Counterexample-Guided Abstraction Refinement for Hybrid Systems

Edmund Clarke, Ansgar Fehnker, Zhi Han,
Bruce Krogh, Olaf Stursberg and Michael Theobald

Presenter: Joël Ouaknine

Outline

- Introduction
- Background: CEGAR
  Counterexample-Guided Abstraction Refinement
- Hybrid Systems
- CEGAR for Hybrid Systems
  - Abstractions
  - Validation and Refinement
  - Multiple Validation and Refinement Schemes
- Example
  - Car Steering Example
  - Cruise Control
- Related Work
- Future Work

Introduction

- Hybrid Systems:
  - Include continuous and discrete state variables
  - Model embedded systems
  - Applications: cars, robots, chemical plants,...
- Key Challenge:
  - Verification of properties of Hybrid System models
- Common Approach:
  - Employ abstraction to reduce complexity
  - Finding a good abstraction is difficult …
- This Talk:
  - Automated search for a good abstraction

Preliminaries

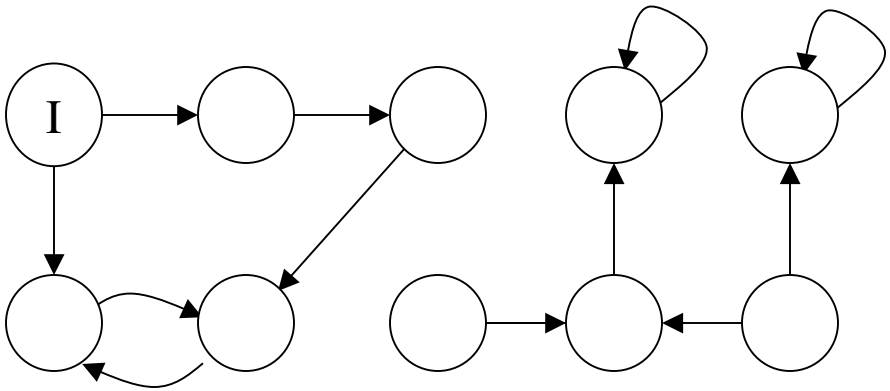
Basic idea of Counterexample-Guided Abstraction Refinement (CEGAR)

1. Given a "complex" ***transition system*** **C**
2. Generate an initial ***abstraction*** **A**
3. Refine **A** as long as it contains ***spurious counterexamples***
4. Use information from counterexample for ***refinement*** of **A**

Preliminaries

Transition system:

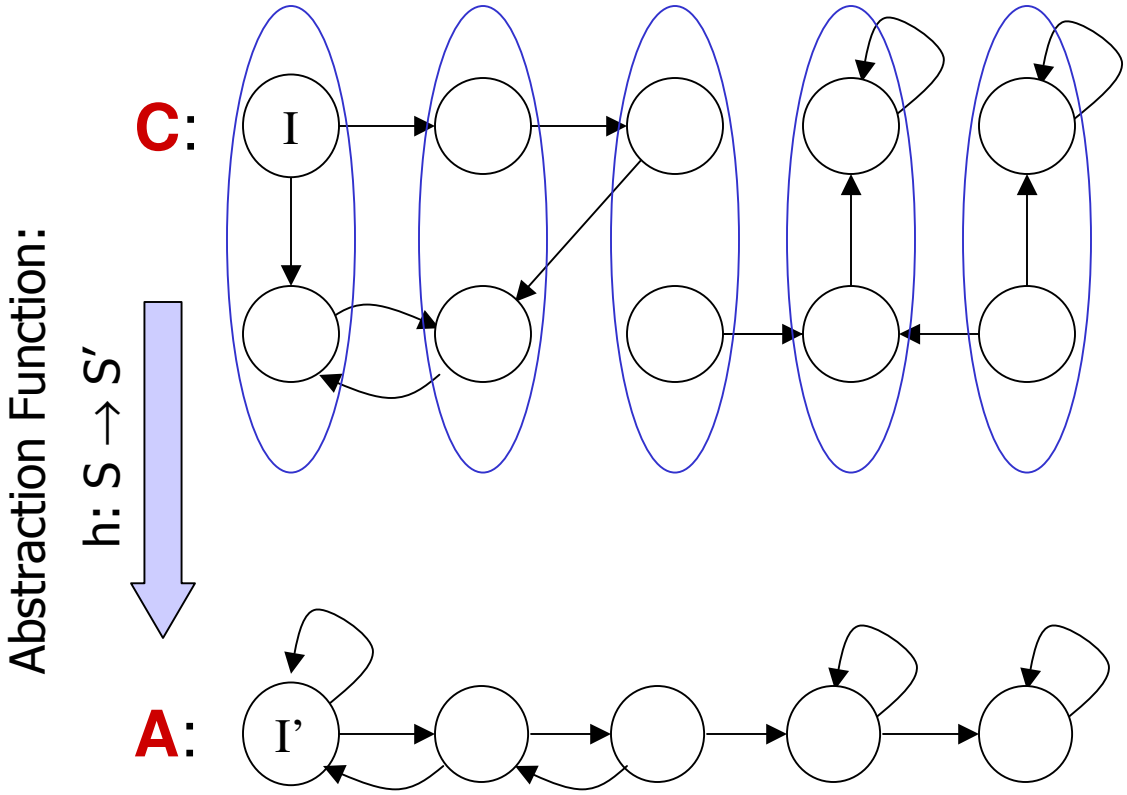
C:



- set of states S
- set of initial states $I \subseteq S$
- transition relation $R \subseteq S \times S$
- transition system T = (S, I, R)

Preliminaries

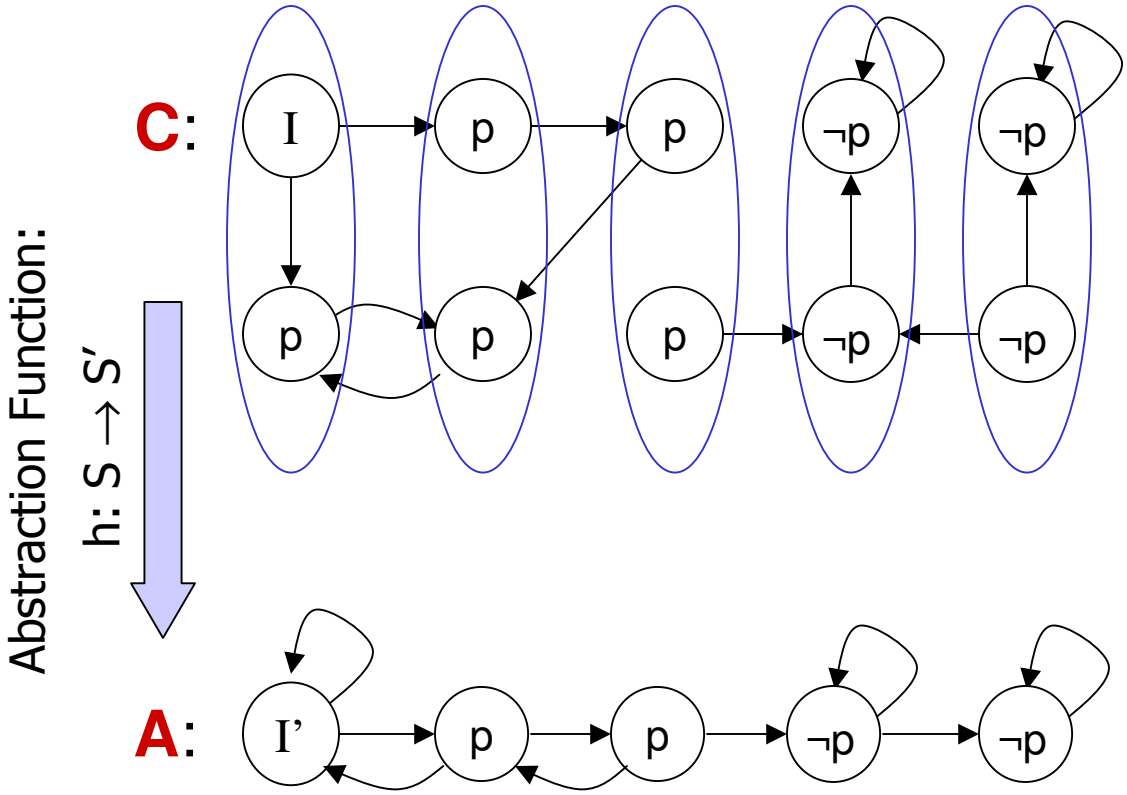
Abstraction: Maps a model onto a less complex model



- set of states S
- set of initial states $I \subseteq S$
- transition relation $R \subseteq S \times S$
- transition system T = (S, I, R)

Preliminaries

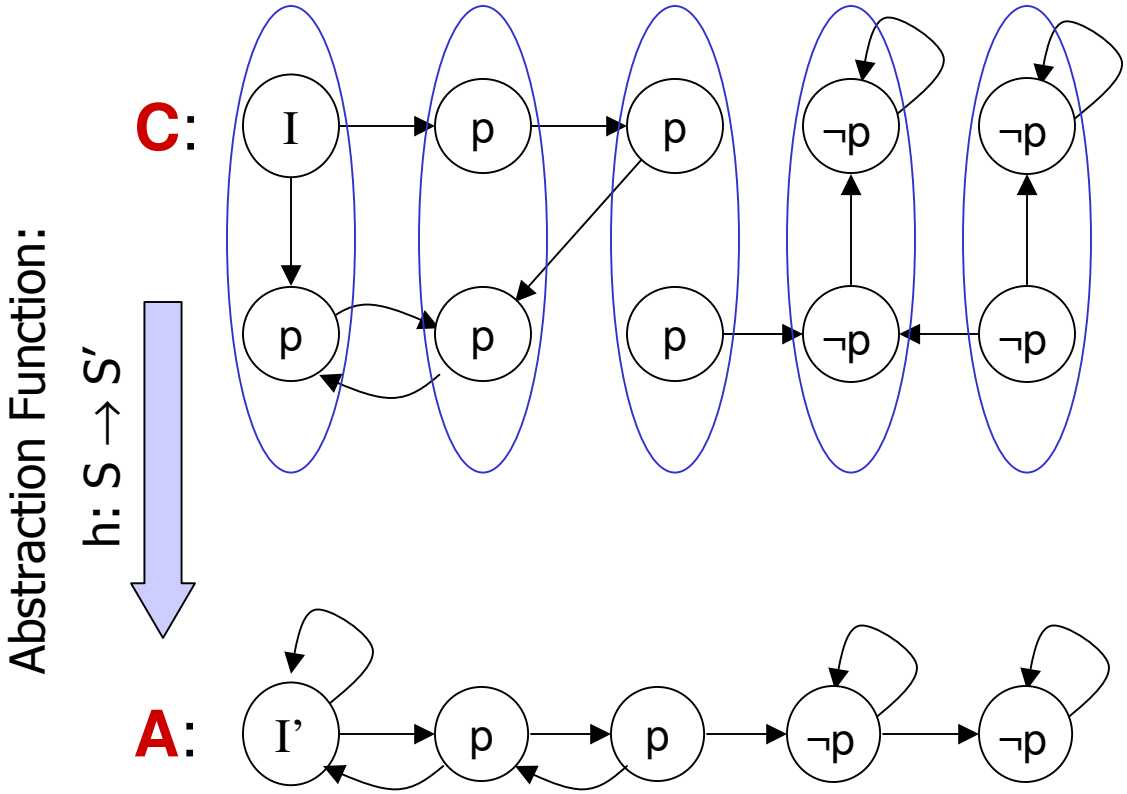
Model Checking: **AG p**



- set of states S
- set of initial states $I \subseteq S$
- transition relation $R \subseteq S \times S$
- transition system T = (S, I, R)

Preliminaries

Model Checking: **AG p**



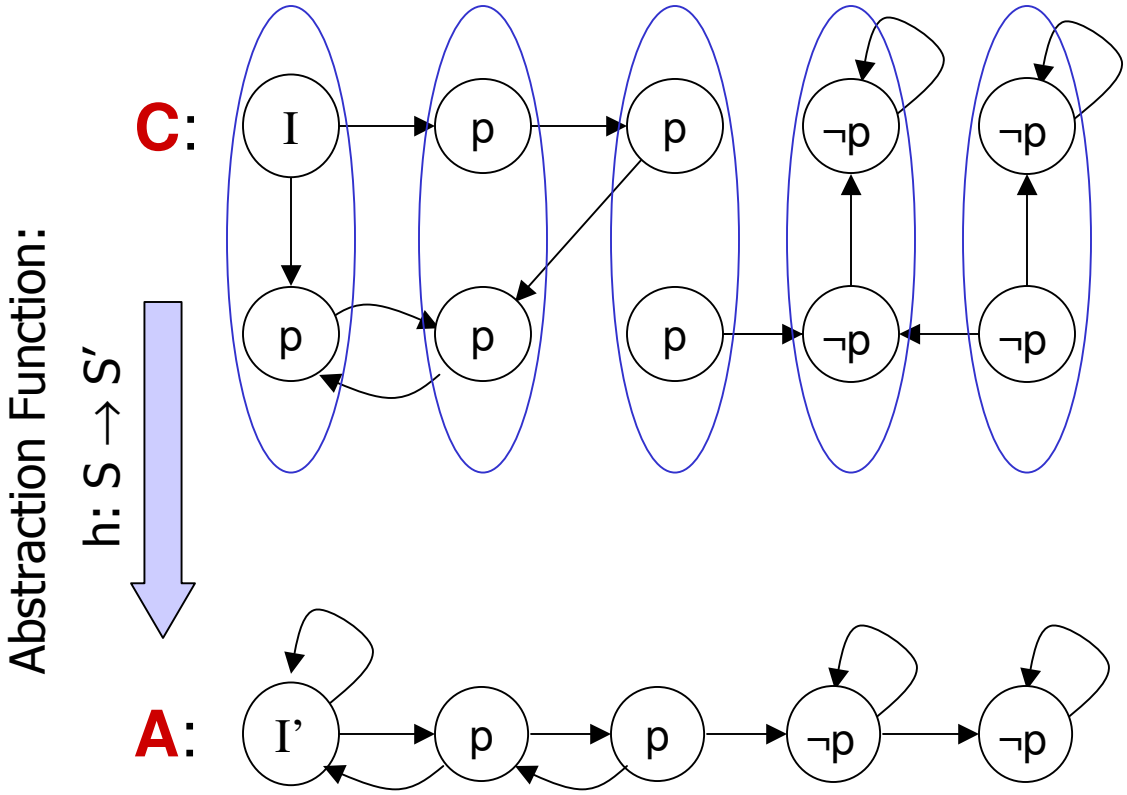
- set of states S
- set of initial states $I \subseteq S$
- transition relation $R \subseteq S \times S$
- transition system T = (S, I, R)

Preservation Theorem:

AG p holds for **A** $\Rightarrow$ AG p holds for **C**

Preliminaries

Model Checking: **AG p**



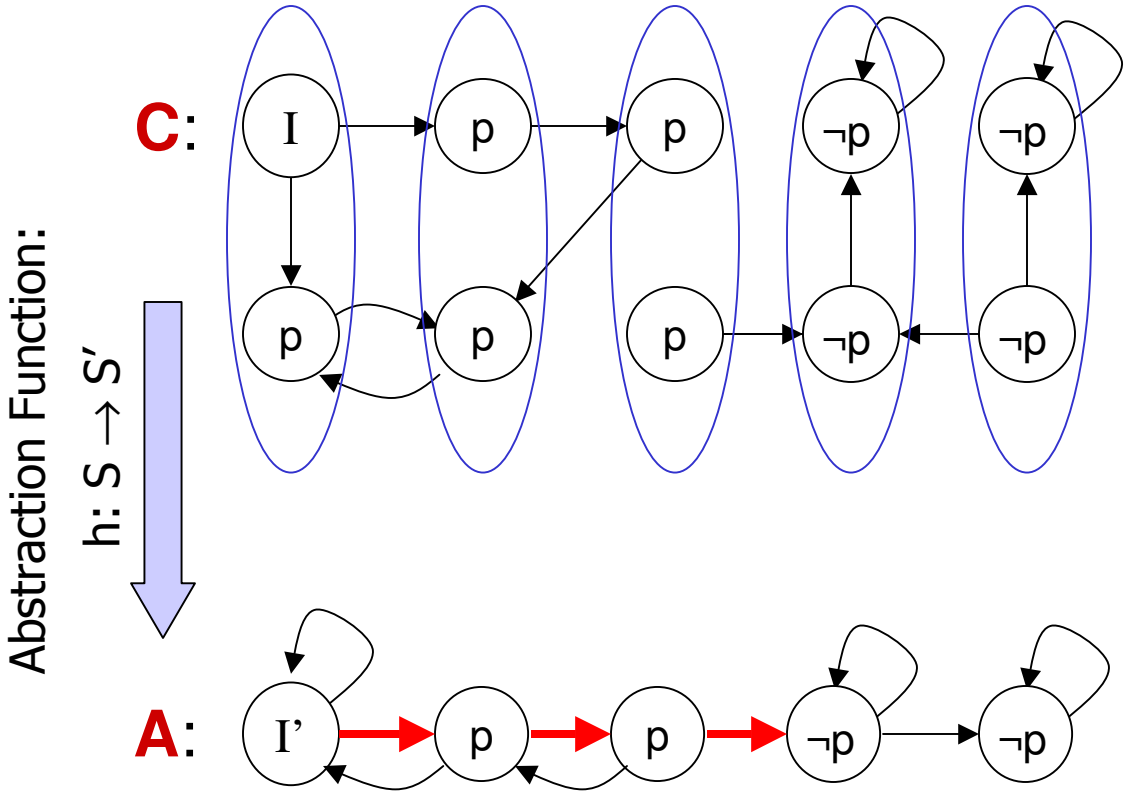
- set of states S
- set of initial states $I \subseteq S$
- transition relation $R \subseteq S \times S$
- transition system T = (S, I, R)

Converse of Preservation Theorem:

AG p holds for **C** $\not\Rightarrow$ AG p holds for **A**

Preliminaries

Model Checking: **AG p**



- set of states S
- set of initial states $I \subseteq S$
- transition relation $R \subseteq S \times S$
- transition system T = (S, I, R)

Counterexample in A may be spurious

$\Rightarrow$ ***Refine abstraction***

Converse of Preservation Theorem:

AG p holds for **C** $\not\Rightarrow$ AG p holds for **A**

Preliminaries

Counterexample-Guided Abstraction Refinement

1. Generate an initial abstraction A
2. Model check property p for A.
   If p holds for A, then RETURN true.
3. Check the counterexample for concrete model C.
   If counterexample is not spurious, then RETURN false.
4. Refine A, and go to step 2.

Outline

- Introduction
- Background: CEGAR
  Counterexample-Guided Abstraction Refinement
- Hybrid Systems
- CEGAR for Hybrid Systems
  - Abstractions
  - Validation and Refinement
  - Multiple Validation and Refinement Schemes
- Example
  - Car Steering Example
  - Cruise Control
- Related Work
- Future Work

Hybrid Systems

- Systems with a non-trivial interaction between continuous and discrete components
- Hybrid automata model
  - Discrete behavior defined by finite number of locations
  - Continuous behavior in each location defined by differential equations
- Hybrid system verification
  - Reachability algorithms
  - Bisimulation algorithms
  - Tools: CheckMate, ddt/Charon, Hytech, …

Hybrid Systems

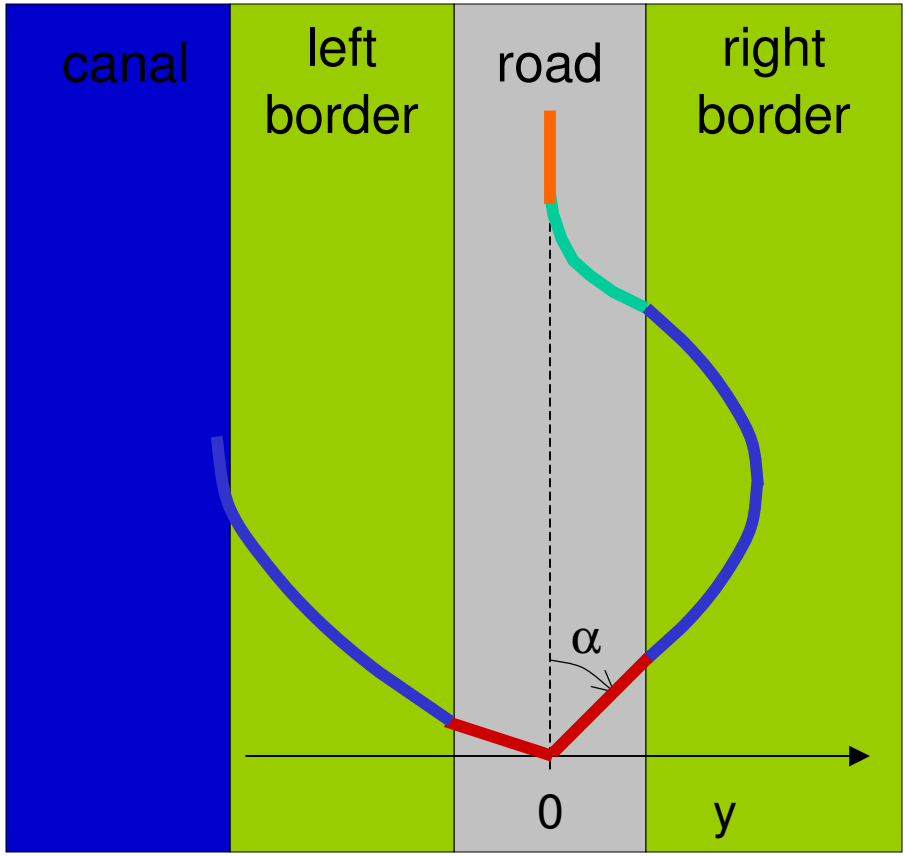
Example: Car Steering Example

Control strategy:

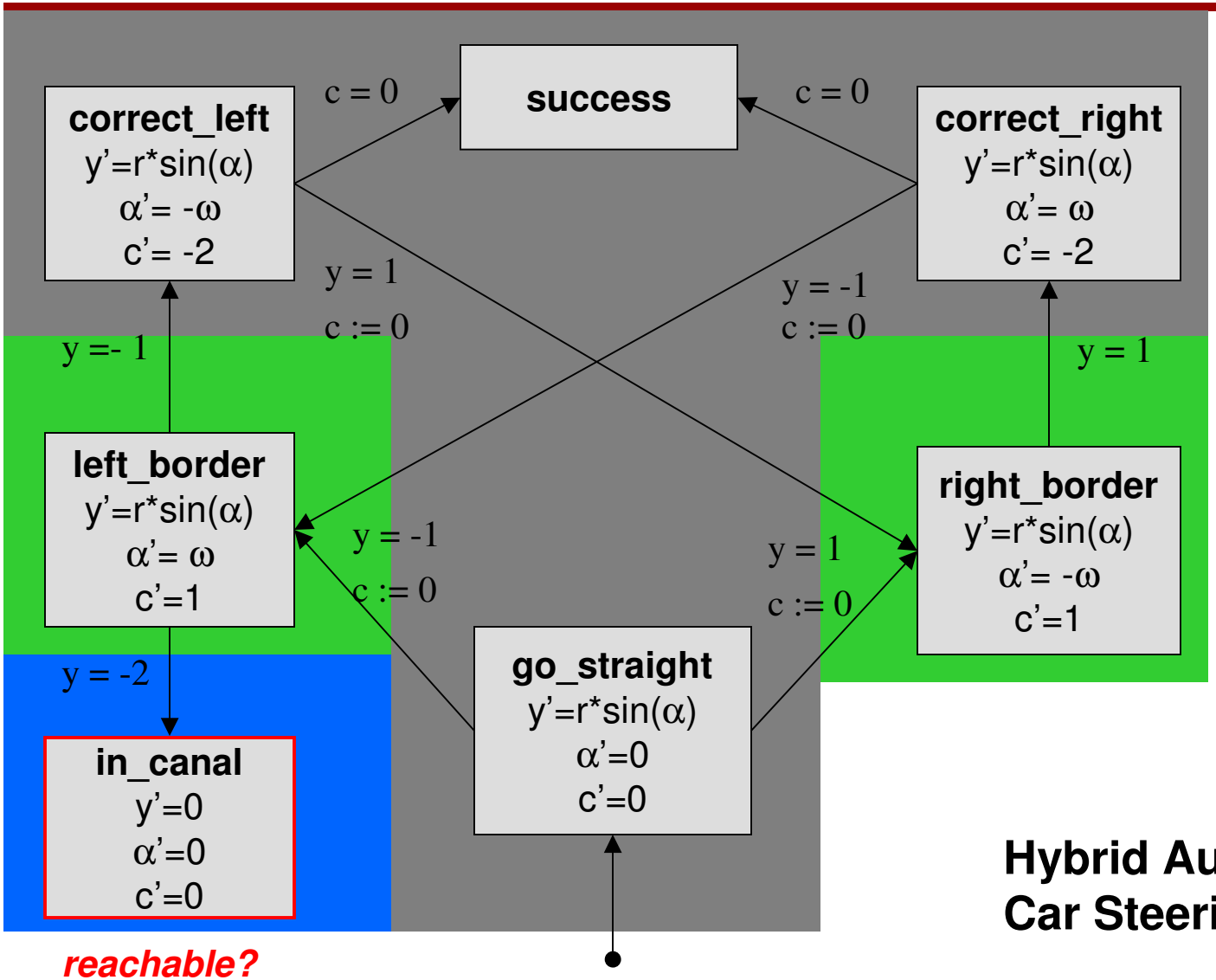
- correct the direction (α) if a border is reached
- after returning to the road, steer to $\alpha = 0$

Safety requirement

- Canal should not be reachable



Hybrid Systems



Variables:

y : deviation
α : angle
c : multi-rate clock

Parameters:

r: velocity
ω: angular velocity

Hybrid Automaton for the Car Steering Example

reachable?

$y_{init} \in [-1, 1]$, $\alpha_{init} \in [-\pi/4, \pi/4]$, $c_{init} = 0$
$r = 2$, $\omega = \pi/4$

Outline

- Introduction
- Background: CEGAR
  Counterexample-Guided Abstraction Refinement
- Hybrid Systems
- CEGAR for Hybrid Systems
  - Abstractions
  - Validation and Refinement
  - Multiple Validation and Refinement Schemes
- Example
  - Car Steering Example
  - Cruise Control
- Related Work
- Future Work

CEGAR for Hybrid Systems

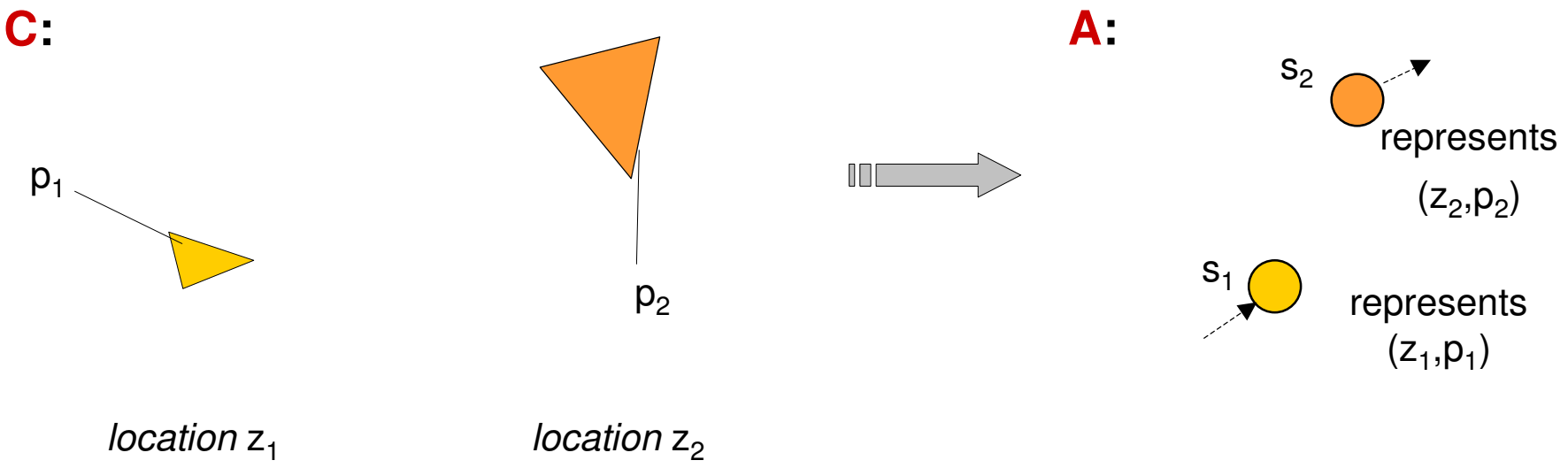
- Abstraction into finite-state model
  - Initial abstraction
- Validation and Refinement
  - Local, i.e. only where necessary
  - Splitting of states, removal of transitions,…
- Multiple Validation and Refinement Schemes
  - Over-approximation methods

CEGAR: Abstraction for Hybrid Systems

Concrete Model: Hybrid Automaton (infinite state)

Abstract Model: Finite State Transition System

Partition Element: Pair consists of control location and (union of) polyhedra



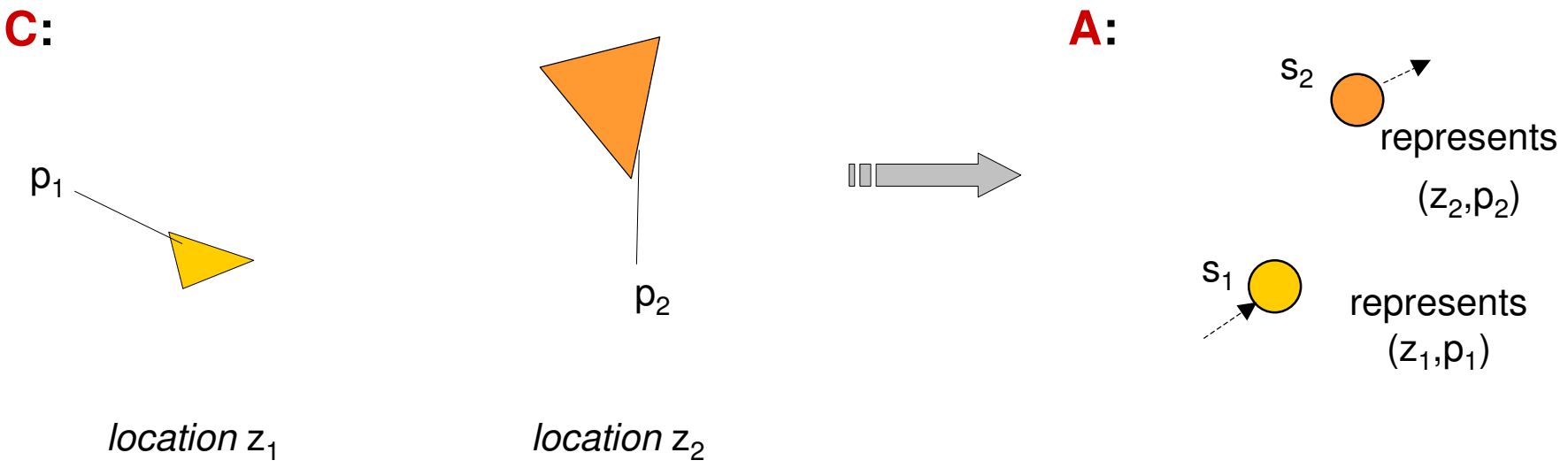
CEGAR: Abstraction for Hybrid Systems

Concrete Model: Hybrid Automaton (infinite state)

Abstract Model: Finite State Transition System

Partition Element: Pair consists of control location and (union of) polyhedra

Abstraction: There exists a transition in **A** **if** there exists a corresponding transition in **C**



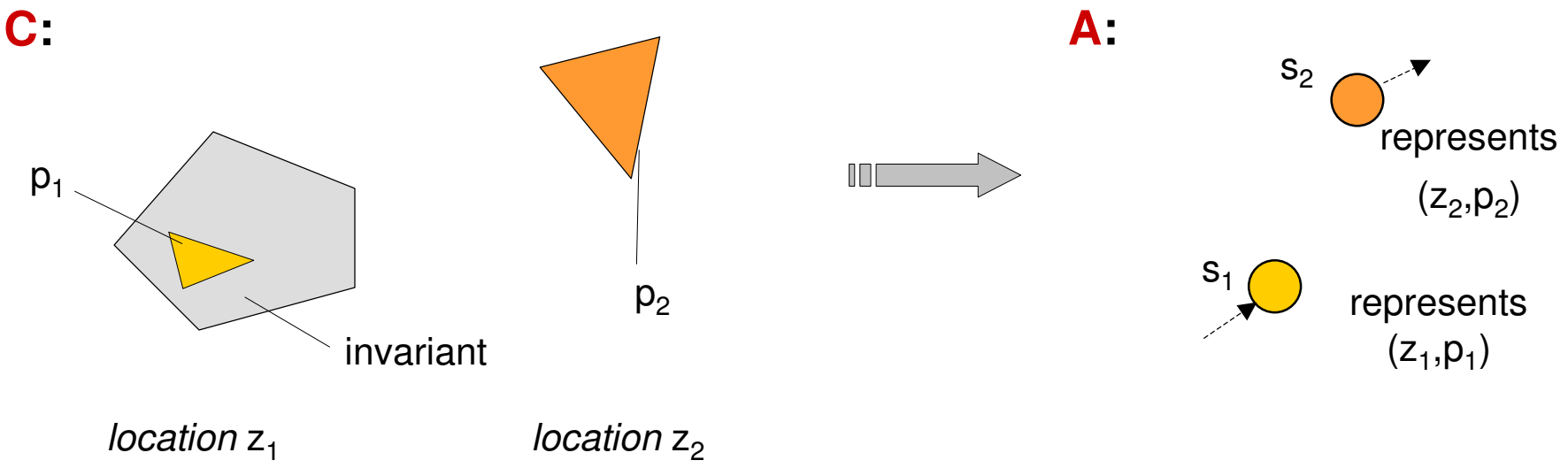
CEGAR: Abstraction for Hybrid Systems

Concrete Model: Hybrid Automaton (infinite state)

Abstract Model: Finite State Transition System

Partition Element: Pair consists of control location and (union of) polyhedra

Abstraction: There exists a transition in **A** **if** there exists a corresponding transition in **C**



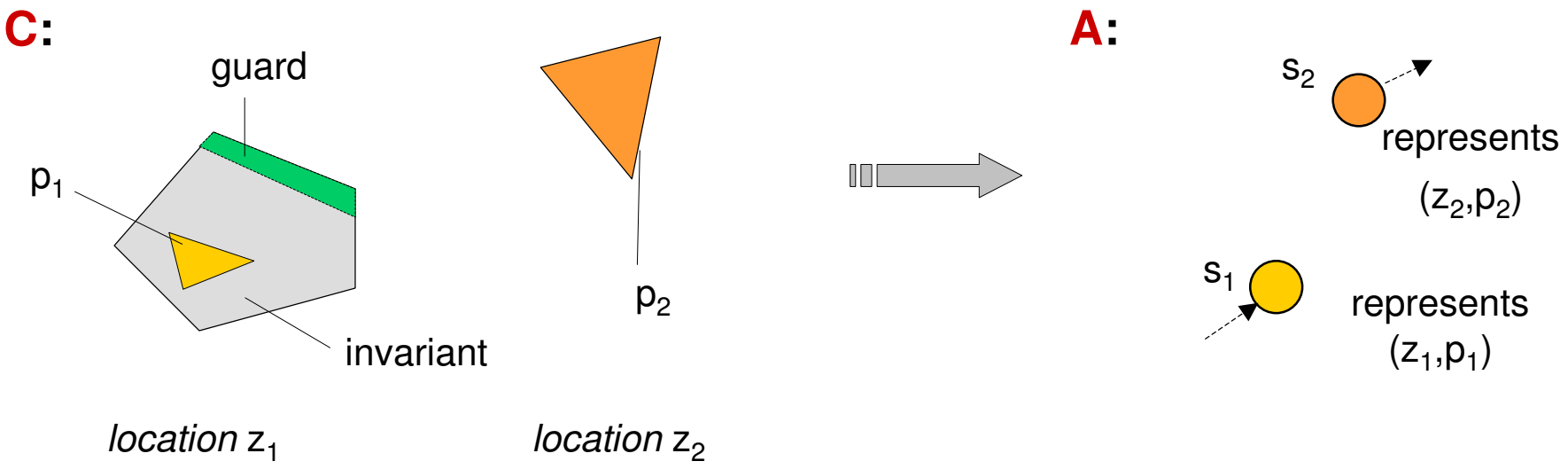
CEGAR: Abstraction for Hybrid Systems

Concrete Model: Hybrid Automaton (infinite state)

Abstract Model: Finite State Transition System

Partition Element: Pair consists of control location and (union of) polyhedra

Abstraction: There exists a transition in **A** **if** there exists a corresponding transition in **C**



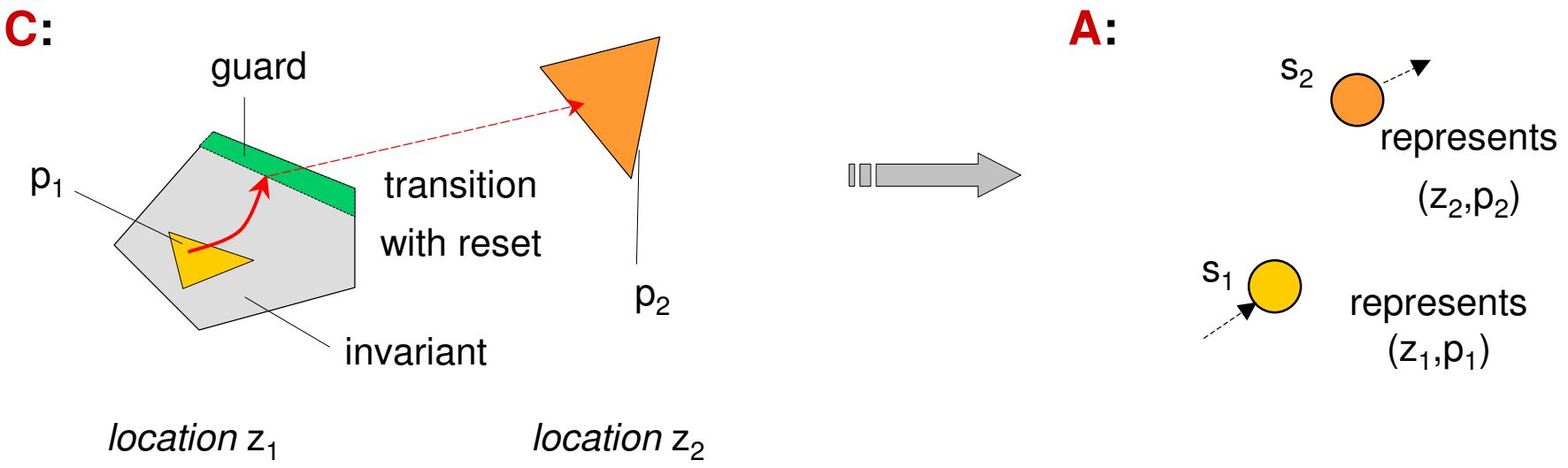
CEGAR: Abstraction for Hybrid Systems

Concrete Model: Hybrid Automaton (infinite state)

Abstract Model: Finite State Transition System

Partition Element: Pair consists of control location and (union of) polyhedra

Abstraction: There exists a transition in **A** **if** there exists a corresponding transition in **C**



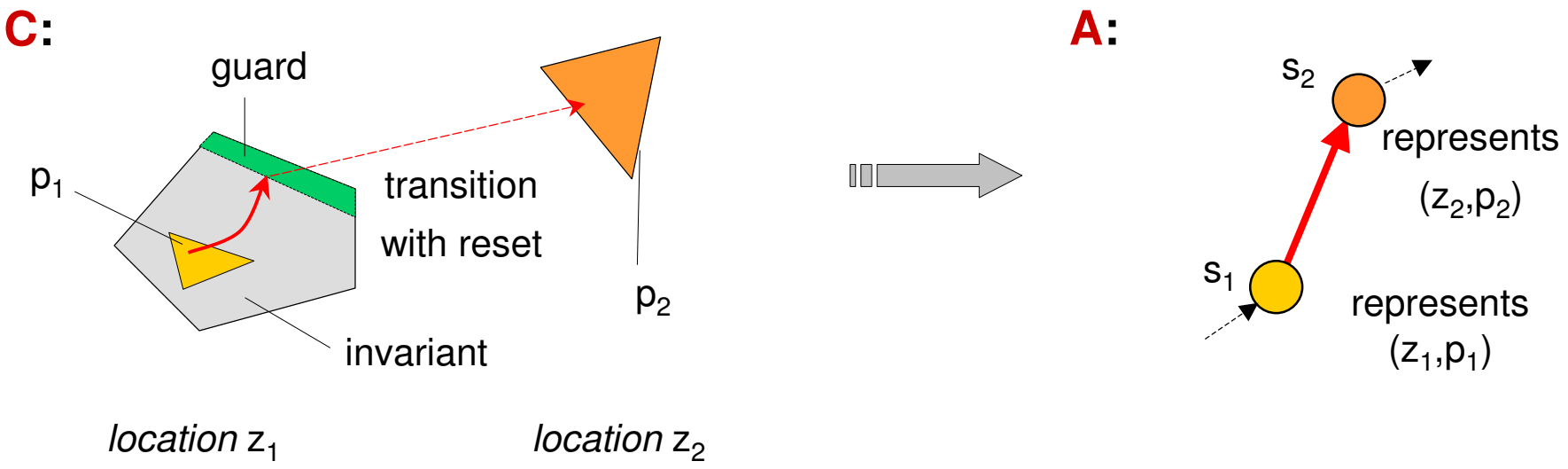
CEGAR: Abstraction for Hybrid Systems

Concrete Model: Hybrid Automaton (infinite state)

Abstract Model: Finite State Transition System

Partition Element: Pair consists of control location and (union of) polyhedra

Abstraction: There exists a transition in **A** **if** there exists a corresponding transition in **C**



CEGAR: Abstraction for Hybrid Systems

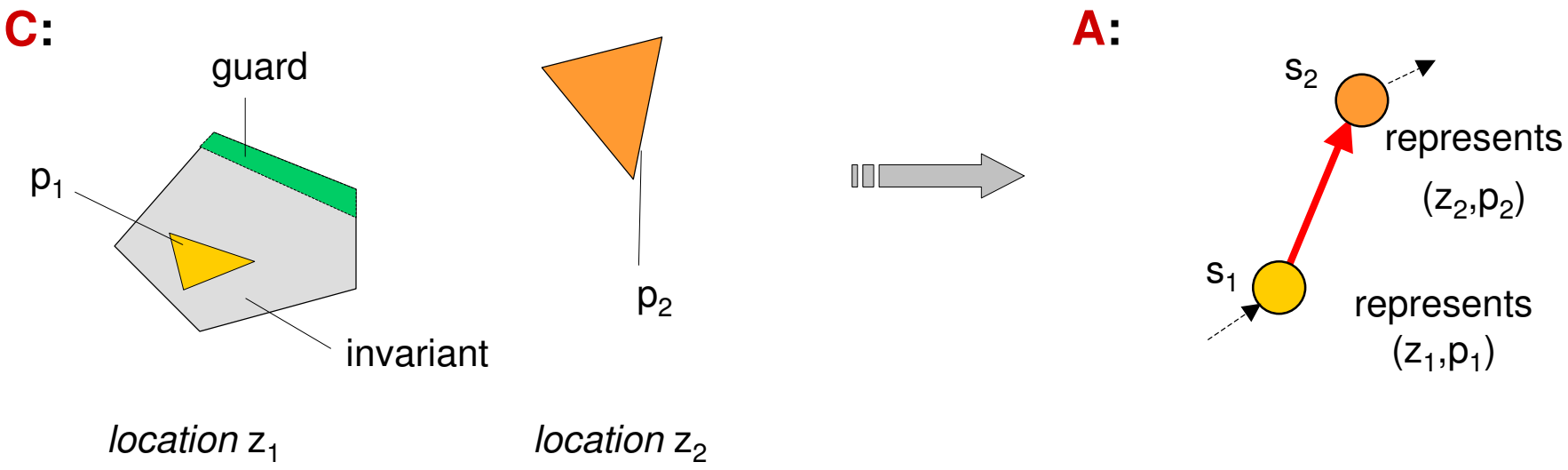
Concrete Model: Hybrid Automaton (infinite state)

Abstract Model: Finite State Transition System

Partition Element: Pair consists of control location and (union of) polyhedra

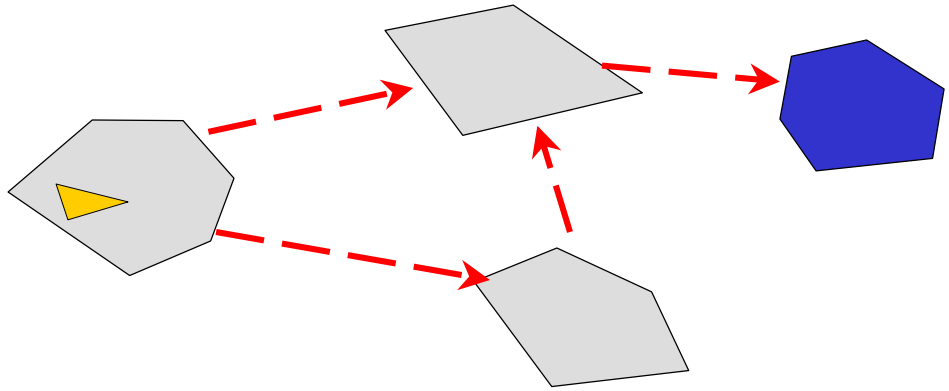
Abstraction: There exists a transition in **A** **if** there exists a corresponding transition in **C**

Spurious Transition: A transition in **A** with no corresponding transition in **C**



CEGAR: Validation and Refinement

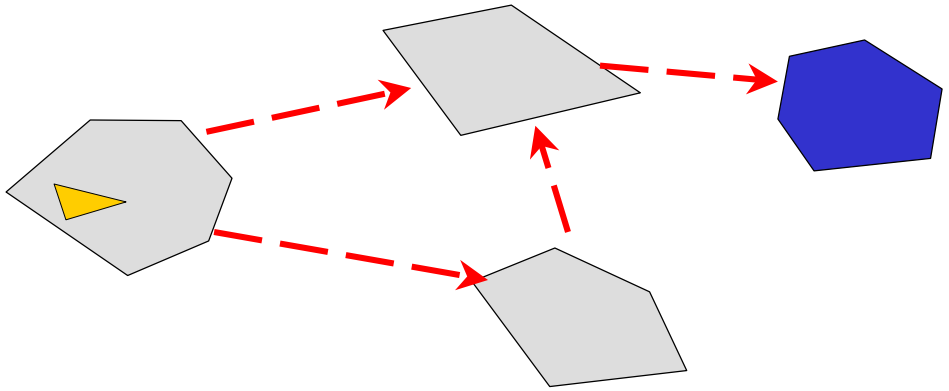
Verification Problem: **Given:** initial location + set and **bad location**
Verify: **bad location** can never be reached

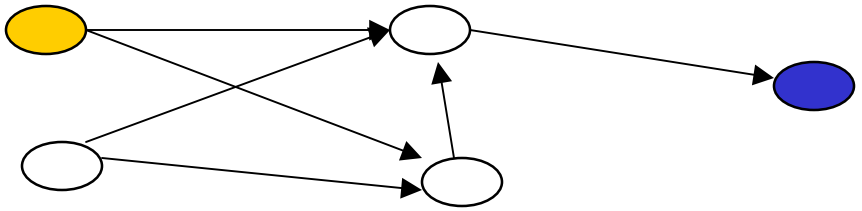


CEGAR: Validation and Refinement

Verification Problem: **Given: initial location + set** and **bad location**
Verify: bad location can never be reached

STEP1: Build initial abstraction



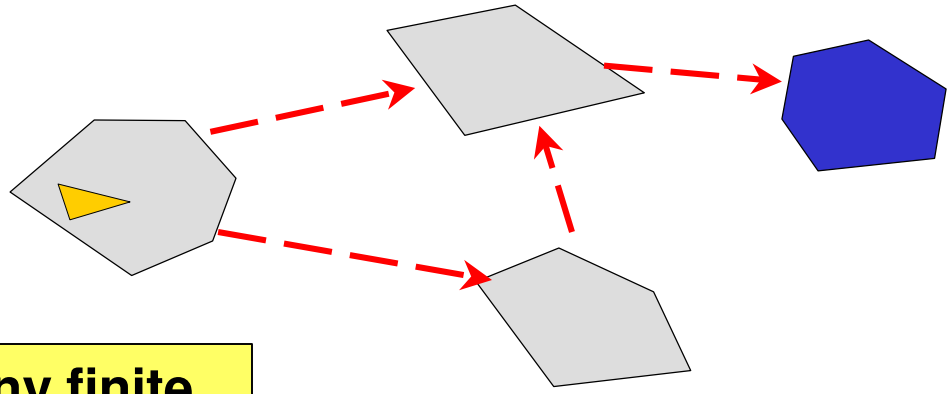


CEGAR: Validation and Refinement

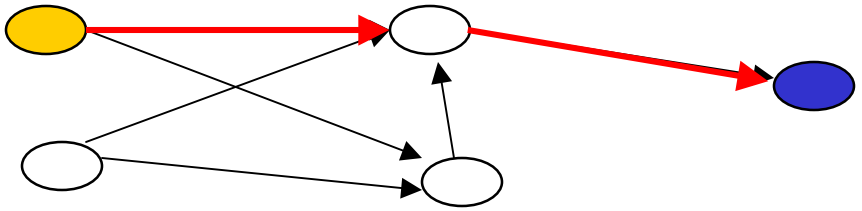
Verification Problem: **Given:** initial location + set and **bad location**
Verify: **bad location** can never be reached

STEP1: Build initial abstraction

STEP2: find **CE** in abstraction



Use any finite state model checker



CEGAR: Validation and Refinement

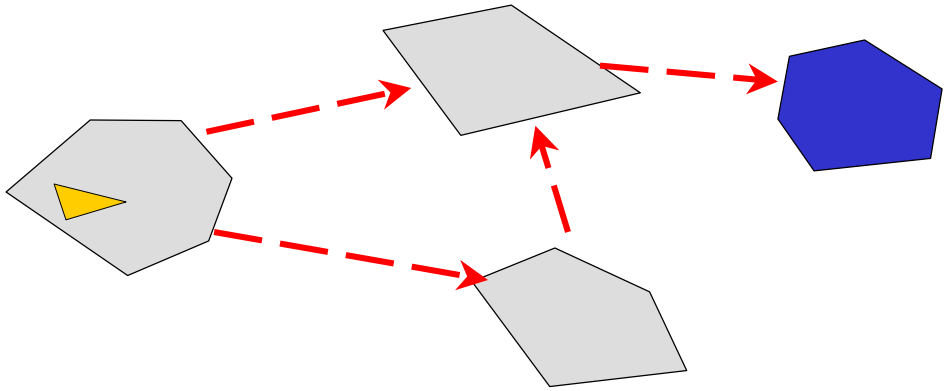
Verification Problem: **Given:** initial location + set and **bad location**
Verify: **bad location** can never be reached

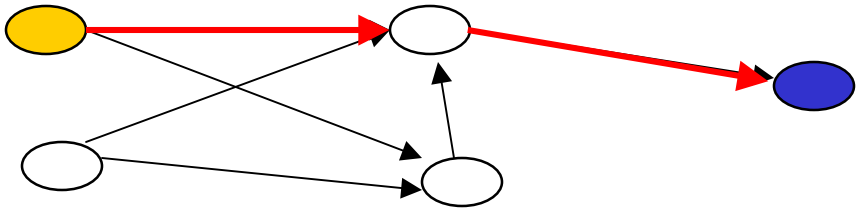
STEP1: Build initial abstraction

STEP2: find **CE** in abstraction

STEP3: for each transition in **CE**

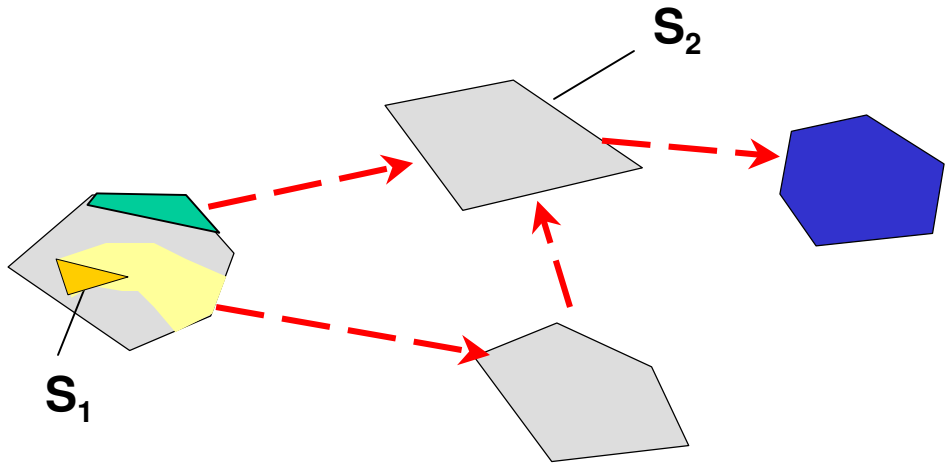
- validate transition
- refine abstraction





CEGAR: Validation and Refinement

Verification Problem: **Given:** initial location + set and **bad location**
Verify: **bad location** can never be reached



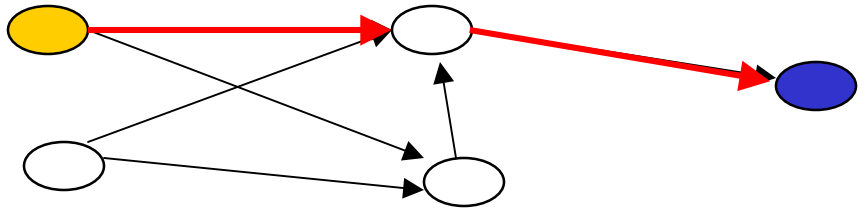
STEP1: Build initial abstraction

STEP2: find **CE** in abstraction

STEP3: for each transition in **CE**

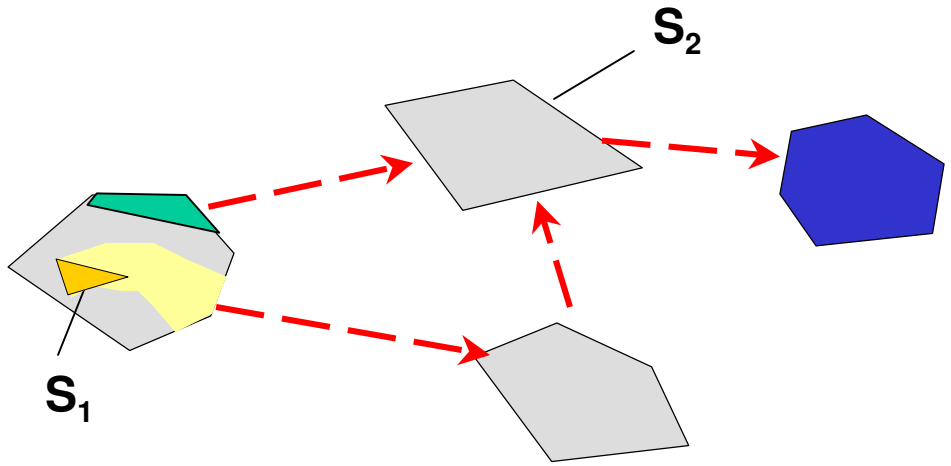
- validate transition
- refine abstraction

Case 1: S_2 is not reachable



CEGAR: Validation and Refinement

Verification Problem: **Given:** initial location + set and **bad location**
Verify: **bad location** can never be reached



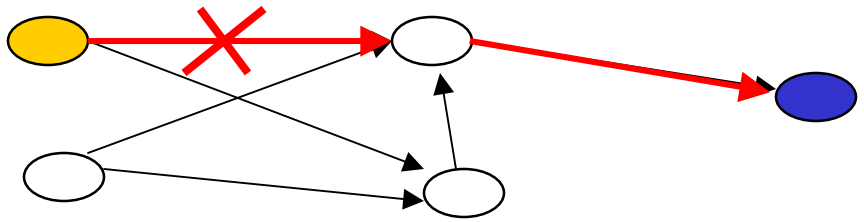
STEP1: Build initial abstraction

STEP2: find **CE** in abstraction

STEP3: for each transition in **CE**

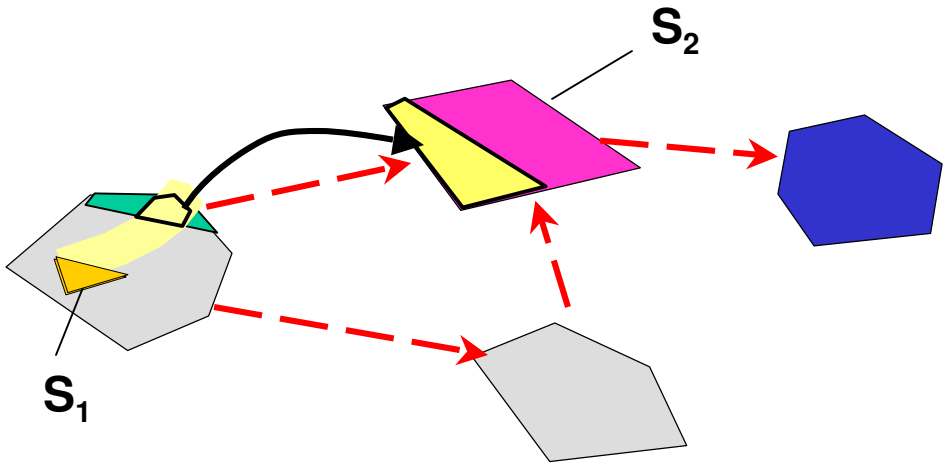
- validate transition
- refine abstraction

Case 1: S_2 is not reachable



CEGAR: Validation and Refinement

Verification Problem: **Given:** initial location + set and **bad location**
Verify: **bad location** can never be reached



STEP1: Build initial abstraction

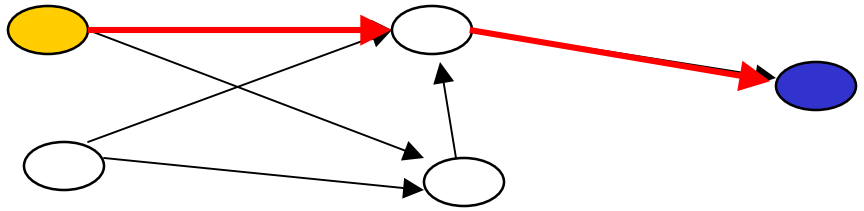
STEP2: find **CE** in abstraction

STEP3: for each transition in **CE**

- validate transition
- refine abstraction

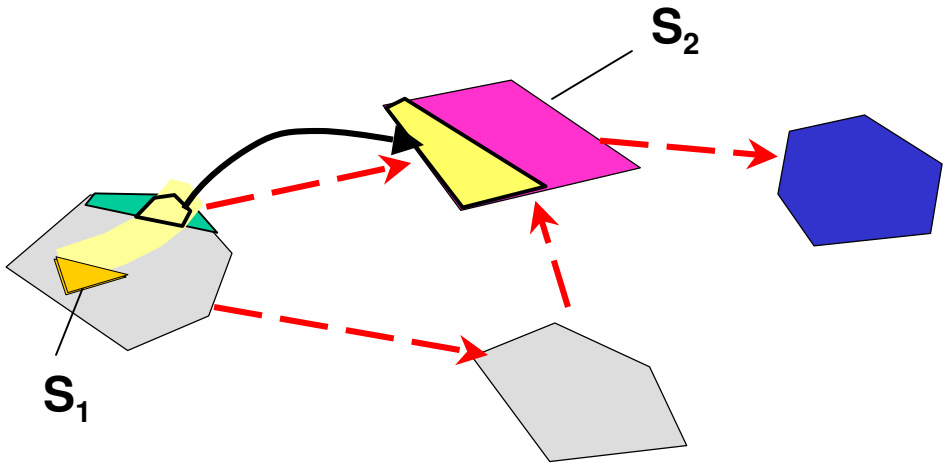
Case 1: S_2 is not reachable

Case 2: **Part** of S_2 is not reachable



CEGAR: Validation and Refinement

Verification Problem: **Given:** initial location + set and **bad location**
Verify: **bad location** can never be reached



STEP1: Build initial abstraction

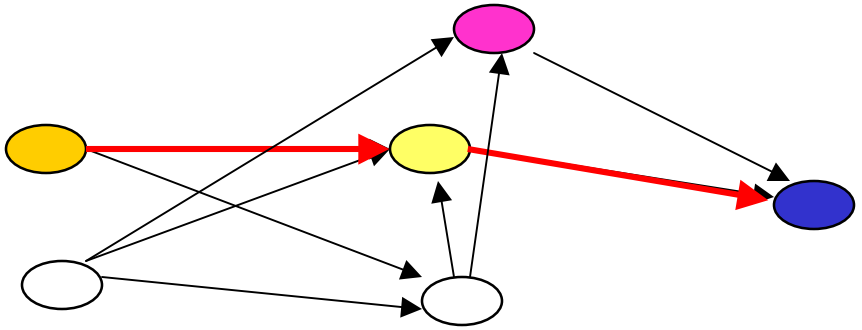
STEP2: find **CE** in abstraction

STEP3: for each transition in **CE**

- validate transition
- refine abstraction

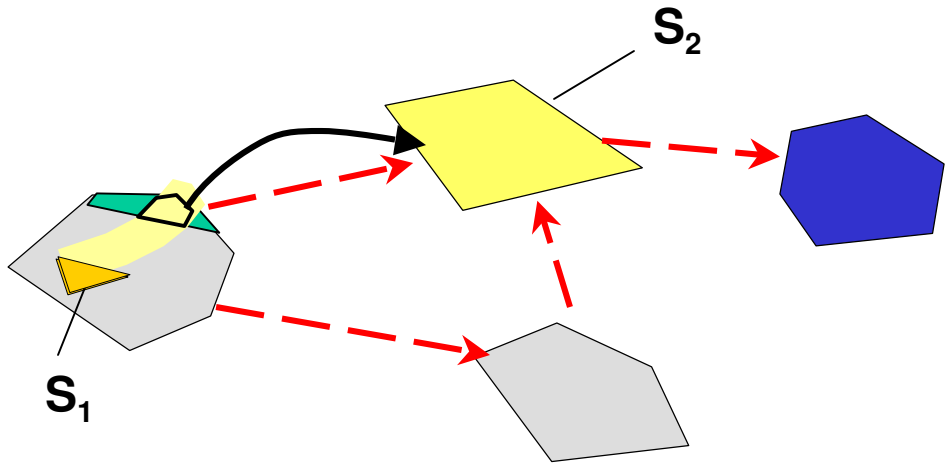
Case 1: S_2 is not reachable

Case 2: **Part** of S_2 is not reachable



CEGAR: Validation and Refinement

Verification Problem: **Given:** initial location + set and **bad location**
Verify: **bad location** can never be reached



STEP1: Build initial abstraction

STEP2: find **CE** in abstraction

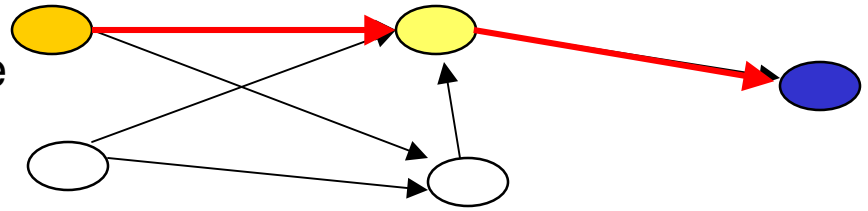
STEP3: for each transition in **CE**

- validate transition
- refine abstraction

Case 1: S_2 is not reachable

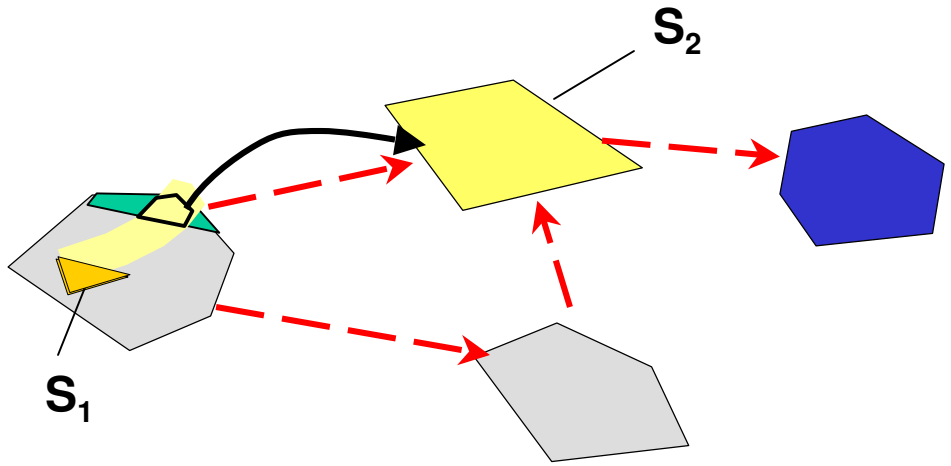
Case 2: **Part** of S_2 is not reachable

Case 3: All of S_2 may be reachable



CEGAR: Validation and Refinement

Verification Problem: **Given:** initial location + set and **bad location**
Verify: **bad location** can never be reached



STEP1: Build initial abstraction

STEP2: find **CE** in abstraction

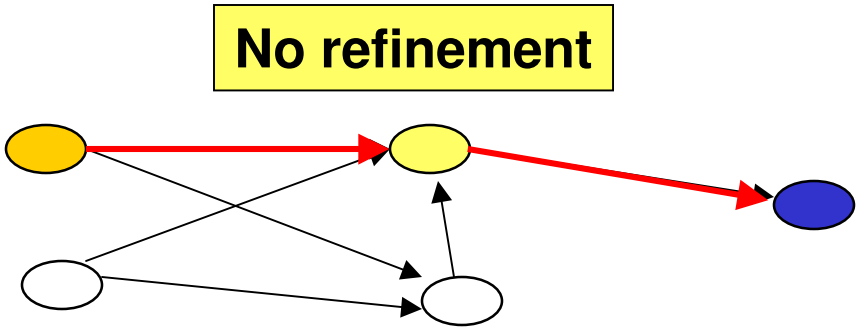
STEP3: for each transition in **CE**

- validate transition
- refine abstraction

Case 1: S_2 is not reachable

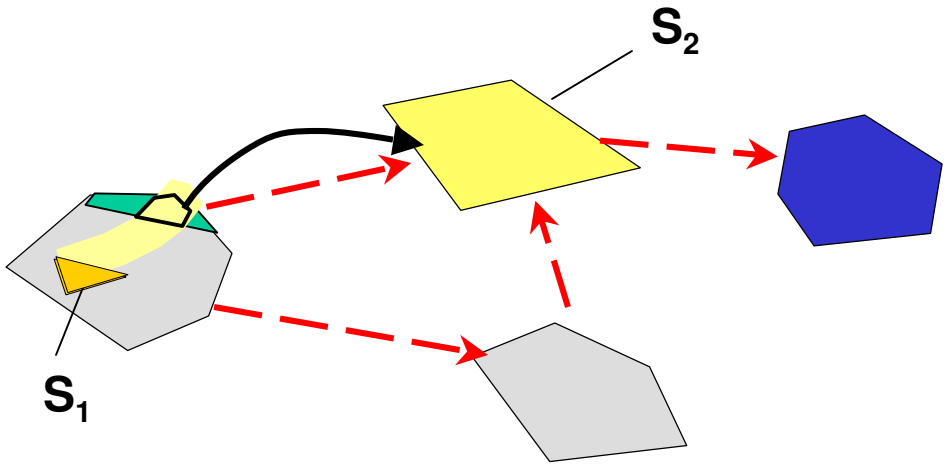
Case 2: **Part** of S_2 is not reachable

Case 3: All of S_2 may be reachable



CEGAR: Validation and Refinement

Verification Problem: **Given:** initial location + set and **bad location**
Verify: **bad location** can never be reached



STEP1: Build initial abstraction

STEP2: find **CE** in abstraction

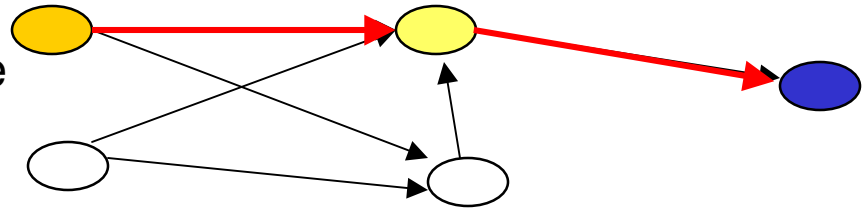
STEP3: for each transition in **CE**

- validate transition
- refine abstraction

Case 1: S_2 is not reachable

Case 2: **Part** of S_2 is not reachable

Case 3: All of S_2 may be reachable

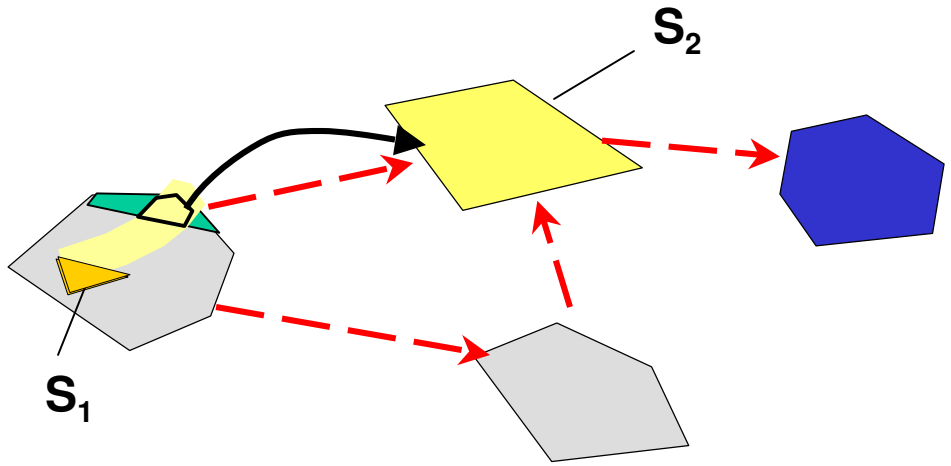


Case 2,3 $\Rightarrow$ next transition

Case 1 $\Rightarrow$ next **CE**

CEGAR: Validation and Refinement

Verification Problem: **Given:** initial location + set and **bad location**
Verify: **bad location** can never be reached



STEP1: Build initial abstraction

STEP2: find **CE** in abstraction

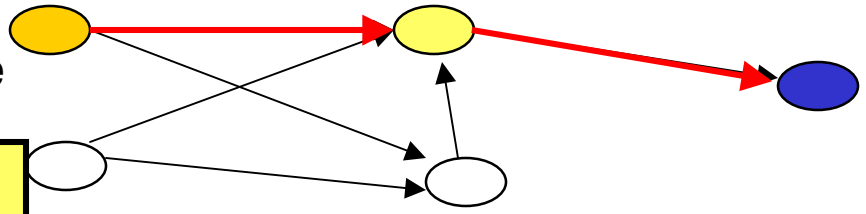
STEP3: for each transition in **CE**

- validate transition
- refine abstraction

Case 1: S_2 is not reachable

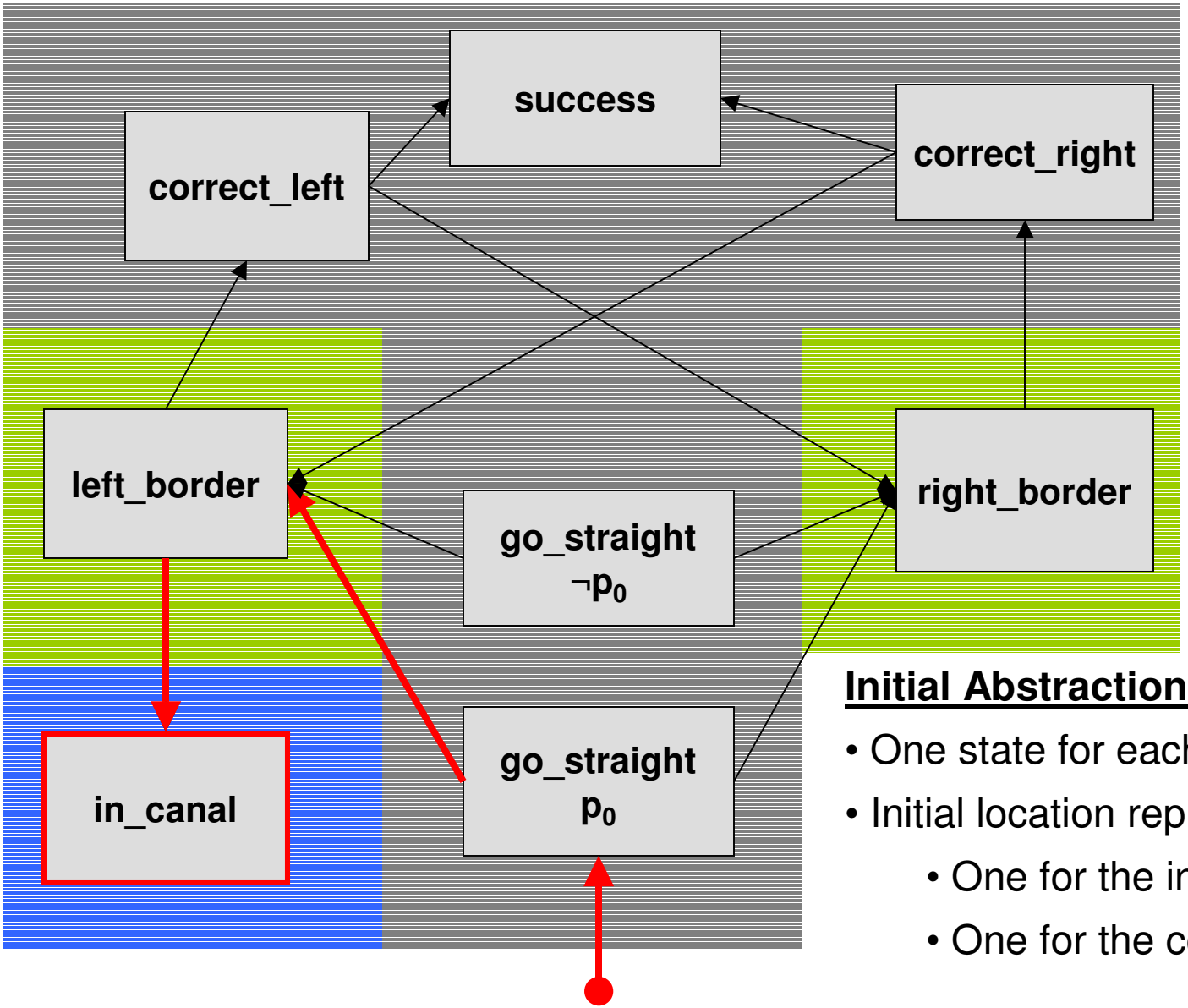
Case 2: **Part** of S_2 is not reachable

Case 3: All of S_2 may be reachable



If all transitions **valid** (case 2,3)
$\Rightarrow$ **CE valid**

CEGAR: Validation and Refinement

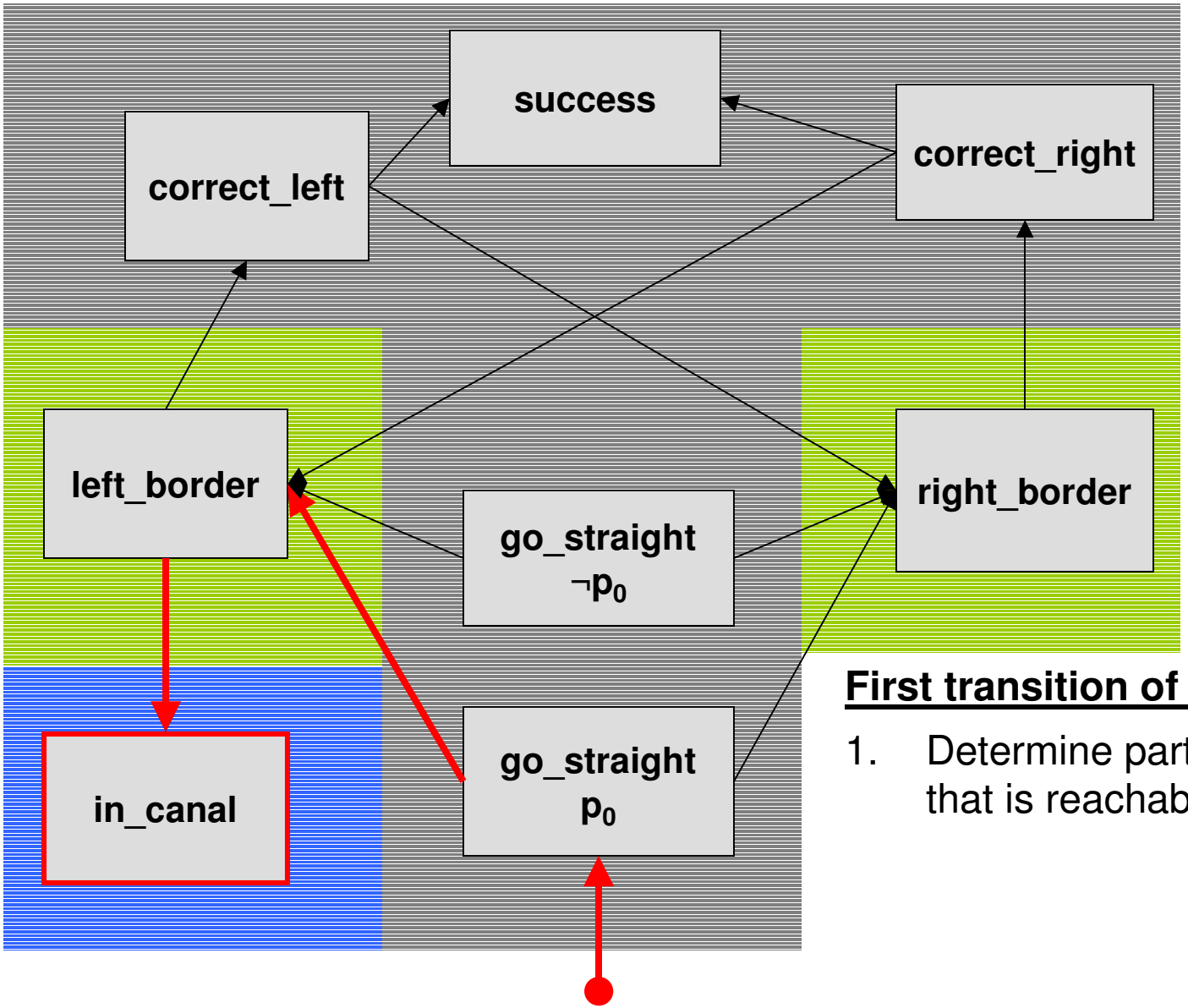


$p_0 = \{-1 < y < 1, c = 0, -\pi/4 < \alpha < \pi/4\}$

Initial Abstraction:

- One state for each location
- Initial location represented by 2 states
  - One for the initial set
  - One for the complement

CEGAR: Validation and Refinement



$p_0 = \{-1 < y < 1, c = 0, -\pi/4 < \alpha < \pi/4\}$

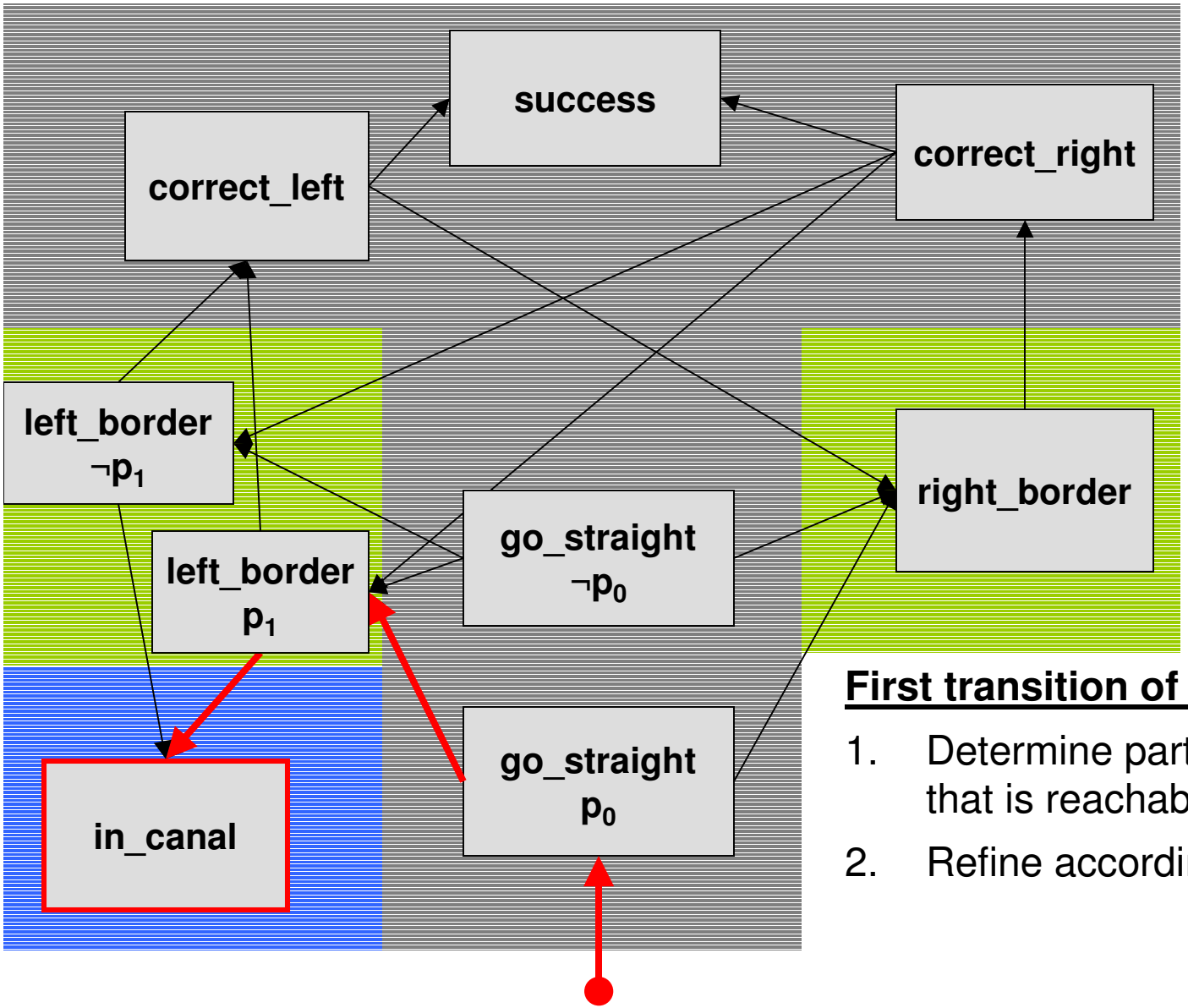
$p_1 = \{ y = -1, c = 0, -\pi/4 < \alpha < \pi/4\}$

p_1 is the part of *left_border* that is reachable from p_0

First transition of CE

1. Determine part of *go_straight and* p_0 that is reachable in *left_border* (p_1)

CEGAR: Validation and Refinement



$p_0 = \{-1 < y < 1, c = 0, -\pi/4 < \alpha < \pi/4\}$

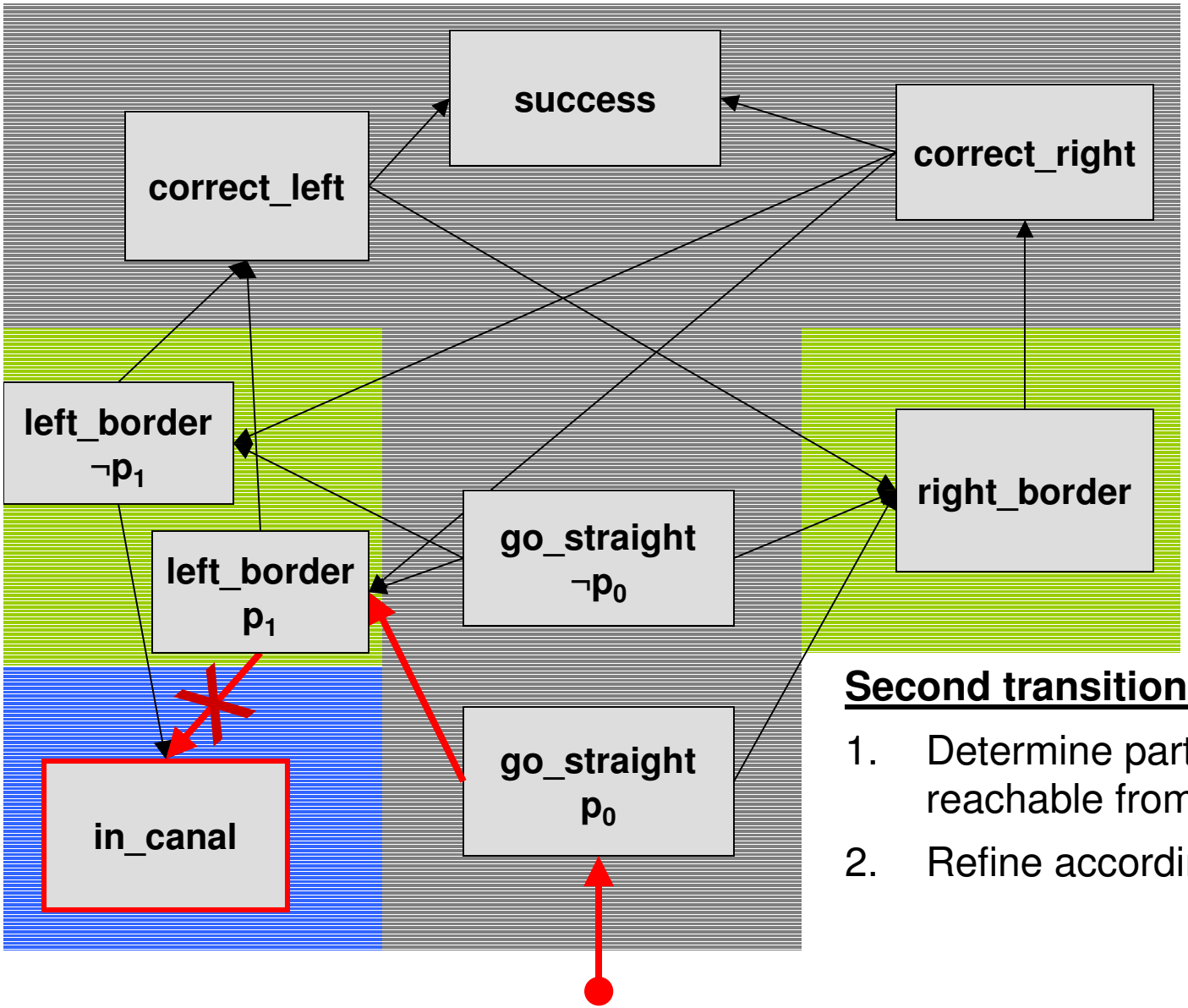
$p_1 = \{ y = -1, c = 0, -\pi/4 < \alpha < \pi/4\}$

p_1 is the part of *left_border* that is reachable from p_0

First transition of CE

1. Determine part of *go_straight and* p_0 that is reachable in *left_border*
2. Refine accordingly

CEGAR: Validation and Refinement



$p_0 = \{-1 < y < 1,\ c = 0,\ -\pi/4 < \alpha < \pi/4\}$

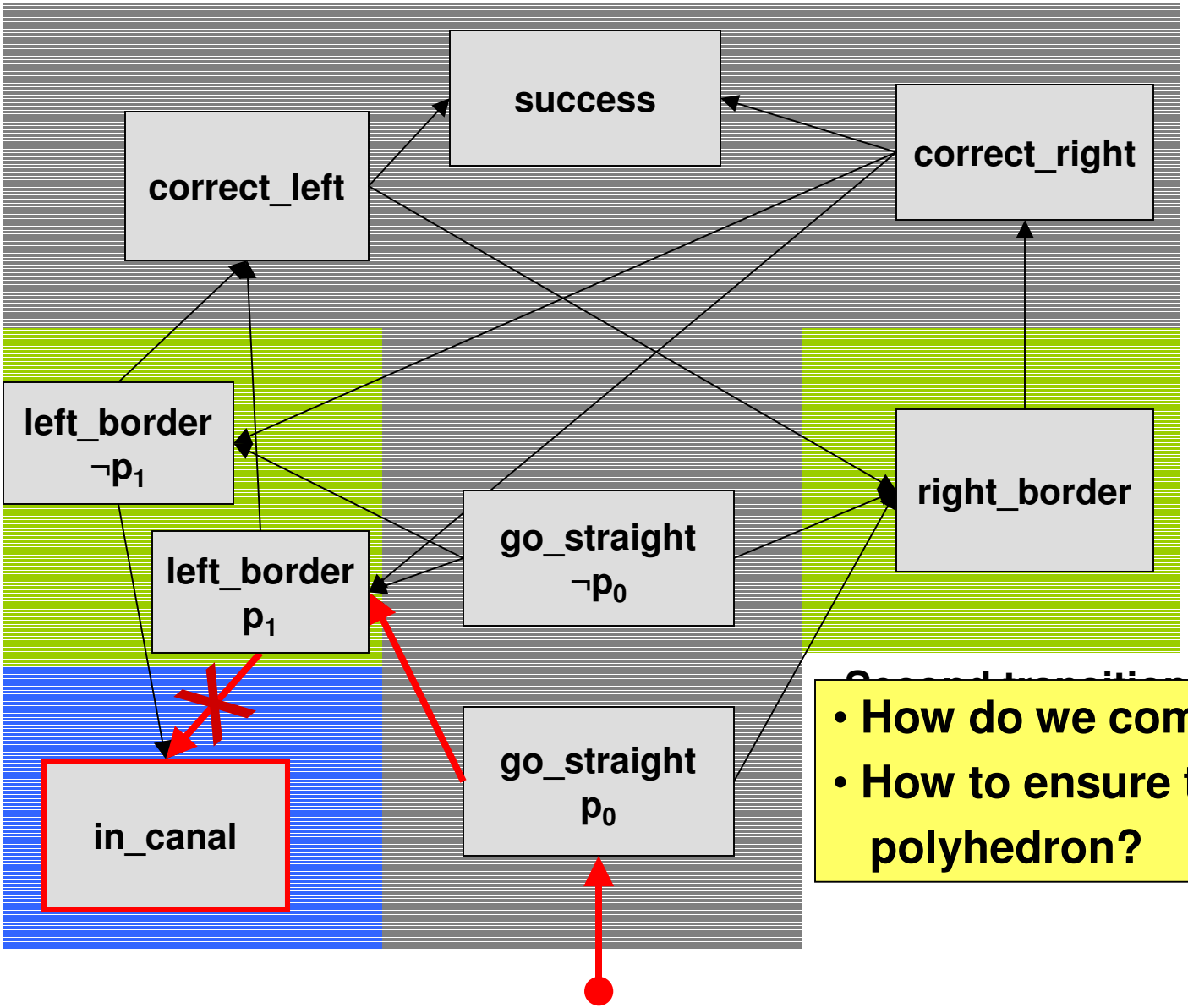
$p_1 = \{\ y = -1,\ \ c = 0,\ -\pi/4 < \alpha < \pi/4\}$

in_canal **not** reachable, given constraints on α and y

Second transition of CE

1. Determine part of *in_canal* that is reachable from *left_border and* p_1
2. Refine accordingly

CEGAR: Validation and Refinement



$p_0 = \{-1 < y < 1,\ c = 0,\ -\pi/4 < \alpha < \pi/4\}$

$p_1 = \{\ y = -1,\ \ c = 0,\ -\pi/4 < \alpha < \pi/4\}$

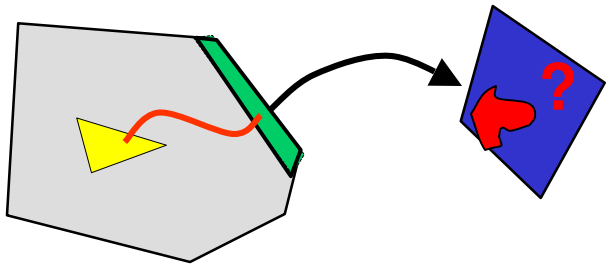
in_canal **not** reachable, given constraints on α and y

- **How do we compute p_1?**
- **How to ensure that p_1 is a polyhedron?**

Multiple Validation and Refinement Schemes

Given: goal set and target set

Compute: subset of the target set that are reachable from goal set



Problem: transition relation is given implicitly by *differential equations*

Approach: over-approximate the reachable states using polyhedra, rectangles, or ellipsoids, etc.

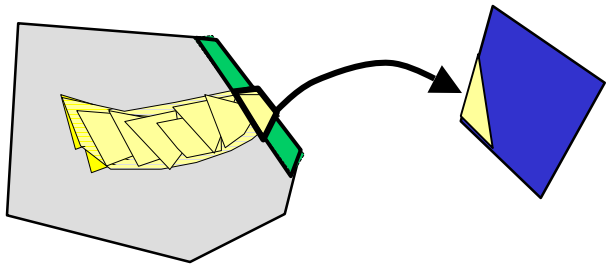
- easier to compute than exact set or computable at all
- results in objects easier to work with, in our case polyhedra

CEGAR provides framework for different approximation methods

Multiple Validation and Refinement Schemes

Given: goal set and target set

Compute: subset of the target set that are reachable from goal set



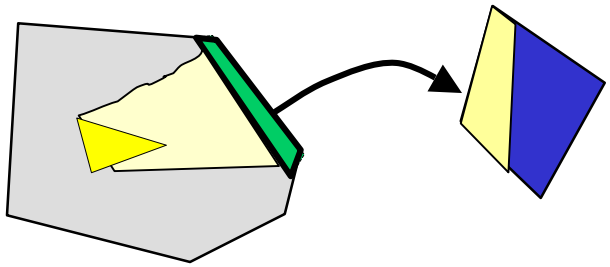
We use 2 methods:

1. Flowpipe approximation a la CheckMate (uses polyhedra for a tight over-approximation)

Multiple Validation and Refinement Schemes

Given: goal set and target set

Compute: subset of the target set that are reachable from goal set

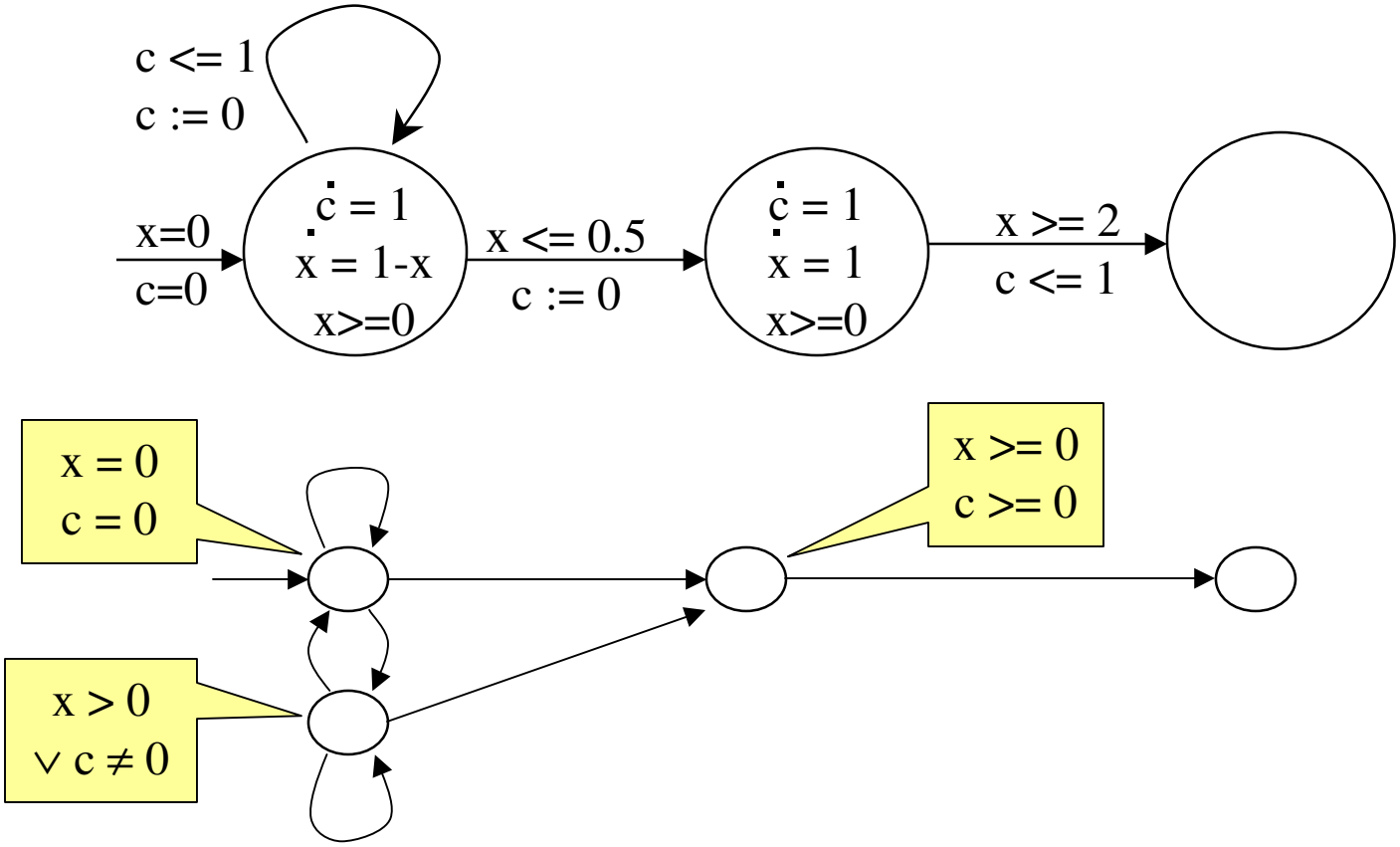


We use 2 methods:

1. Flowpipe approximation ala CheckMate (uses p[illegible]edra for a tight over-approximation)
2. Coarse optimization based method (cheap)

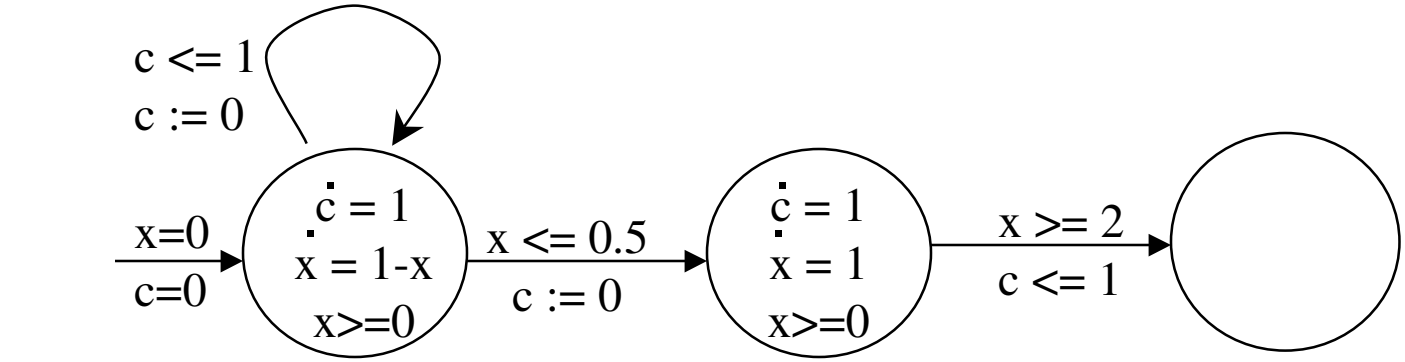
Formulates existence of a transition as optimization problem

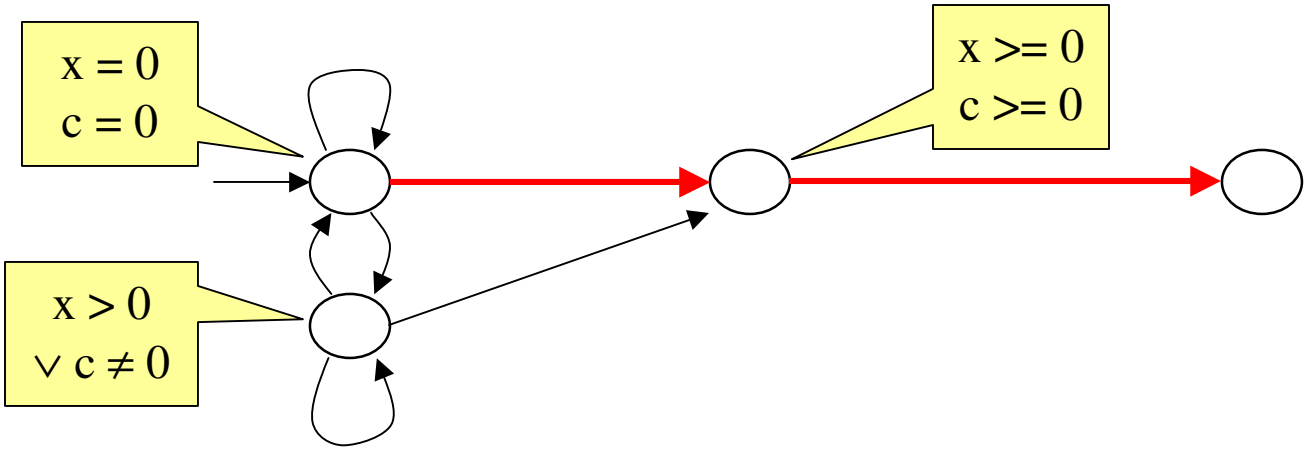
Multiple Validation and Refinement Schemes



Initial abstraction

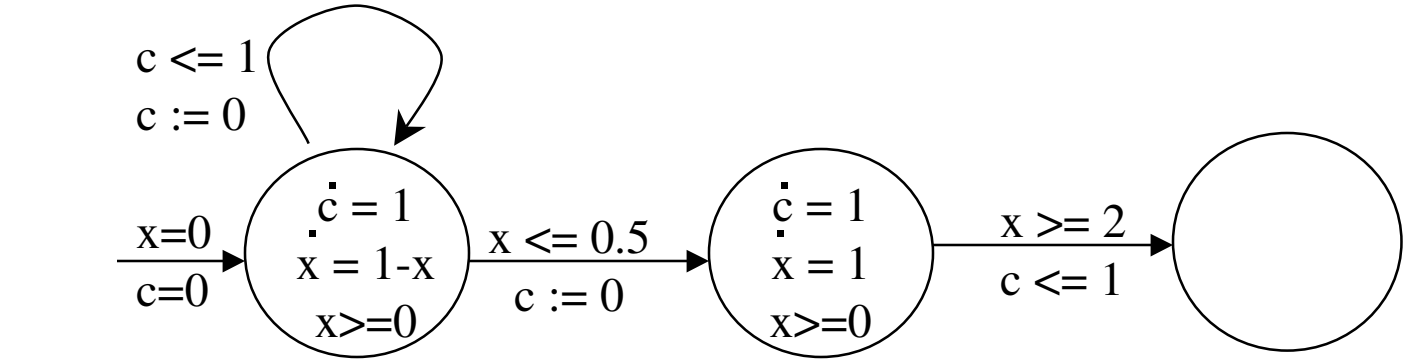
Multiple Validation and Refinement Schemes

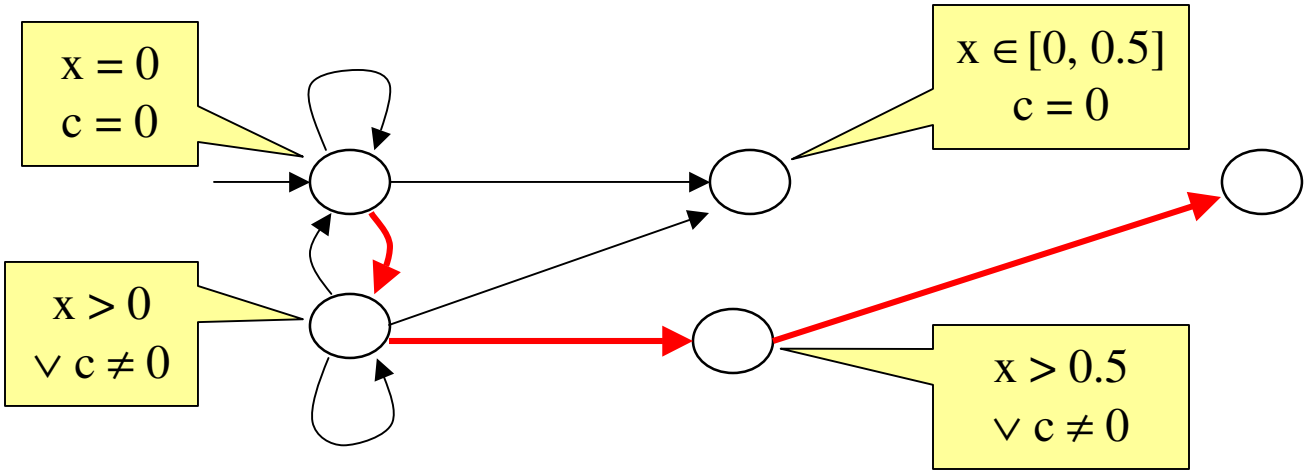




First counterexample

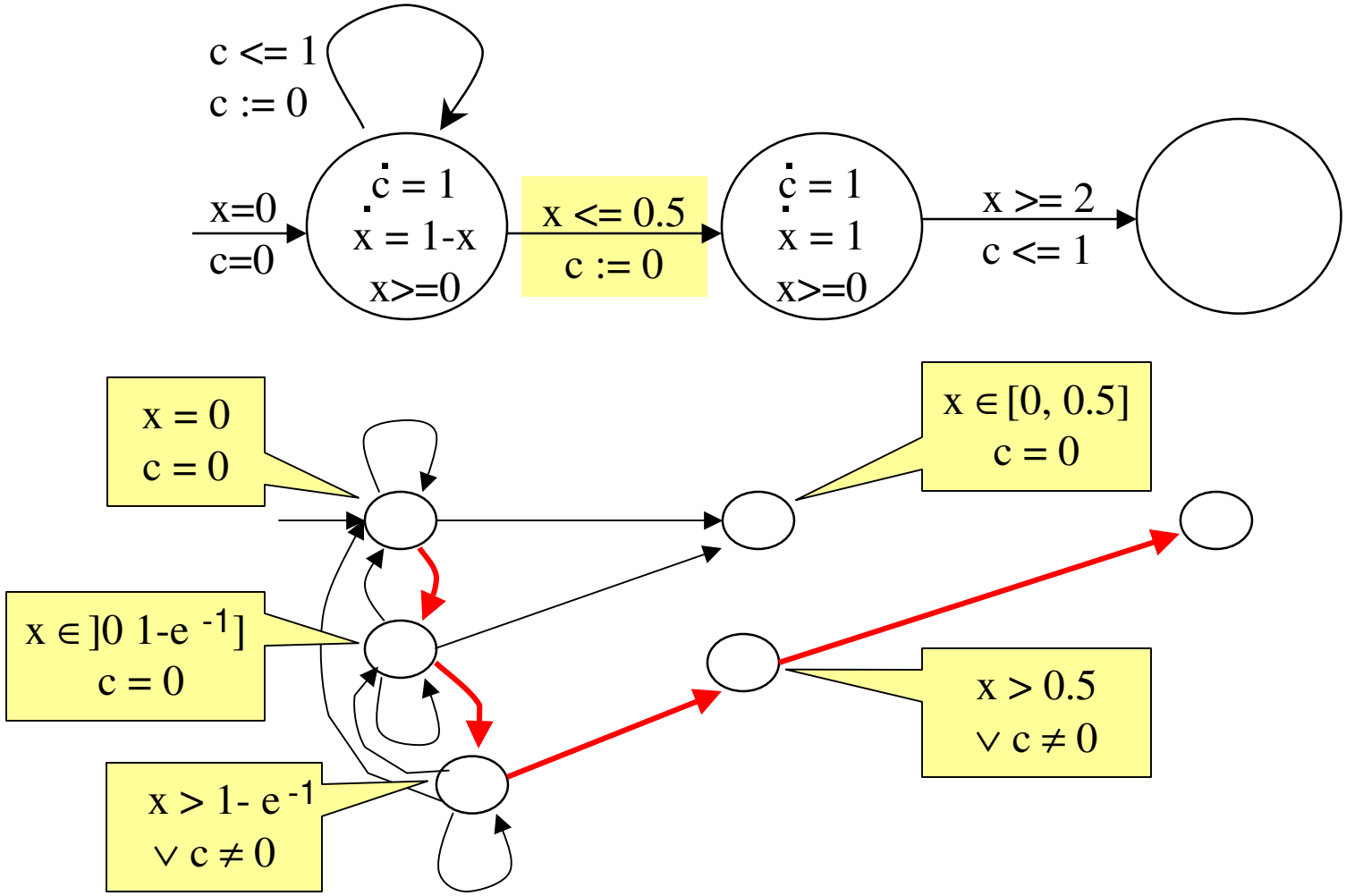
Multiple Validation and Refinement Schemes





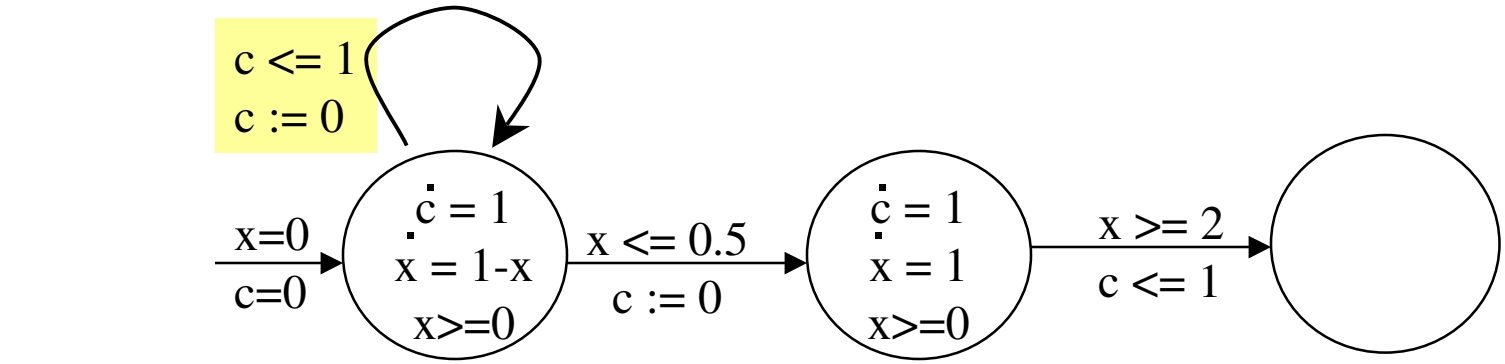
Second counterexample

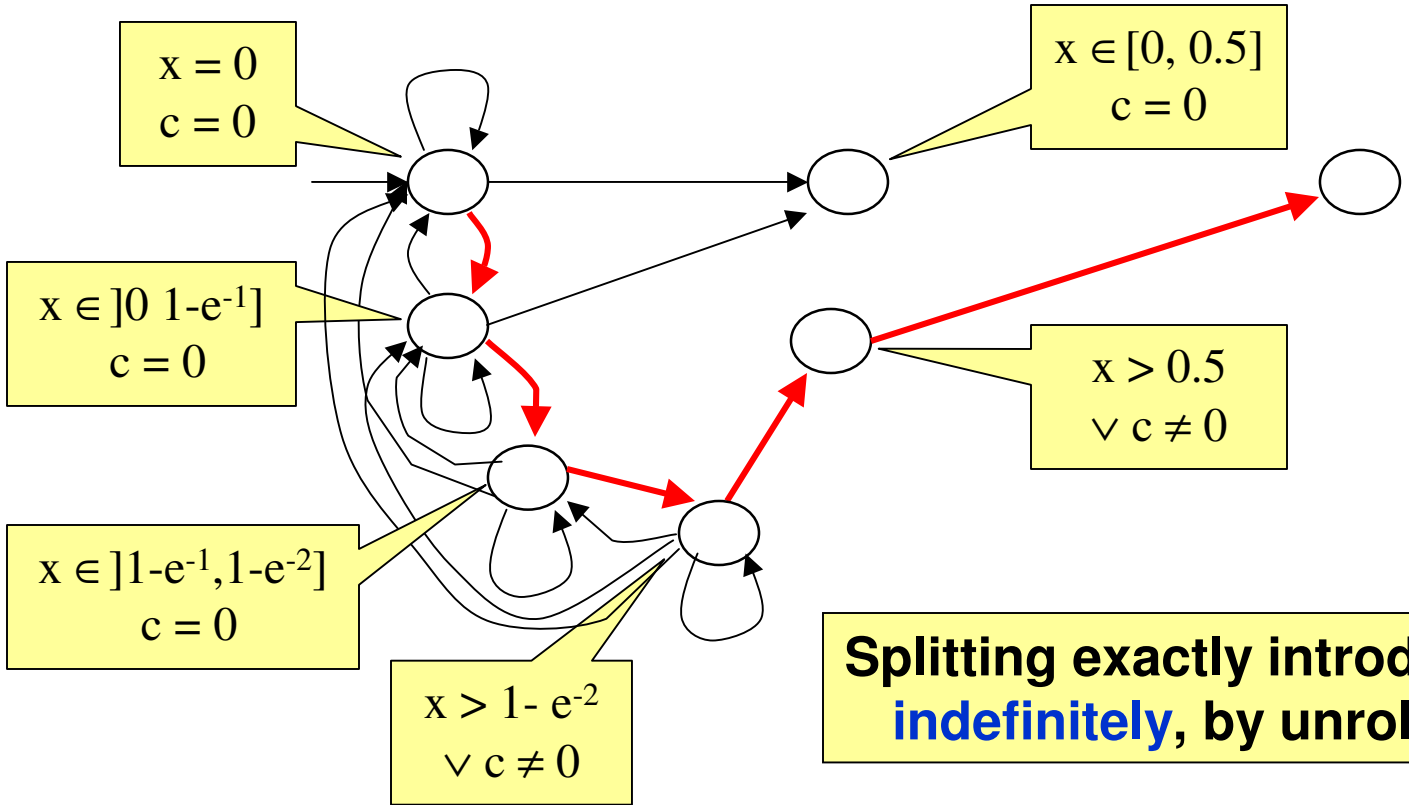
Multiple Validation and Refinement Schemes



Third counterexample after splitting with the exact sets

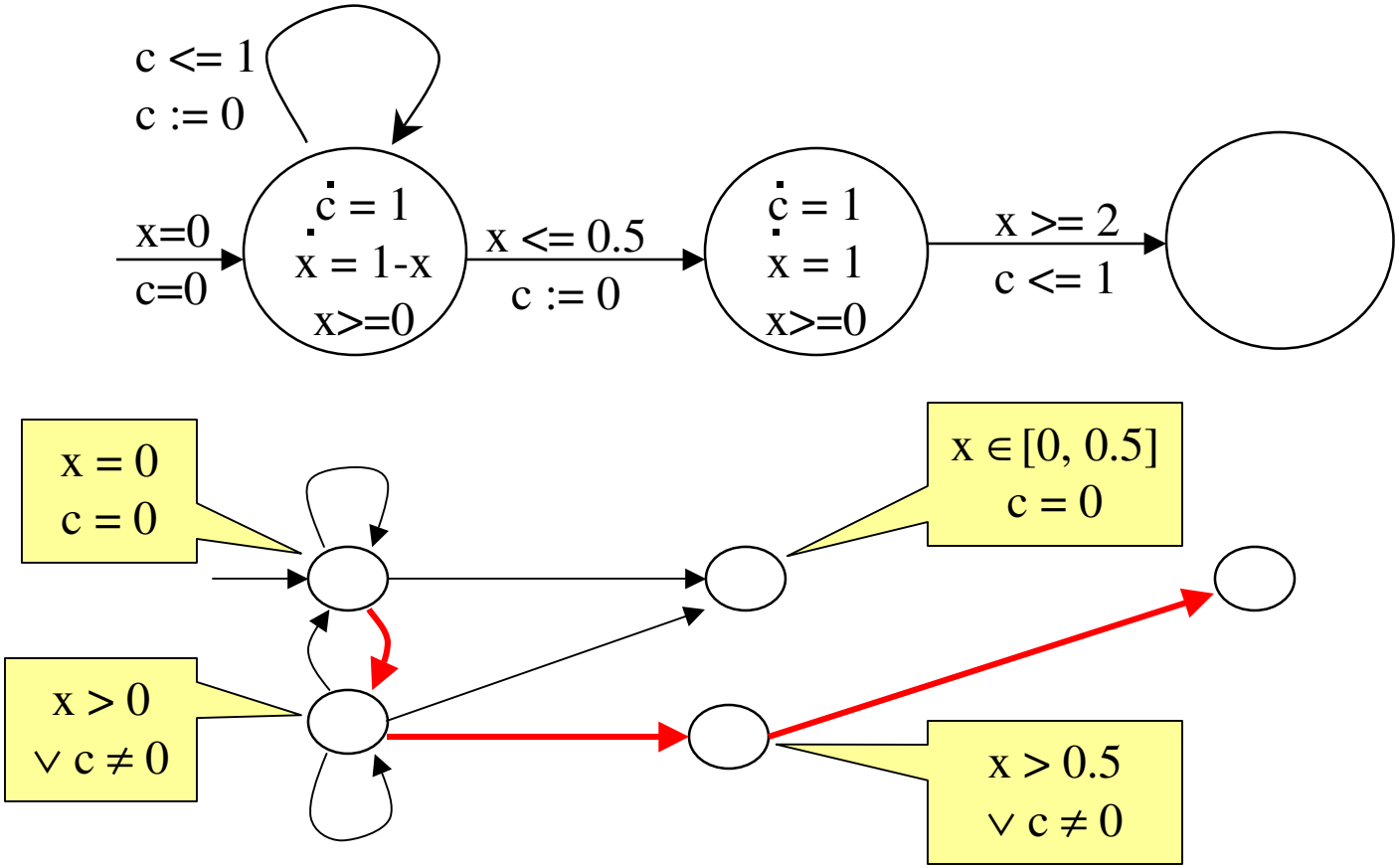
Multiple Validation and Refinement Schemes





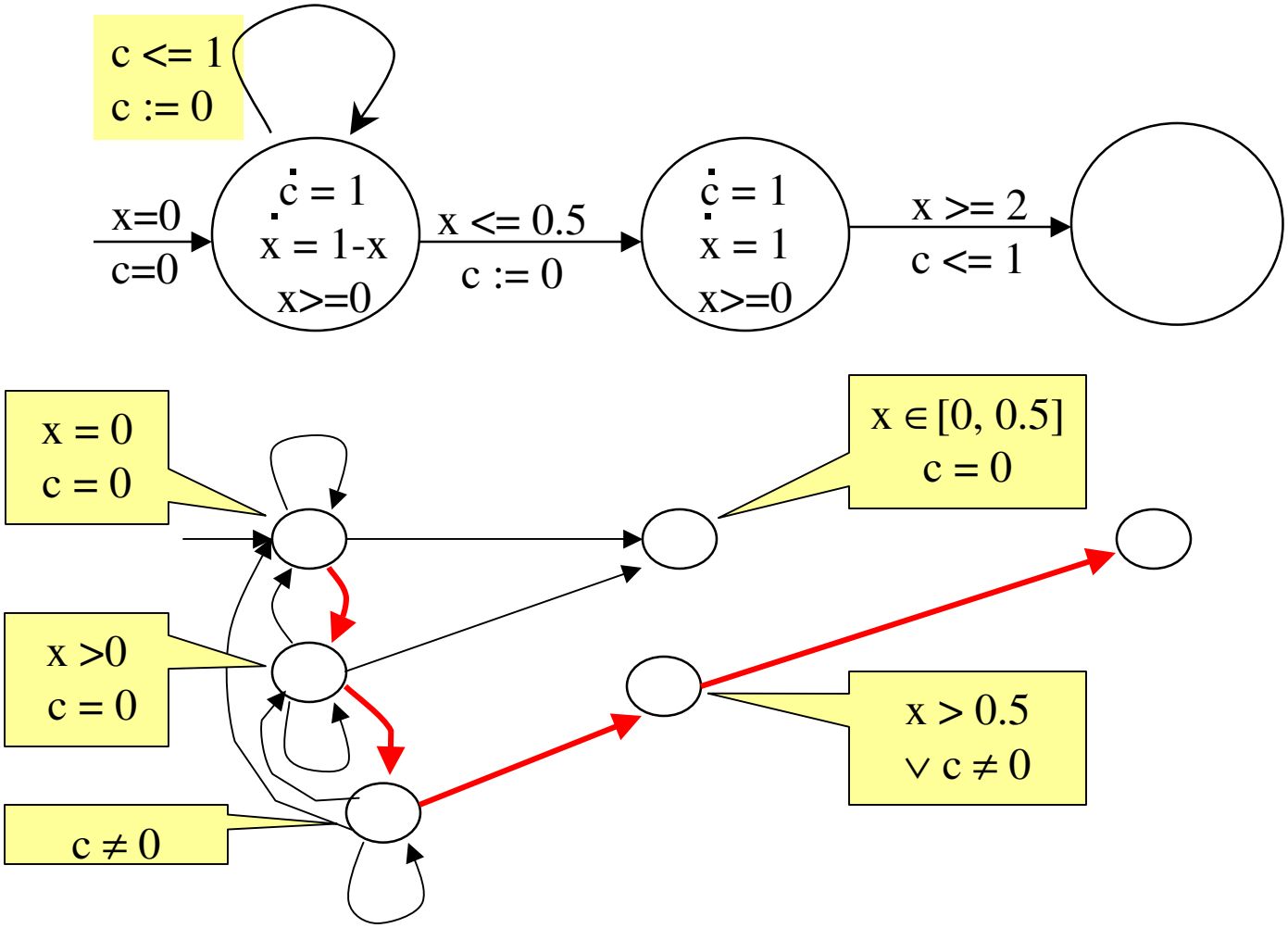
Splitting exactly introduces new CEs indefinitely, by unrolling the loop

Multiple Validation and Refinement Schemes



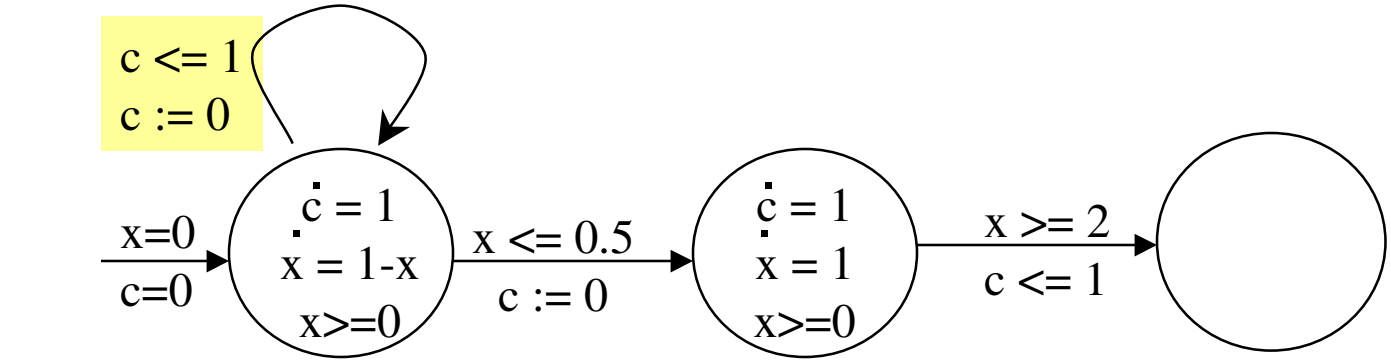
Second counterexample as before

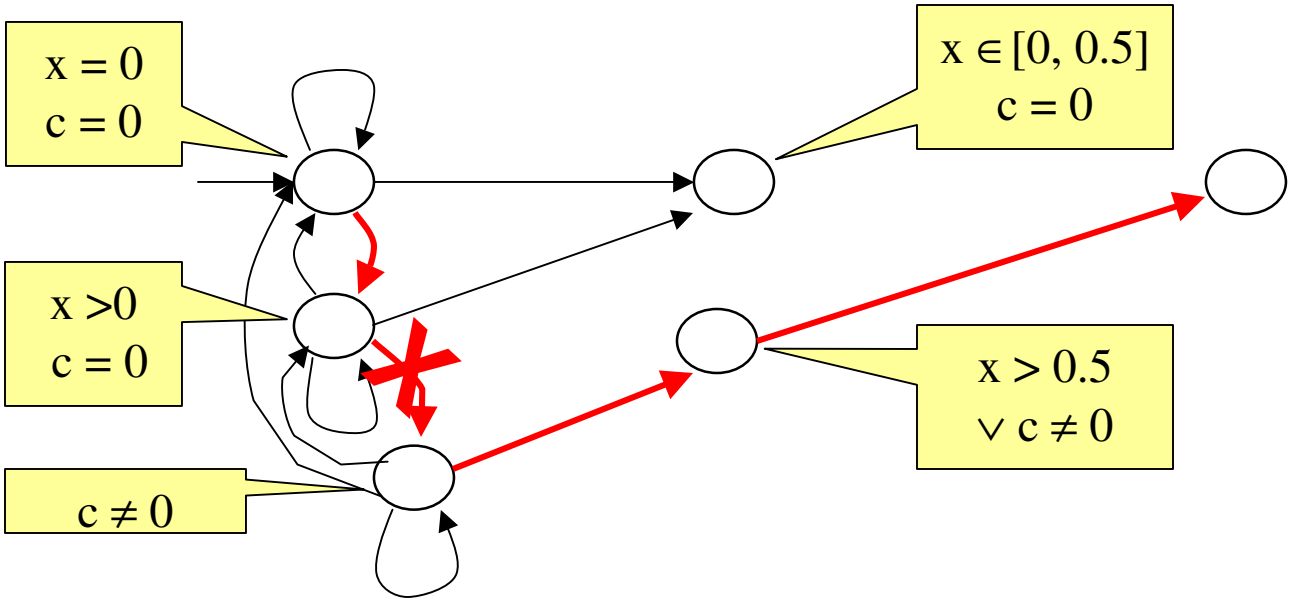
Multiple Validation and Refinement Schemes



Splitting using a coarse over-approximation

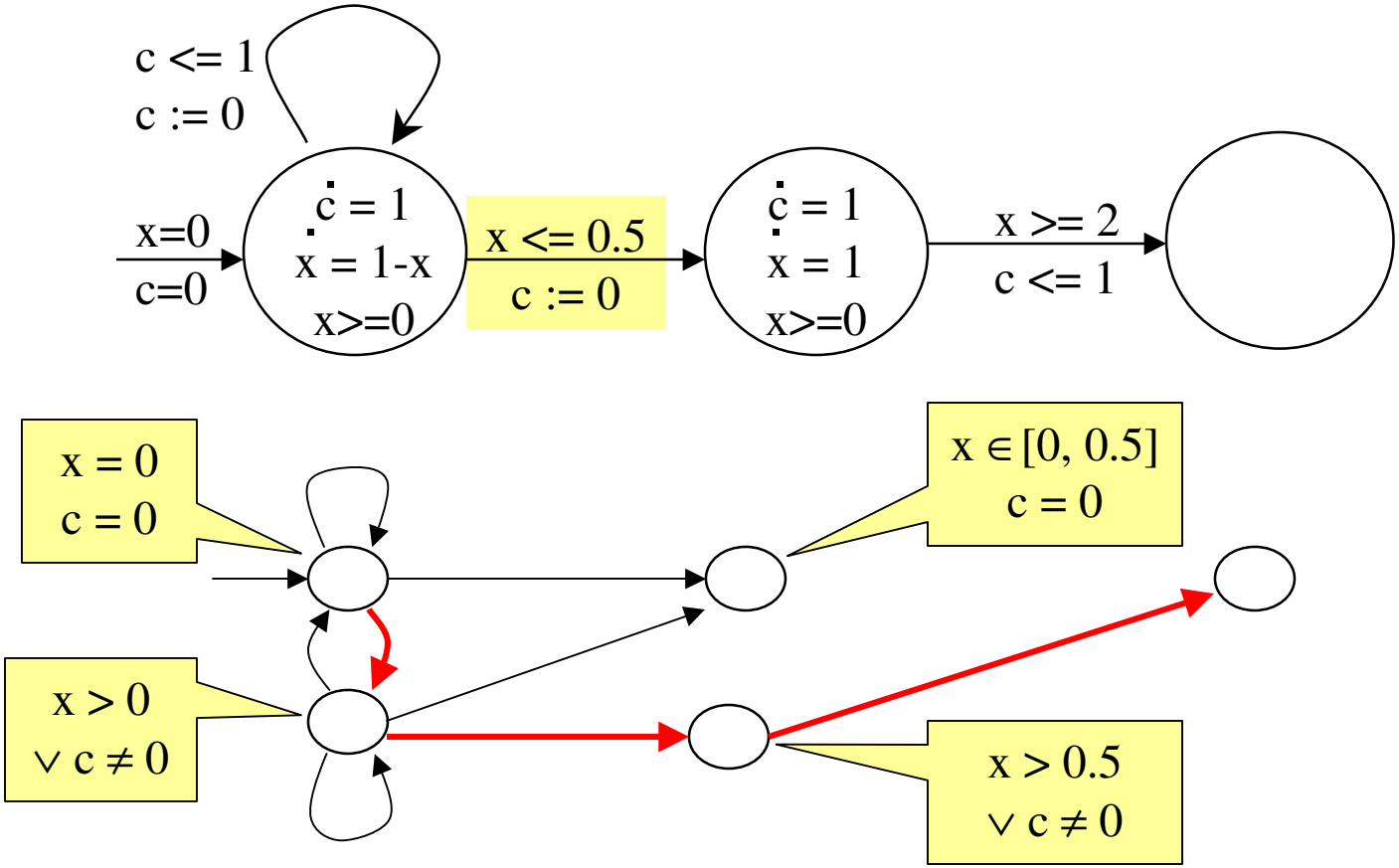
Multiple Validation and Refinement Schemes





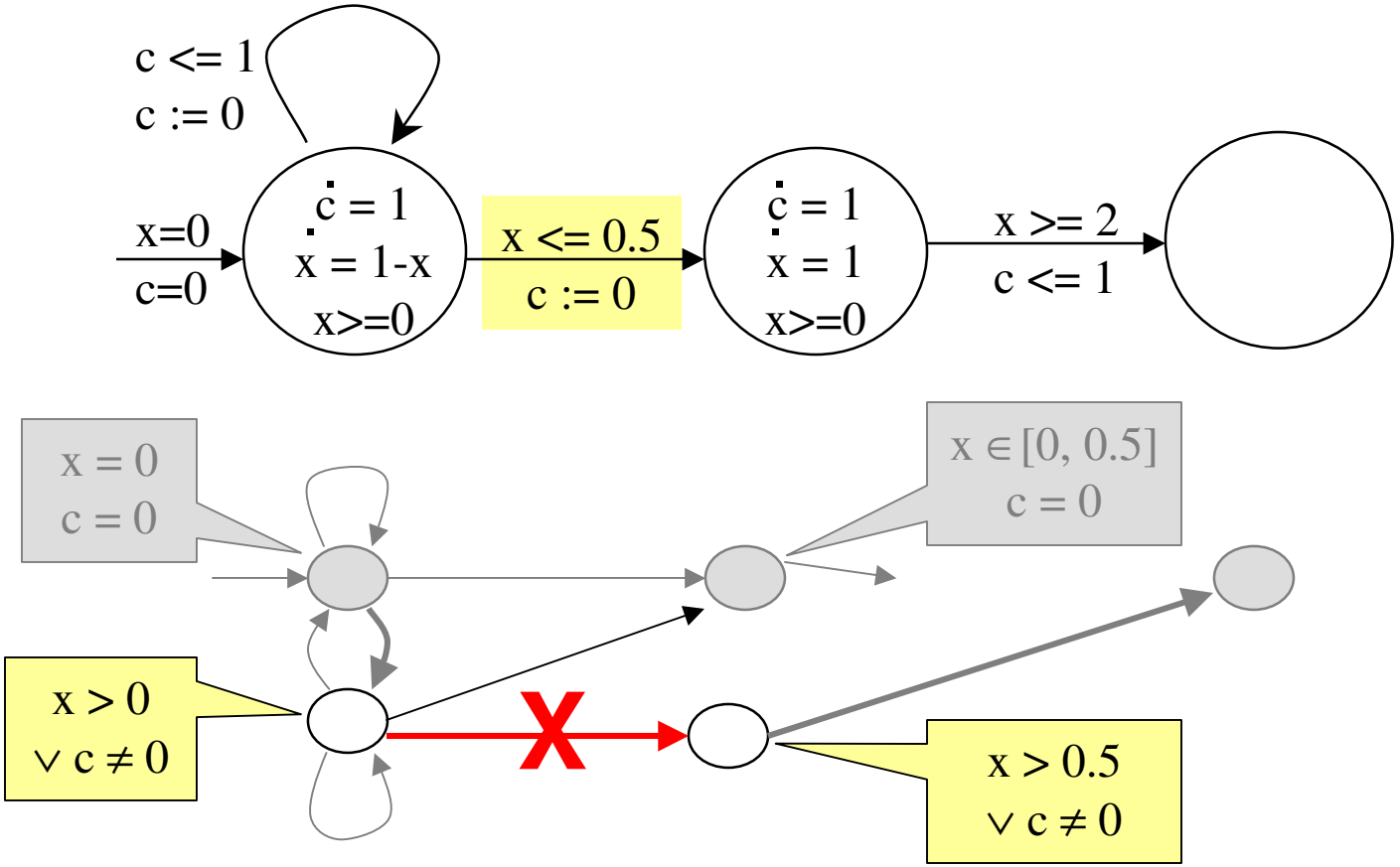
Counterexample removed without splitting ⇒ DONE!

Validation of Fragments



Second counterexample as before

Validation of Fragments



The second transition can be found to be spurious by considering it on its own.

Outline

- Introduction
- Background: CEGAR
  Counterexample-Guided Abstraction Refinement
- Hybrid Systems
- CEGAR for Hybrid Systems
  - Abstractions
  - Validation and Refinement
  - Multiple Validation and Refinement Schemes
- Example
  - Car Steering Example
  - Cruise Control
- Related Work
- Future Work

Results

Car Steering Example

Common reachability analysis:	117 sec
CEGAR with multiple over-approximations:	70 sec
As before with considering fragments:	10 sec

Results

Car Steering Example

If coarse approximation fails, fall back on better method

Common reachability analysis:	117 sec
CEGAR with multiple over-approximations:	70 sec
As before with considering fragments:	10 sec

Results

Car Steering Example

Common reachability analysis:	117 sec
CEGAR with multiple over-approximations:	70 sec
As before with considering fragments:	10 sec

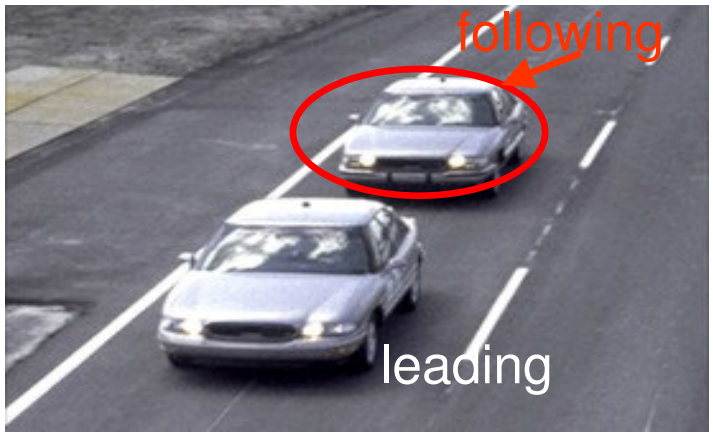
If a CE cannot be refuted with fragment, consider complete CE

Results

Car Steering Example

Common reachability analysis:	117 sec
CEGAR with multiple over-approximations:	70 sec
As before with considering fragments:	10 sec

Adaptive Cruise Control

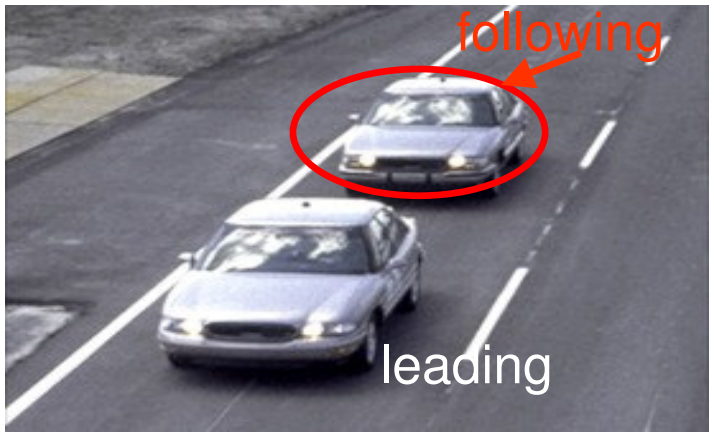


Results

Car Steering Example

Common reachability analysis:	117 sec
CEGAR with multiple over-approximations:	70 sec
As before with considering fragments:	10 sec

Adaptive Cruise Control



Common reachability analysis:	770 sec
CEGAR with multiple over-approximations:	450 sec
As before with considering fragments:	39 sec

Related work

- Series of Abstractions
  - Jeannet, Halbwachs et al., *Declarative synchronous programs.* [1999]
  - Krogh and Chutinam, *Quotient transition systems for HS.* [2001]
  - Tiwari and Khanna, *Abstractions for Hybrid Systems*. [2002]
- Counterexample-Guided Abstraction Refinement
  - Kurshan, Clarke et al., *CEGAR for finite systems*, [2000]
  - Alur, Dang, Ivancic , *soon in this theater*

Summary and Future Work

Extension of CEGAR to Hybrid Systems

- explore dynamics only in part of the system relevant to the property
- local refinement within a location
- framework for different over-approximation methods
- also applies to general infinite state systems
- promising application of prototype (10 times faster on first examples)

Future Work

- evaluate different validation and refinement schemes
- other local criteria for improved validation and refinement
- tool development
- more examples

Thank you

Contact:	Ansgar Fehnker	<ansgar@ece.cmu.edu>
	Joël Ouaknine	<ouaknine@cs.cmu.edu>
	Olaf Stursberg	<olaf.stursberg@uni-dortmund.de>
	Michael Theobald	<theobald@cs.cmu.edu>