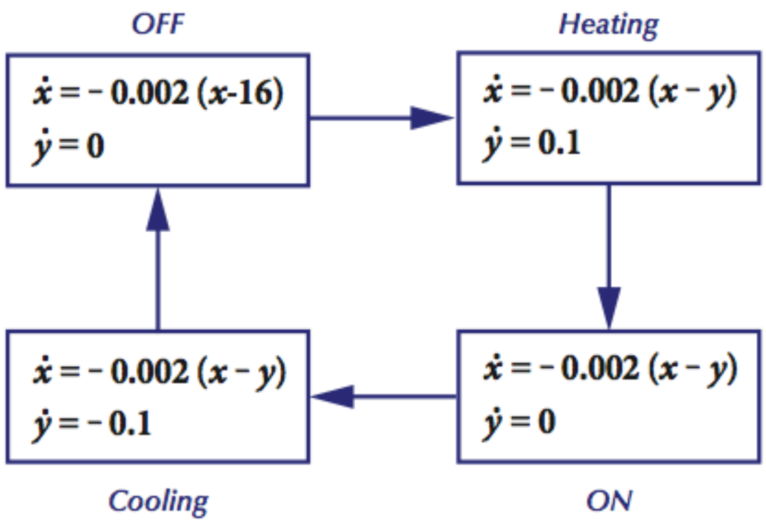


Given the automaton, make sure they can be implemented:
- Need to be careful about Zenoness

Details in TAC 2013, ACC 2013.

Example: Heater

A four-mode thermostat:



x: room temperature
y: heater temperature
mode: {off, on, heating, cooling}

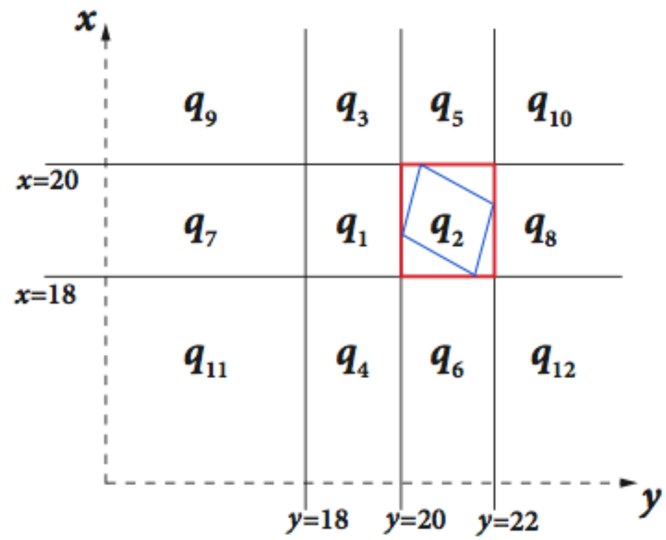
Specification

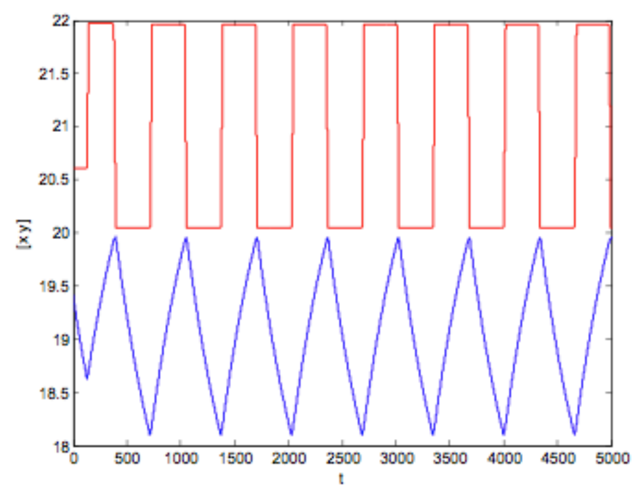
Design a switching sequence such that

P1 $\Diamond(18 \leq x \leq 20 \wedge 20 \leq y \leq 22)$

P2 $(18 \leq x \leq 20 \wedge 20 \leq y \leq 22) \rightarrow \Box(18 \leq x \leq 20 \wedge 20 \leq y \leq 22))$

Over-approximation

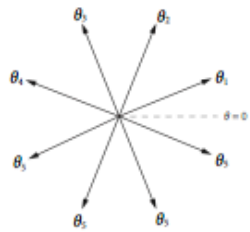




Example: Motion Planning

2d robot motion planning:

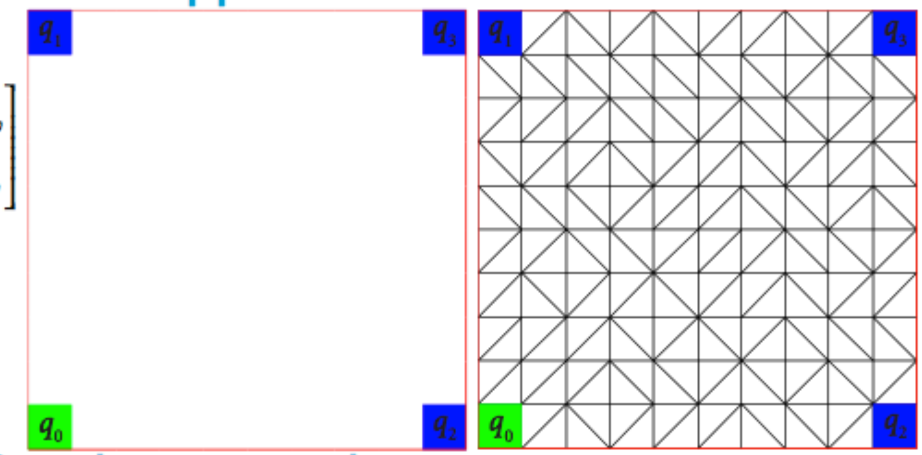
$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}\cos\theta & 0\\ \sin\theta & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}v\\ w\end{bmatrix}$$



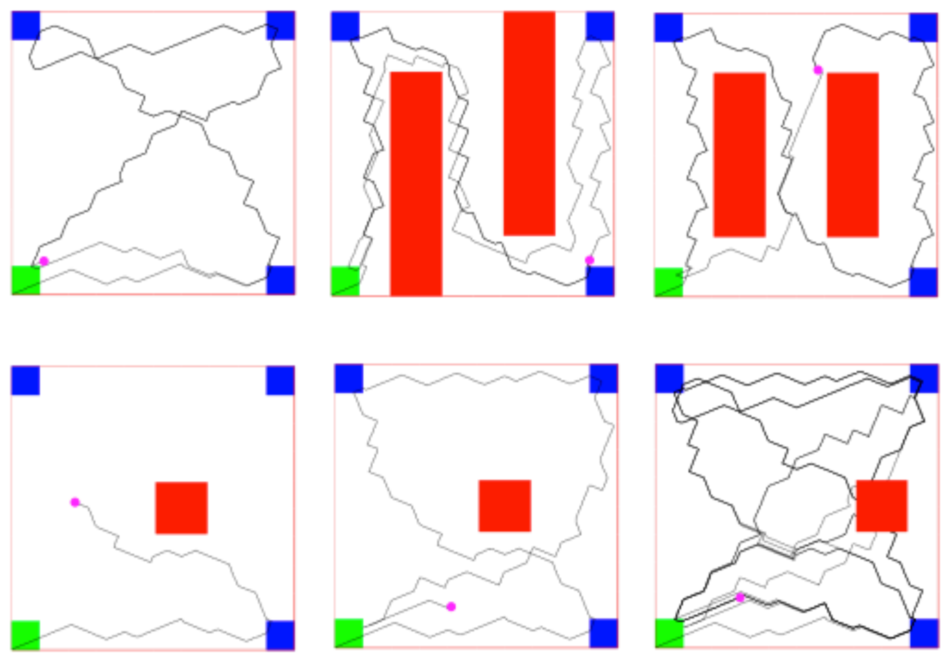
$$\begin{bmatrix}\dot{x}\\ \dot{y}\end{bmatrix} = \begin{bmatrix}v_0\cos\theta_p\\ v_0\sin\theta_p\end{bmatrix}$$

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

Over-approximation



Simulation results



Specification

Design a switching sequence such that

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

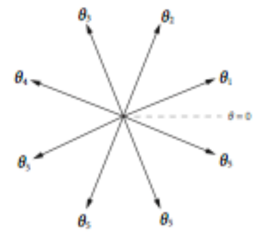
P2 $\Box(\text{PARK} \rightarrow \Diamond q_0)$

while avoiding static and moving obstacles.

Example: Motion Planning

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

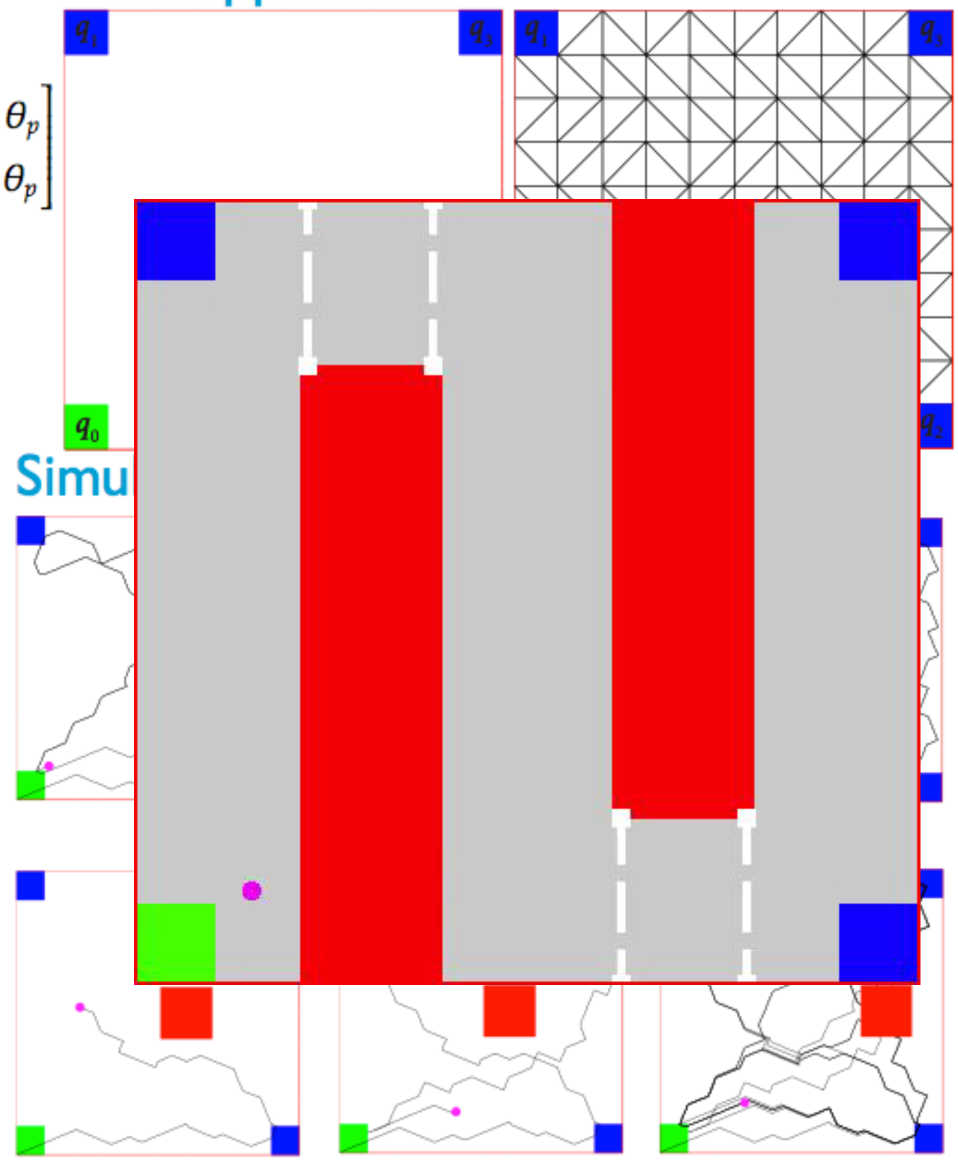
Design a switching sequence such that

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

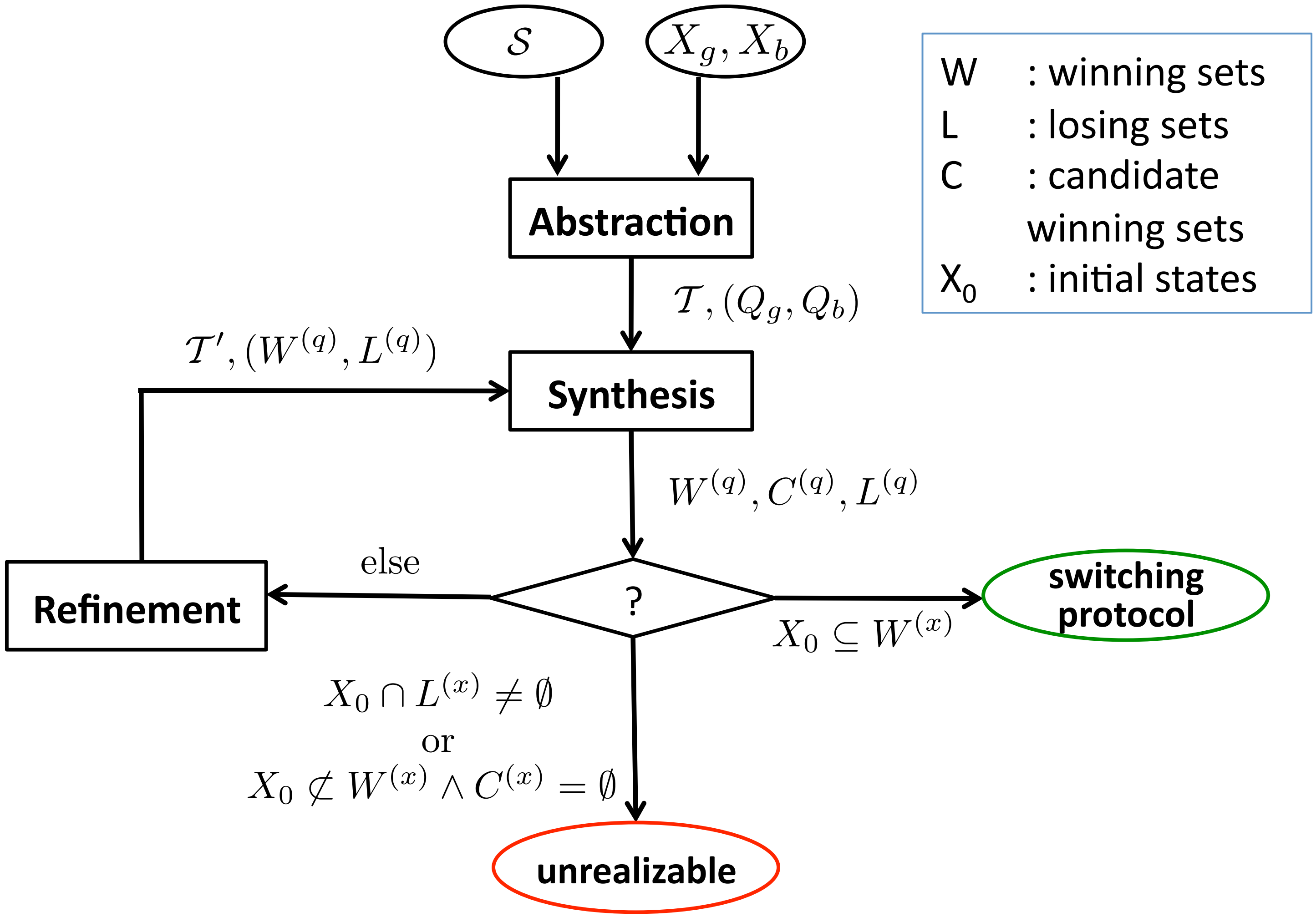
P2 $\square(\text{PARK} \rightarrow \Diamond q_0)$

while avoiding static and moving obstacles and pedestrians.

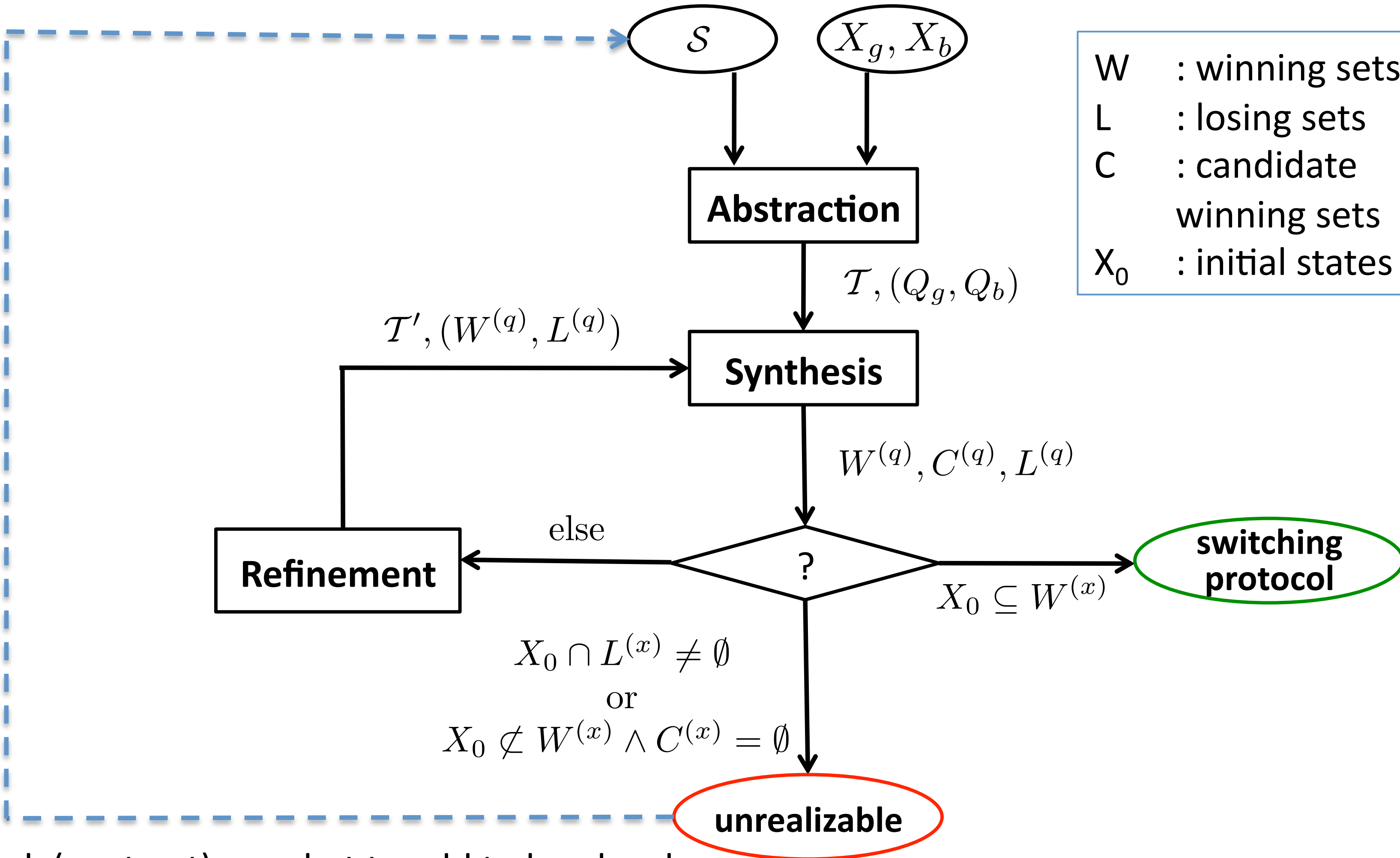
Over-approximation



Ongoing work: Extended CEGAR for Synthesis



Ongoing work: Extended CEGAR for Synthesis



W	: winning sets
L	: losing sets
C	: candidate winning sets
X_0	: initial states

feedback (contract) on what to add to low-level

Summary

- Switching protocol synthesis:
  - A novel abstract model by augmented finite transition systems (more achievable behavior with same sized abstractions)
  - Efficient computation of abstraction and refinements

- Current & Future work:
  - CEGAR (initial results -> abstractions driven by specs -> for scalability)
  - Send feedback to the low-level control designers in case of impossibility
  - Incorporate implementation uncertainties, allow digital implementations
  - Beyond LTL? Hard-time constraints.
  - Feedback on spec's -> analyzing potential reasons of unrealizability (e.g. Bloem et al.)

Questions?

- Thanks to:
  - Organizers
  - Collaborators: Jun Liu (Sheffield), Richard M. Murray (Caltech), Ufuk Topcu (Penn), Pavithra Prabhakar (IMDEA)
  - Funding: IBM and UTC through iCyPhy consortium
  - Audience ☺