Abstractions

Sayan Mitra

Verifying cyberphysical systems

mitras@illinois.edu

Outline

- Abstractions
- Simulation relations
- Composition and substitutivity

Abstractions and Simulations

Consider models that have the same external interface (input/output variables and actions)

We would like to *approximate* one (hybrid) automaton H_1 with another one H_2

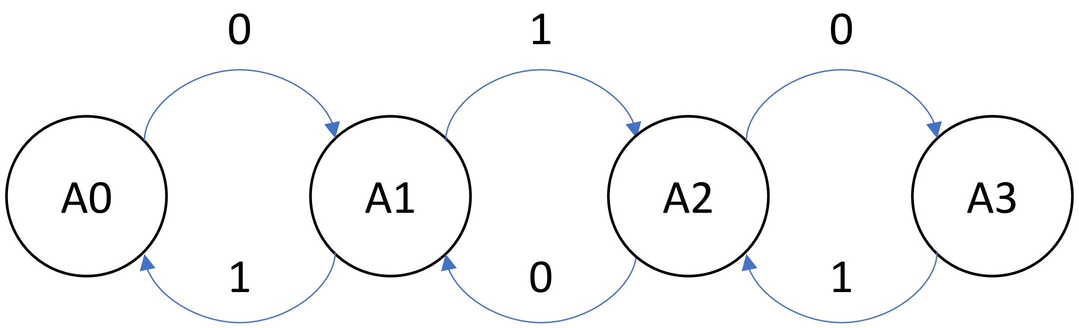
- We can over-approximate the reachable states of H_1 with those of H_2
- This would ensure that invariants of H_2 *carry over* to H_1
- We would like to go beyond invariants, and want to have more general requirements (e.g., CTL) carry over

H_2 should be *simpler* (smaller description, fewer states, transitions, linear dynamics, etc.) and preserve some properties of H_1 (and not others)

Verifying some requirements of H_2 can then carry over requirements to H_1

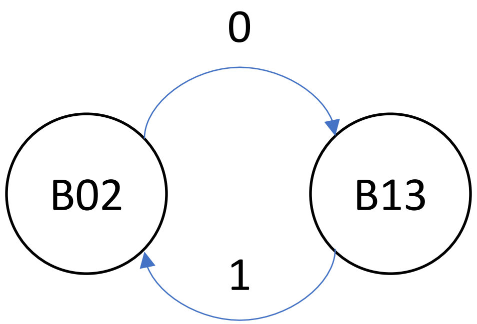
Finite state examples

A



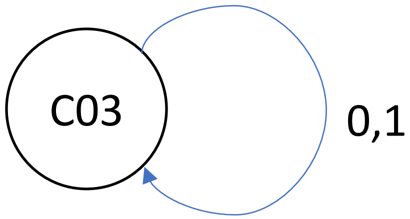
$\text{Traces}_A = (01)^*$

B



$\text{Traces}_B = 01^*$

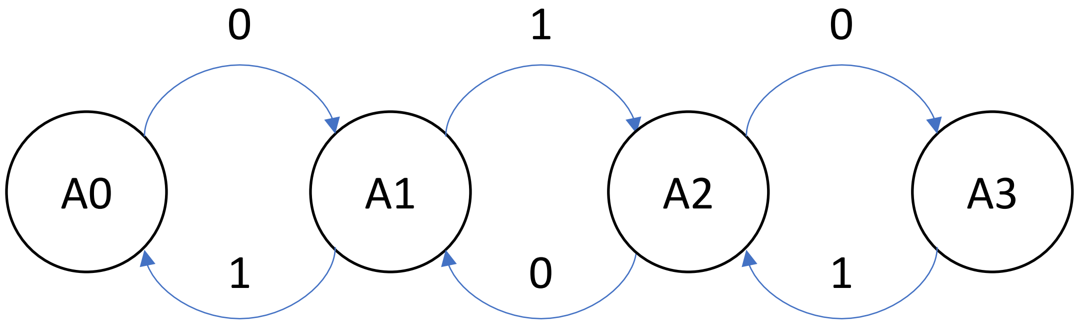
C



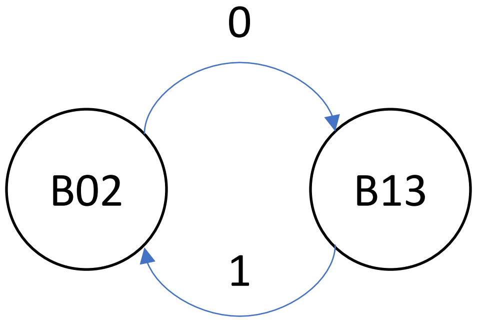
$\text{Traces}_C = \{0,1\}^*$

Finite state examples

A

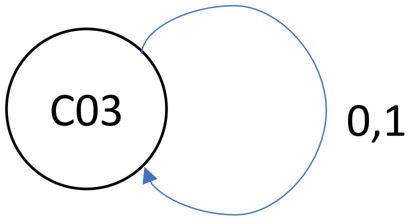


B



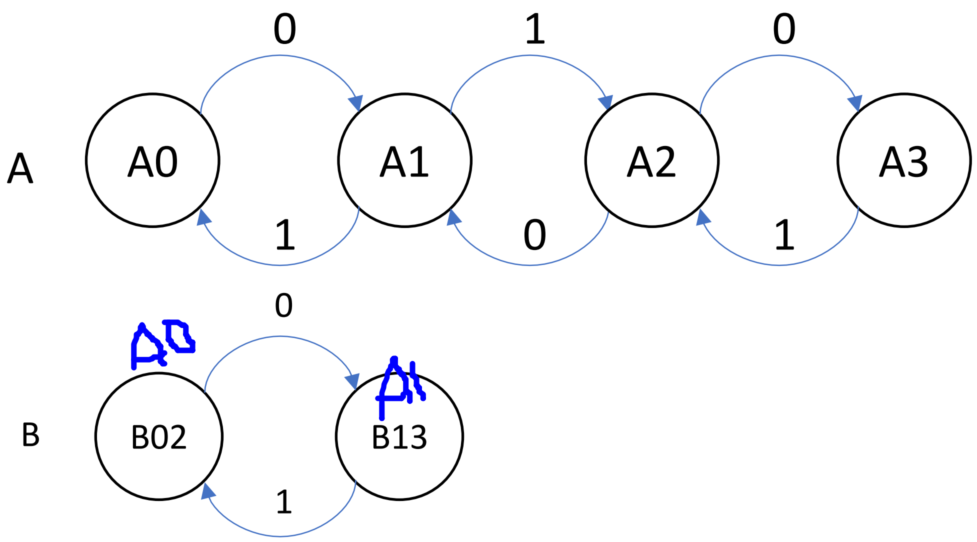
B **simulates** A and vice versa.
A and B are **bisimilar**.

C



C simulates both A and B.
C is an abstraction of both A and B.

How to prove B simulates A?



Show there exists a **simulation relation** from states of A to states of B. Say, $R\ =$ $((A0, B02), (A2, B02), (A1, B13), (A3, B13))$

Show that for every transition $Ai \to_A Ai'$ and $(Ai, Bj) \in R$ there exists Bj' such that

1. $Bj \to_B Bj'$
2. $(Ai', Bj') \in R$
3. $Trace(Bj \to_B Bj') = Trace(Ai \to_A Ai')$

Forward simulation relation

Consider a pair of automat $\mathcal{A}_1 = \langle Q_1, \Theta_1, A_1, D_1 \rangle$ and $\mathcal{A}_2 = \langle Q_2, \Theta_2, A_2, D_2 \rangle$.
Recall *trace* of an execution preserves the visible part of an execution

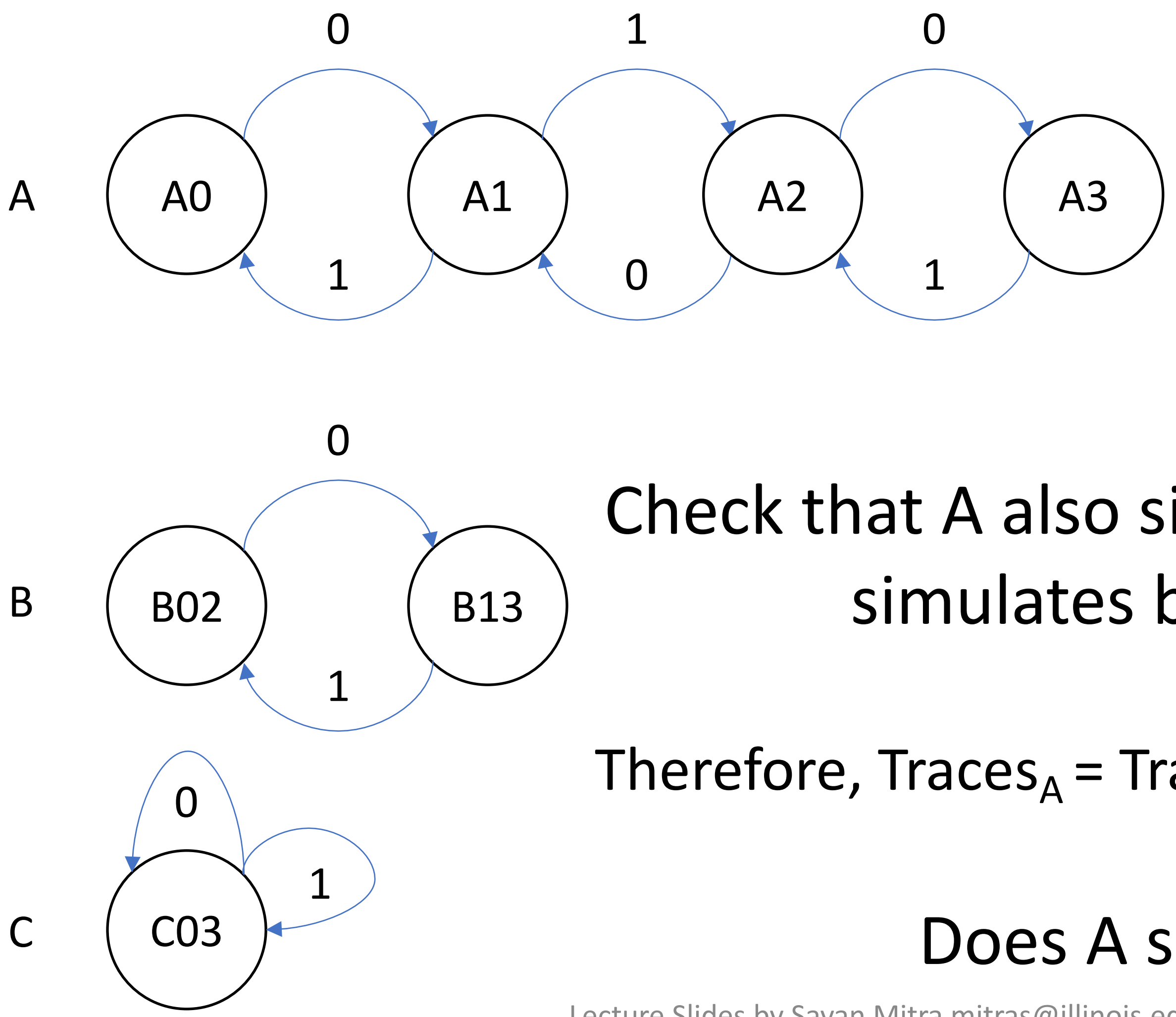
Definition. A **relation** $R \subseteq Q_1 \times Q_2$ is a forward simulation relation from $\mathcal{A}_1$ to $\mathcal{A}_2$ if

1. For every $q_1 \in \Theta_1$ there exists a $q_2 \in \Theta_2$ such that $q_1 R q_2$
2. For every transition $q_1 \rightarrow_1^{a_1} q_1'$ and $q_1 R q_2$ there exists q_2', a_2 such that
    - $q_2 \rightarrow_2^{a_2} q_2'$
    - $q_1' \; R \; q_2'$
    - $Trace(q_1, a_1, q_1') = Trace(q_2, a_2, {q_2}')$

Theorem. If there exists a forward simulation from $\mathcal{A}_1$ to $\mathcal{A}_2$ then $Traces_1 \subseteq Traces_2$.

Theorem. If there exists a forward simulation from $\mathcal{A}_1$ to $\mathcal{A}_2$ then $Traces_1 \subseteq Traces_2$.

Finite state examples



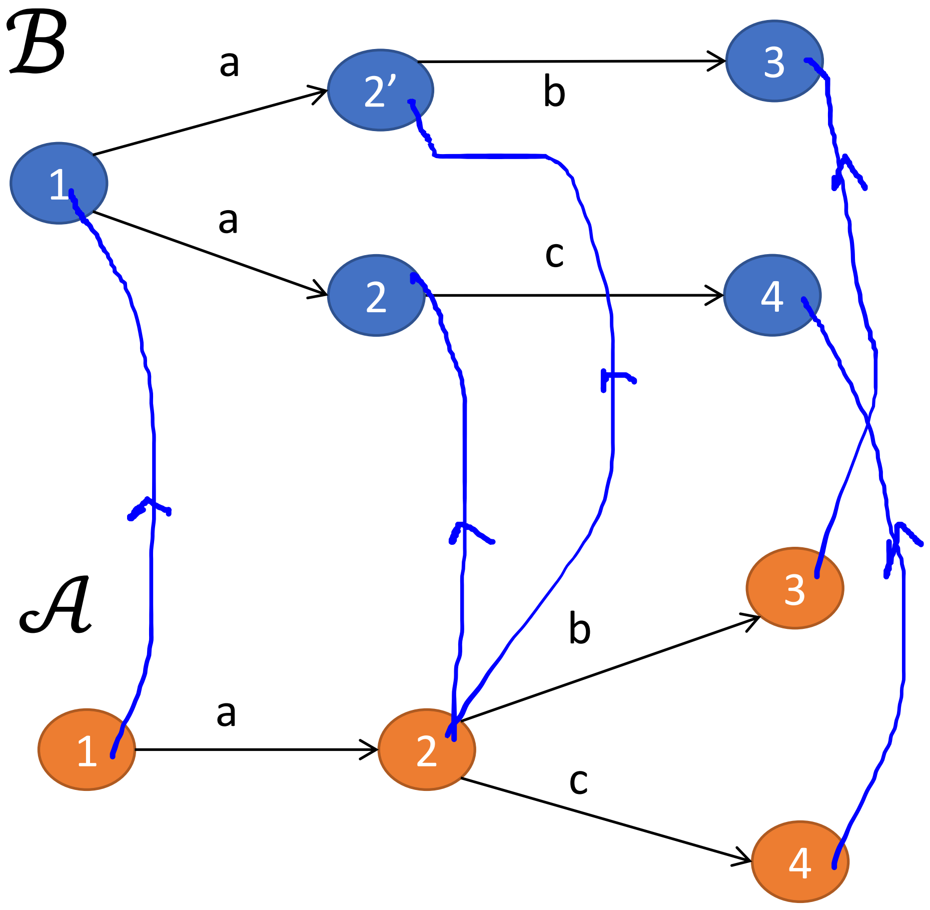
Check that A also simulates B and that C simulates both A and B.

Therefore, $\text{Traces}_A = \text{Traces}_B \subseteq Traces_C$?

Does A simulate C?

A Simulation Example

- $\mathcal{A}$ is an implementation of $\mathcal{B}$
- Is there a forward simulation from $\mathcal{A}$ to $\mathcal{B}$?
- Consider the forward simulation relation
- $\mathcal{A}$: $2 \rightarrow_c 4$ cannot be simulated by $\mathcal{B}$ from 2' although (2,2') are related.



Simulations for hybrid systems

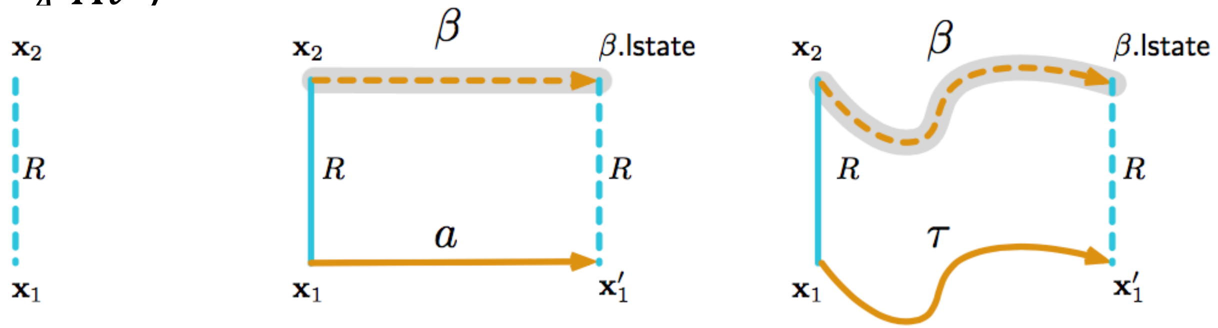
Forward simulation relation from $\mathcal{A}_1$ to $\mathcal{A}_2$ is a relation R $\subseteq val(X_1) \times val(X_2)$ such that

1. For every $\mathbf{x}_1 \in \Theta_1$ there exists $\mathbf{x}_2 \in \Theta_2$ such that $\mathbf{x}_1$ R $\mathbf{x}_2$
2. For every $\mathbf{x}_1 \rightarrow_{\boldsymbol{a_1}} \mathbf{x}_1' \in \mathcal{D}$ and $\mathbf{x}_2$ such that $\mathbf{x}_1$ R $\mathbf{x}_2$, there exists $\mathbf{x}_2'$ such that
   - $x_2 \rightarrow_{\boldsymbol{a_1}} x_2'$ and
   - x_1' R x_2'
3. For every $\boldsymbol{\tau_1} \in \mathcal{T}_1$ and $\mathbf{x}_2$ such that $\tau_1.fstate$ R $\mathbf{x}_2$, there exists $\tau_2 \in \mathcal{T}_2$ that
   - $x_2 = \tau_2.fstate$ and
   - x_1' R $\tau_2.lstate$
   - $\tau_2.\mathrm{dom} = \tau_1.dom$

Theorem. If there exists a forward simulation relation from hybrid automaton $\mathcal{A}_1$ to $\mathcal{A}_2$ then for every execution of $\mathcal{A}_1$ there exists a corresponding execution of $\mathcal{A}_2$.

Simulation relations for hybrid automata

- Recall condition 3 in definition of simulation relation: $Trace(Bj \rightarrow_B Bj') = Trace(Ai \rightarrow_A Ai')$



- Hybrid automata have transitions and trajectories
- Different types of simulation depending on different notions for "Trace"
  - Match for all variable values, action names, and time duration of trajectories (abstraction)
  - Match variables but not time (time abstract simulation)
  - Match a subset (external) of variables and actions (trace inclusion)
  - Match single action/trajectory of A with a sequence of actions and trajectories of B

Timer simulates Ball (w.r.t. timing of bounce actions)

Automaton Ball(c, v_0, g)
  variables:
    x: Reals := 0
    v: Reals := v_0
  actions: bounce
    transitions:
      bounce
        pre $x = 0 \wedge v < 0$
        eff v := -cv
    trajectories:
        evolve d(x) = v; d(v) = -g
        invariant $\boldsymbol{x \geq 0}$

Automaton Timer(c, v_0, g)
  variables: analog
  timer: Reals := $2v_0/g$,
  n:Naturals=0;
  actions: bounce
    transitions:
      bounce
        pre *timer = 0*
        eff n:=n+1; timer := $\frac{2v_0}{gc^n}$
    trajectories:
        evolve d(timer) = -1
        invariant timer ≥ 0

Some nice properties of Forward Simulation

Let $\mathcal{A}, \mathcal{B}$, and $\mathcal{C}$ be comparable TAs. If R_1 is a forward simulation from $\mathcal{A}$ to $\mathcal{B}$ and R_2 is a forward simulation from $\mathcal{B}$ to $\mathcal{C}$, then $R_1 \circ R_2$ is a forward simulation from $\mathcal{A}$ to $\mathcal{C}$

$\mathcal{A}$ implements $\mathcal{C}$

The **implementation relation** is a preorder of the set of all (comparable) hybrid automata

(A preorder is a reflexive and transitive relation)

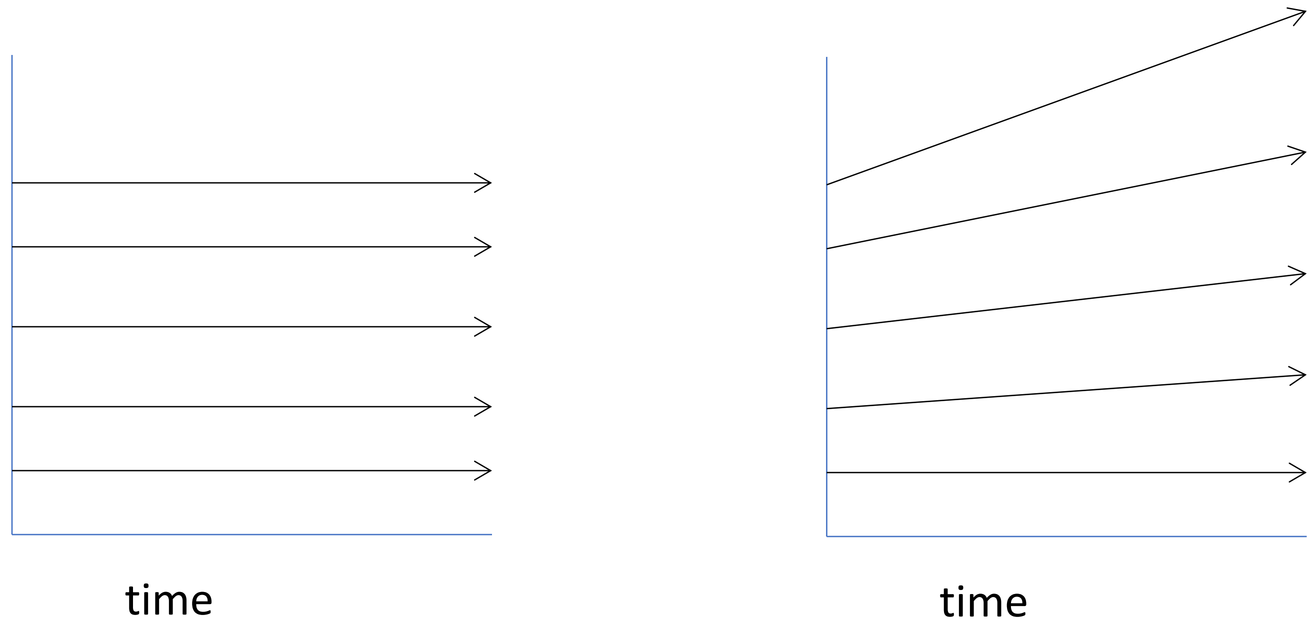
If R is a forward simulation from $\mathcal{A}$ to $\mathcal{B}$ and R^{-1} is a forward simulation from $\mathcal{B}$ to $\mathcal{A}$ then R is called a **bisimulation** and $\mathcal{B}$ are $\mathcal{A}$ **bisimilar**

Bisimilarity is an **equivalence relation**

(reflexive, transitive, and symmetric)

Remark on Simulations and Stability

Stability not preserved by ordinary simulations and bisimulations [Prabhakar, et. al 15]



Stability Preserving Simulations and Bisimulations for Hybrid Systems, Prabhakar, Dullerud, Viswanathan IEEE Trans. Automatic Control 2015

Backward Simulations

Backward simulation relation from $\mathcal{A}_1$ to $\mathcal{A}_2$ is a relation $R \subseteq Q_1 \times Q_2$ such that

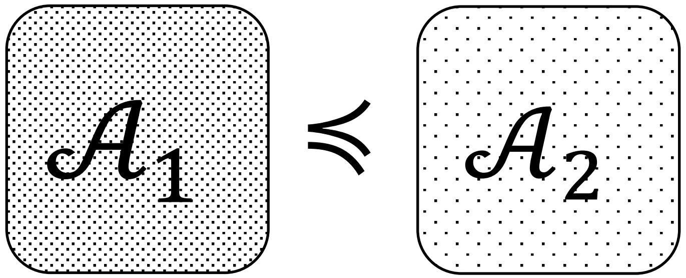
1. If $x_1 \in \Theta_1$ and $x_1 \, R \, x_2$ then $x_2 \in \Theta_2$ such that
2. If $x'_1 \, R \, x'_2$ and $x_1 \xrightarrow{a} x_1'$ then
   - $x_2 \xrightarrow{\beta} x_2'$ and
   - $x_1 \, R \, x_2$
   - Trace(β) = a_1
3. For every $\tau \in \mathcal{T}$ and $x_2 \in Q_2$ such that $x_1' \, R \, x_2'$, there exists x_2 such that
   - $x_2 \xrightarrow{\beta} x_2'$ and
   - $x_1 \, R \, x_2$
   - Trace(β) = τ

Theorem. If there exists a backward simulation relation from $\mathcal{A}_1$ to $\mathcal{A}_2$ then ClosedTraces$_1 \subseteq$ ClosedTraces$_2$

Abstractions II

Abstraction recap

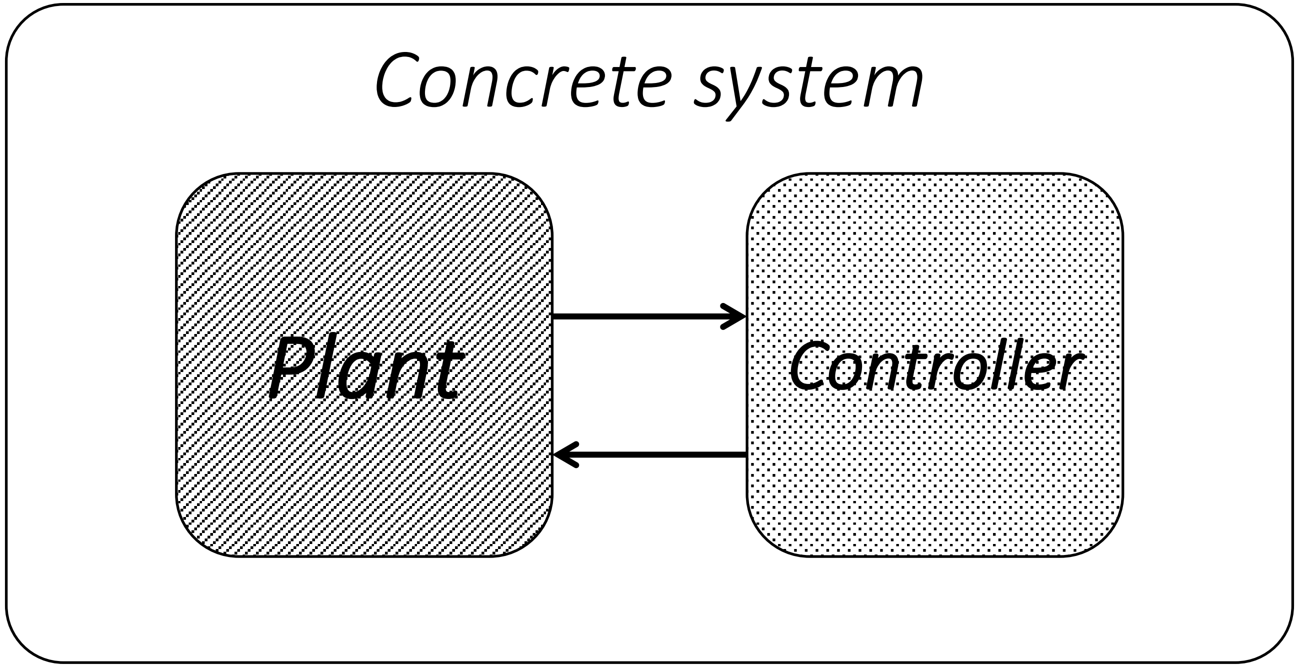
- Defined what it means for $\mathcal{A}_2$ to be abstraction of$\mathcal{A}_1$
- $Traces_{\mathcal{A}_1} \subseteq Traces_{\mathcal{A}_2}$
- $\mathcal{A}_1 \preccurlyeq_T \mathcal{A}_2$
- If $\mathcal{A}_1 \preccurlyeq_T \mathcal{A}_2$ and $\mathcal{A}_2 \preccurlyeq_T \mathcal{A}_1$ then $\mathcal{A}_1 \preccurlyeq_T \mathcal{A}_3$
- Transitive, $\preccurlyeq_T$ defines a preordering on compatible automata
- We saw methods for proving $\mathcal{A}_1 \preccurlyeq_T \mathcal{A}_2$
  - *Forward simulation and backward simulation*
- $\preccurlyeq_T$ defines a preorder



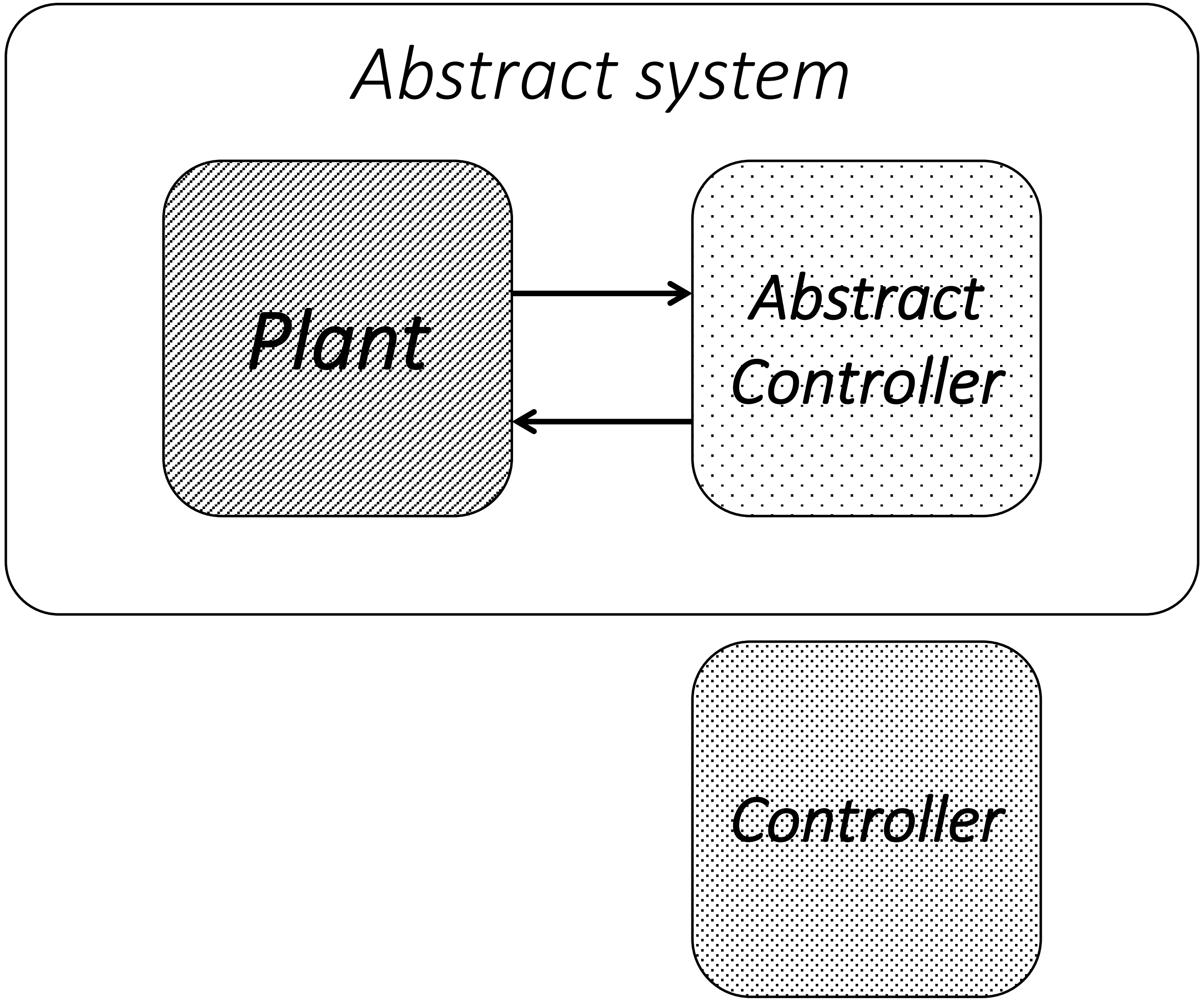
Outline

- Abstractions and composition
- CEGAR

Substituting an automaton with its abstraction

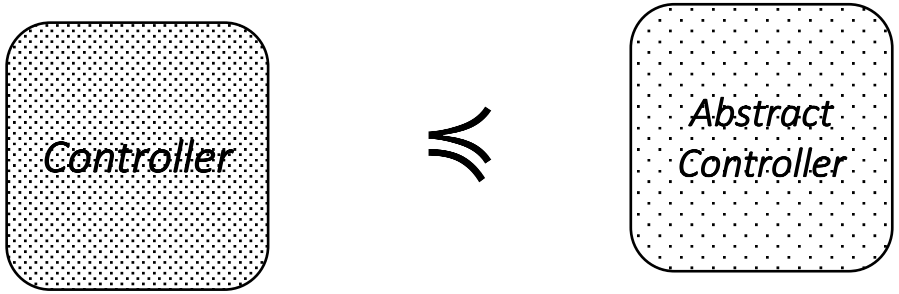


Substituting an automaton with its abstraction

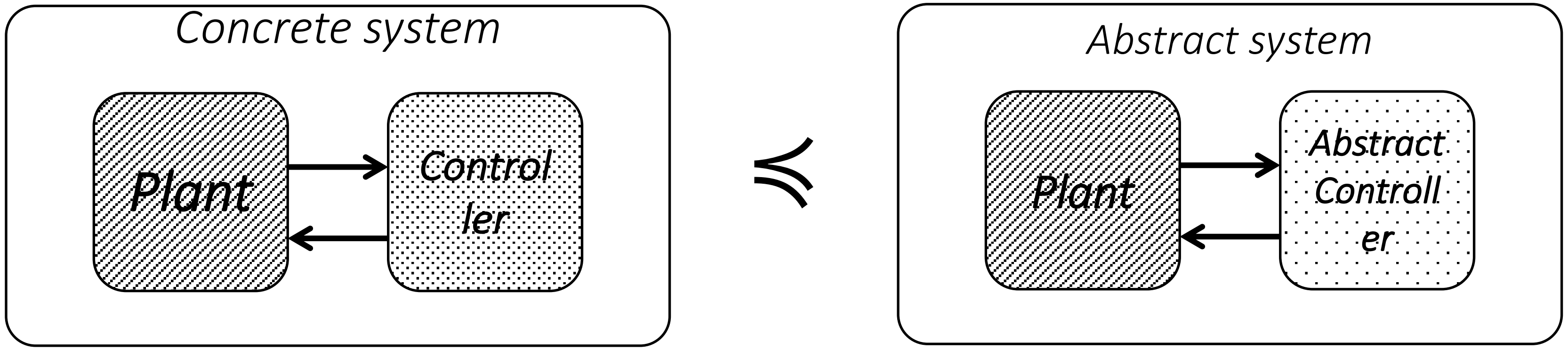


How is the abstract system related to the concrete system?

Does



imply



?

Hybrid IO Automaton

Recall a hybrid automaton $\mathcal{A} = \langle V, \Theta, A, \boldsymbol{D}, \boldsymbol{T} \rangle$

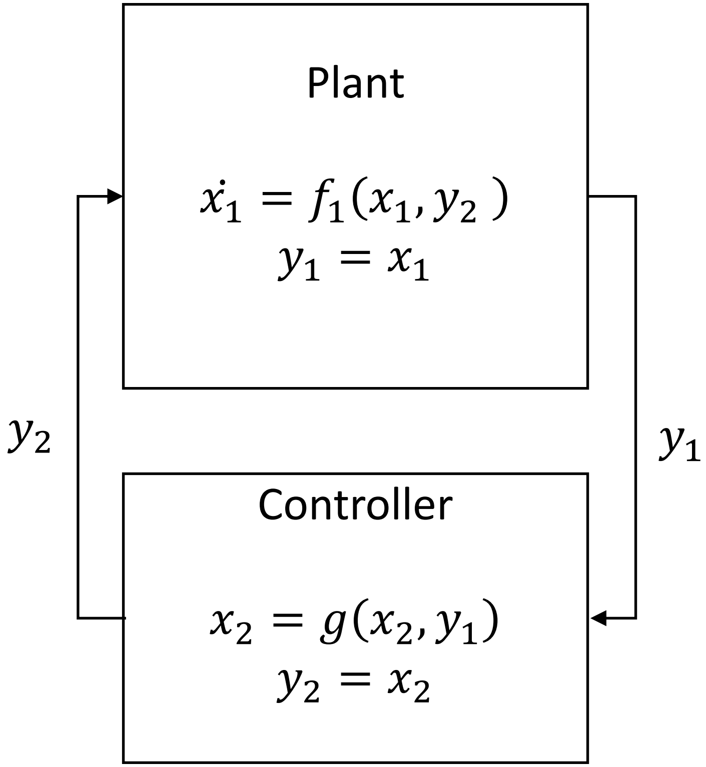
We will partition the set of variables V of $\mathcal{A}$ into

- X: internal or state variables (do not interact)
- Y: output variables
- U: input variables

- $V = X \cup Y \cup U$

This gives rise to hybrid I/O automata (HIOA) **[Lynch, Segala, Vaandrager 2002]**

We defined composition of compatible HIOA $\mathcal{A} = \mathcal{A}_1 || \mathcal{A}_2$



Composition of Hybrid Automata

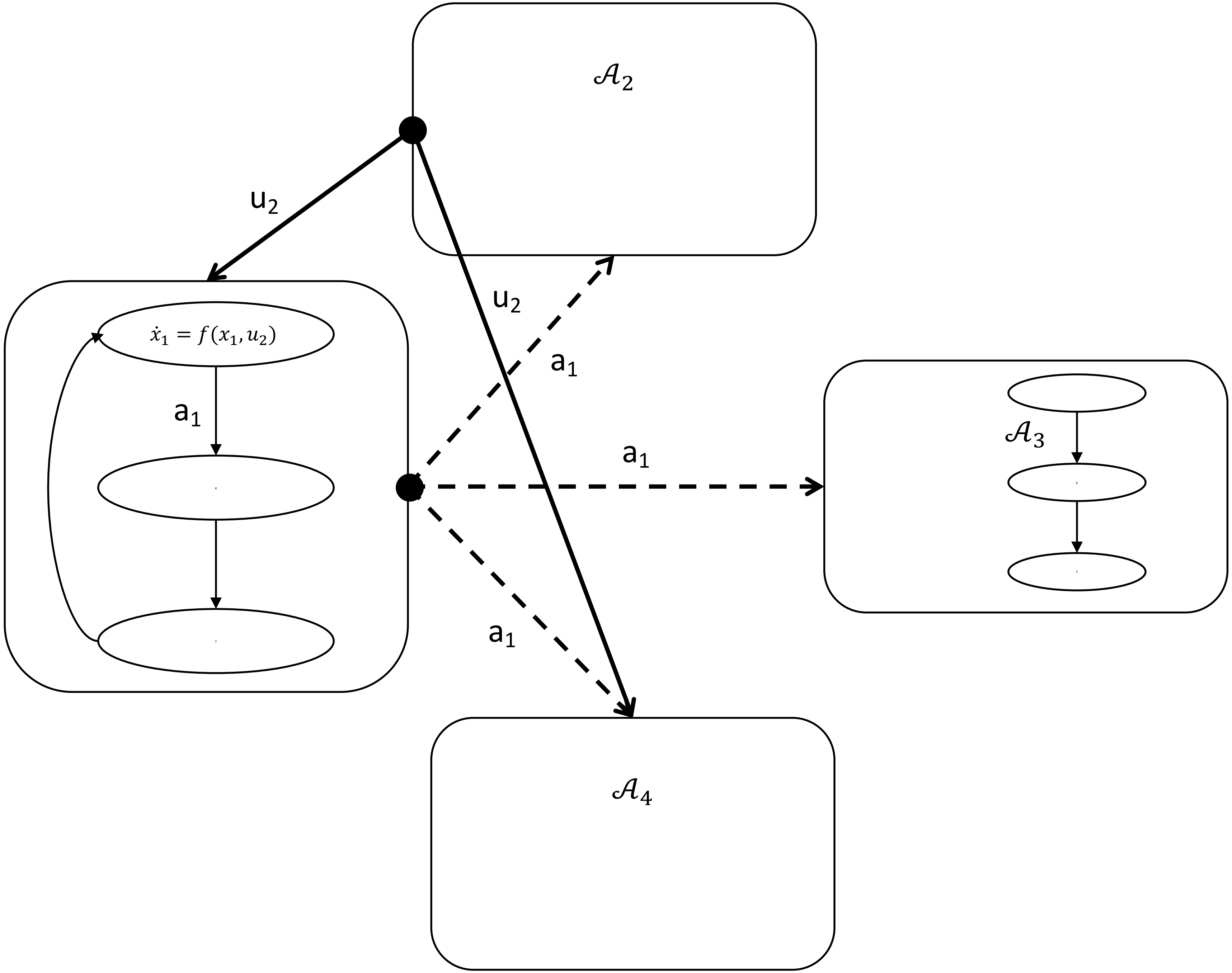
The parallel composition operation on automata enable us to construct larger and more complex models from simpler automata modules

$\mathcal{A}_1$ to $\mathcal{A}_2$ are compatible if $X_1 \cap X_2 = H_1 \cap A_2 = H_2 \cap A_1 = \emptyset$

Variable names are disjoint; Action names of one are disjoint with the internal action names of the other

Composition

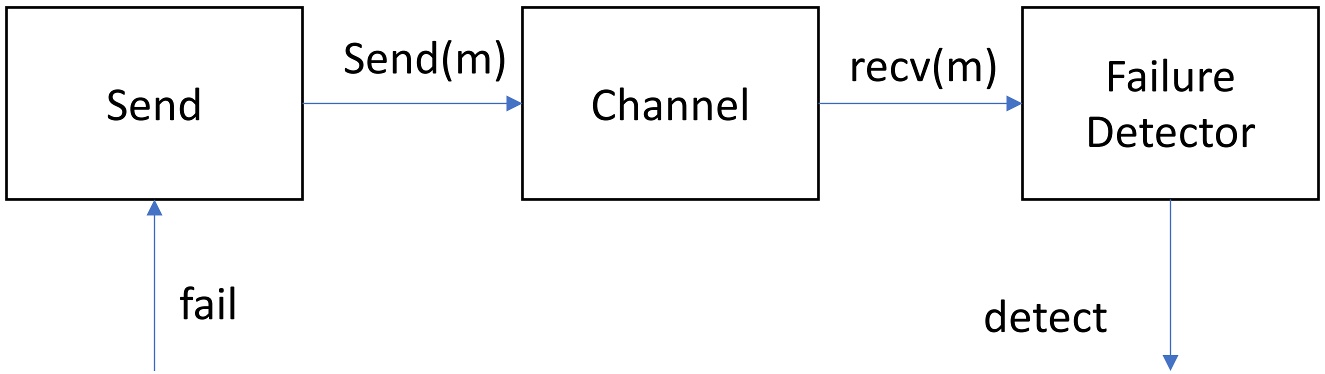
- For compatible $\mathcal{A}_1$ and $\mathcal{A}_2$ their composition $\mathcal{A}_1 \mid\mid \mathcal{A}_2$ is the structure $\boldsymbol{\mathcal{A}}= (V, \Theta, A, \mathcal{D}, \mathcal{T})$
- Variables $V = X \cup Y \cup U$
  - $X = X_1 \cup X_2, \mathrm{Y} = Y_1 \cup Y_2, U = U_1 \cup U_2 \setminus Y$
- $\Theta = \{ \boldsymbol{x} \in val(X) | \, \forall i \ \in \{1,2\}\colon \boldsymbol{x}\lceil X_{\boldsymbol{i}} \in \Theta_i\}$
- Actions $A = H \cup O \cup I$
  - $H = H_1 \cup H_2, \mathrm{O} = O_1 \cup O_2, \mathrm{I} = I_1 \cup I_2 \setminus O,$
- $(\boldsymbol{x}, a, \boldsymbol{x}') \in \ \mathcal{D}$ *iff for* $i \in \{1,2\}$
  - $a \in A_i$ *and* $(\boldsymbol{x}\lceil X_i, a, \boldsymbol{x}'\lceil X_i) \in \ \mathcal{D}_i$
  - $a \notin A_i \ \boldsymbol{x}\lceil X_i = \boldsymbol{x}\lceil X_i$
- $\mathcal{T}$*:* set of **trajectories** for V
  - $\tau \in \ \mathcal{T}$ iff $\forall i \ \in \{1,2\}, \ \tau \downarrow V_i \in \mathcal{T}_{\mathrm{i}}$



$\mathcal{A}_2$
u_2
u_2
a_1
$\dot{x}_1 = f(x_1, u_2)$
a_1
a_1
$\mathcal{A}_3$
a_1
$\mathcal{A}_4$

Modeling a Simple Failure Detector System

- Periodic send
- Channel
- Timeout



Composition

- For compatible $\mathcal{A}_1$ and $\mathcal{A}_2$ their composition $\mathcal{A}_1 \mid\mid \mathcal{A}_2$ is the structure $\boldsymbol{\mathcal{A}}= (X, Q, \Theta, E, H, \mathcal{D}, \mathcal{T})$
- $X = X_1 \cup X_2$ (disjoint union)
- $Q \subseteq val(X)$
- $\Theta = \{ \boldsymbol{x} \in Q \mid \forall i \in \{1,2\}: \boldsymbol{x}.Xi \in \Theta_i\}$
- $H = H_1 \cup H_2$ *(disjoint union)*
- $E = E_1 \cup E_2$ *and* A= E ∪ H
- $(\boldsymbol{x}, a, \boldsymbol{x}') \in \mathcal{D}$ *iff*
  - $a \in H_1$ *and* $(\boldsymbol{x}.X_1, a, \boldsymbol{x}'.X_1) \in \mathcal{D}_1$ *and* $\boldsymbol{x}.X_2 = \boldsymbol{x}.X_2$
  - $a \in H_2$ *and* $(\boldsymbol{x}.X_2, a, \boldsymbol{x}'.X_2) \in \mathcal{D}_2$ *and* $\boldsymbol{x}.X_1 = \boldsymbol{x}.X_1$
  - *Else,* $(\boldsymbol{x}.X_1, a, \boldsymbol{x}'.X_1) \in \mathcal{D}_1$ *and* $(\boldsymbol{x}.X_2, a, \boldsymbol{x}'.X_2) \in \mathcal{D}_2$
- $\mathcal{T}$: set of trajectories for X
  - $\tau \in \mathcal{T}$ iff $\forall i \in \{1,2\}$, $\tau.Xi \in \mathcal{T}_i$

Theorem . $\boldsymbol{\mathcal{A}}$ is also a hybrid automaton.

Example: Send || TimedChannel

Automaton Channel(b,M)

variables: internal

queue: **Queue**[M,Reals] := {}

clock1: Reals := 0

actions: external send(m:M), receive(m:M)

transitions:

send(m)

pre true

eff queue := append(<m, clock1+b>, queue)

receive(m)

pre head(queue)[1] = m

eff queue := queue.tail

trajectories:

evolve d(clock1) = 1

stop when ∃ m, d, <m,d> ∈ queue

⋀ clock=d

Automaton PeriodicSend(u, M)

variables: internal clock: Reals := 0

actions: external send(m:M)

transitions:

send(m)

pre clock = u

eff clock := 0

trajectories:

evolve d(clock) = 1

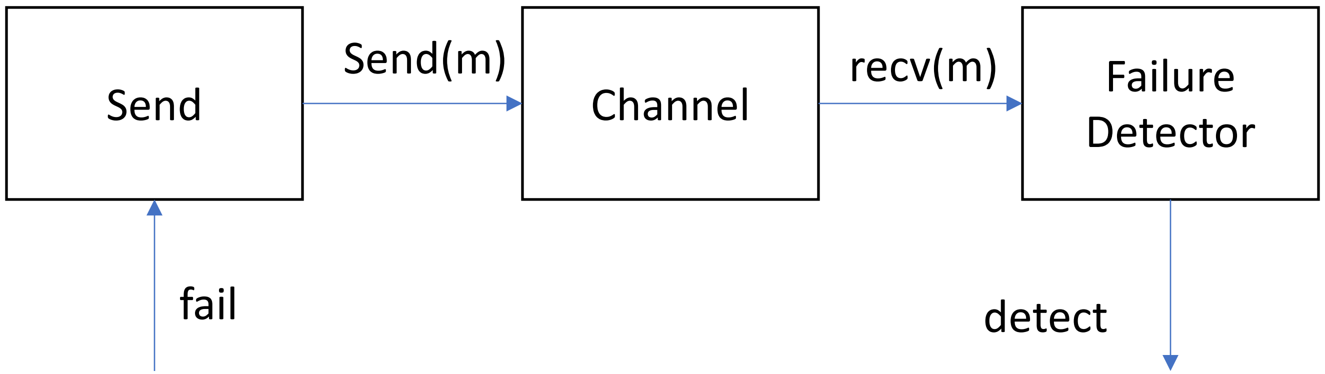
stop when clock=u

Composed Automaton

```
Automaton SC(b,u)
  variables: internal queue: Queue[M,Reals] := {}
            clock_s, clock_c: Reals := 0
  actions: external send(m:M), receive(m:M)
  transitions:
    send(m)
    pre clock_s = u
    eff  queue := append(<m, clock_c+b>, queue); clock_s := 0
    receive(m)
    pre head(queue)[1] = m
    eff queue := queue.tail
  trajectories:
    evolve d(clock_c) = 1; d(clock_s) = 1
    stop when
        (∃ m, d, <m,d> ∈ queue /\  clock_c=d)
        \/ (clock_s=u)
```

Modeling a Simple Failure Detector System

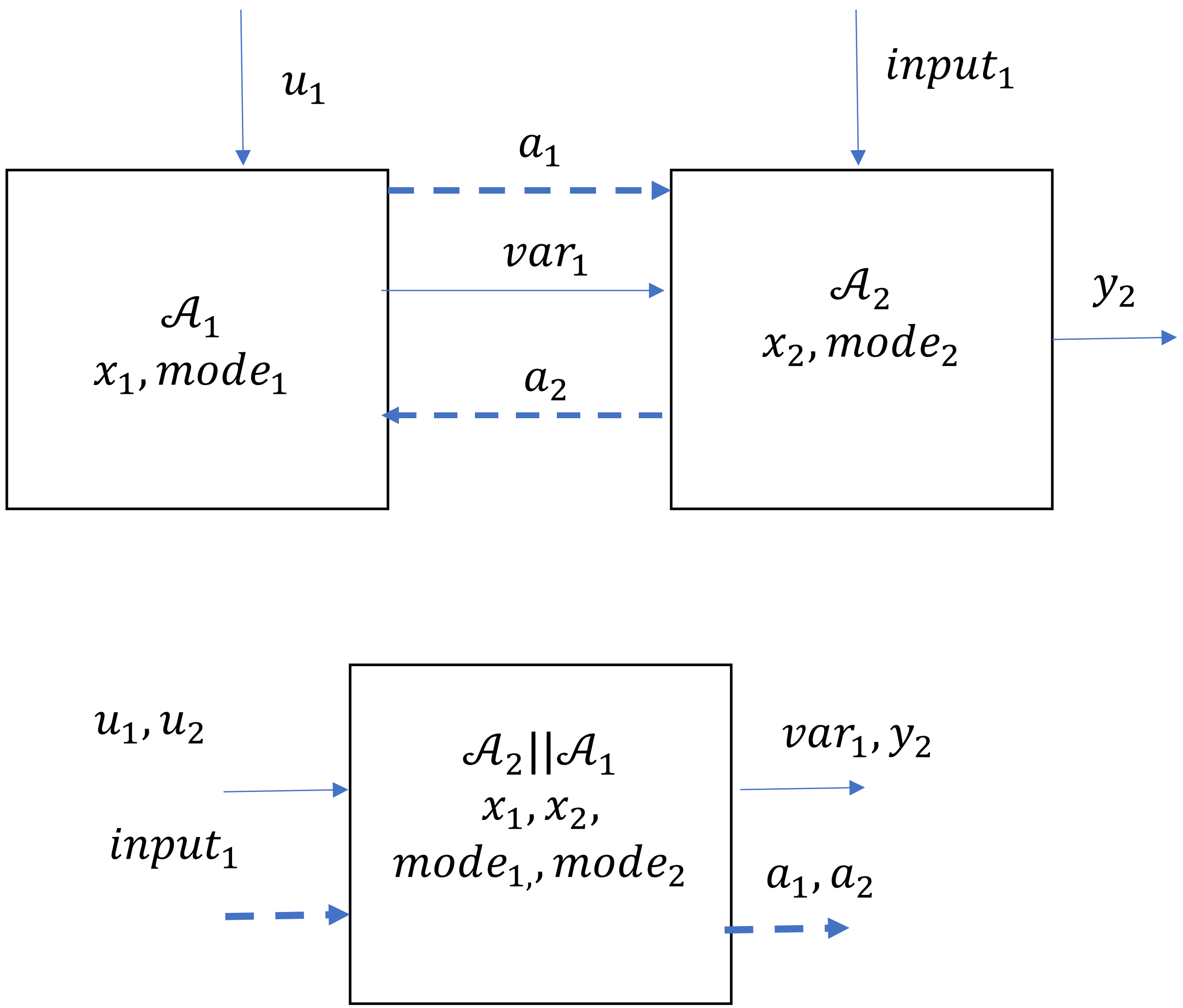
- Periodic send || Channel
- Periodic send || Channel || Timeout



Time bounded channel & Simple Failure Detector

```
Automaton Timeout(u,M)
  variables: internal suspected: Boolean := F,
             clock: Reals := 0
  actions: external receive(m:M), timeout
  transitions:
     receive(m)
     pre true
     eff  clock := 0; suspected := false;
     timeout
     pre ~suspected /\ clock = u
     eff suspected := true
  trajectories:
     evolve d(clock) = 1
     stop when clock = u /\ ~suspected
```

General composition



Some properties about composed automata

- Let $\boldsymbol{\mathcal{A}} = \mathcal{A}_1 \,||\, \mathcal{A}_2$ and let α be an execution fragment of $\boldsymbol{\mathcal{A}}$.
  - Then $\alpha_i = \alpha|(A_i, X_i)$ is an execution fragment of $\mathcal{A}_i$
  - α is time-bounded iff both α_1 and α_2 are time-bounded
  - α is admissible iff both α_1 and α_2 are admissible
  - α is closed iff both α_1 and α_2 are closed
  - α is non-Zeno iff both α_1 and α_2 are non-Zeno
  - α is an execution iff both α_1 and α_2 are executions
- Traces $_{\mathcal{A}} = \{ \boldsymbol{\beta} \mid \boldsymbol{\beta} | E_i \in \text{Traces } \mathcal{A}_i \}$
- See examples in the TIOA monograph

A trace theorem restriction from composition of I/O automata

Theorem 5.5 (from Theory of Timed I/O Automata by Lynch et. al.)

Suppose $\mathcal{A} = \mathcal{A}_1||\mathcal{A}_2$ and let E be the set of input/output actions of A. Then $\mathbf{Traces}_{\mathcal{A}}$ is exactly the set of (E, Ø)-sequences whose restrictions to $\mathcal{A}_1$ and $\mathcal{A}_2$ are traces of $\mathcal{A}_1$ **and** $\mathcal{A}_2$, respectively. That is,

$\mathbf{Traces}_{\mathcal{A}}$ = {β | β is an (E, Ø)-sequence and $\beta\lceil(E_{\mathcal{A}_1}||\emptyset) \in \mathbf{Traces}_{\mathcal{A}_i}$, i ∈ {1, 2}}.

Substitutivity

Theorem 1. Suppose $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{B}$ have the same external interface and $\mathcal{A}_1$, $\mathcal{A}_2$ are compatible with $\mathcal{B}$. If $\mathcal{A}_{\mathbf{1}} \preccurlyeq \mathcal{A}_2$ then $\mathcal{A}_1 || \mathcal{B} \preccurlyeq \mathcal{A}_2 || \mathcal{B}$

Substutivity

Theorem 2. Suppose $\mathcal{A}_1$ $\mathcal{A}_2$ $\mathcal{B}_1$ and $\mathcal{B}_2$ are HAs and $\mathcal{A}_1$ $\mathcal{A}_2$ have the same external actions and $\mathcal{B}_1$ $\mathcal{B}_2$ have the same external actions and $\mathcal{A}_1$ $\mathcal{A}_2$ is compatible with each of $\mathcal{B}_1$ and $\mathcal{B}_2$.

If $\mathcal{A}_1 \preccurlyeq \mathcal{A}_2$ and $\mathcal{B}_1 \preccurlyeq \mathcal{B}_2$ then $\mathcal{A}_1 \,||\, \mathcal{B}_1 \preccurlyeq \mathcal{A}_2 || \mathcal{B}_2$.

- Proof. $\mathcal{A}_1 \,||\, \mathcal{B}_1 \preccurlyeq \mathcal{A}_2 || \mathcal{B}_1$

  $\mathcal{A}_2 || \mathcal{B}_1 \preccurlyeq \mathcal{A}_2 || \mathcal{B}_2$

  By transitivity of implementation relation

  $\mathcal{A}_1 \,||\, \mathcal{B}_1 \preccurlyeq \mathcal{A}_2 || \mathcal{B}_2$

A stronger substitutivity result

Theorem 3. $\mathcal{A}_1 \,||\, \mathcal{B}_2 \preccurlyeq \mathcal{A}_2 || \mathcal{B}_2$ and $\mathcal{B}_1 \preccurlyeq \mathcal{B}_2$ then $\mathcal{A}_1 \,||\, \mathcal{B}_1 \preccurlyeq \mathcal{A}_2 || \mathcal{B}_2$.

A stronger substitutivity result

Theorem 3. $\mathcal{A}_1 \,||\, \mathcal{B}_2 \preccurlyeq \mathcal{A}_2||\mathcal{B}_2$ and
$\mathcal{B}_1 \preccurlyeq \mathcal{B}_2$ then $\mathcal{A}_1 \,||\, \mathcal{B}_1 \preccurlyeq \mathcal{A}_2||\mathcal{B}_2$.

Proof. Let β ∈ $\mathbf{Traces}_{\mathcal{A}_1 \,||\, \mathcal{B}_1}$.
By Theorem 5.5 (of LVS TIOA), $\beta\lceil(E_{\mathcal{A}_1}||\emptyset) \in Traces_{\mathcal{A}_1}$ and $\beta\lceil(E_{\mathcal{B}_1}||\emptyset) \in Traces_{\mathcal{B}_1}$.
Since $\mathcal{B}_1 \preccurlyeq \mathcal{B}_2$

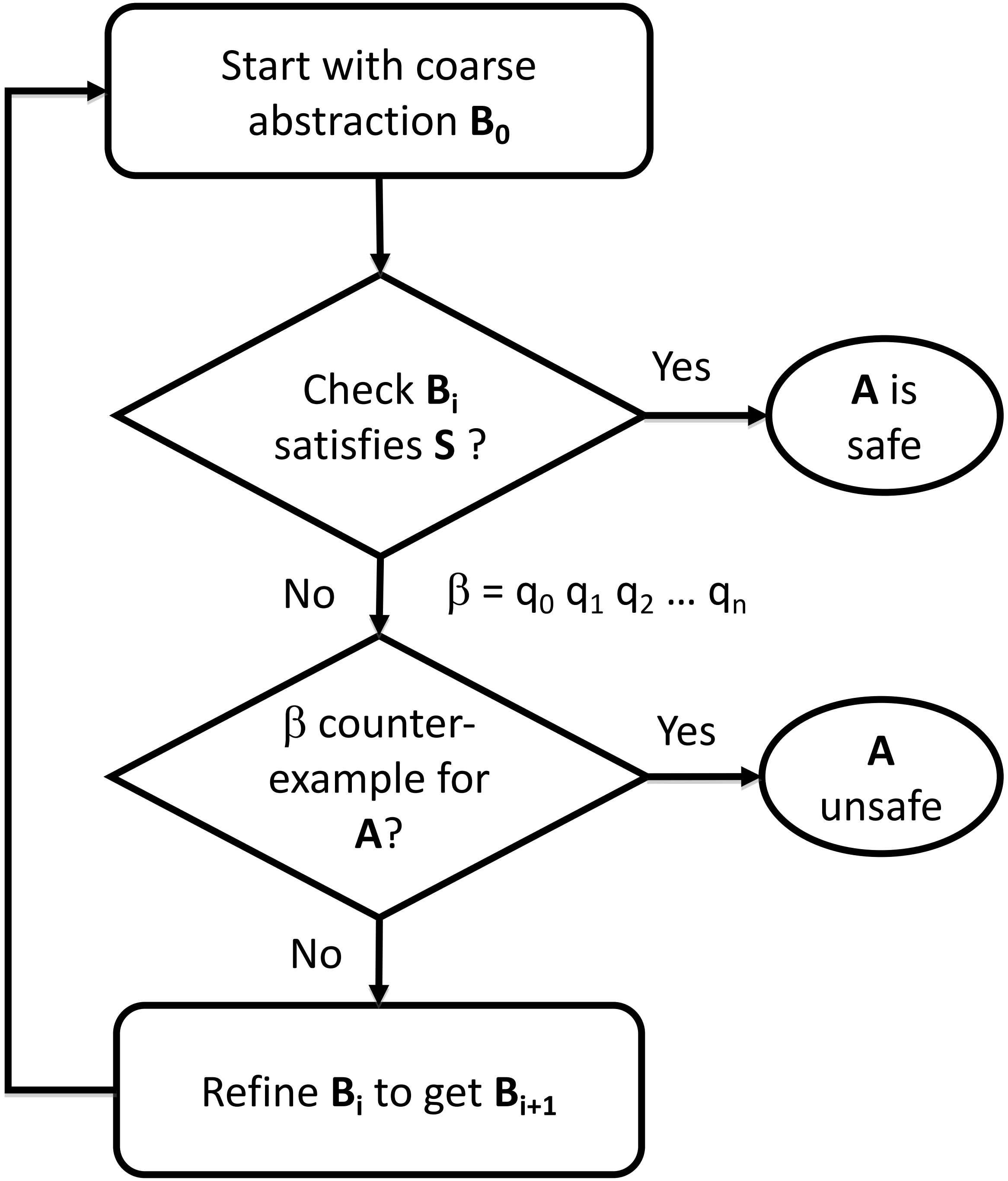
$\beta\lceil\left(E_{\mathcal{B}_2}||\emptyset\right) \in Traces_{\mathcal{B}_2}$

By Theorem 5.5 , β ∈ $\mathbf{Traces}_{\mathcal{A}_1 \,||\, \mathcal{B}_2}$
Since $\mathcal{A}_1 \,||\, \mathcal{B}_2 \preccurlyeq \mathcal{A}_2||\mathcal{B}_2$ by assumption, β ∈ $\mathbf{Traces}_{\mathcal{A}_2 \,||\, \mathcal{B}_2}$

Counter-example guided abstraction-refinement

Counterexample guided abstraction refinement (CEGAR)

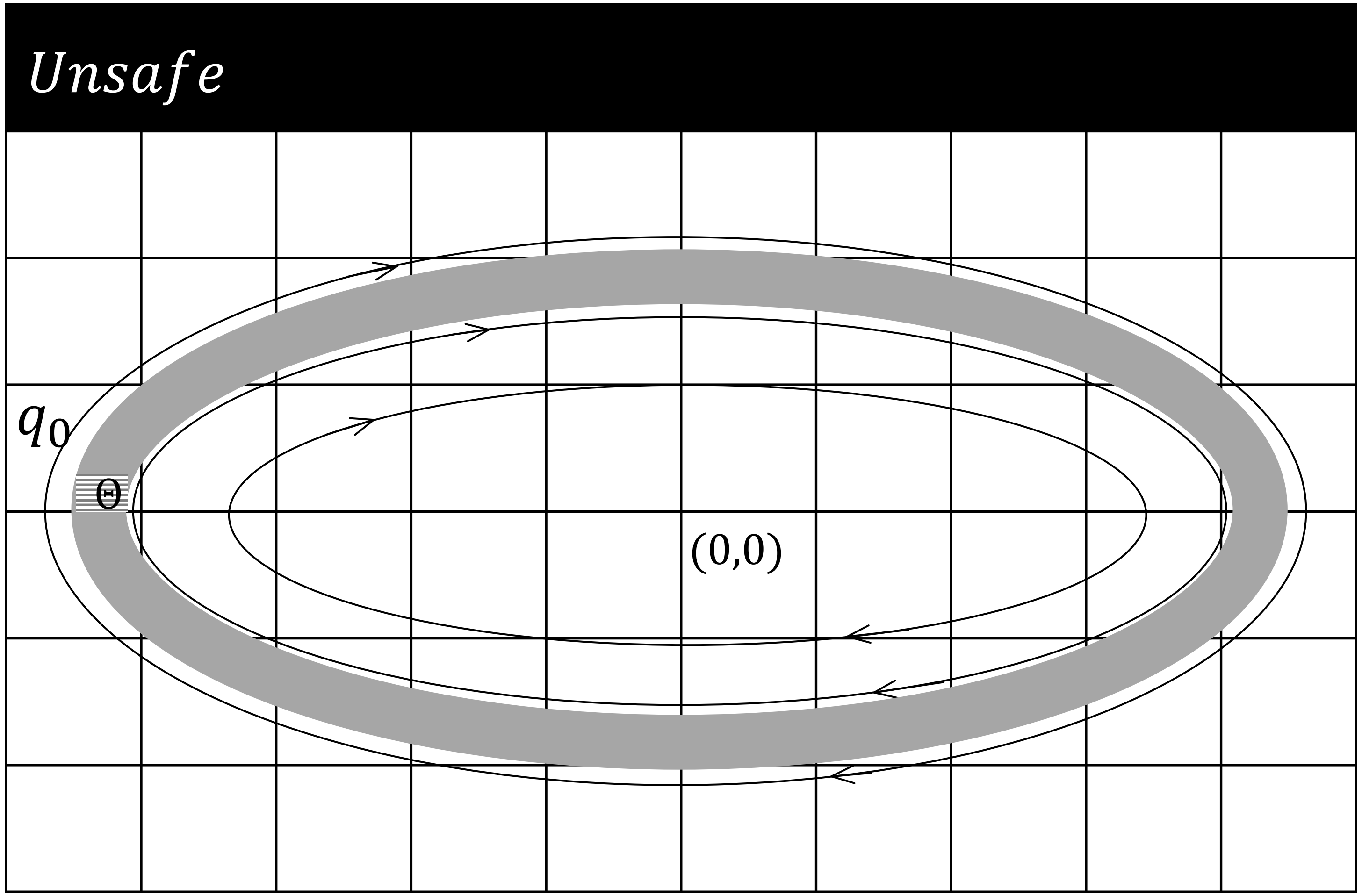
- A general algorithmic framework for automatically constructing and verifying property-specific abstractions [Clarke:2000]
- CEGAR has been applied to discrete automata, software, and hybrid systems [Holzman 00,Ball 01, Alur 2006,Clarke 2003, Fehnker2005, Prabhakar 15, Roohi 17]
- We will discuss the basic idea of the CEGAR and the key design choices, and their implications.



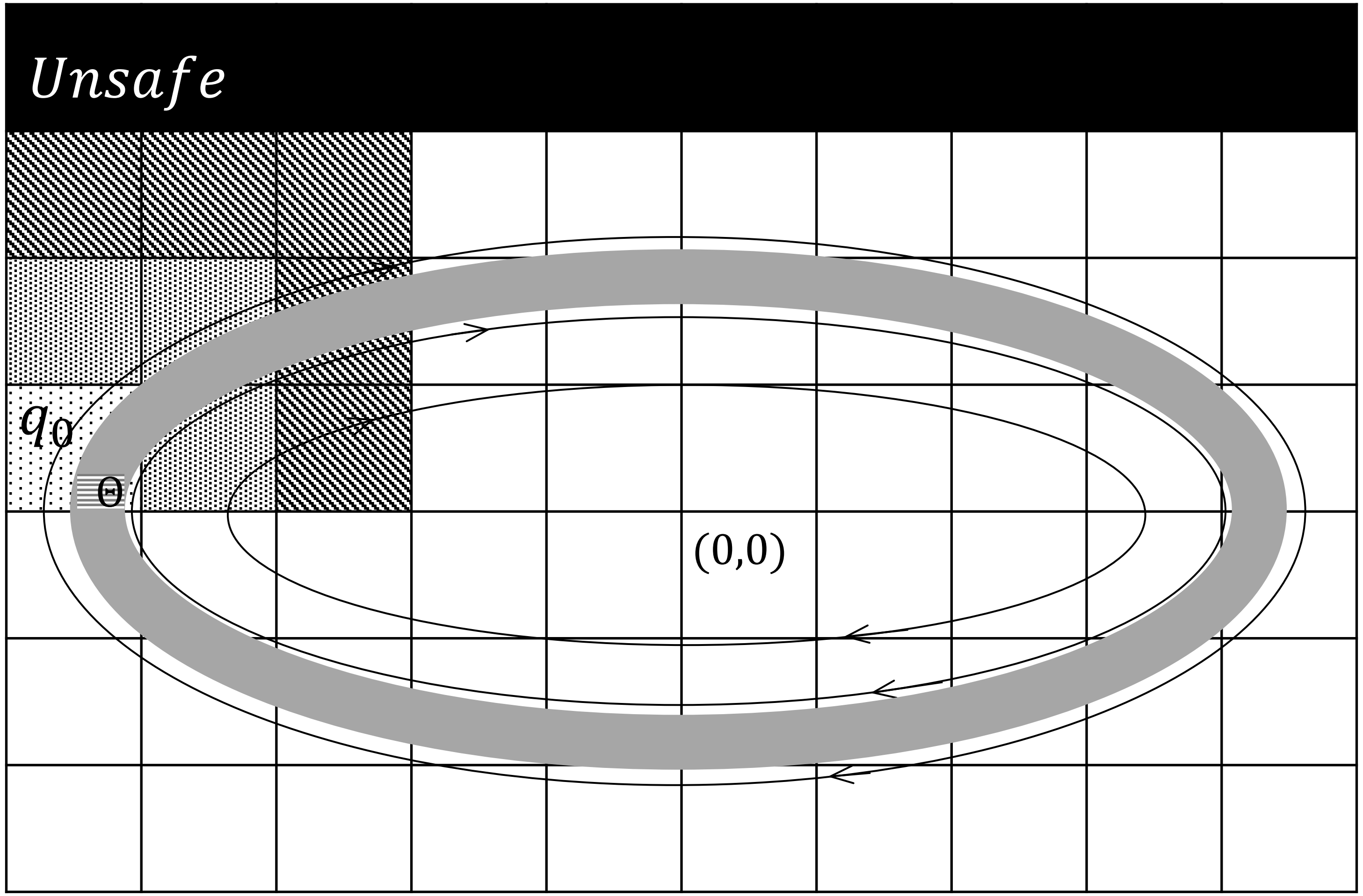
Idea of CEGAR

Key design choices

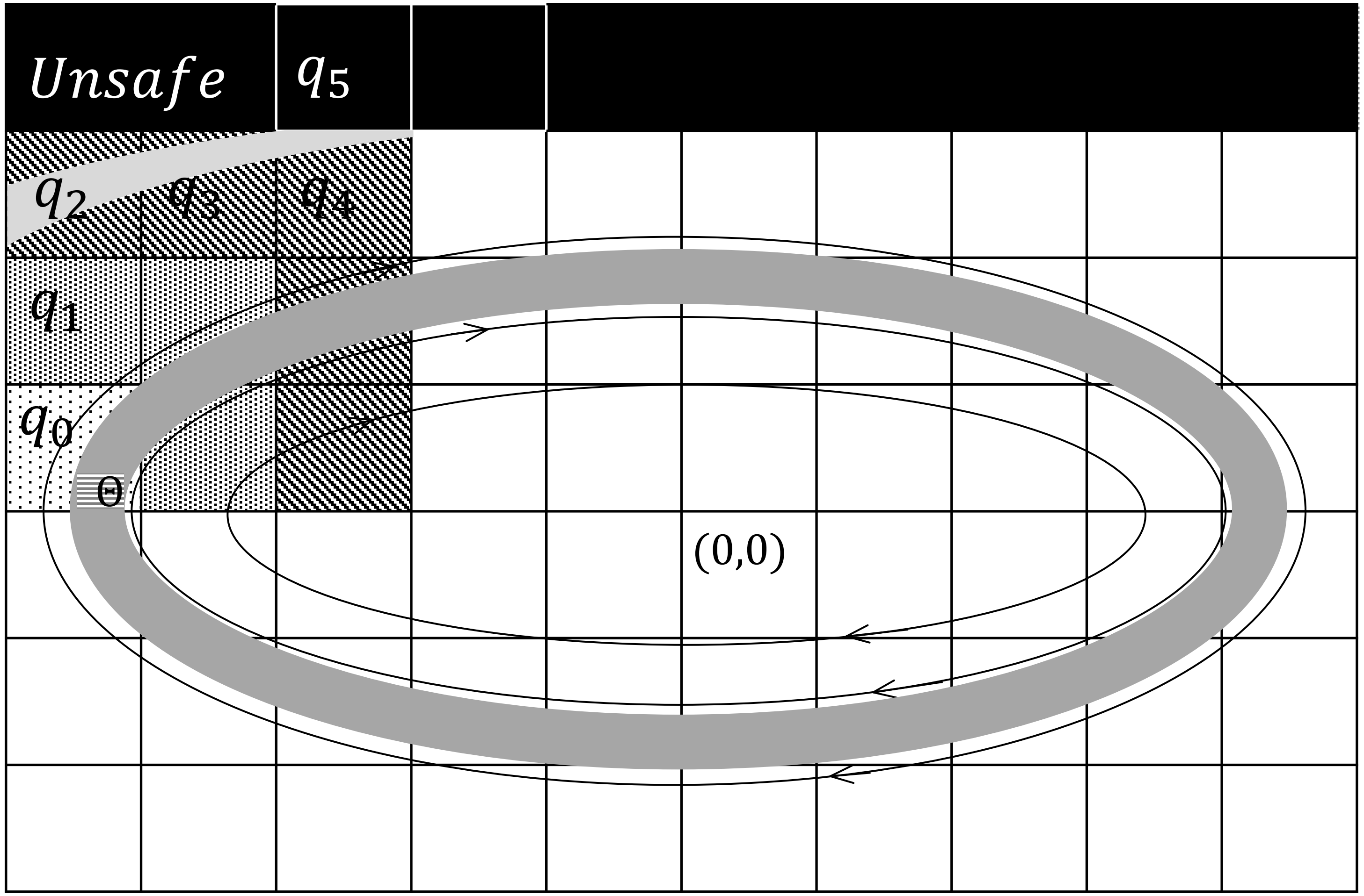
- Space of the abstract automata (finite, timed, linear)
- Model checker for abstract automaton
- Counter-example validation procedure
- Refinement strategy



Unsafe
q0
Θ
(0,0)



$Unsafe$
q_0
Θ
(0,0)



$$S_4 = Pre_A(S_5) \cap R^{-1}(q_4) \neq \emptyset$$
$$S_3 = Pre_A(S_4) \cap R^{-1}(q_3) \neq \emptyset$$
$$S_2 = Pre_A(S_3) \cap R^{-1}(q_2) \neq \emptyset$$
$$S_1 = Pre_A(S_2) \cap R^{-1}(q_1) = \emptyset$$