Integrating Induction, Deduction and Structure for Synthesis

Sanjit A. Seshia

Associate Professor

EECS Department

UC Berkeley

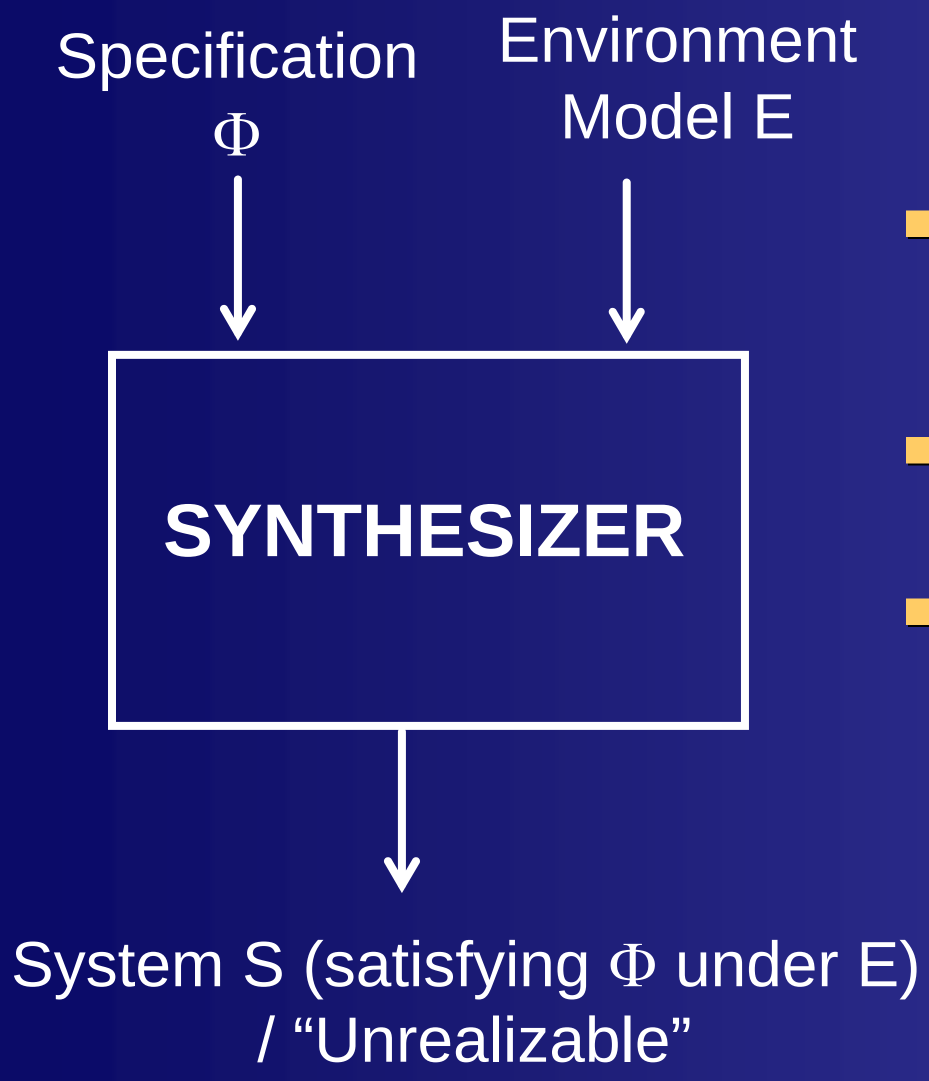
Students: **S. Jha, W.Li, L. Dworkin, D. Sadigh**

Collaborators: A. Tiwari, S. Gulwani

NSF ExCAPE e-Seminar

November 5, 2012

Synthesis and Its Challenges



- **Specification can be incomplete or difficult to use**
- **Environment model might be deficient**
- **High Complexity of synthesis**

Reverse Engineering Malware

Obfuscated code:

<u>Input:</u> y   <u>Output:</u> modified value of y

```
{ a=1; b=0; z=1; c=0;
  while(1) {
   if (a == 0) {
   if (b == 0) { y=z+y;
   b=~b;  c=~c; if (~c)
  else {
    z=z+y; a=~a; b=~b; c=~c;
    if (~c) break; } }
   else if (b == 0) {z=y << 2; a=~a;}
    else { z=y << 3; a=~a; b=~b;}
 } }
```

CHALLENGE:
SPECIFICATION CONTROLLED BY 'ATTACKER'

What it does:

y = y * 45

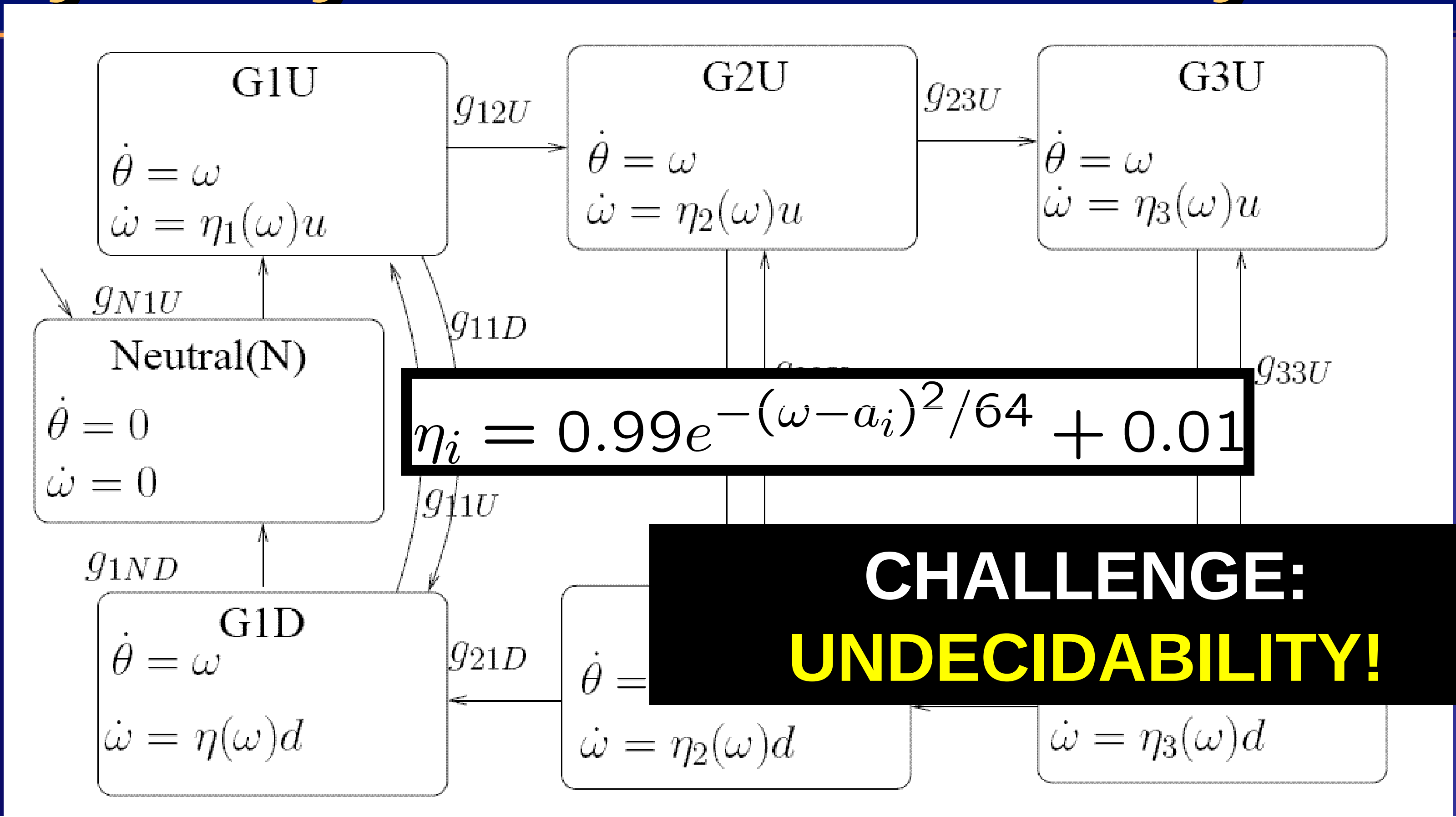
using

esis.

FROM
CONFICKER WORM

Paper: S. Jha et al., "Oracle-Guided Component-Based Program Synthesis", ICSE 2010.

Synthesizing Switching Logic (Hybrid Systems with Nonlinear Dynamics)



$$\eta_i = 0.99e^{-(\omega - a_i)^2/64} + 0.01$$

CHALLENGE:
UNDECIDABILITY!

Synthesize switch between the gears such that
- Efficiency η_i always exceeds 0.5 when $\omega \geq 5$
- Distance to be covered takes minimum time.

Paper: S. Jha et al., "Synthesizing Switching Logic for Safety and Dwell-Time Requirements", Intl. Conf. on Cyber-Physical Systems, April 2010.

Reactive Synthesis from LTL

CHALLENGE:
ENVIRONMENT SPECIFICATION

Often due to missing environment assumptions!

Paper: W. Li et al., "Mining Assumptions for Synthesis", MEMOCODE 2011.

Strategies to Address Challenges

- **Human Input / Insight**
  - Hypothesis on the form of artifact to be synthesized
  - "Structure Hypothesis"
  - Examples:
    - Sketch [Bodik, Solar-Lezama, et al.]
    - Component library [Jha et al., …]
- **Induction + Deduction**
  - Induction: specific examples → general rules
    - Learning from examples
  - Deduction: general rules → specific conclusions
    - Logical inference and constraint solving
  - Purely deductive approach is inefficient / inapplicable
  - Purely inductive approach gives no guarantees

Approach: Sciduction

Structure-Constrained Induction and Deduction

Structure Hypotheses
(on artifacts to be synthesized)

SKETCH,
TEMPLATE,
Etc.

Deductive Procedure
"Lightweight": solves lower complexity problem or special case of original decision problem

Inductive Procedure
Active Learning: selects examples to learn from

S. A. Seshia, "Sciduction: Combining Induction, Deduction, and Structure for Verification and Synthesis", DAC 2012

Outline

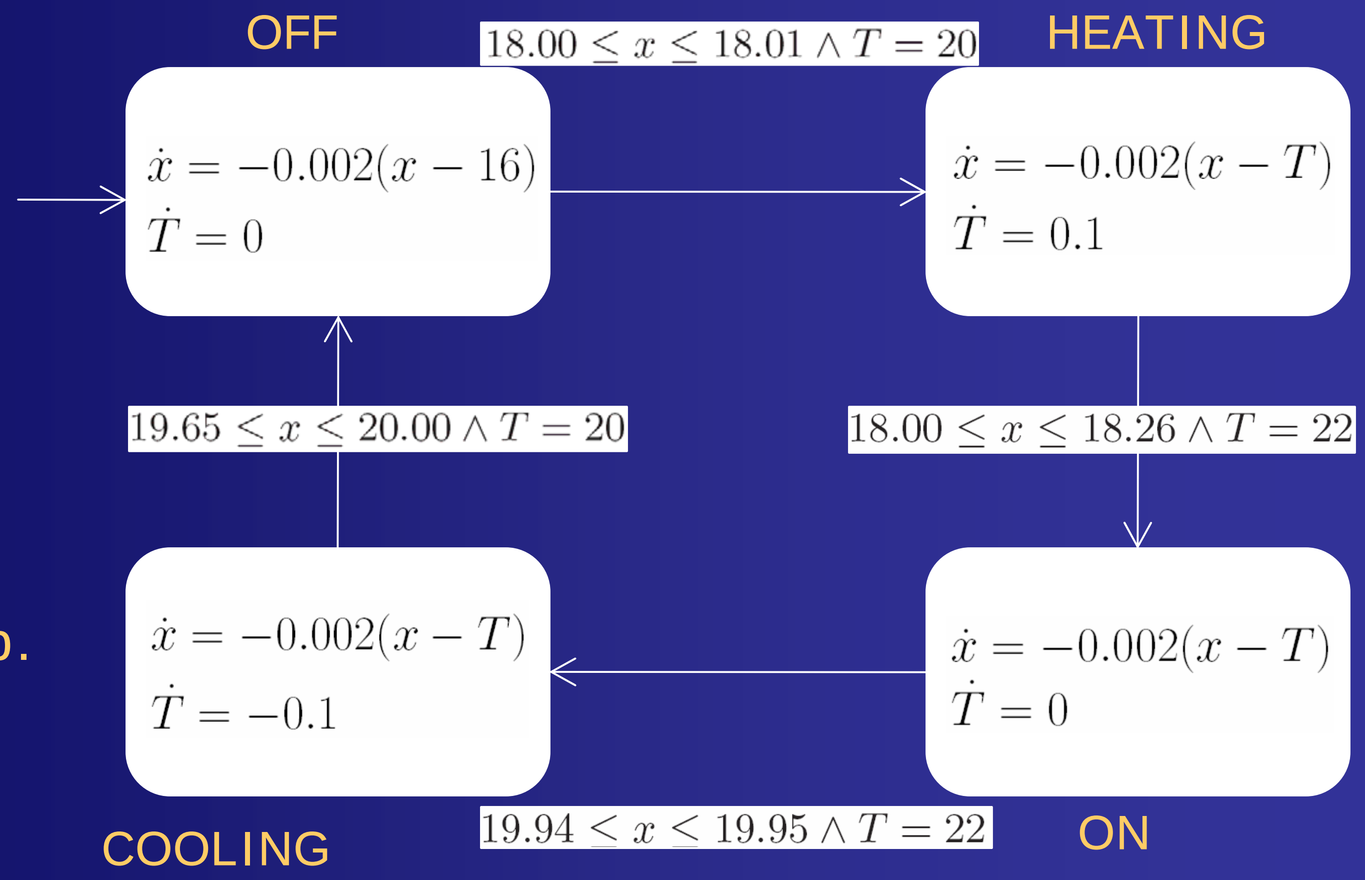
- Switching Logic Synthesis for Hybrid Systems
- Reflections on the Approach
- Synthesizing Environment Assumptions for Reactive Synthesis from LTL
- Conclusions and Future Directions

Hybrid Automata

Dynamics: Location → Ordinary Differential Equations (ODEs)

Switching Logic ⊆ Location X Location → Predicates

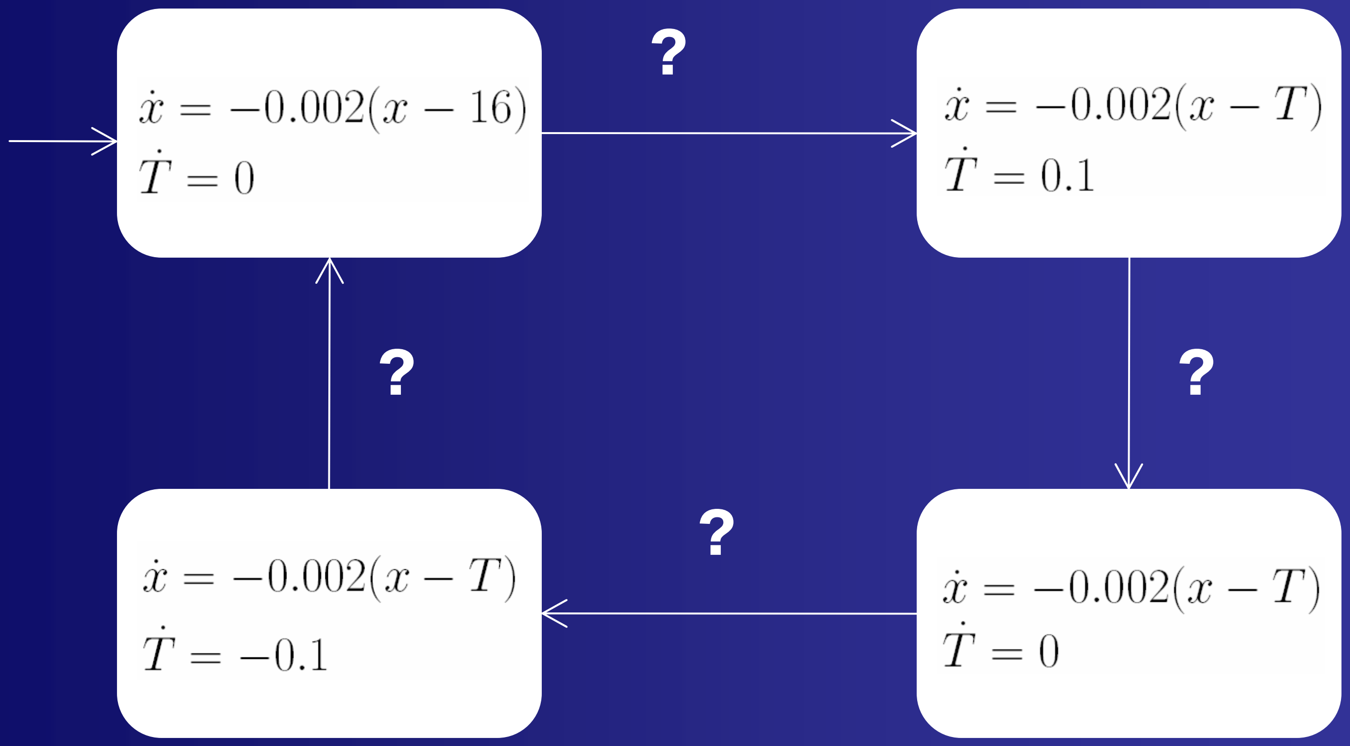
A guard is taken as soon as it is enabled.



x – room temp.

T – coil temp.

Switching Logic Synthesis Problem



- SAFETY: Temperature x must lie between 18 and 20 C.
  - In general: safe region is a hyperbox
- OPTIMALITY: Minimize switching between the modes

Our Solution

Structure Hypothesis:
Guards are Hyperboxes

+

Inductive Inference:
Learning Hyperbox from Examples

+

Deductive Procedure:
Numerical Simulation within a mode

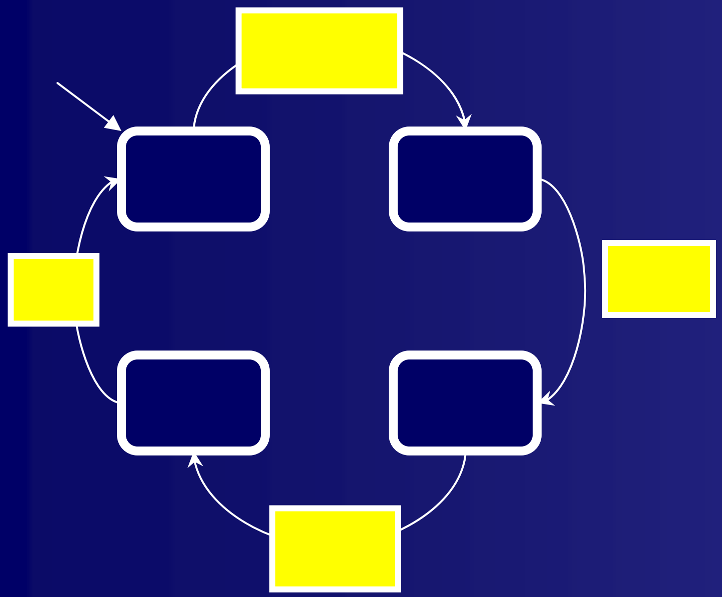
Related Work

- Synthesis through Game Solving or Reachability [Asarin et al, IEEE 2000; Koo et al., HSCC 01]
- Reduce to finite state automata synthesis [Tabuada, SCL 08]
- Constraint-based approach [Taly et al. VMCAI 09]

- Our approach is different in:
  - Hypothesis about guard structure
  - Use of hyperbox learning from examples
  - Numerical simulation as a 'decision procedure'

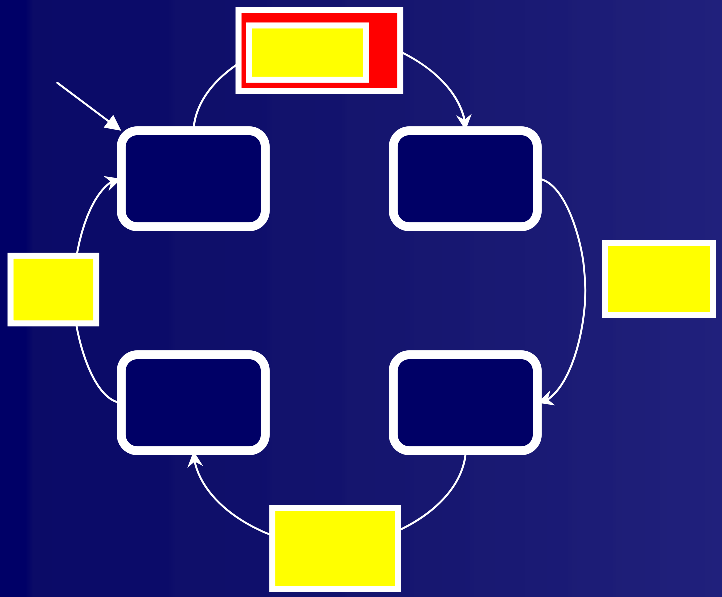
Overview of Approach

Fixpoint computation connects
Learning and Numerical Simulation



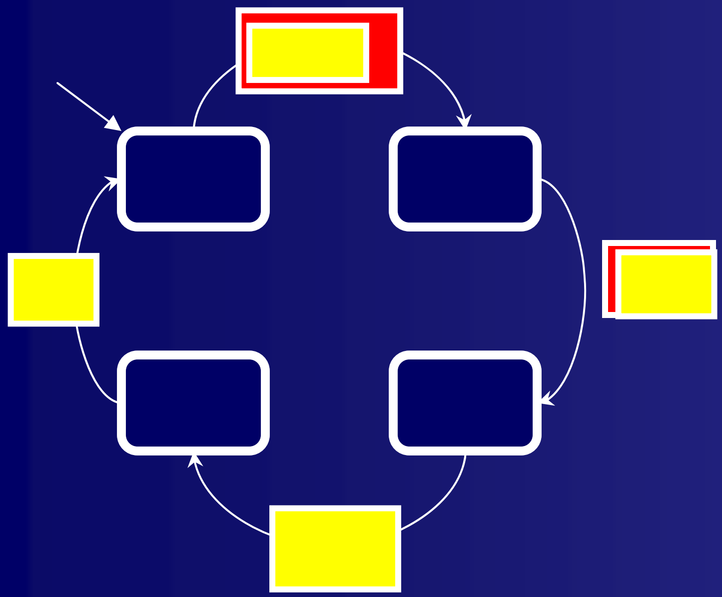
Overview of Approach

Fixpoint computation connects
Learning and Numerical Simulation



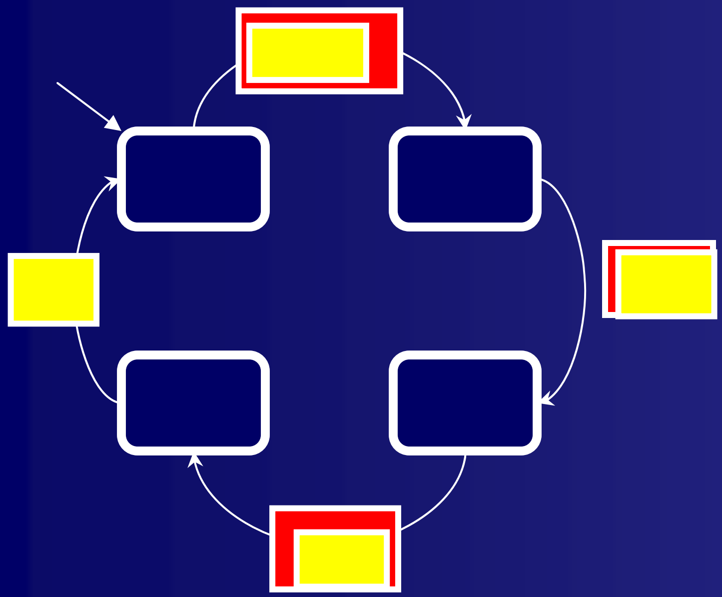
Overview of Approach

Fixpoint computation connects
Learning and Numerical Simulation



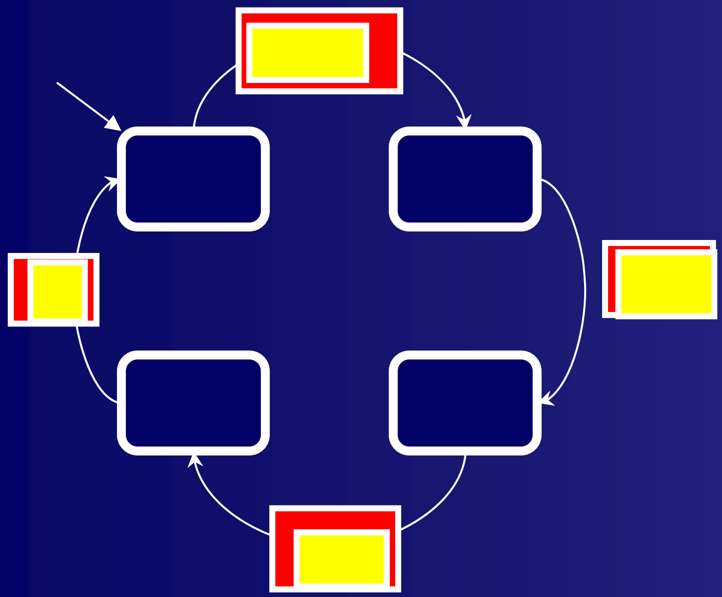
Overview of Approach

Fixpoint computation connects
Learning and Numerical Simulation



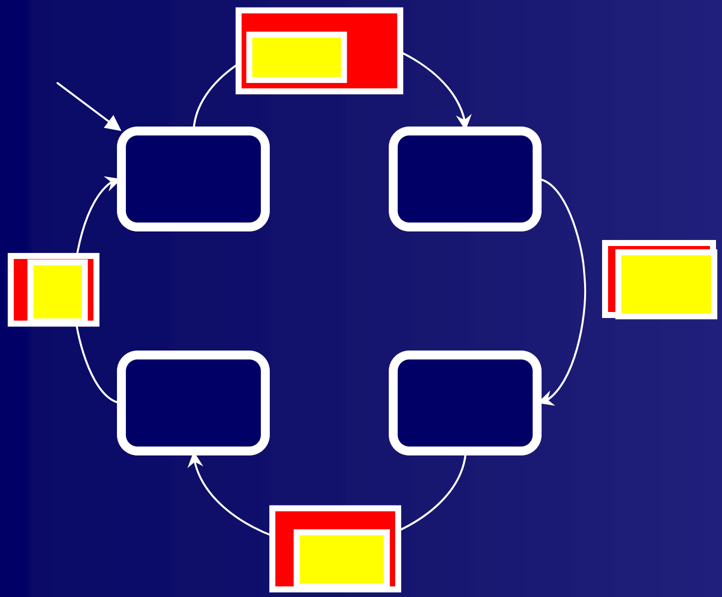
Overview of Approach

Fixpoint computation connects
Learning and Numerical Simulation



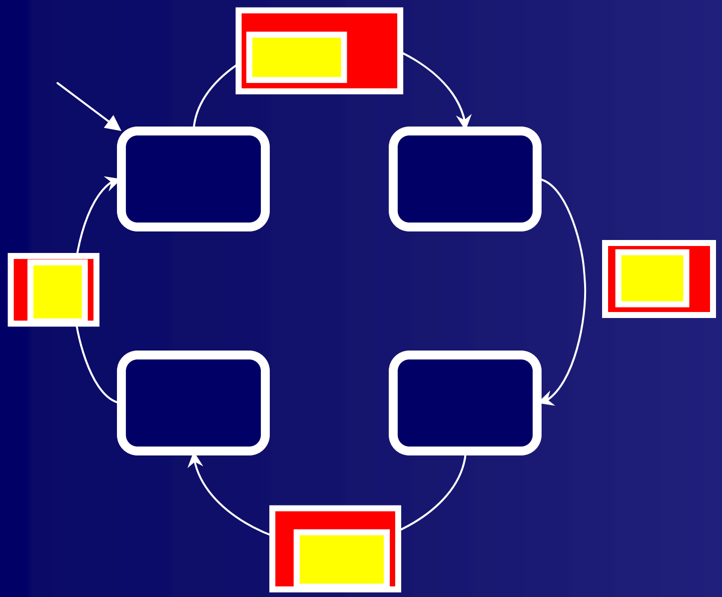
Overview of Approach

Fixpoint computation connects
Learning and Numerical Simulation



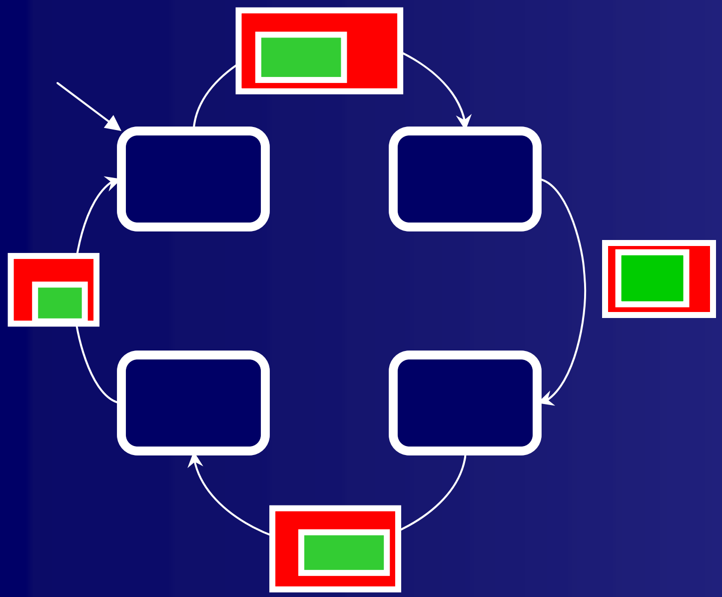
Overview of Approach

Fixpoint computation connects
Learning and Numerical Simulation



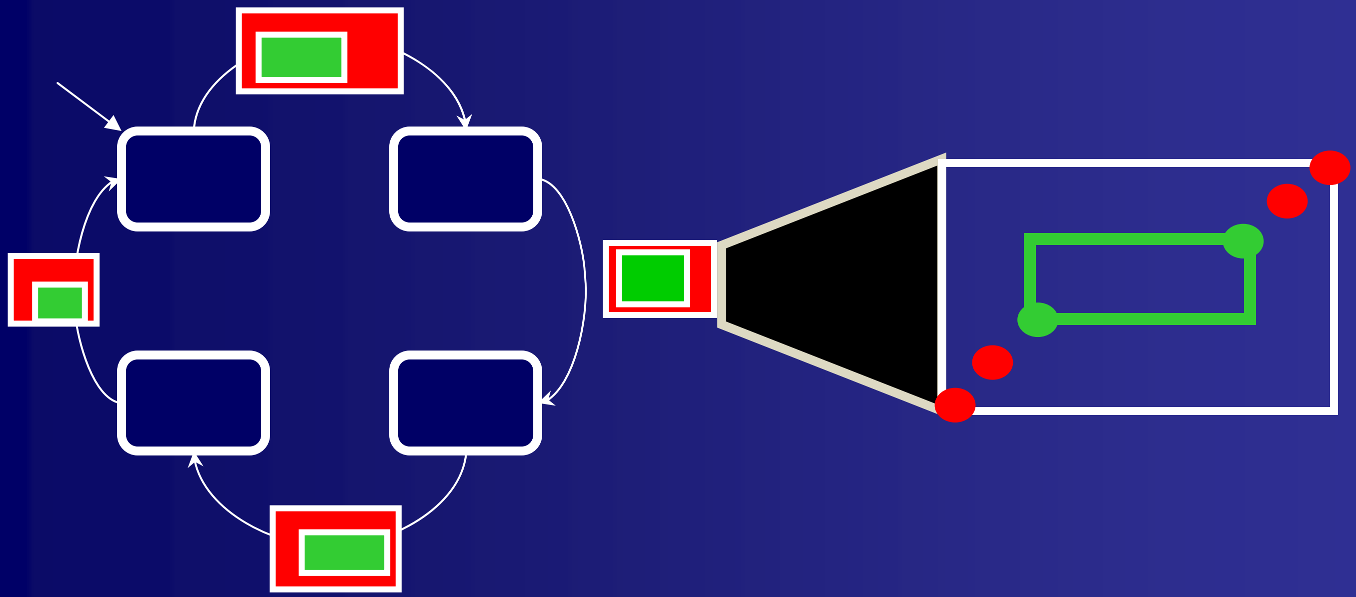
Overview of Approach

Fixpoint computation connects
Learning and Numerical Simulation



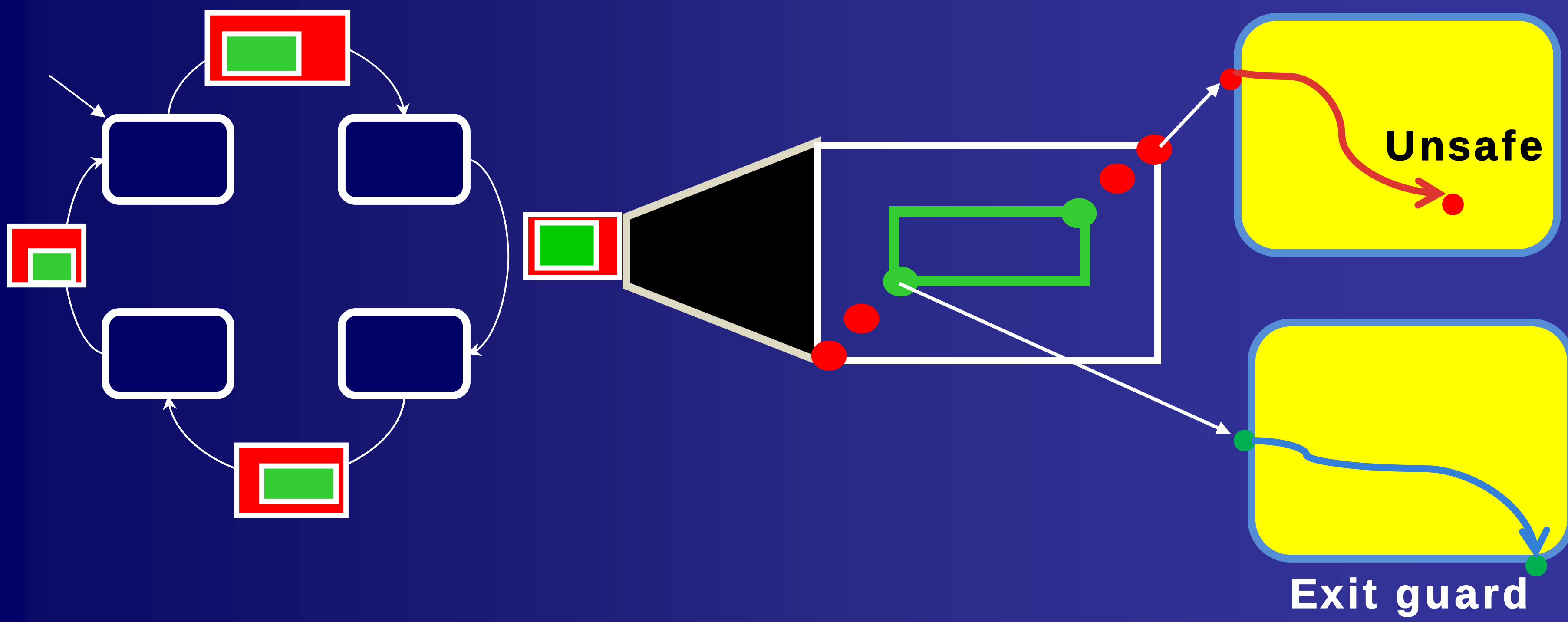
Overview of Approach

Fixpoint computation connects
Learning and Numerical Simulation



Overview of Approach

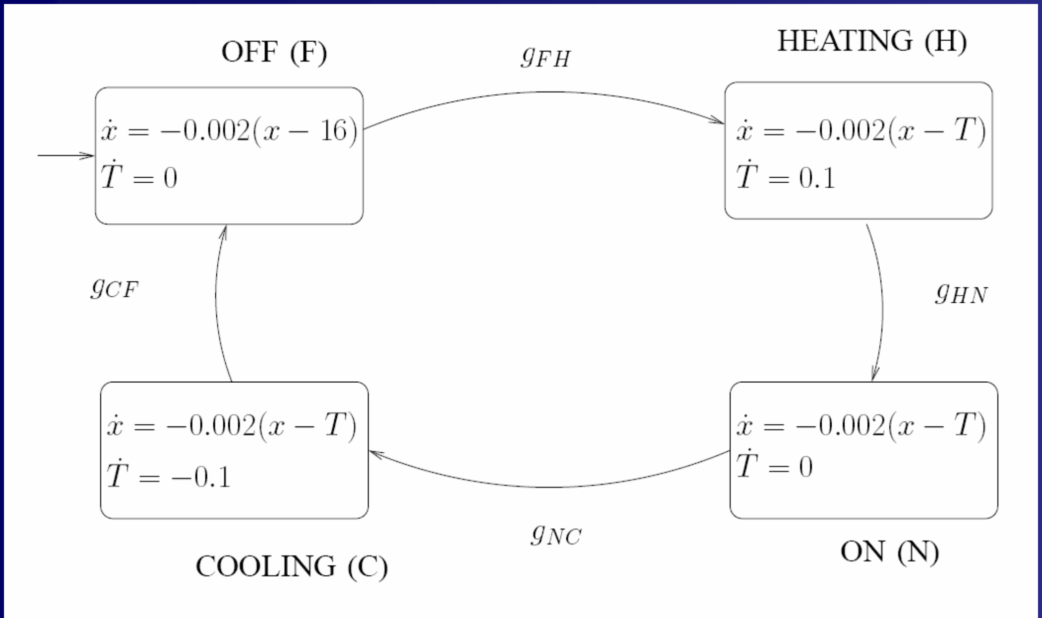
Fixpoint computation connects
Learning and Numerical Simulation



Fixpoint Computation Constraints

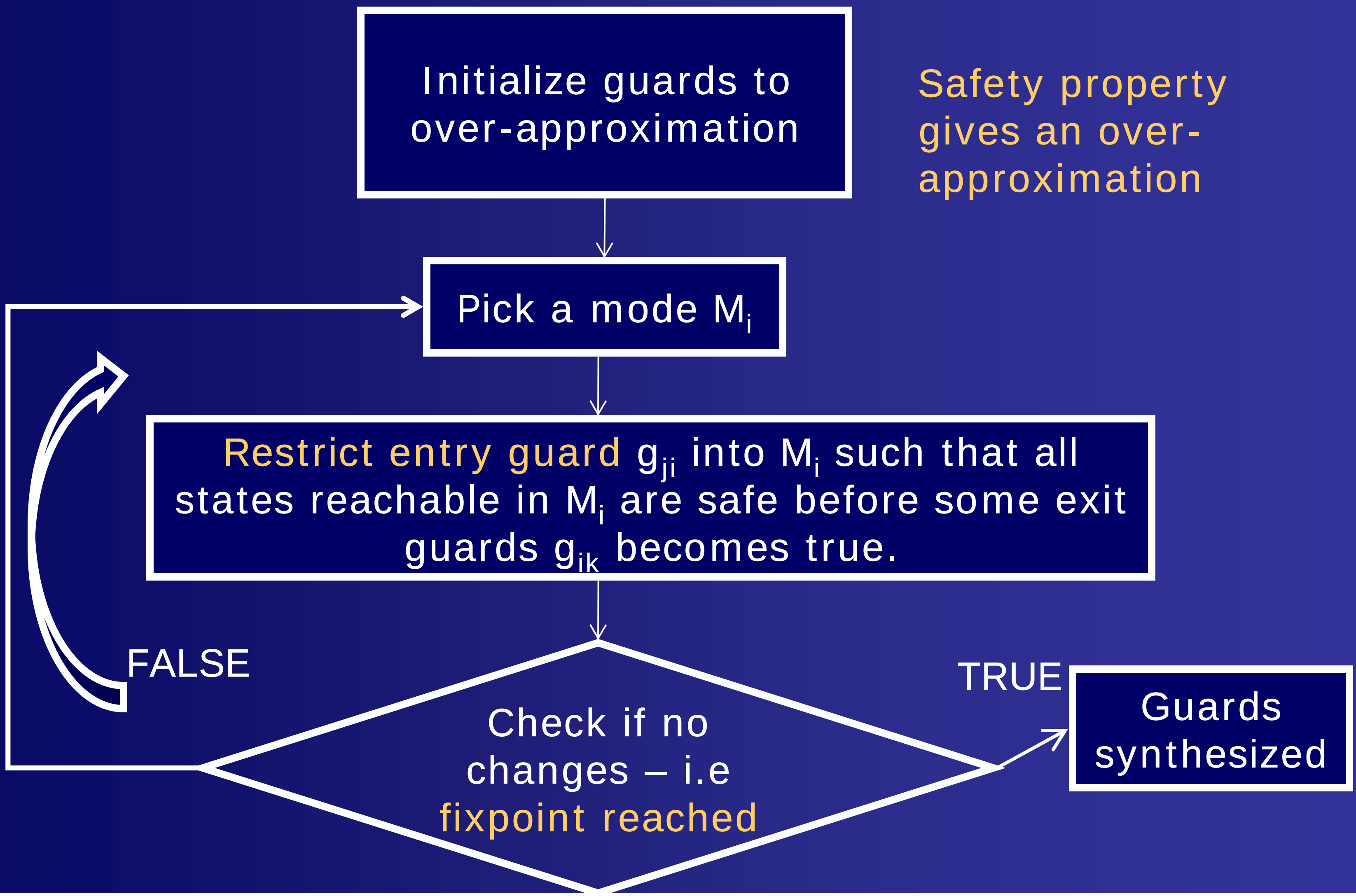
$$Mode_1,\ I \models \phi_S \mathbf{W}(\bigvee_{k \in M} g_{1k})$$

$$Mode_i, \bigvee_{j \in M} g_{ji} \models \phi_S \mathbf{W}(\bigvee_{k \in M} g_{ik}) \text{ for } i = 1..k$$



$$F, I \models \phi_S\ \mathbf{W}\ g_{FH}$$
$$F, g_{CF} \models \phi_S\ \mathbf{W}\ g_{FH}$$
$$H, g_{FH} \models \phi_S\ \mathbf{W}\ g_{HN}$$
$$N, g_{HN} \models \phi_S\ \mathbf{W}\ g_{NC}$$
$$C, g_{NC} \models \phi_S\ \mathbf{W}\ g_{CF}$$

Fixpoint Computation



Structure Hypothesis

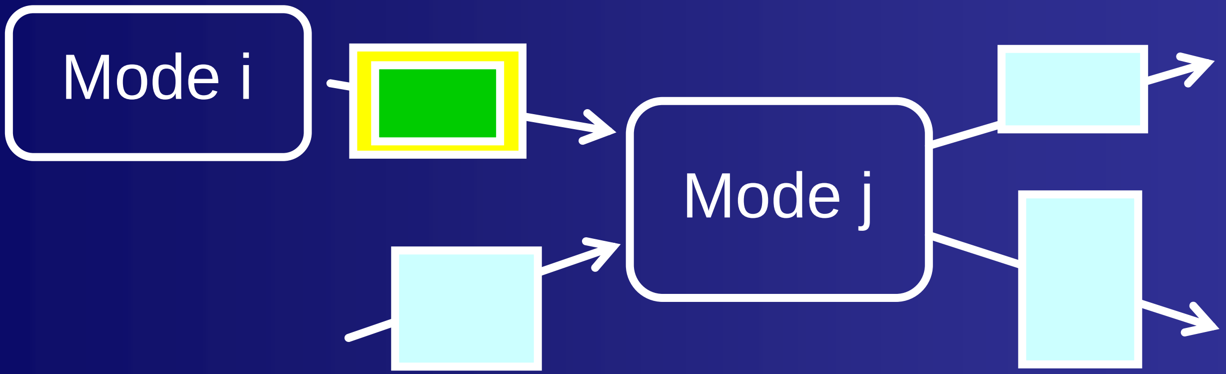
- Guard region has a restricted geometric form
- Equivalently: guard is a restricted type of logical formula
- Hyperbox:

  Interval logic: $c1 \leq x \leq c2 \wedge c3 \leq y \leq c4$

Learning Problem

- **Given:**
  - An overapproximate hyperbox for guard g_{ij}
  - Overapproximate hyperboxes for all other guards in/out of Mode j
  - One safe switching state from Mode i to Mode j
  - Reachability oracle for single switching pts into Mode j

- **Find:** Safe hyperbox restriction of g_{ij}

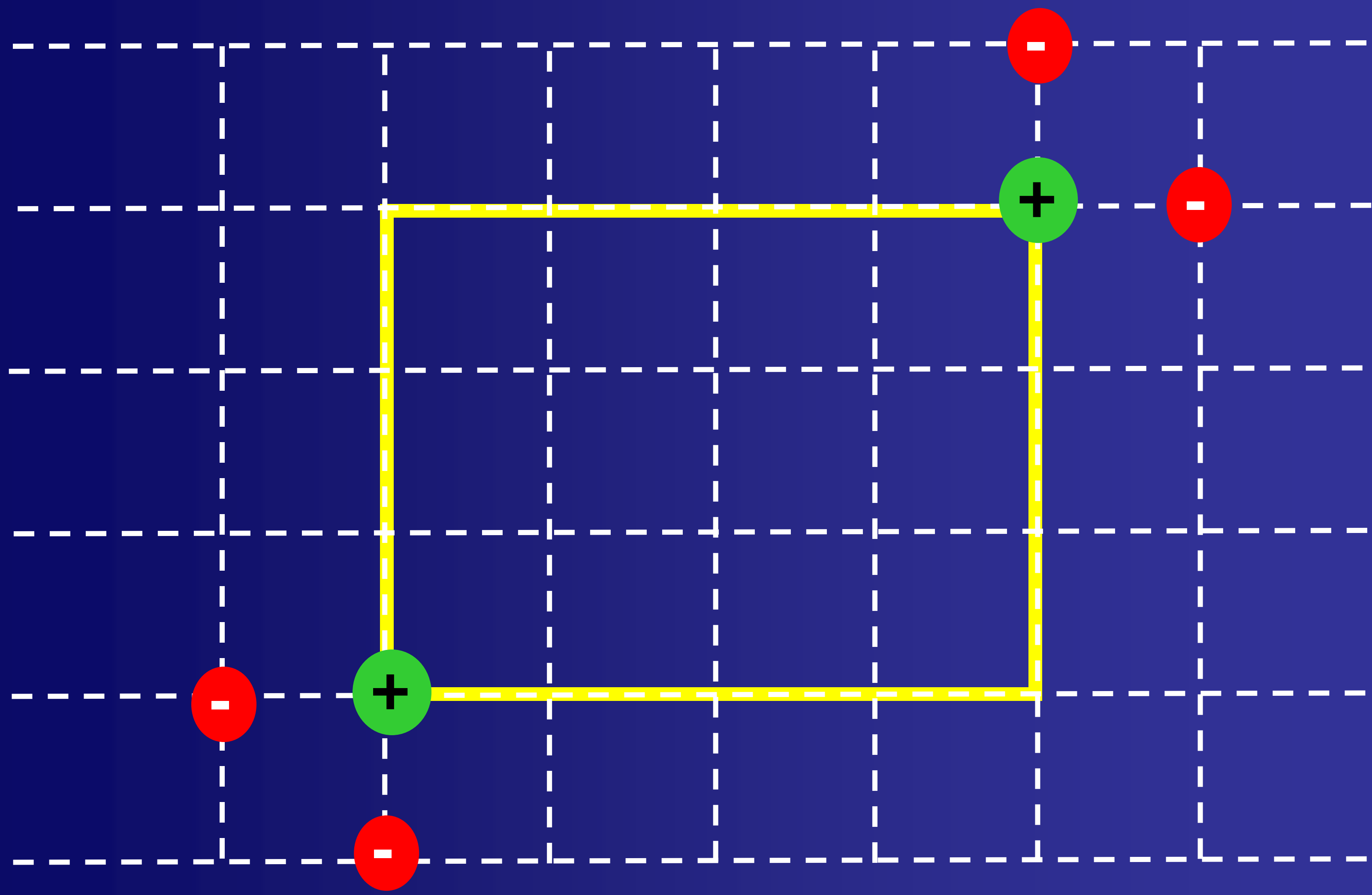


Optimal Teaching Sequence

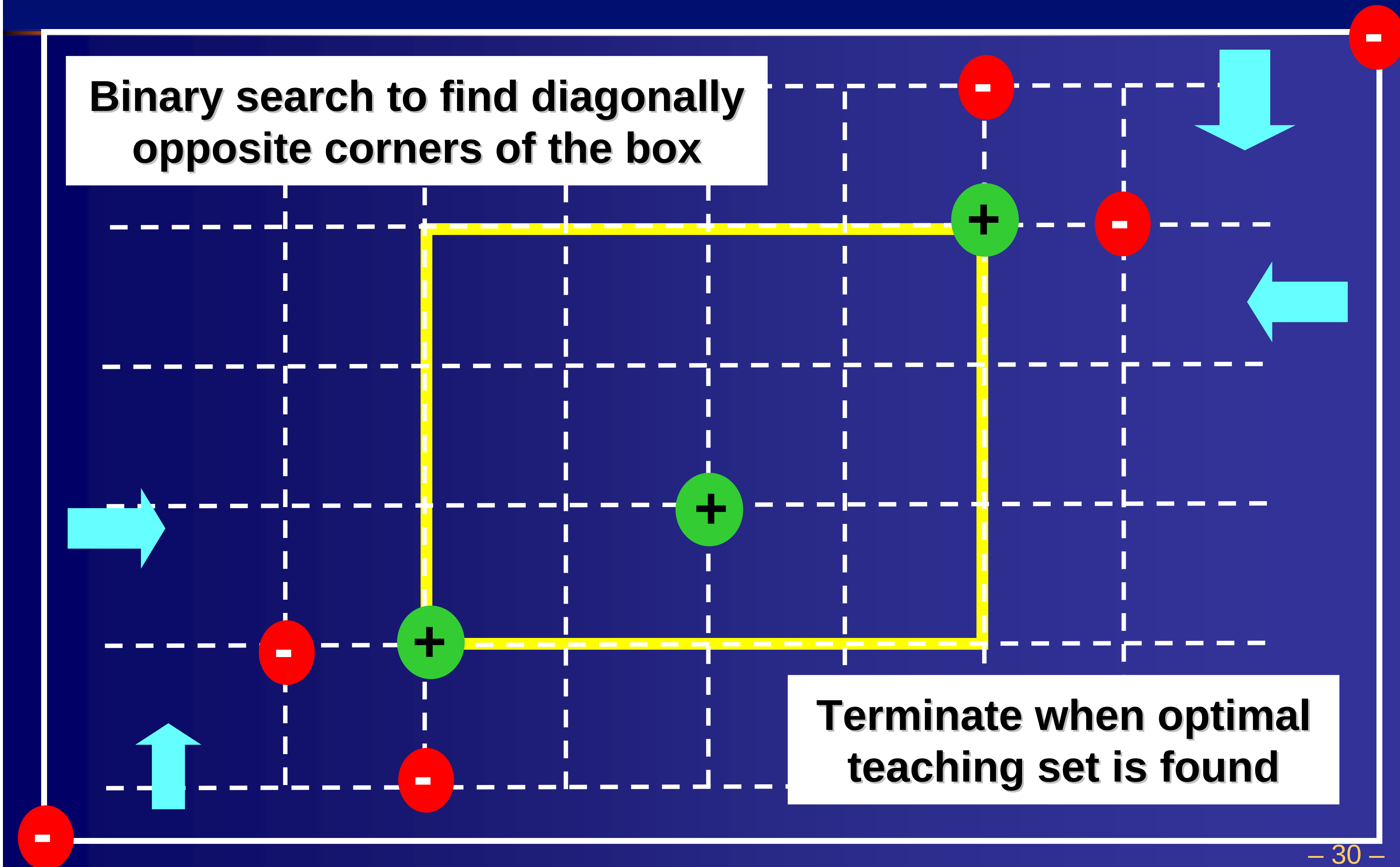
- **Teaching problem:**

  If we knew what the guard was, what is the smallest set of examples to give to a learner to uniquely identify it?

Teaching an N-dimensional Box



Learning an N-dimensional Box



Complexity

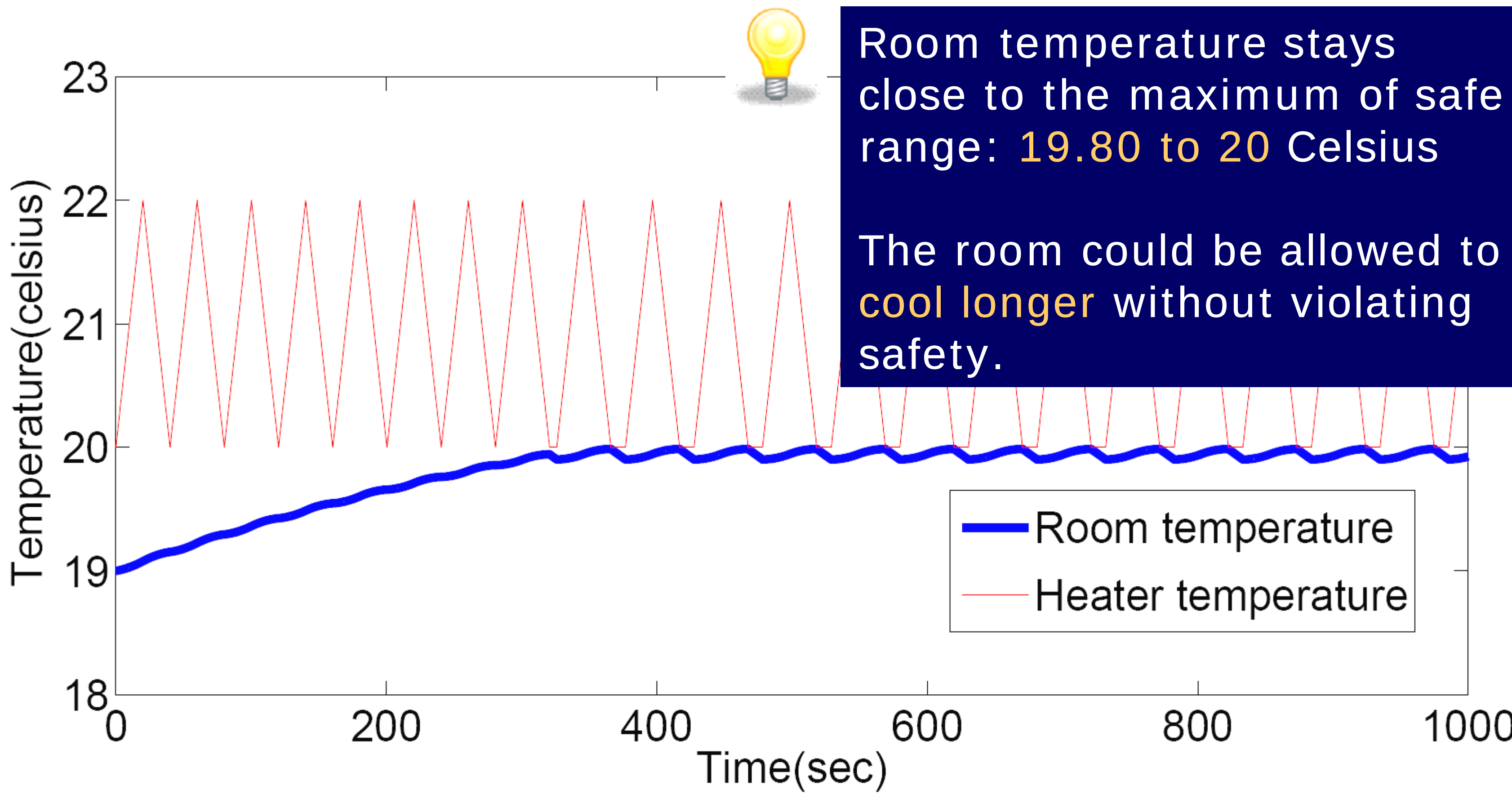
Number of simulations needed:

- Box (Conjunction of intervals): poly(log R, d)
- Linearly independent linear inequalities: poly(log R, d, n)
- Conjunction of arbitrary linear constraints: Exponential in d (best known)

(learning convex polygons is NP-hard – COLT94,98)

R is range of variables, d is number of variables (dimension), n is the number of constraints.

Synthesized Thermostat for Safety Requirement



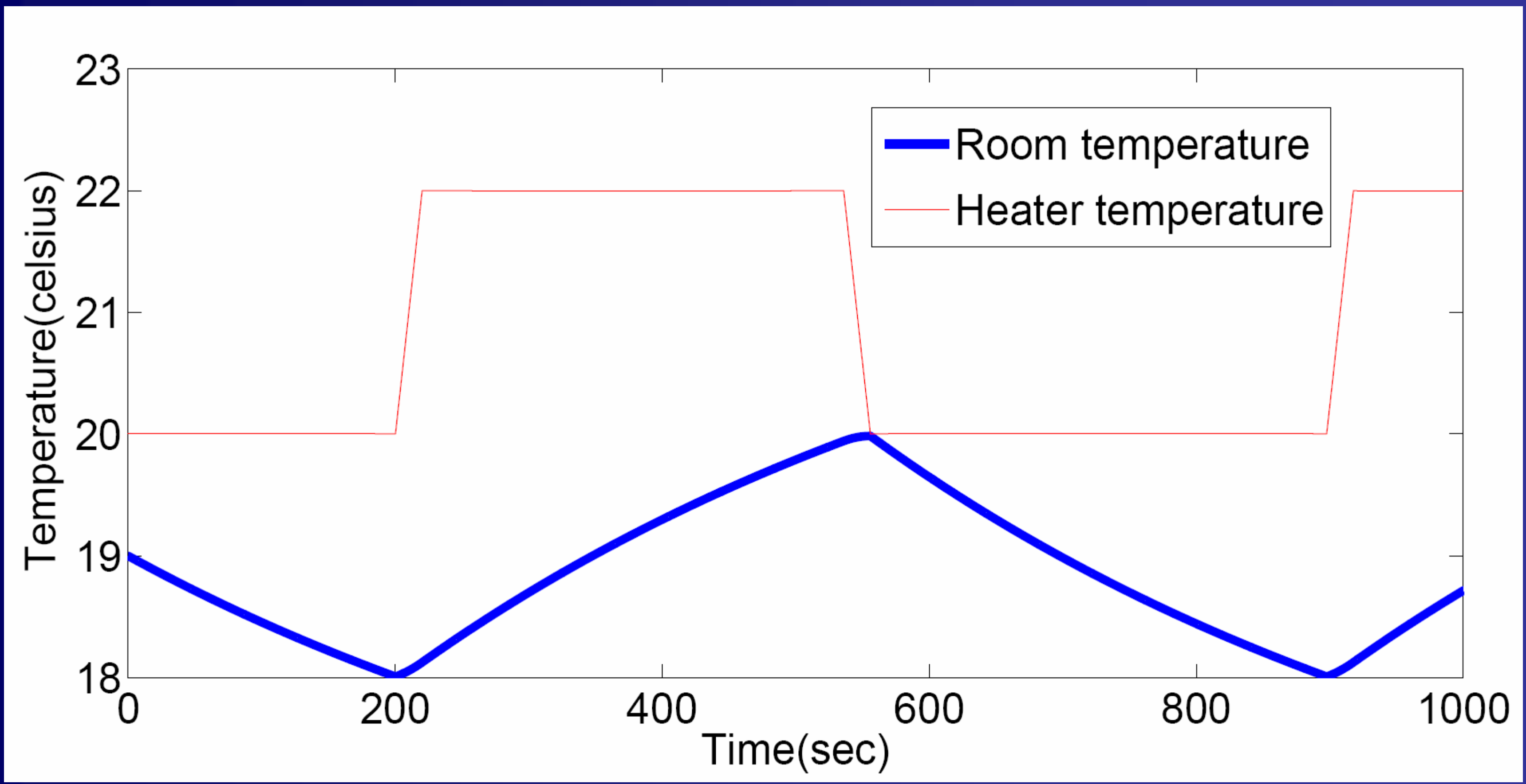
Room temperature stays close to the maximum of safe range: 19.80 to 20 Celsius

The room could be allowed to cool longer without violating safety.

Dwell Time Requirement

- Lower and/or upper bound on time spent in a mode.
- Approximates performance properties: minimize switching, reduce stay in “costly” modes etc.
- Key idea:
  - Restrict entry guards to decrease dwell-time
  - Restrict exit guards to increase dwell-time
  - Recompute fixed point to enforce safety properties

Synthesized Thermostat
300 s min dwell time in off and on mode



Theorems

- **Soundness:**

  Synthesized Design satisfies given safety objectives.

- **Completeness:**

  No Design Synthesized if and only if No Design Feasible with given safety objectives.

Assuming:

(1) Ideal Reachability Oracle for ODEs

- Necessary assumption

(2) Structure Hypothesis

- if ANY switching logic exists, there exists one with hyperbox guards (on a known grid reflecting precision of sensing)
- Assumption of “convenience”

Case Studies

Example	# of Iterations	Runtime (seconds)
Thermostat Controller		
v1	5	21.6
v2 Case A	6	26.2
v2 Case B	6	25.7
TCAS		
Case A	4	55.3
Case B	5	59.1
Automatic Transmission	6	83.6
Train Gate Controller		
Case A	3	22.5
Case B	4	28.3

Synthesis for Optimality [EMSOFT'11]

- **Minimize long-run cost** $\lim_{t \to \infty} \sum_{i=1}^{n} \frac{P_i}{R_i}(t)$

STRUCTURE HYPOTHESIS

Form of guard — Halfspace

+

DEDUCTIVE ENGINE

Synthesizes for special case — Numerical Simulation/ Optimization

+

INDUCTIVE INFERENCE (LEARNING)

Generalizes from special case results — PAC-learning of Halfspaces

Outline

- Switching Logic Synthesis for Hybrid Systems
- Reflections on the Approach
- Synthesizing Environment Assumptions for Reactive Synthesis from LTL
- Conclusions and Future Directions

Structure Hypothesis gives only Conditional Soundness/Completeness

- **Synthesis procedure is sound (complete) assuming the structure hypothesis holds.**
  - Switching logic synthesis: If ANY switching logic exists, there exists one with hyperbox guards (on a known grid)
  - In practice: how do we know if hyperbox guards exist?

- **Can we restate this assumption in terms of the inputs to the synthesis process?**

YES. Sufficient conditions:

- **monotonicity of continuous dynamics in each mode**
- **safe region is hyperbox on the known grid**

Structure Hypothesis gives only Conditional Soundness/Completeness

- Another approach: Use a verifier to check that the synthesized system satisfies its specification
  – Verifier invoked in the loop or at the end
    - E.g. CEGIS loop used with Sketching
  – Requires efficient verification

Interaction between Inductive and Deductive Procedures in Switching Logic Synthesis

Structure Hypothesis defines Concept Class for Learning

INDUCTIVE ENGINE

Hyperbox Learning
(Active: selects examples)

What is the label (+/-) for this example

→

←

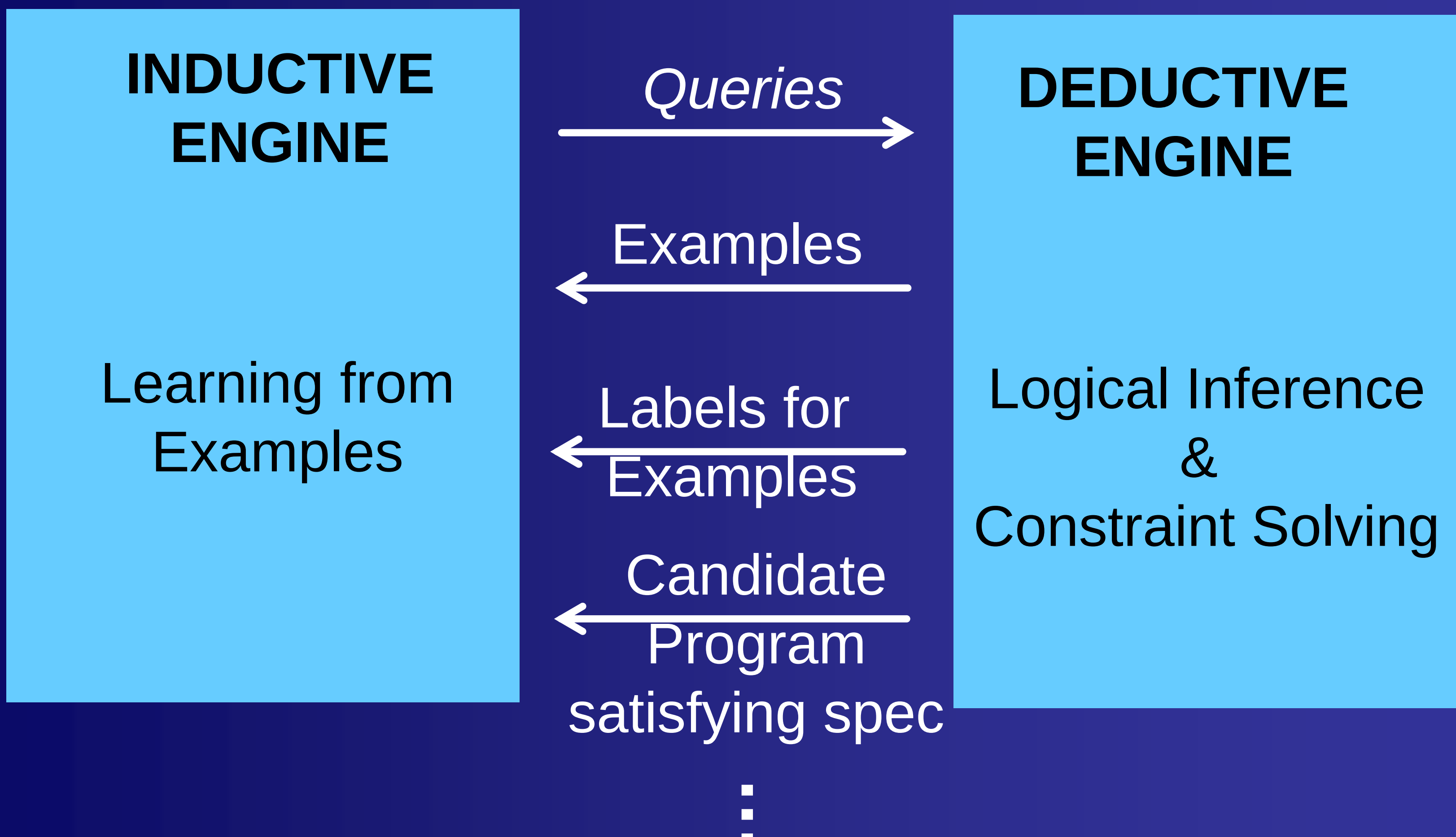
Labels
(safe/unsafe)

DEDUCTIVE ENGINE

ODE Reachability Oracle
(uses numerical simulation; special case of synthesis problem)

Inductive Strategy at the Top-Level

Structure Hypothesis defines Concept (Program) Class



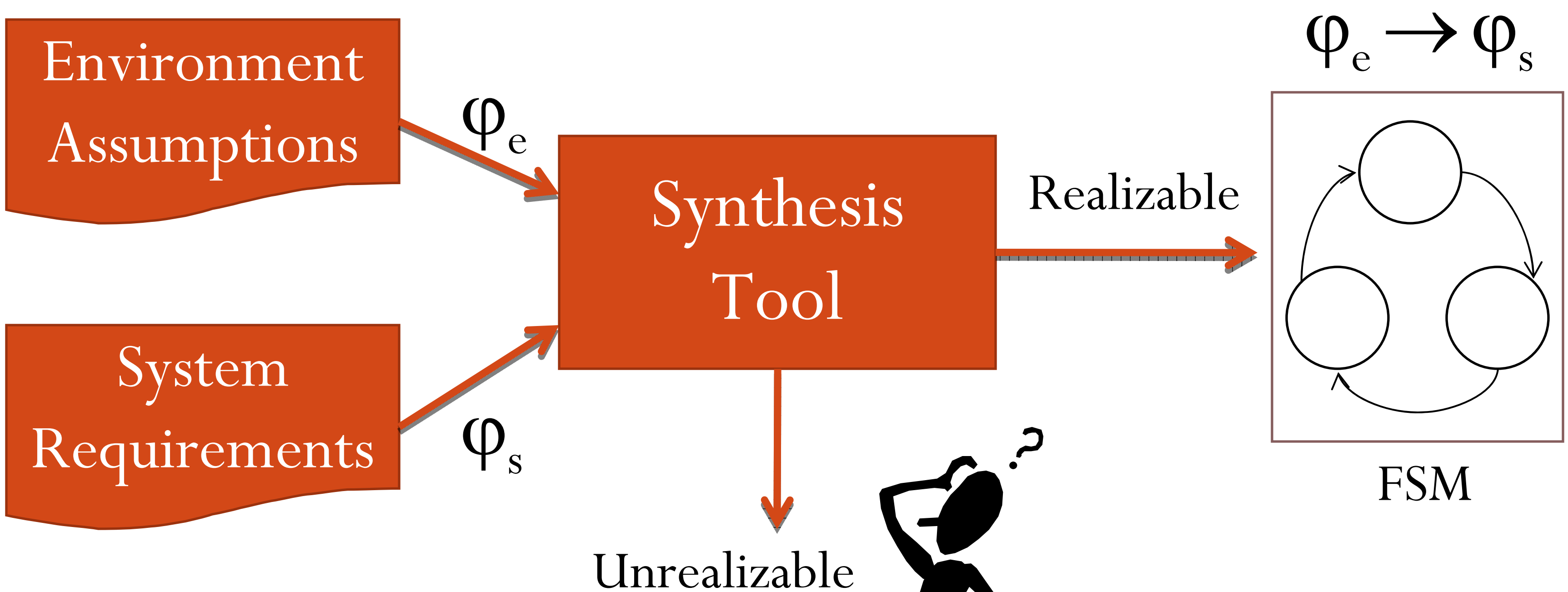
Outline

- Switching Logic Synthesis for Hybrid Systems
- Reflections on the Approach
- Synthesizing Environment Assumptions for Reactive Synthesis from LTL
- Conclusions and Future Directions

A Personal Story of Synthesis from LTL

- Long, long ago…
- … we embarked on a "synthesis from LTL" project.
- Verified Electronic Voting Machine
  - Verilog design, list of LTL properties [Sturton et al.,CCS 2009]
  - http://uclid.eecs.berkeley.edu/vvm/
- Attempt 1: Synthesizer *ran forever* (~a week), no output
- Idea: synthesize individual modules (selections within contests, navigation between contests, etc.)
- Attempt 2,3,…: Unrealizable! Too many *environment assumptions* needed (at interfaces between modules)

Problem



Often due to incomplete environment assumption!

Satisfiability and Realizability

- A LTL formula φ is satisfiable if there exists an infinite word (i.e. sequence of inputs and outputs) that satisfies φ.

- A LTL specification φ is realizable if for all inputs, there exists a finite-state transducer M (e.g. a Moore machine) which generates computations that satisfies φ.

Problem Description

- <u>Goal:</u>

  Generate additional assumptions to enable synthesis

- <u>Context:</u>

  Original specification is *satisfiable* but *unrealizable*

- <u>Assume:</u>

  - Given only a few interesting user scenarios (satisfying traces)

  - Specifications are in the GR(1) class

- <u>Challenge:</u>

  - Space of possible additional assumptions is huge

  - Want assumptions that can be understood and analyzed by a human user

Our Contribution [MEMOCODE 2011]

Counterstrategy-guided synthesis of environment assumptions

- Demonstrated to generate useful/intuitive environment assumptions for digital circuits and robotic controllers

Sciduction for Synthesizing Environment Assumptions

Structure Hypothesis:
Environment Assumptions are Restricted GR(1) properties

+

Inductive Inference:
Version Space Learning

+

Deductive Engine:
(Finite-state) Model Checking

GR(1) Synthesis [Piterman, Pnueli, Saar]

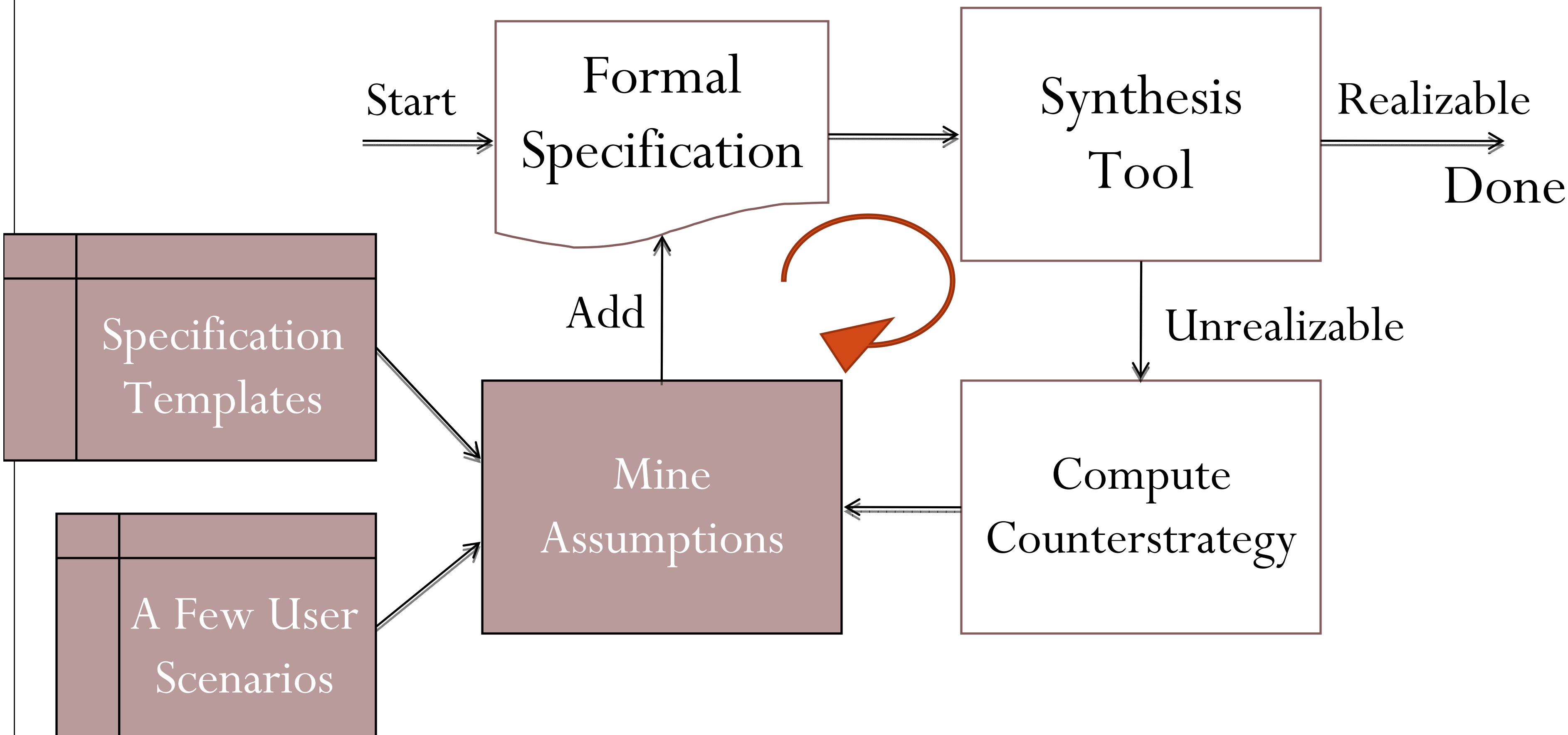
- Formulas in the form: $\varphi_e \rightarrow \varphi_s$
  - Input and output partitions ***I*** and ***O*****.**
  - φ_α^i: initial state formulas. $\alpha \in \{e, s\}$
  - φ_α^t: transition formulas, in the form of **G** *B*, where *B* is a Boolean combination of variables in $\boldsymbol{I} \cup \boldsymbol{O}$ and expressions **X** *u*, $u \in \boldsymbol{I}$ if $\alpha = e$ and $u \in \boldsymbol{I} \cup \boldsymbol{O}$ if $\alpha = s$.
  - φ_α^f: fairness formulas, in the form of **G F** *B*, where *B* is a Boolean formula over $\boldsymbol{I} \cup \boldsymbol{O}$.
- Synthesis as a turn-based <u>two-player game between the system and the environment</u>
  - Realizable if the system has a winning strategy, otherwise env wins; Strategy representable as finite-state transducer

Counter-strategy and counter-trace

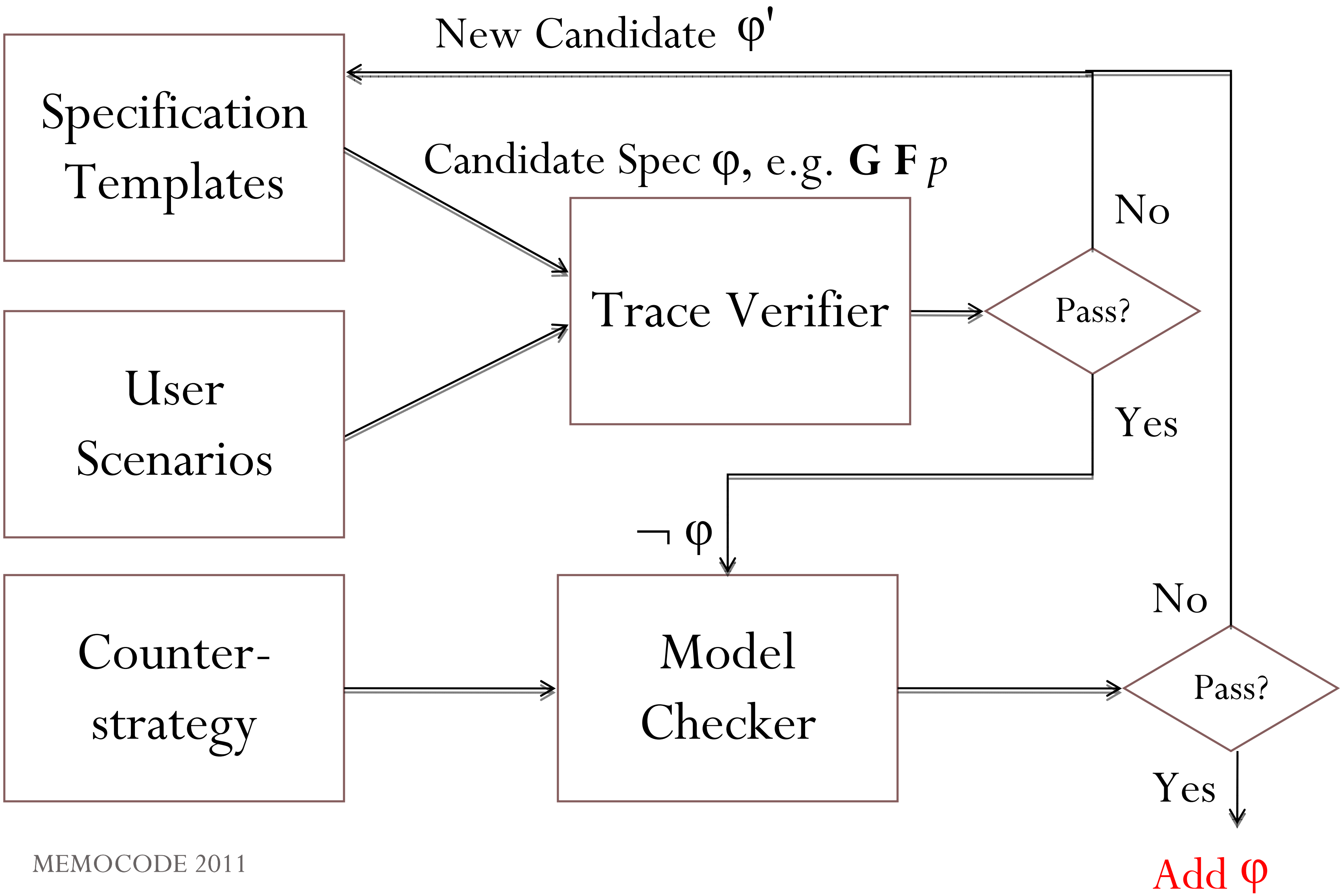
- Counter-strategy is a strategy for the environment to force violation of the specification.

- Counter-trace is a fixed input sequence such that the specification is violated regardless of the outputs generated by the system.

Assumption Mining to Assist Synthesis



Mining Algorithm

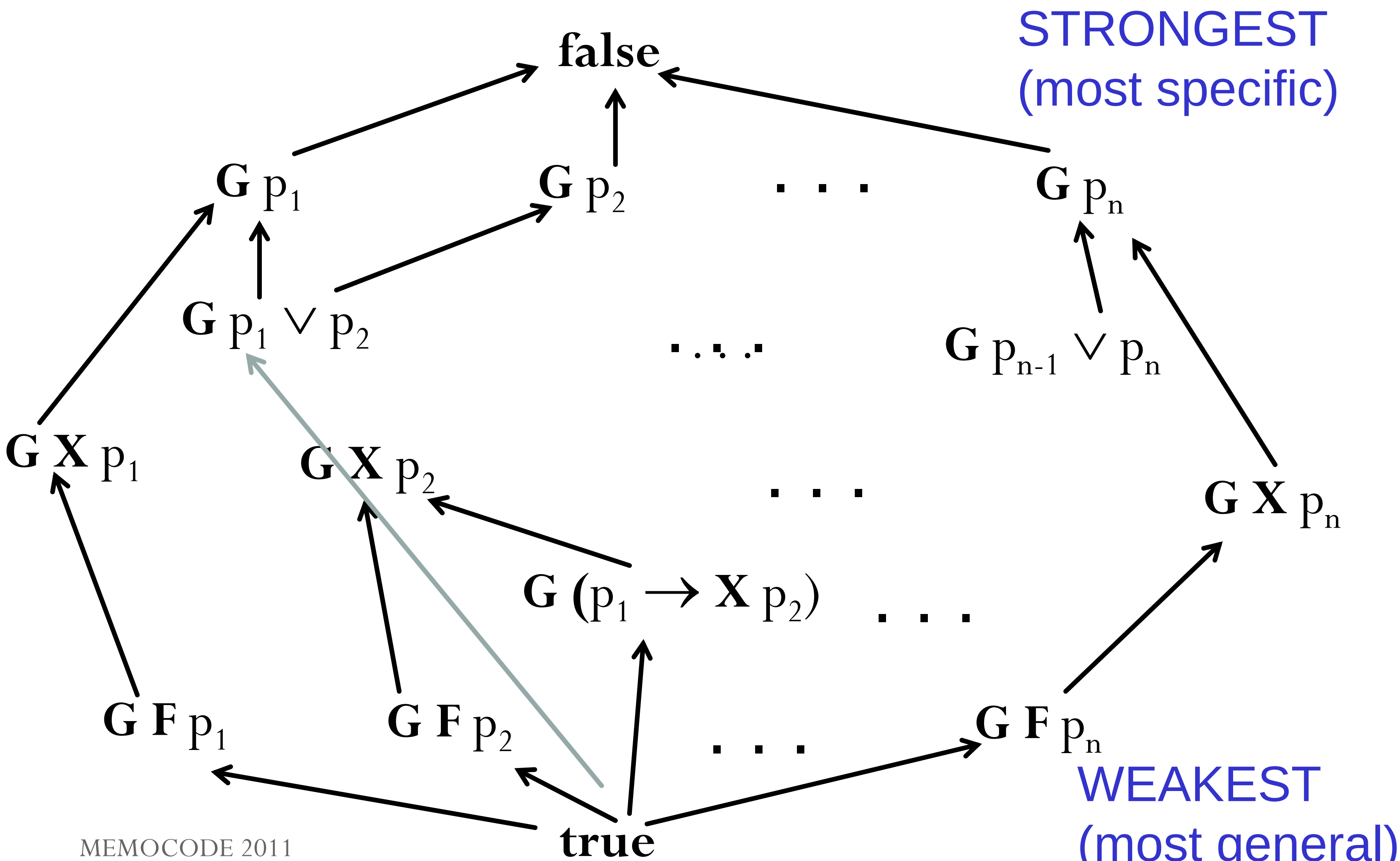


Assumption Templates follow GR(1)

- γ^1: $\mathbf{G}\ \mathbf{F}\ ?b$, where $b \in \boldsymbol{I}$
- γ^2: $\mathbf{G}\ (?b_1 \rightarrow \mathbf{X}\ ?b_2)$, where $b_1 \in \boldsymbol{I} \cup \boldsymbol{O}$ and $b_2 \in \boldsymbol{I}$
- γ^3: $\mathbf{G}\ (?b_1 \vee ?b_2)$, where $b_1, b_2 \in \boldsymbol{I}$

- The specification remains in GR(1) after the addition of new assumptions.
- How to pick candidate assumptions: (next slide)
  - Weakest to Strongest, with heuristics
  - **G F** b weaker than **G X** b weaker than **G** b
  - IMPORTANT: Check each assumption for consistency with existing set

Version Space Learning



Theoretical Results

- Theorem 1: A redundant environment assumption (implied by the existing assumptions) is never added.
  - Proof sketch: guaranteed by the counter-strategy guided approach.
- Corollary: The set of environment assumptions is minimal (but not minimum, in terms of number of properties).
  - Example: We may find G p and G q rather than G $(p \wedge q)$

- Theorem 2: [Completeness] If there exist environment assumptions under our structure hypothesis that make the spec realizable, then the procedure finds them (terminates successfully).
  - "conditional completeness" guarantee

- Theorem 3: [Soundness] The procedure never adds inconsistent environment assumptions.

Example

- Inputs: request r and cancel c
- Outputs: grant g
- System specification φ_s:
  - $\mathbf{G}\ (r \rightarrow \mathbf{X}\ \mathbf{F}\ g)$
  - $\mathbf{G}(c \vee g \rightarrow \mathbf{X}\ \neg g)$
- Environment assumption φ_e:
  - True
- No user scenarios.
- Not realizable because the environment can force c to be high all the time

Example

- System φ_s:
  - $\mathbf{G}\ (r \rightarrow \mathbf{X}\ \mathbf{F}\ g)$
  - $\mathbf{G}(c \lor g \rightarrow \mathbf{X} \neg g)$
- A counter-trace:

  r: 1 1 (1)

  c: 1 1 (1)
- Test assumption candidates by checking its negation:
  - $\mathbf{G}\ (\mathbf{F}\ c)$ ------------ ×
  - $\mathbf{G}\ (\mathbf{F} \neg c)$ ------------ √

- System φ_s:
  - $\mathbf{G}\ (r \rightarrow \mathbf{X}\ \mathbf{F}\ g)$
  - $\mathbf{G}(c \lor g \rightarrow \mathbf{X} \neg g)$
- Environment φ_e:
  - $\mathbf{G}\ (\mathbf{F} \neg c)$

Realizable!

Result Summary

- Experiment Setup:
  - Remove assumptions from an originally realizable specification
  - Use a few (often a single) satisfying traces of the original specification as representative user scenarios
  - Mine additional assumptions until the specification is realizable

- Use Cadence SMV to generate the satisfying traces and model check the counter-strategies
- Use RATSY [Bloem et al. 2010] to check realizability of the specifications and compute the counter-strategies and counter-traces in case of unrealizability

Result Summary

- Result Summary:
  - Case studies in existing literature: AMBA AHB, IBM Gen Buffer, robotic vehicle controller, etc.
  - Recovered the missing assumption in most cases
  - AMBA AHB Example:

Original assumption:

G (HLOCK[0] = 1 → HBUSREQ[0] = 1)

Master 0 requests locked access to the bus (HLOCK[0] = 1)

Master 0 requests access to the bus (HBUSREQ[0] = 1)

Mined assumption:

G (**F** HLOCK[0] = 0)

Related Work

- Constructing the "weakest" assumption needed for realizability from the game graph [Chatterjee et al. 2008]
  - Computes a safety assumption that removes a minimal set of environment edges form the game graph
  - Computes a liveness assumption that puts fairness on the remaining environment edges
  - The additional environment assumption is a single Büchi automaton which can be difficult for a normal human user to analyze or even understand.
- Computing counter-strategy for GR(1) synthesis [Konighofer et al. 2009]
  - Counter-strategies and counter-traces as explanations for unrealizability

Discussion

- Specification mining can generate useful environment assumptions to cope with unrealizability.
- Limitation: choice of templates
  - Can extend same approach with broader set of templates
- Can the same idea of learning from counter-strategies and counter-traces be applied to other synthesis tasks?
  - i.e., not just synthesis from LTL

Outline

- Switching Logic Synthesis for Hybrid Systems
- Reflections on the Approach
- Synthesizing Environment Assumptions for Reactive Synthesis from LTL
- Conclusions and Future Directions

Induction + Deduction + Structure

- Structure Hypothesis encodes human insight about form of artifact
- Synthesis procedure combines inductive inference with deductive reasoning
- Several demonstrations
  - Synthesis of loop-free programs [Jha et al, '10]
  - Switching logic synthesis for safety and optimality [Jha et al '10,'11]
  - Environment assumptions for LTL synthesis [Li et al '11]
  - Fixed-point code from floating-point [Jha, Seshia, '11]

Future Directions

- Exploring Structure Hypotheses
  - Need for flexibility in hypotheses
  - Requires flexible (extensible) learning algorithms that can accommodate changes to hypotheses
- Providing unconditional guarantees
  - Techniques to verify if structure hypothesis holds for a given input
- Investigate strategies to combine induction & deduction
  - Interplay between Teaching & Learning