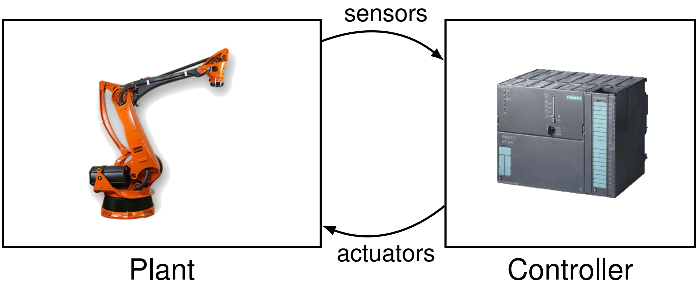
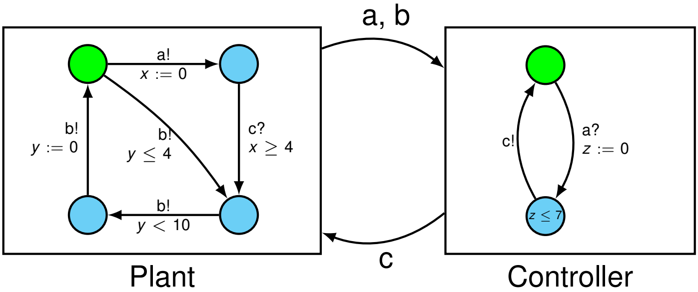
Counterexample-Guided Synthesis of Observation Predicates

Rayna Dimitrova and Bernd Finkbeiner
Universität des Saarlandes

Control of Real-Time Systems

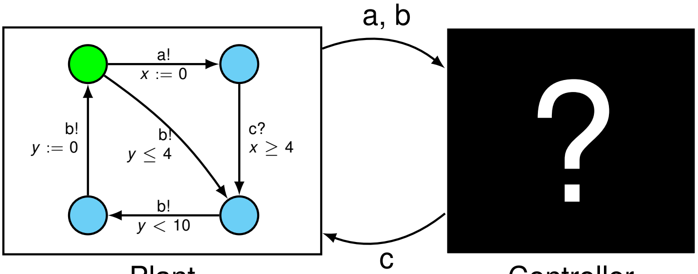


Timed Controller Verification



Model checking timed automata is PSPACE-complete
[Alur & Dill, 1990]

Timed Controller Synthesis

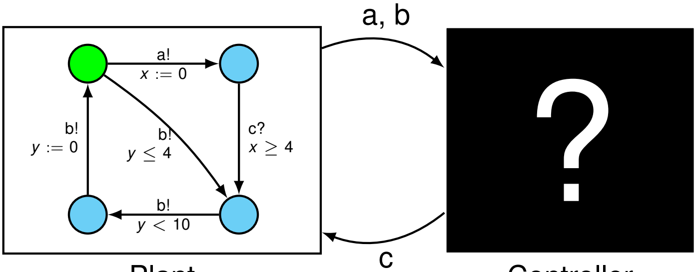


Plant

Controller
(**complete observability**)

Solving a game of perfect information
[Maler et al., 1995]

Timed Controller Synthesis

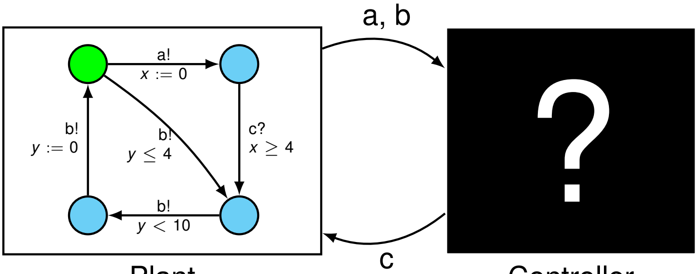


Plant

Controller
(**complete observability**)

Safety synthesis is EXPTIME-complete
[Henzinger & Kopke, 1999]

Timed Controller Synthesis

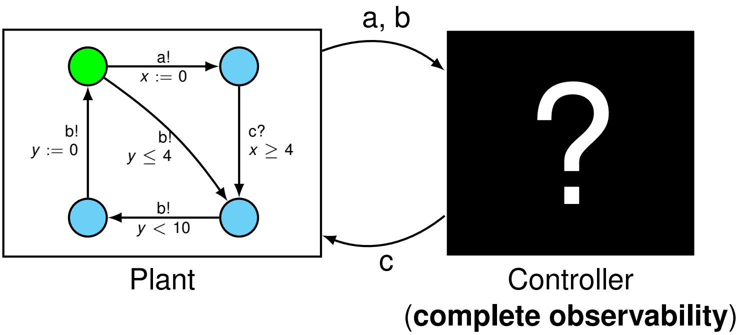


Plant

Controller
(**complete observability**)

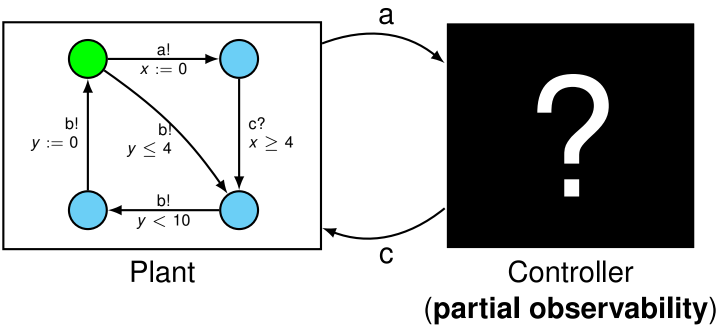
Effective game solving algorithm
[Cassez et al., 2005]

Timed Controller Synthesis



The controller needs to **perfectly observe** the behavior of the plant

Timed Controller Synthesis with Partial Observation



More realistic assumption:

The controller can only **partially observe** the behavior of the plant

Timed Controller Synthesis with Partial Observation

- The problem is in general undecidable
- Decidable when a granularity is fixed

  a maximal constant for each clock and a constant $m \in \mathbb{N}$ such that each constant used is an integral multiple of $\frac{1}{m}$

  - 2EXPTIME-complete

[Bouyer et al, 2003]

- Decidable for fixed finite set of observation predicates

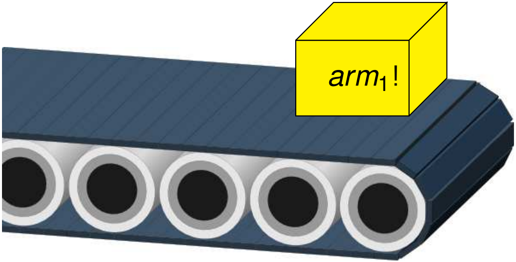
  $\{o_1, o_2, \ldots, o_n\}$, $o_i = (L_i, \varphi_i)$: set of locations and clock constraint

[Cassez, 2007]

The Role of Observations in Timed Control

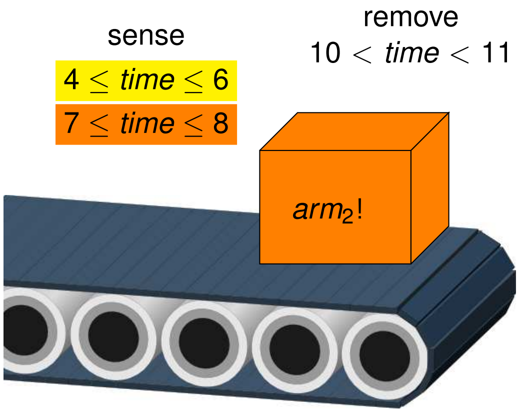
remove

$10 < \mathit{time} < 11$



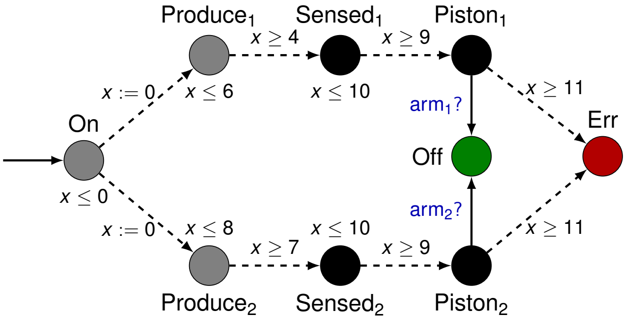
- The controller has to remove the box strictly between 10 and 11 time units after the start of the production phase

The Role of Observations in Timed Control



- The controller has to remove the box strictly between 10 and 11 time units after the start of the production phase
- To choose the right action, the controller needs to distinguish the type of the box, based on the time the box was sensed

The Role of Observations in Timed Control

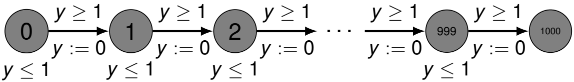


- Clock x unobservable, own (observable) clock y
- Observing $y \leq 6$ (and hence $y > 6$) and $y = \frac{21}{2}$ suffices
- Observation predicates: constraints over the observable clocks
  - decision predicate $y \leq 6$ (decide which action should be taken)
  - action point $y = \frac{21}{2}$ (when a controllable action can be taken)

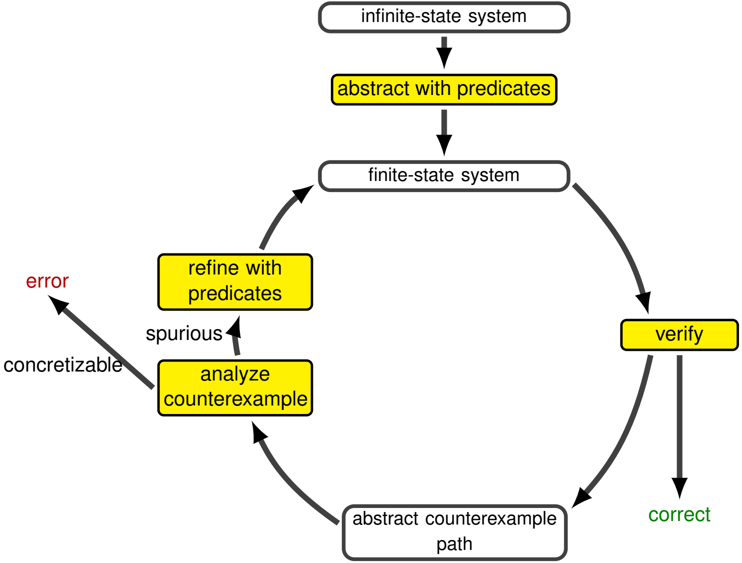
How to Find Observation Predicates?

- Manually fix a finite set of observable predicates
- Refine granularity by brute-force enumeration $1, \frac{1}{2}, \frac{1}{4}, \ldots$?

  Not a good idea:

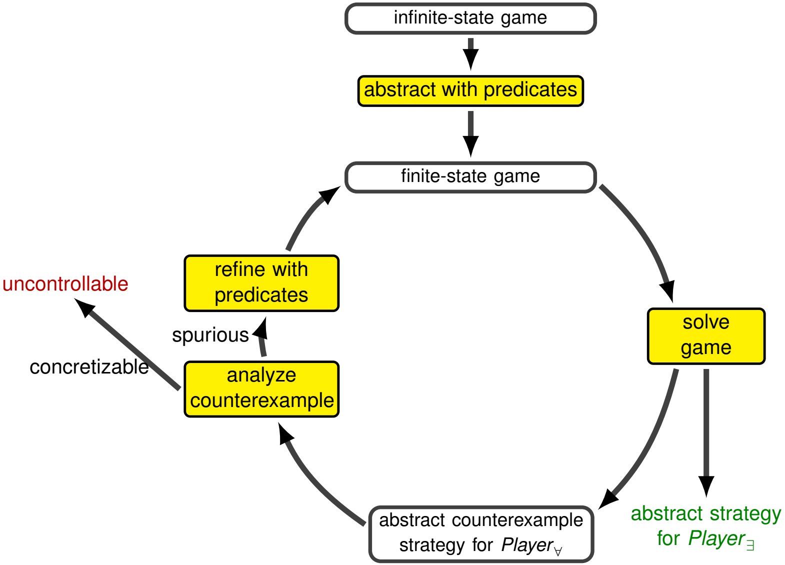


CEGAR



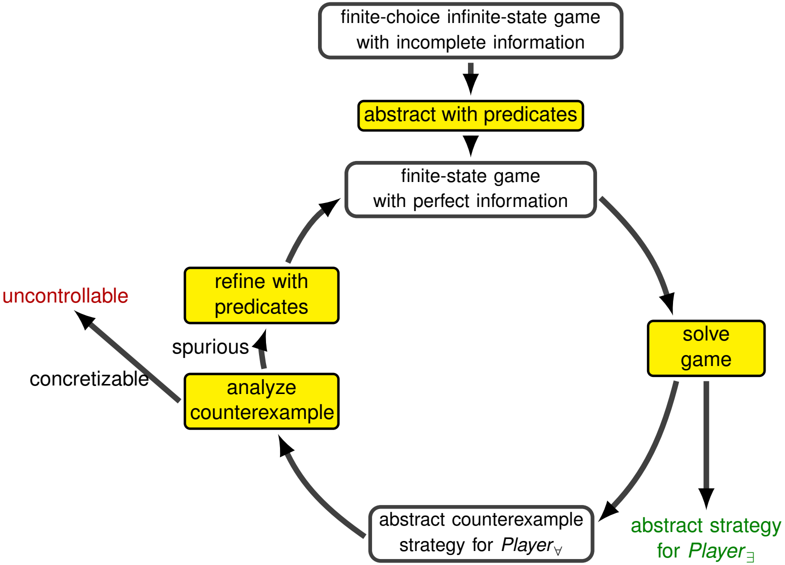
[Clarke et al., 2000]

CEGAR for Games



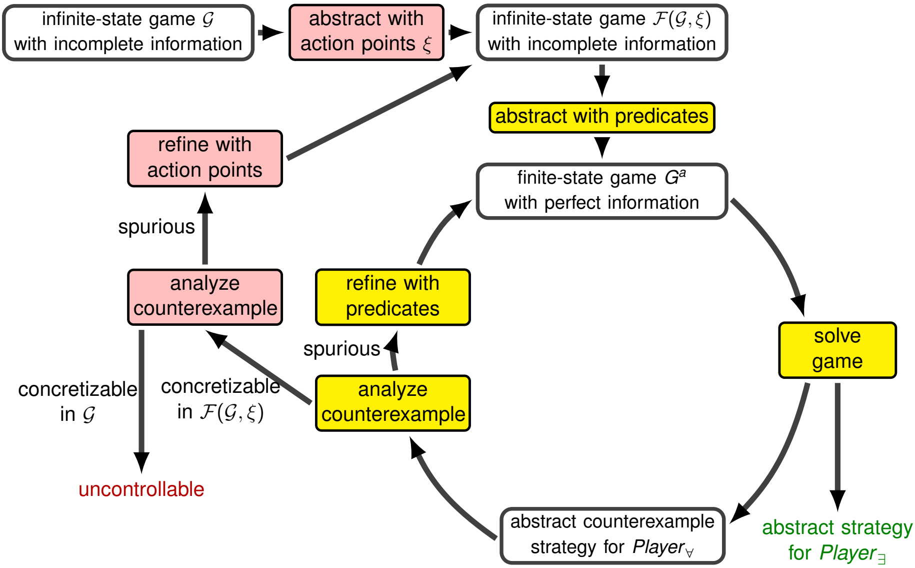
[Henzinger et al., 2003]

CEGAR for Games with Incomplete Information

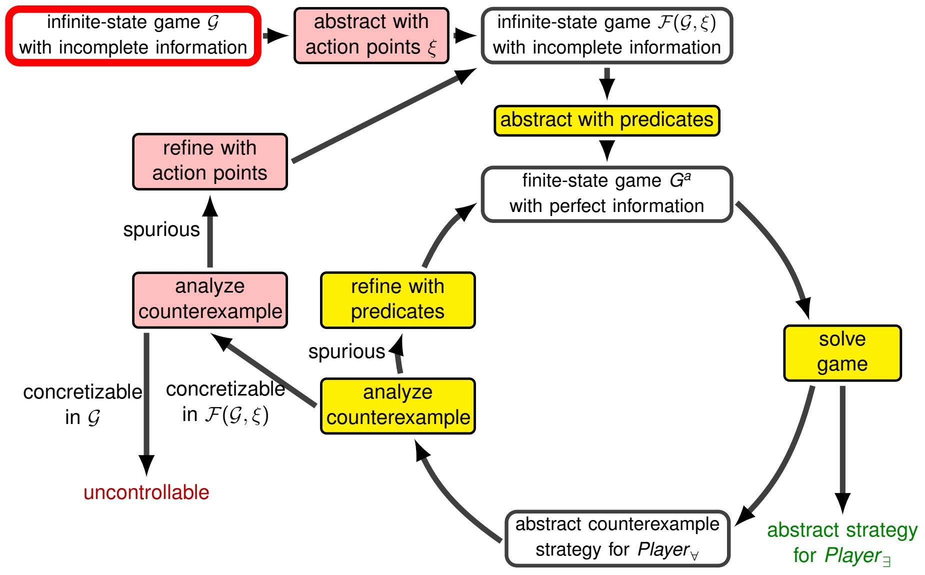


[DF, 2008]

Counterexample-Guided Synthesis of Observations

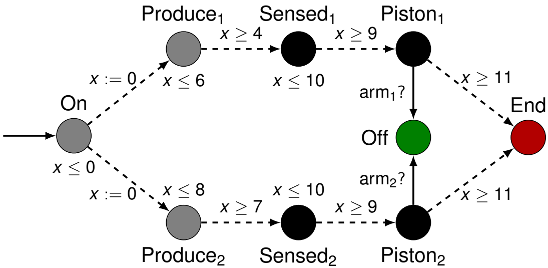


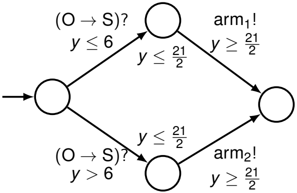
Overview



infinite-state game $\mathcal{G}$ with incomplete information
abstract with action points ξ
infinite-state game $\mathcal{F}(\mathcal{G}, \xi)$ with incomplete information
abstract with predicates
finite-state game G^a with perfect information
solve game
abstract strategy for $Player_\exists$
abstract counterexample strategy for $Player_\forall$
analyze counterexample
spurious
refine with predicates
concretizable in $\mathcal{F}(\mathcal{G}, \xi)$
analyze counterexample
spurious
refine with action points
concretizable in $\mathcal{G}$
uncontrollable

The Synthesis Problem





Plant $\mathcal{P}$

- unobservable clocks X_u
  - $X_u = \{x\}$
- observable clocks X_o
  - $X_o = \emptyset$
- equivalence on locations
  F $= \{\text{Off}\}$, E $= \{\text{Err}\}$,
  O $= \{\text{On}, \text{Produce}_1, \text{Produce}_2\}$,
  S $= \{\text{Sensed}_1, \text{Sensed}_2, \text{Piston}_1, \text{Piston}_2\}$

Controller $\mathcal{C}$

- own set of clocks X_c
  - $X_c = \{y\}$
- controllable actions Σ_c (uncontrollable Σ_u)
  - $\Sigma_c = \{\text{arm}_1, \text{arm}_2\}$
- observes transitions
  - $(O \to S), \ldots$

The Synthesis Game

- Two-player infinite-state games with incomplete information
- *Player*$_\exists$ represents the controller, *Player*$_\forall$ the environment (plant)
- Safety objective encodes safety timed controller synthesis

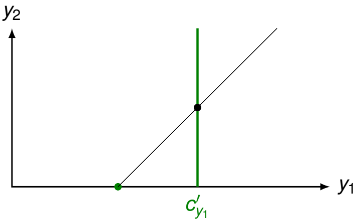
Symbolic Game Representation

Variables updated by $\mathit{Player}_\exists$:

- finite-range: t, $V_\exists^{<\infty} = \{\mathit{act}, \mathit{wait}, \mathit{reset}\}$
- symbolic constants: $V_\exists^{\infty} = \{c_x \mid x \in X_{\mathrm{o+c}}\}$ $(X_{\mathrm{o+c}} = X_{\mathrm{o}} \dot\cup X_{\mathrm{c}})$

Transition relation for $\mathit{Player}_\exists$:

- choose to remain idle and let time elapse
- choose to reset a set of controllable clocks
- choose to execute an action in Σ_{c} immediately
- choose to execute an action in Σ_{c} after a delay



$$\frac{t}{\begin{array}{c} \neg t' \quad \mathit{act}' \in \Sigma_{\mathrm{c}} \quad \mathit{wait}' \quad \mathit{reset}' = \emptyset \\ \exists x \in X_{\mathrm{o+c}}.\ c'_x > 0 \\ \forall x \in X_{\mathrm{o+c}}.\ c'_x > 0 \rightarrow c'_x > x \end{array}}$$

Symbolic Game Representation

Variables updated by $\mathit{Player}_\forall$:

- observable by $\mathit{Player}_\exists$: t, $V_\forall^o = X_\mathsf{o} \dot{\cup} X_\mathsf{c} \dot{\cup} \{\mathit{oloc}, \mathit{reset\_enabled}\}$
- unobservable by $\mathit{Player}_\exists$: $V_\forall^u = X_\mathsf{u} \dot{\cup} \{\mathit{loc}, \mathit{delay\_enabled}\}$

Transition relation for $\mathit{Player}_\forall$:

- execute an enabled uncontrollable action
- execute the controllable action chosen by $\mathit{Player}_\exists$

$$\frac{\neg t \quad (\neg \mathit{wait} \vee \neg \mathit{delay\_enabled}) \quad \mathit{act} = \sigma \quad (\mathit{loc}, v) \xrightarrow{\sigma} (\mathit{loc}', v')}{t' \quad \mathit{oloc}' = [\mathit{loc}'] \quad \mathit{reset\_enabled}' \quad \mathit{delay\_enabled}'}$$

- let time elapse by making a delay transition
- reset the controllable clocks chosen by $\mathit{Player}_\exists$
- give the turn to $\mathit{Player}_\exists$ to make a move
- do a *skip* transition leaving all variables unchanged

Symbolic Game Representation

Transition relation for $\mathit{Player}_\forall$:

- execute the controllable action chosen by $\mathit{Player}_\exists$
- execute an enabled uncontrollable action
- let time elapse by taking a delay transition

$$\frac{\neg t \quad \mathit{wait} \quad \mathit{delay\_enabled} \quad \sigma \in \mathbb{R}_{>0} \quad (\mathit{loc}, v) \xrightarrow{\sigma} (\mathit{loc}', v')}{\begin{array}{c}\neg t' \quad \mathit{oloc}' = \mathit{oloc} \quad \forall x \in X_\mathrm{c}.\ x' = x + \sigma \\ (\mathit{act} \neq \bot \rightarrow \forall x \in X_{\mathrm{o+c}}.\ x < c_x \rightarrow x' < c_x)\end{array}}$$

$$\frac{\neg t \quad \mathit{wait} \quad \mathit{delay\_enabled} \quad \sigma \in \mathbb{R}_{>0} \quad (\mathit{loc}, v) \xrightarrow{\sigma} (\mathit{loc}', v')}{\begin{array}{c}\neg t' \quad \mathit{oloc}' = \mathit{oloc} \quad \forall x \in X_\mathrm{c}.\ x' = x + \sigma \\ (\mathit{act} \neq \bot \rightarrow \forall x \in X_{\mathrm{o+c}}.\ x < c_x \rightarrow x' \leq c_x) \\ \neg \mathit{delay\_enabled}' \quad (\mathit{act} \neq \bot \rightarrow \exists x \in X_{\mathrm{o+c}}.\ c_x > 0 \wedge x' \geq c_x \wedge x < c_x)\end{array}}$$

- reset the controllable clocks chosen by $\mathit{Player}_\exists$
- give the turn to $\mathit{Player}_\exists$ to make a move
- do a *skip* transition leaving all variables unchanged

Strategies

- Strategy:
  maps a prefix of a play (sequence of variable valuations) to a successor valuation of the variables updated by the player, which must result in a successor state for the last state
- Winning strategy:
  achieves the objective (e.g., avoid or reach a given set of (error) states) regardless of the opponent's behavior
- A strategy for $Player_\exists$ must be consistent:
  reacts in the same way for prefixes that agree on the valuations of the variables that $Player_\exists$ can observe

The definition of the game guarantees that $Player_\exists$ does not gain information from observing individual moves of $Player_\exists$ other than those that correspond to observable transitions in the plant $\mathcal{P}$.

Control Strategies

winning strategy for $Player_\exists$

$\Leftrightarrow$

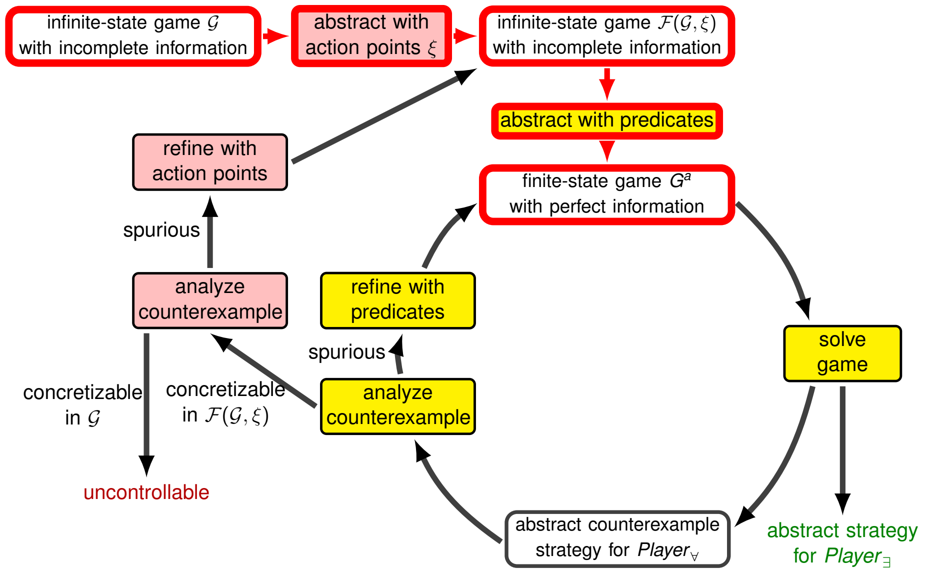
control strategy to avoid error location

Strategy for $Player_\exists$ with

- finite set of memory states
- finite set of outputs
- rectangular or diagonal observations

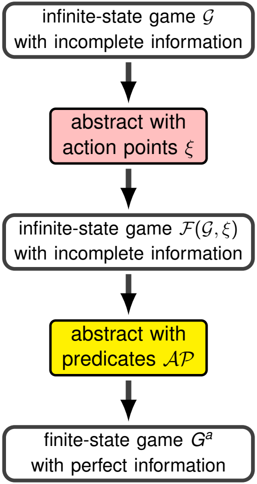
can be transformed into a controller automaton

Overview



infinite-state game $\mathcal{G}$
with incomplete information
abstract with
action points ξ
infinite-state game $\mathcal{F}(\mathcal{G}, \xi)$
with incomplete information
abstract with predicates
finite-state game G^a
with perfect information
solve
game
abstract strategy
for $Player_\exists$
abstract counterexample
strategy for $Player_\forall$
analyze
counterexample
spurious
refine with
predicates
concretizable
in $\mathcal{F}(\mathcal{G}, \xi)$
analyze
counterexample
spurious
refine with
action points
concretizable
in $\mathcal{G}$
uncontrollable

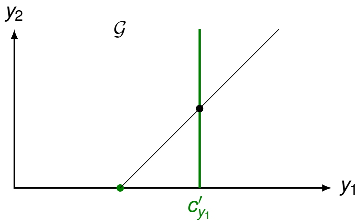
Abstraction Steps

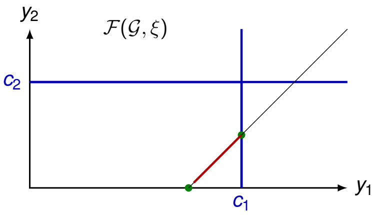


1. fix a finite set of action points ξ
   - restrict the timing precision of the controllable actions
   - the resulting nondeterminism is resolved by *Player*$_\forall$
2. abstract the game w.r.t. a finite set of abstraction predicates $\mathcal{AP}$
   - fix/restrict the (observation) power of *Player*$_\exists$
   - overapproximate the possible behaviors of *Player*$_\forall$

Fixing the Action Points

- $\xi(y) \subset \mathbb{Q}_{>0}$ – finite set of action points for each clock $y \in X_{\text{o+c}}$
- In $\mathcal{G}$, $Player_{\exists}$ can choose when a controllable action will be executed
- In $\mathcal{F}(\mathcal{G}, \xi)$, $Player_{\exists}$ can choose to execute the action immediately or within a certain interval determined by the current clock values and ξ





- Construct $\mathcal{F}(\mathcal{G}, \xi)$ from $\mathcal{G}$ by
  - removing the variables $V_{\exists}^{\infty} = \{c_x \mid x \in X_{\text{o+c}}\}$
  - adjusting the transition relations accordingly (using ξ)

Fixing the Action Points

Game with fixed action points $\mathcal{F}(\mathcal{G}, \xi)$

Variables updated by $Player_\exists$: t, $V_\exists^{<\infty} = \{act, wait, reset\}$

- $Player_\exists$ chooses to execute an action in Σ_c after a delay

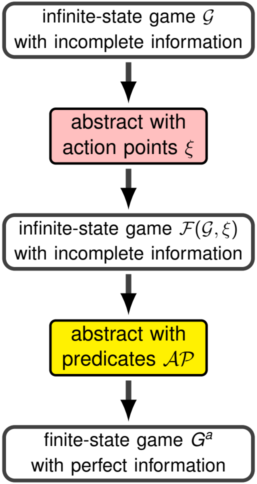
$$\frac{t}{\neg t' \quad act' \in \Sigma_c \quad wait' \quad reset' = \emptyset}$$

- $Player_\forall$ lets time elapse by making a delay transition

$$\frac{\neg t \quad wait \quad delay\_enabled \quad \sigma \in \mathbb{R}_{>0} \quad (loc, v) \xrightarrow{\sigma} (loc', v')}{\begin{array}{c} \neg t' \quad oloc' = oloc \quad \forall x \in X_c.\ x' = x + \sigma \\ (\forall x \in X_{o+c}.\ \forall c \in \xi(x).\ x < c \rightarrow x' < c) \\ delay\_enabled' \in \{\text{true}, \text{false}\} \end{array}}$$

$$\frac{\neg t \quad wait \quad delay\_enabled \quad \sigma \in \mathbb{R}_{>0} \quad (loc, v) \xrightarrow{\sigma} (loc', v')}{\begin{array}{c} \neg t' \quad oloc' = oloc \quad \forall x \in X_c.\ x' = x + \sigma \\ (\forall x \in X_{o+c}.\ \forall c \in \xi(x).\ x < c \rightarrow x' \leq c) \\ \neg delay\_enabled' \quad (\exists x \in X_{o+c}.\ \exists c \in \xi(x).\ x' \geq c \wedge x < c) \end{array}}$$

Abstraction Steps



1. fix a finite set of action points ξ
   - restrict the timing precision of the controllable actions
   - the resulting nondeterminism is resolved by *Player*$_\forall$
2. abstract the game w.r.t. a finite set of abstraction predicates $\mathcal{AP}$
   - fix/restrict the (observation) power of *Player*$_\exists$
   - overapproximate the possible behaviors of *Player*$_\forall$

Predicate Abstraction

Abstraction predicates

- $\mathcal{AP}$ – finite set of abstraction predicates over all variables
- valuation a – boolean vector with one element for each predicate in $\mathcal{AP}$

Goal: overapproximate the observation equivalence relation

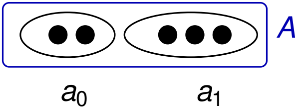
Abstract states are **sets of valuations**

which have the same values for the observable predicates in $\mathcal{AP}$

Example:

$$\mathcal{AP} = \{x = 0, y \leq 1\}$$

in a_0, $x = 0$ is *false*
in a_1, $x = 0$ is *true*



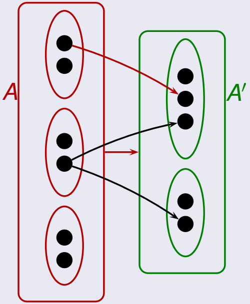
$y \leq 1$ is *true*

Predicate Abstraction

Goal: give more power to $Player_\forall$, restrict the power of $Player_\exists$

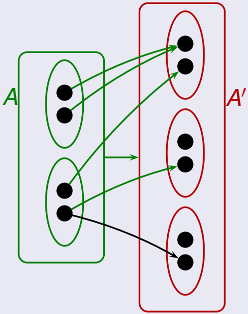
For $Player_\forall$

$(A, A') \in T^a_\forall$ if there exist states s for A and s' for A': $(s, s') \in T^{\mathcal{F}}_\forall$



For $Player_\exists$

$(A, A') \in T^a_\exists$ if for every s for A there exists s' for A': $(s, s') \in T^{\mathcal{F}}_\exists$



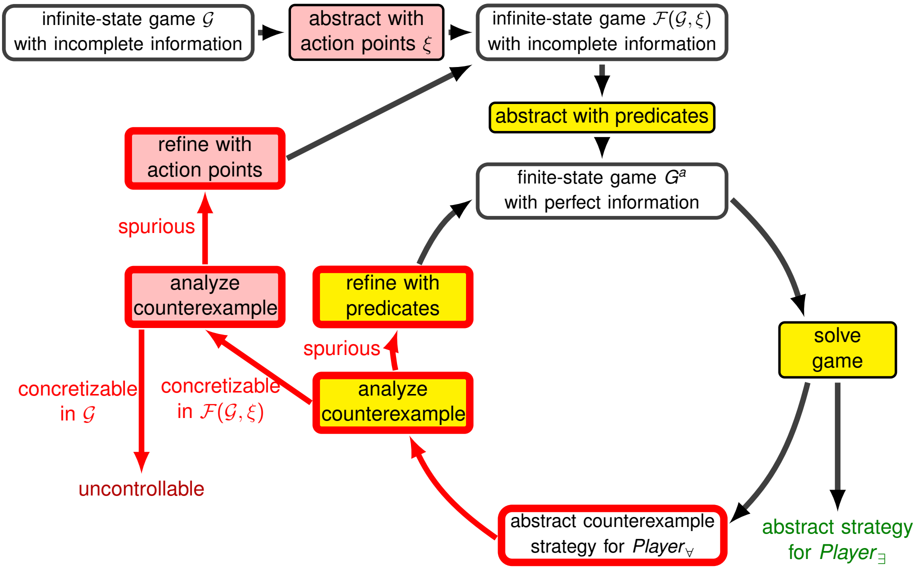
Sound Abstraction

winning strategy for $\mathit{Player}_\exists$ in G^a

$$\Rightarrow$$

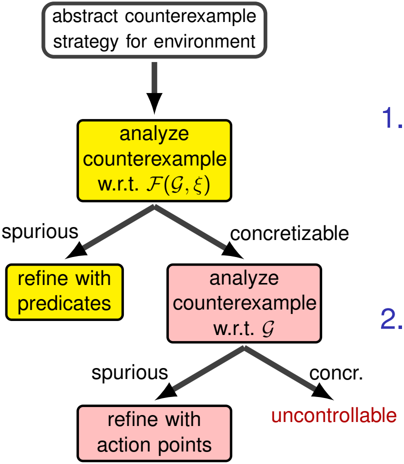
consistent winning strategy for $\mathit{Player}_\exists$ in $\mathcal{G}$

Overview



infinite-state game $\mathcal{G}$
with incomplete information
abstract with
action points ξ
infinite-state game $\mathcal{F}(\mathcal{G}, \xi)$
with incomplete information
abstract with predicates
finite-state game G^a
with perfect information
refine with
action points
spurious
analyze
counterexample
refine with
predicates
spurious
solve
game
concretizable
in $\mathcal{G}$
concretizable
in $\mathcal{F}(\mathcal{G}, \xi)$
analyze
counterexample
uncontrollable
abstract counterexample
strategy for $Player_\forall$
abstract strategy
for $Player_\exists$

Refinement Steps

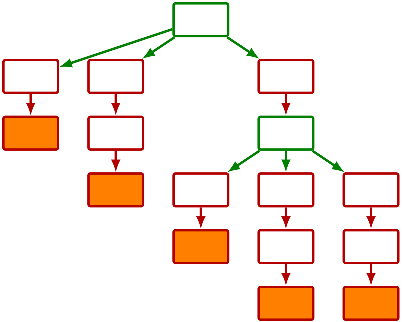


1. New abstraction predicates for $\mathcal{AP}$
   - are computed using interpolation
   - to refine the abstract transition and observation equivalence relations
2. New action points for ξ
   - are extracted from witnesses for satisfiability of the negation of a formula characterizing concretizability in the game $\mathcal{G}$
   - to allow for better timing precision of controllable actions

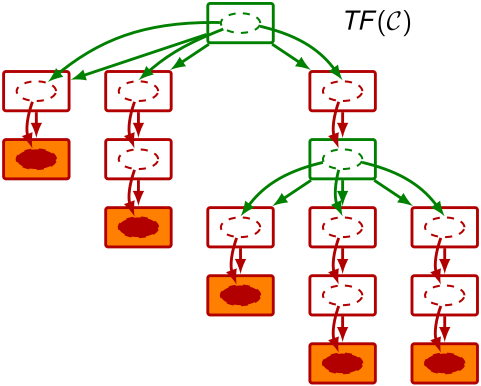
Abstract Counterexample

Abstract counterexample $\mathcal{C}$:

- branches according to all abstract successors of the states belonging to $Player_{\exists}$
- root: initial state
- leaves: error states



Abstract Counterexample Analysis for $\mathcal{F}$



$TF(\mathcal{C})$

$\exists V_{\forall}^{o0}, V_{\exists}^{<\infty 0}$

$\exists V_{\forall}^{o2}$

$\exists V_{\forall}^{o4}$

$\exists V_{\forall}^{o5}$

$\exists V_{\forall}^{u0}, V_{\forall}^{u1}, V_{\forall}^{u2}, \ldots$

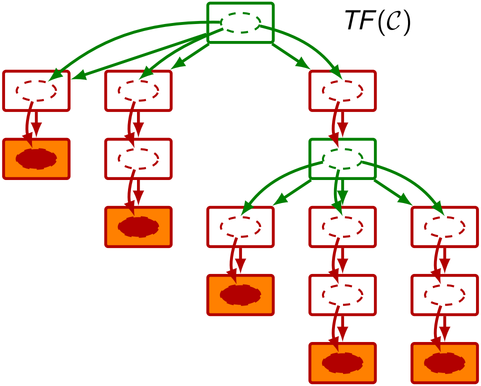
Tree formula $TF(\mathcal{C})$

The abstract counterexample $\mathcal{C}$ is concretizable in $\mathcal{F}$

$\Leftrightarrow$

the tree formula $TF(\mathcal{C})$ is satisfiable.

Abstract Counterexample Analysis for $\mathcal{F}$



$\exists V_\forall^{o\,0}, V_\exists^{<\infty\,0}$

$\forall V_\exists^{<\infty\,1} \longrightarrow$ eliminate via $\wedge$

$\exists V_\forall^{o\,2}$

$\forall V_\exists^{<\infty\,3} \longrightarrow$ eliminate via $\wedge$

$\exists V_\forall^{o\,4}$

$\exists V_\forall^{o\,5}$

$\exists V_\forall^{u\,0}, V_\forall^{u\,1}, V_\forall^{u\,2}, \ldots$

Tree formula $TF(\mathcal{C})$

- $\tau_1, \tau_2, \ldots$: finite sequences of valuations of the variables $V_\exists^{<\infty}$
- since the domain of $V_\exists^{<\infty}$ is finite, finitely many sequences for $\mathcal{C}$
- tree formula $TF(\mathcal{C}) = F(\mathcal{C}, \tau_1) \wedge \ldots \wedge F(\mathcal{C}, \tau_n)$

Refining the Predicate Abstraction

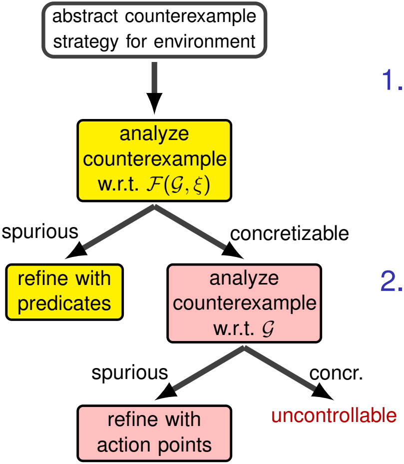
Case I: Abstract transition relation too coarse

- Some $TF(\mathcal{C}, \tau)$ is not satisfiable
- Find interpolant by splitting τ into prefix and suffix in the usual way.

Case II: Abstract observation equivalence too coarse

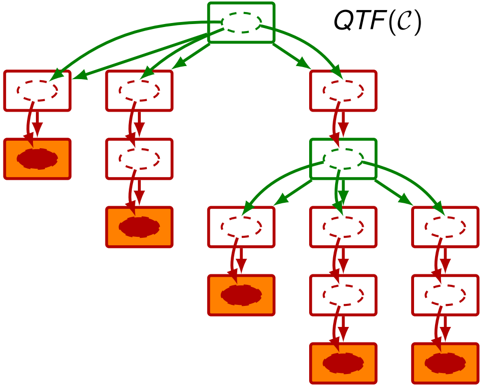
- Each of $TF(\mathcal{C}, \tau_1)$ and $TF(\mathcal{C}, \tau_2)$ is satisfiable, $TF(\mathcal{C}, \tau_1) \wedge TF(\mathcal{C}, \tau_2)$ not
- Compute interpolant I: $TF(\mathcal{C}, \tau_1) \Rightarrow I$ and $I \wedge TF(\mathcal{C}, \tau_2)$ unsatisfiable
- I is over shared variables and, thus, only observable variables

Refinement Steps



1. New abstraction predicates for $\mathcal{AP}$
   - are computed using interpolation
   - to refine the abstract transition and observation equivalence relations
2. New action points for ξ
   - are extracted from witnesses for satisfiability of the negation of a formula characterizing concretizability in the game $\mathcal{G}$
   - to allow for better timing precision of controllable actions

Abstract Counterexample Analysis for $\mathcal{G}$



$QTF(\mathcal{C})$

$\exists V_{\forall}^{o\,0}, V_{\exists}^{<\infty\,0}, V_{\exists}^{\infty\,0}$

$\forall V_{\exists}^{<\infty\,1}, V_{\exists}^{\infty\,1}$

$\exists V_{\forall}^{o\,2}$

$\forall V_{\exists}^{<\infty\,3}, V_{\exists}^{\infty\,3}$

$\exists V_{\forall}^{o\,4}$

$\exists V_{\forall}^{o\,5}$

$\exists V_{\forall}^{u\,0}, V_{\forall}^{u\,1}, V_{\forall}^{u\,2}, \ldots$

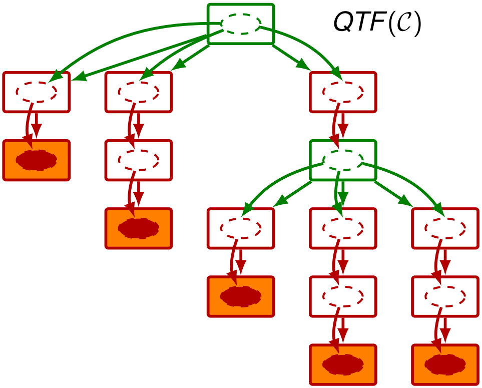
Quantified tree formula $QTF(\mathcal{C})$

The abstract counterexample $\mathcal{C}$ is concretizable in $\mathcal{G}$

$\Leftrightarrow$

the quantified tree formula $QTF(\mathcal{C})$ is satisfiable.

Abstract Counterexample Analysis for $\mathcal{G}$



$QTF(\mathcal{C})$

$\exists V_{\forall}^{o\,0}, V_{\exists}^{<\infty\,0}, V_{\exists}^{\infty\,0}$

$\forall V_{\exists}^{<\infty\,1}, V_{\exists}^{\infty\,1}$

$\exists V_{\forall}^{o\,2}$

$\forall V_{\exists}^{<\infty\,3}, V_{\exists}^{\infty\,3}$

$\exists V_{\forall}^{o\,4}$

$\exists V_{\forall}^{o\,5}$

$\exists V_{\forall}^{u\,0}, V_{\forall}^{u\,1}, V_{\forall}^{u\,2}, \ldots$

Quantified tree formula $QTF(\mathcal{C})$

$QTF(\mathcal{C}) = \ldots \forall V_{\exists}^{\infty\,n}. \ \ldots$ unsatisfiable

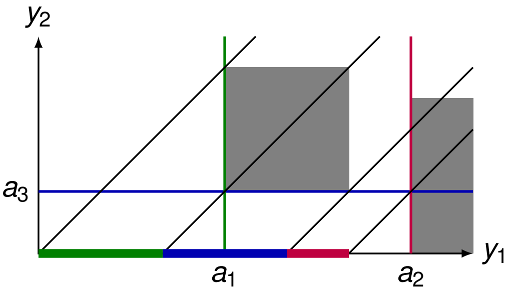
$\Leftrightarrow$

$\Phi = \neg QTF(\mathcal{C}) = \ldots \exists V_{\exists}^{\infty\,n} \ldots$ satisfiable

Computing Action Points

Restriction on the witness functions for $V_\exists^\infty$

- for $k \in \mathbb{N}_{>0}$, Φ_k requires that Φ is satisfied and for each block $\exists {V_\exists^\infty}^n$
  - there are constants $a_1, \ldots, a_k$ such that the witness function selects one of them and a corresponding variable in each of k cases



$a_1 = 3$, variable y_1
$a_2 = 6$, variable y_1
$a_3 = 1$, variable y_2

$$
\begin{array}{lcl}
y_1^n \in [0,2) & \mapsto & c_{y_1}^n = 3, c_{y_2}^n = 0 \\
y_1^n \in [2,4) & \mapsto & c_{y_1}^n = 0, c_{y_2}^n = 1 \\
y_1^n \in [4,5] & \mapsto & c_{y_1}^n = 6, c_{y_2}^n = 0
\end{array}
$$

Computing Action Points

Strengthening Φ

- strengthening for $k \in \mathbb{N}_{>0}$
- free variables for action points and corresponding variable indices

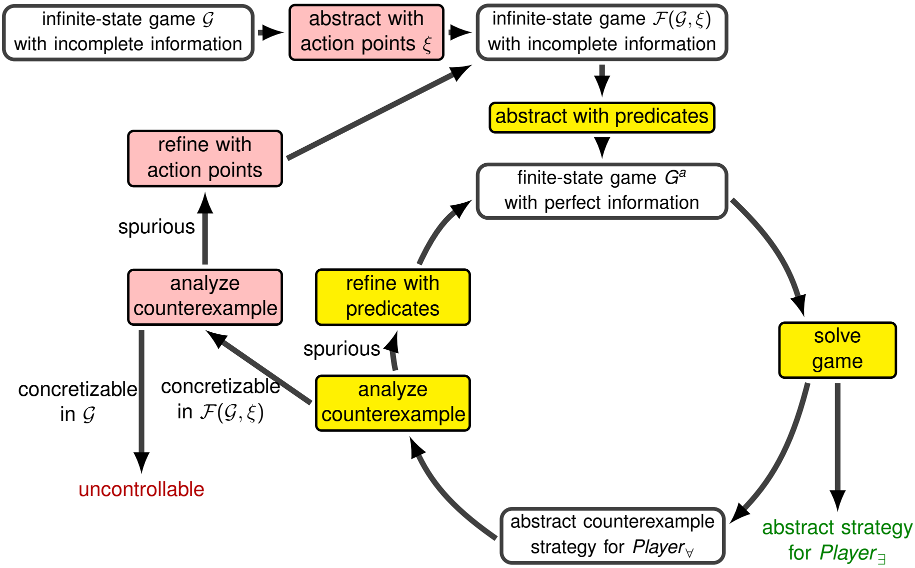
Refinement procedure

- Start with $k = 1$ and iterate incrementing k while Φ_k is unsatisfiable
- If some Φ_k is satisfiable, then extract the action points from model

Progress

If the loop terminates, the action points in ξ' suffice to eliminate from $\mathcal{F}(\mathcal{I}, \xi')$ all winning strategies for $Player_\exists$ that are subsumed by f_e.

Overview



infinite-state game $\mathcal{G}$ with incomplete information
abstract with action points ξ
infinite-state game $\mathcal{F}(\mathcal{G},\xi)$ with incomplete information
abstract with predicates
finite-state game G^a with perfect information
solve game
abstract strategy for $Player_\exists$
abstract counterexample strategy for $Player_\forall$
analyze counterexample
spurious
refine with predicates
concretizable in $\mathcal{F}(\mathcal{G},\xi)$
analyze counterexample
spurious
refine with action points
concretizable in $\mathcal{G}$
uncontrollable

Experimental Results

- Experiments with a prototype implementation
- Problem is out of the scope of state-of-the-art tools for timed controller synthesis $\Rightarrow$ no fully relevant comparison possible
- UPPAAL-TIGA: timed games with partial observability and fixed observable predicates, two examples from [CDLLR07]
- Compare to UPPAAL-TIGA using fixed granularity (observable predicates $0 \leq y < 1$, or, respectively, $0 \leq y < 0.5$ scaled accordingly)
- Scale examples to demonstrate the advantage of our approach over using fine granularity instead of (automatically) discovered predicates

Experimental Results

	A. Strategy	Act. Points	Obs. Preds	Time (s)	TIGA (s)
Paint	55	2	16	72.44	0.08
Paint-100	49	2	15	29.52	3.57
Paint-1000	49	2	15	29.57	336.34
Paint-10000	71	2	16	52.82	$>$ 1800
Paint-100000	71	2	16	52.21	$>$ 1800
Bricks	166	3	17	24.69	0.05
Bricks-100	174	3	17	24.43	2.63
Bricks-1000	174	3	17	24.21	302.08
Bricks-10000	154	3	17	24.62	$>$ 1800
Bricks-100000	174	3	17	24.00	$>$ 1800

Conclusions

- Counterexample-guided synthesis of observation predicates
- Two nested refinement loops:
  - analytical inner loop, computes decision predicates
    based on interpolation
  - constructive outer loop, computes action points
    based on witnesses of satisfiability
- Automatic synthesis of timed controllers with partial observation
- Pattern for reactive synthesis under incomplete information