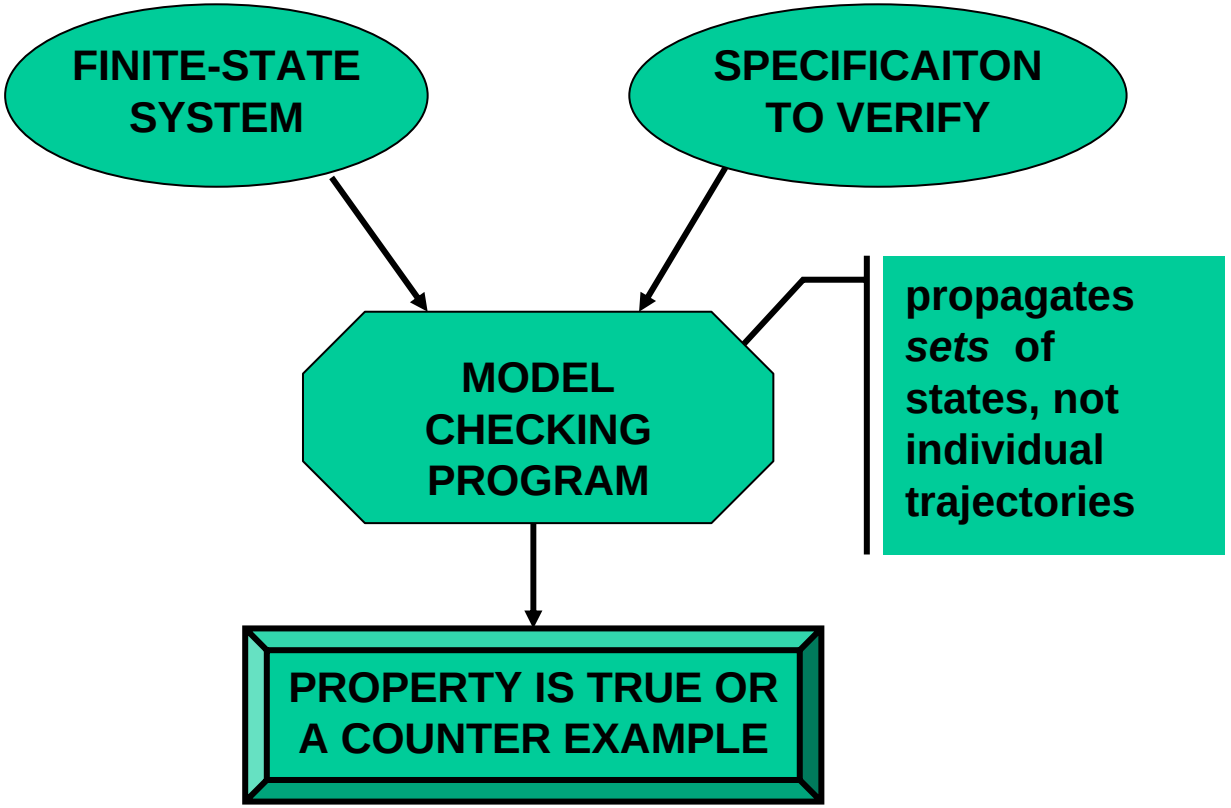
Iterative Relaxation Abstraction for Verification and Design of Hybrid Systems

Bruce H. Krogh
Dept. of Electrical and Computer Engineering
Carnegie Mellon University

Verification via Model Checking

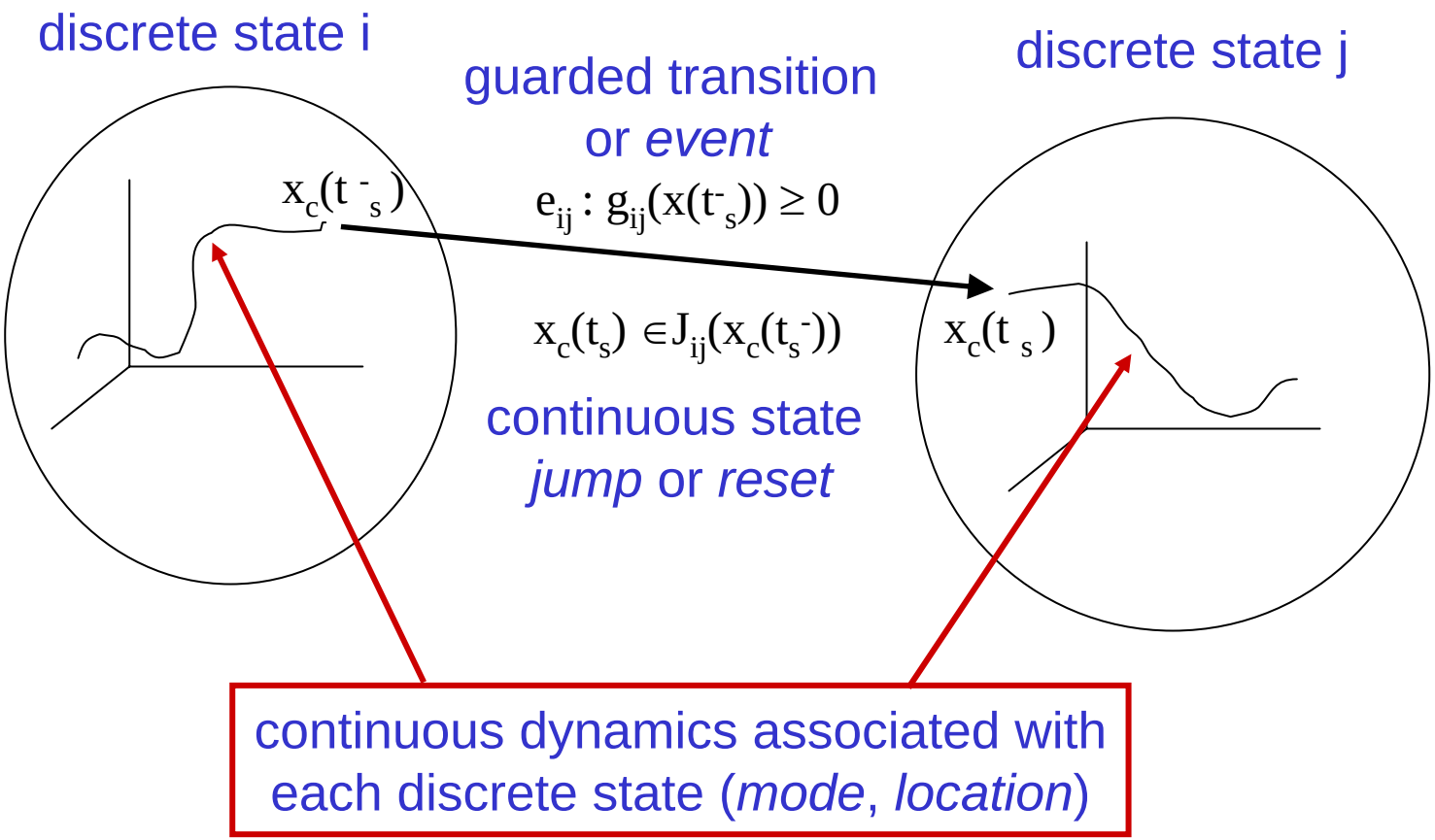


Model Checking vs. Simulation

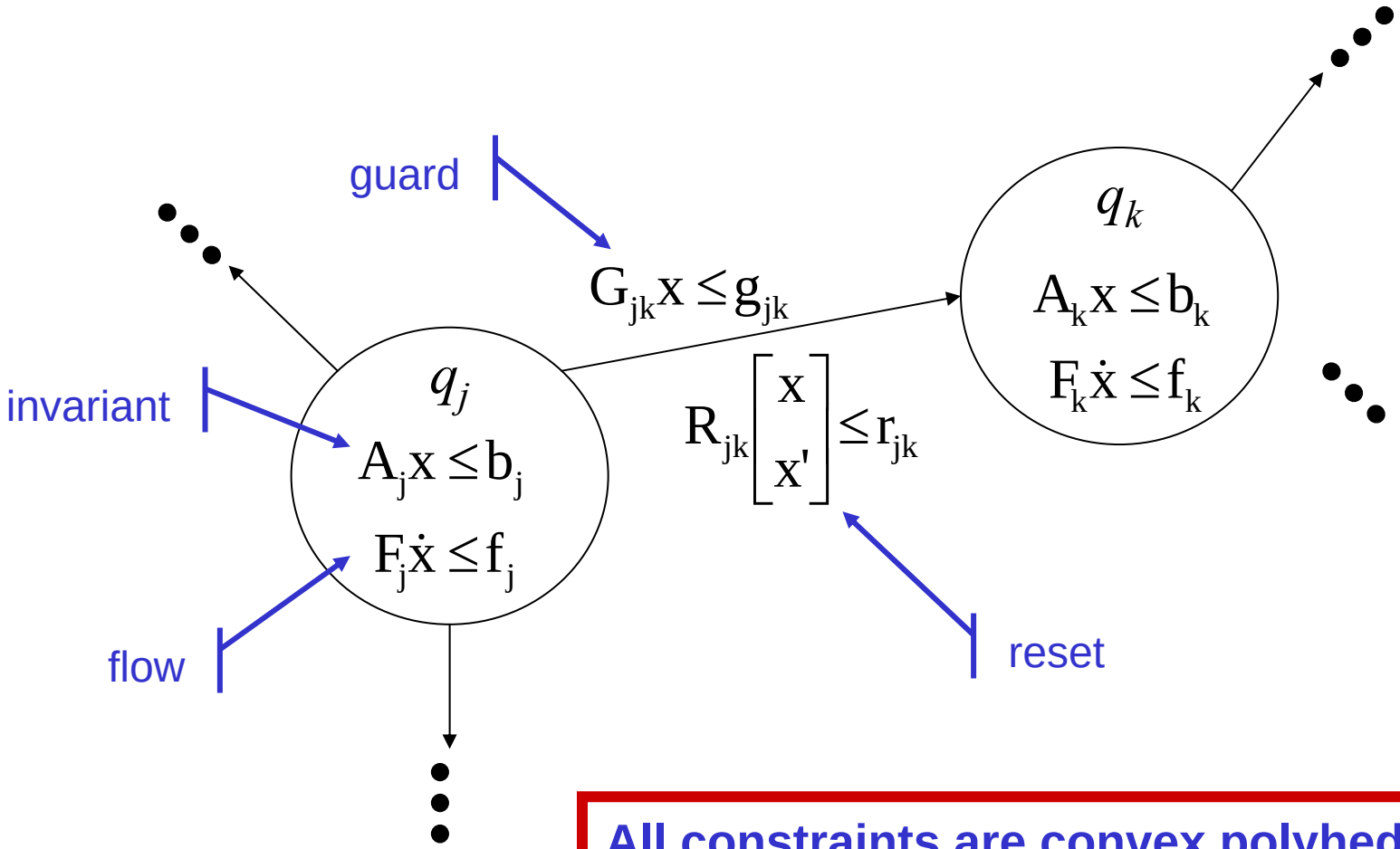
Model checking investigates *every possible behavior* of the system *all* initial conditions and input signals

... a simulator generates *only one trajectory* for a *particular* initial condition and input signal.

Hybrid Automata

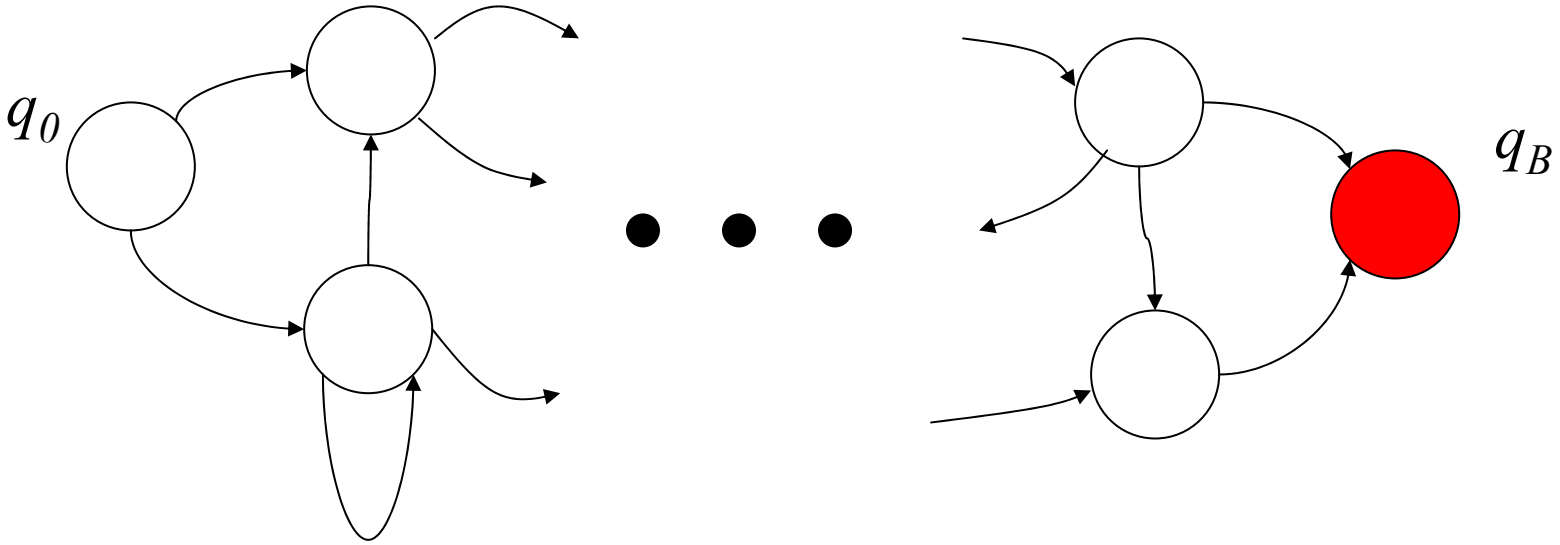


Linear Hybrid Automata (LHA)



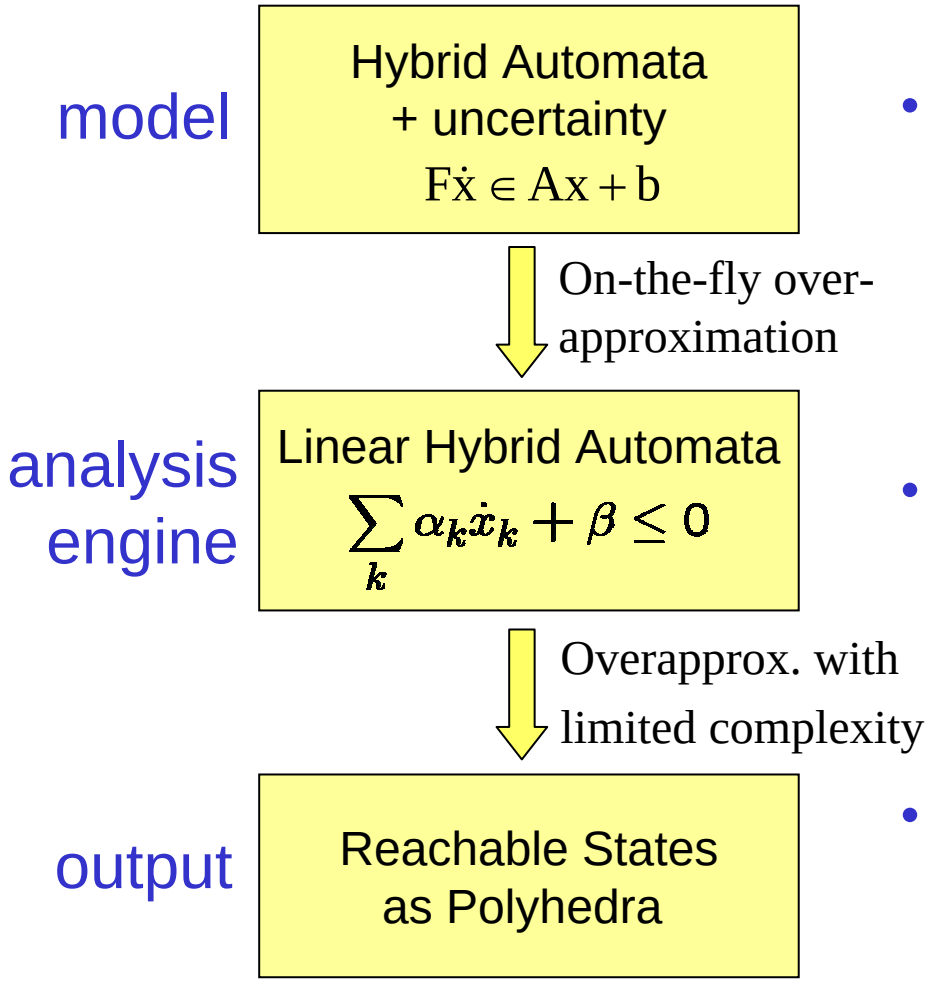
All constraints are convex polyhedra.

Reachability Problems in LHA



***Specification*: Guarantee no paths in the location graph from initial location(s) q_0 *to bad* location(s) q_B are *feasible*.**

Polyhedral Hybrid Automaton Verifier (PHAVer*)



- **Reachability Analysis**
  - exact arithmetic
  - guaranteed conservative overapproximations
  - complexity management by limiting of bits and constraints
- **Uses State-of-the-Art Polyhedra Library**
  - Parma Polyhedra Library (PPL)
  - robust implementation: C++, GMP
- **Compositional Reasoning**
  - computing simulation relations

*developed by G. Frehse (now at VERIMAG)

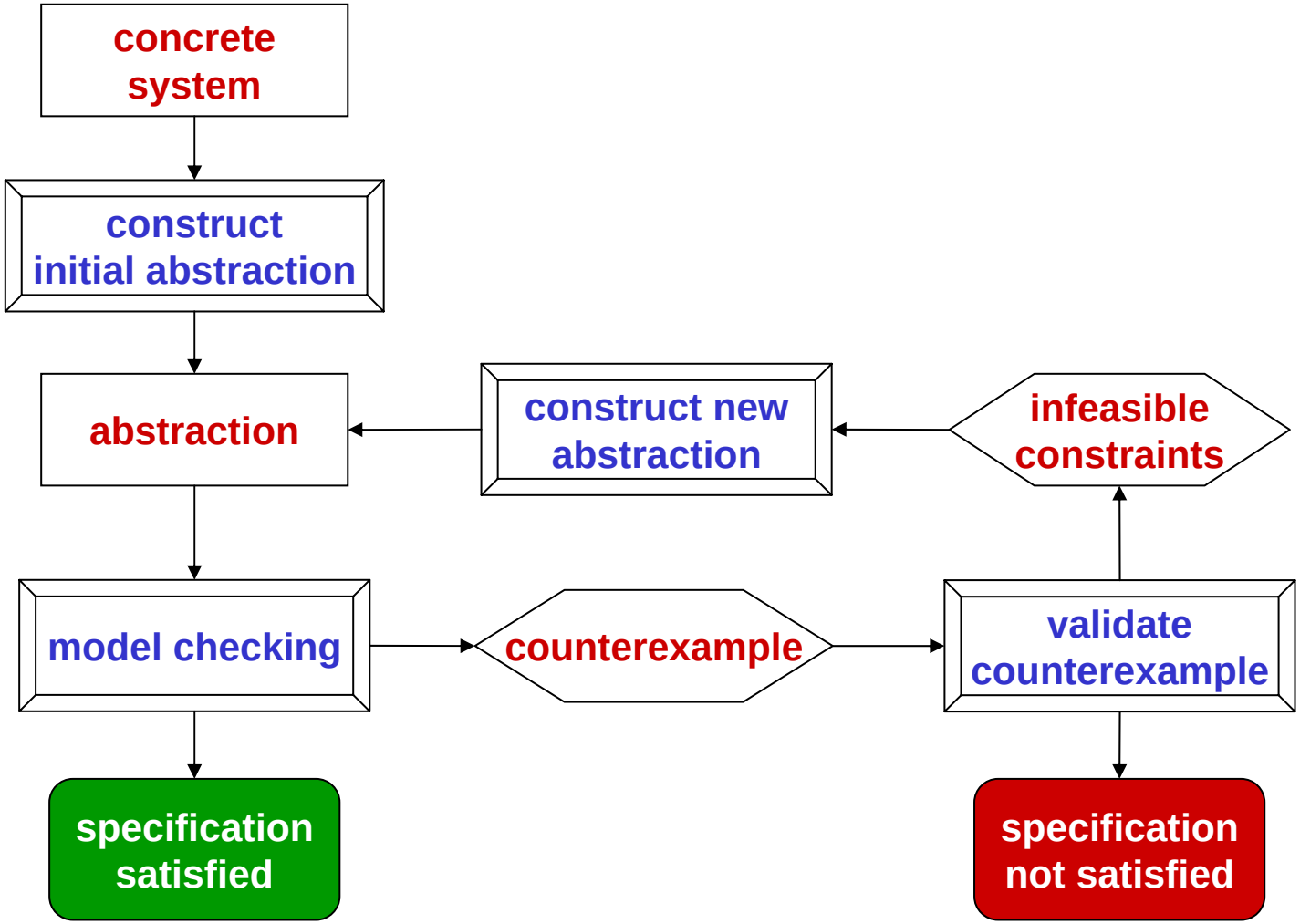
PHAVer References

- Reachability Analysis
  - PHAVer: Algorithmic Verification of Hybrid Systems past HyTech
    Frehse. HSCC'05
  - Time Domain Verification of Oscillator Circuit Properties
    Frehse, Krogh, Rutenbar, Maler. FAC'05
  - Verification of Hybrid Systems using Iterative Refinement
    Frehse, Krogh, Rutenbar. SRC Techcon'05. *Best Paper in Session Award*
  - Verifying Analog Oscillator Circuits Using Forward/Backward Abstraction Refinement
    Frehse, Krogh, Rutenbar. DATE'06
- Compositional Reasoning
  - Assume-Guarantee Reasoning for Hybrid I/O-Automata by Over-Approximation of Continuous Interaction
    Frehse, Han, Krogh. CDC'04

PHAVer available at http://www.cs.ru.nl/~goranf/

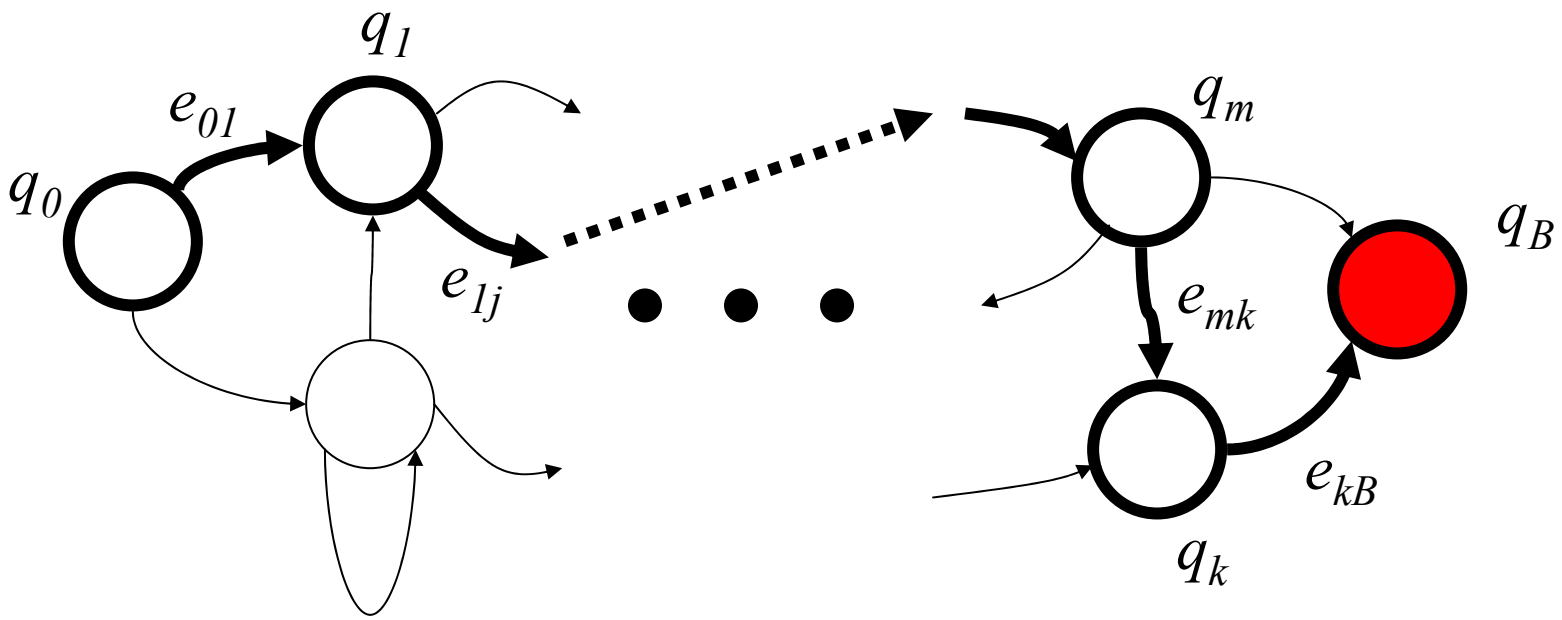
CEGAR

(CounterExample Guided Abstraction Refinement)



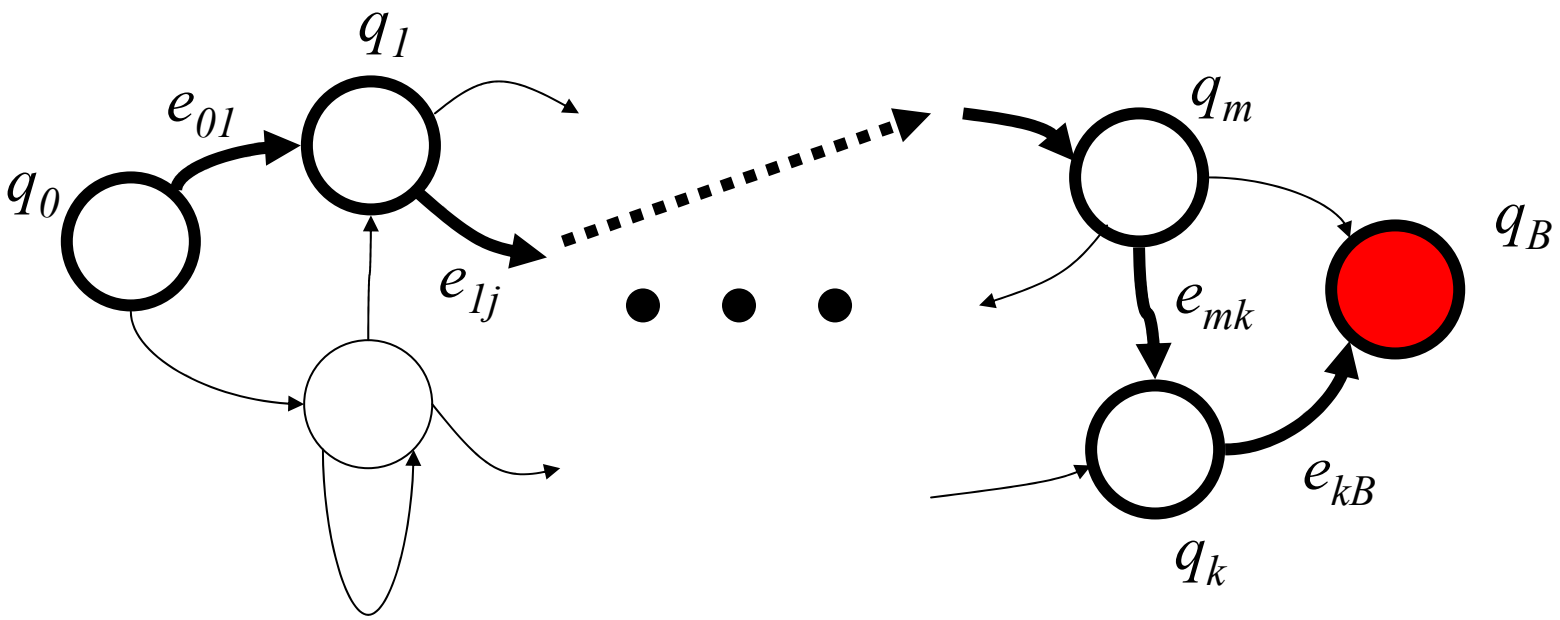
Counterexamples

paths to bad states: $\pi = q_0\, e_{01}\, q_1\, e_{1j} \ldots e_{mk}\, q_k\, e_{kj}\, q_B$



Feasible Counterexamples

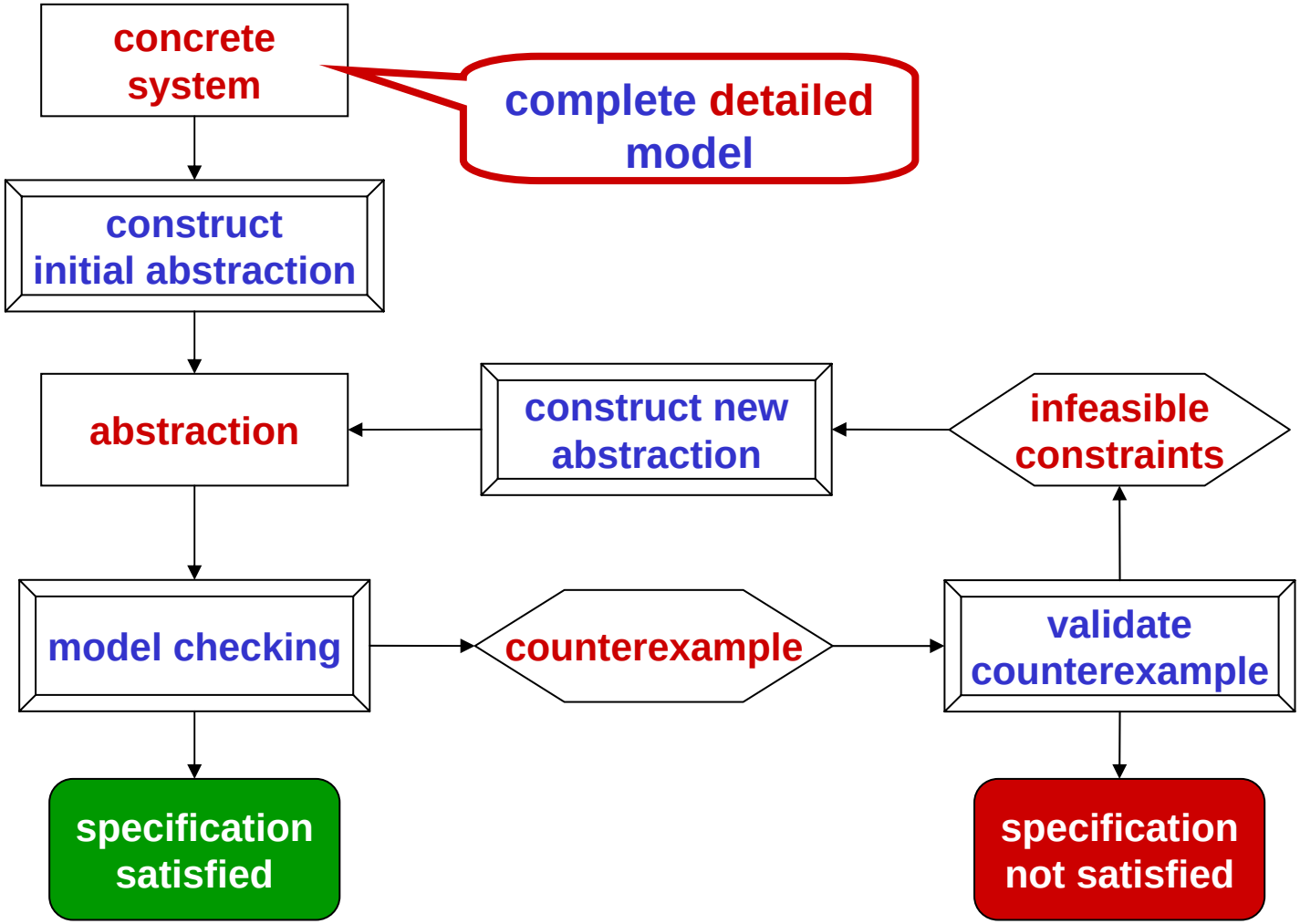
paths to bad states: $\pi = q_0\, e_{01}\, q_1\, e_{1j} \ldots e_{mk}\, q_k\, e_{kj}\, q_B$



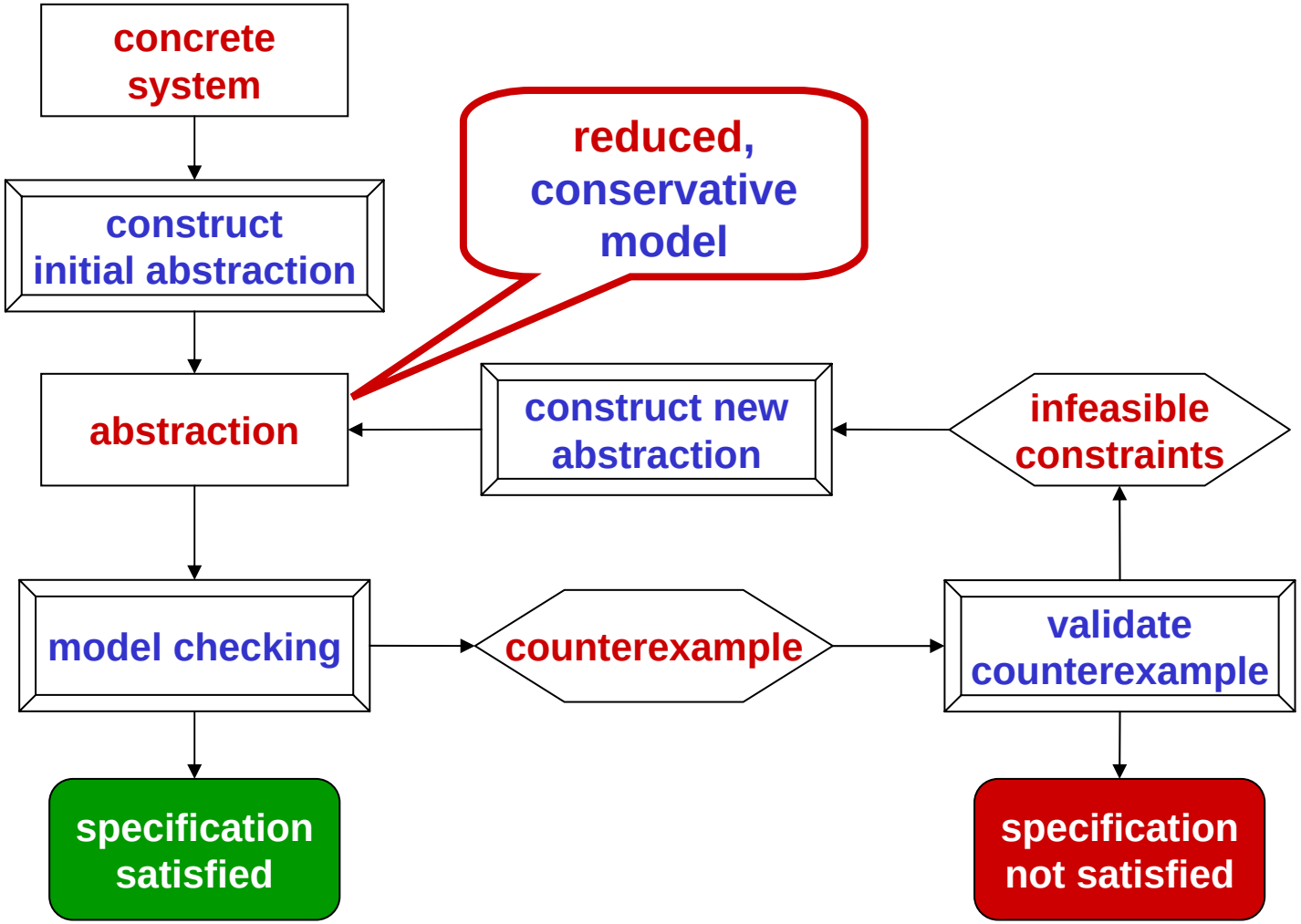
{r^r_r>r₁'$ {4^4_r>4₁'$ fff$ {n^n_r>n₁'$ {E^E_r>E₁

There exists a sequence of continuous trajectories satisfying the constraints along the path.

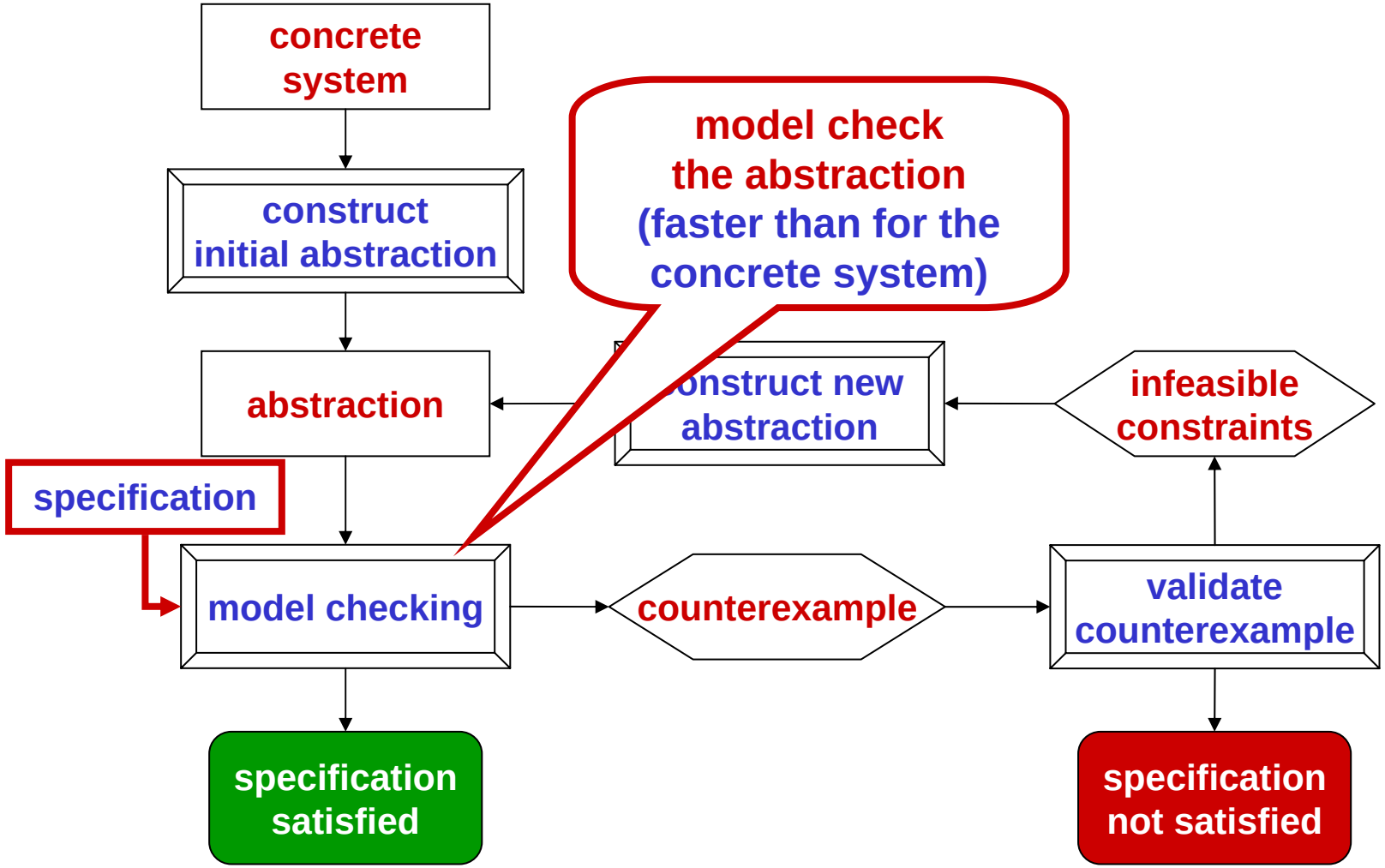
CEGAR


concrete system
complete detailed model
construct initial abstraction
abstraction
construct new abstraction
infeasible constraints
model checking
counterexample
validate counterexample
specification satisfied
specification not satisfied

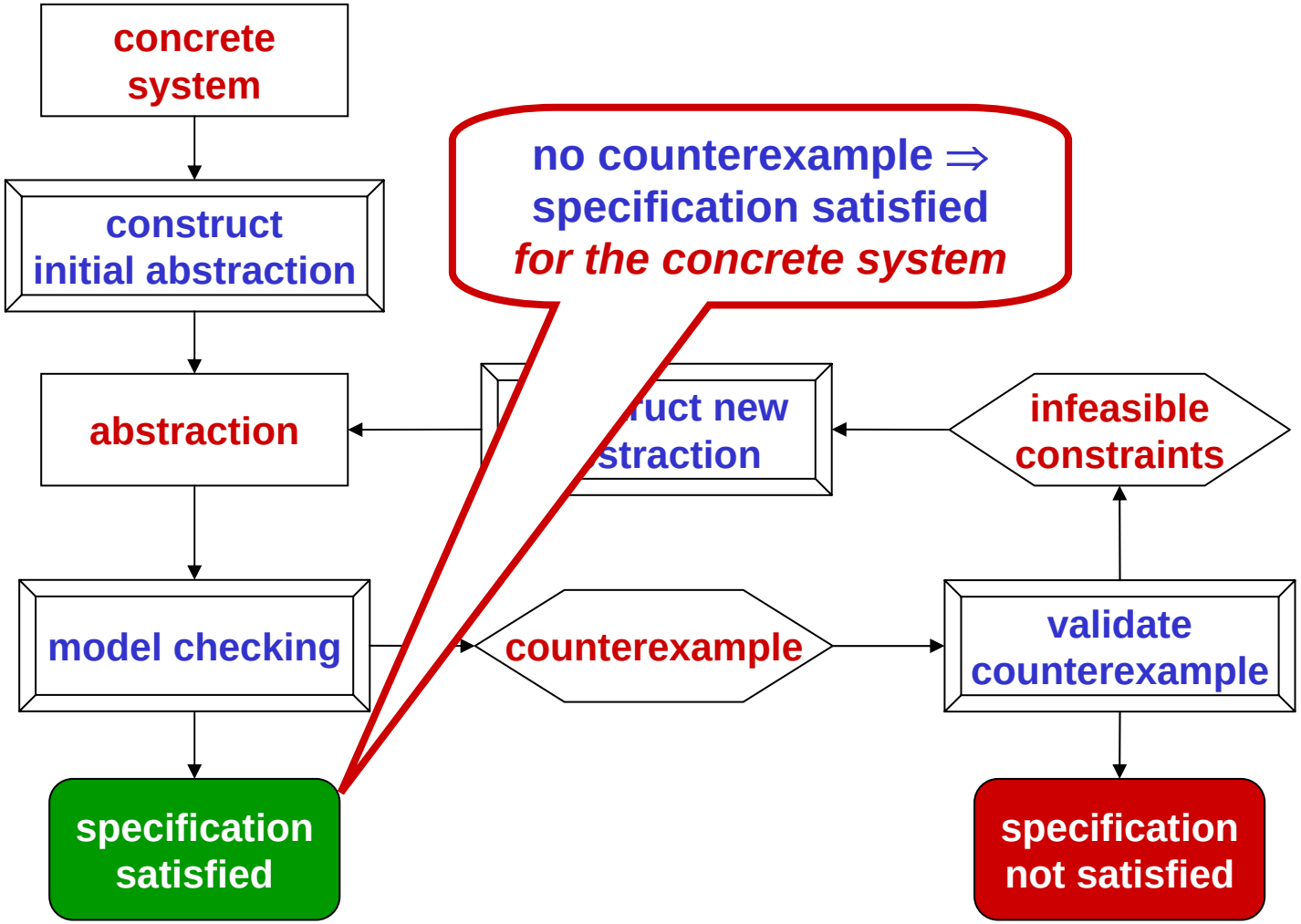
CEGAR


concrete system
construct initial abstraction
reduced, conservative model
abstraction
construct new abstraction
infeasible constraints
model checking
counterexample
validate counterexample
specification satisfied
specification not satisfied

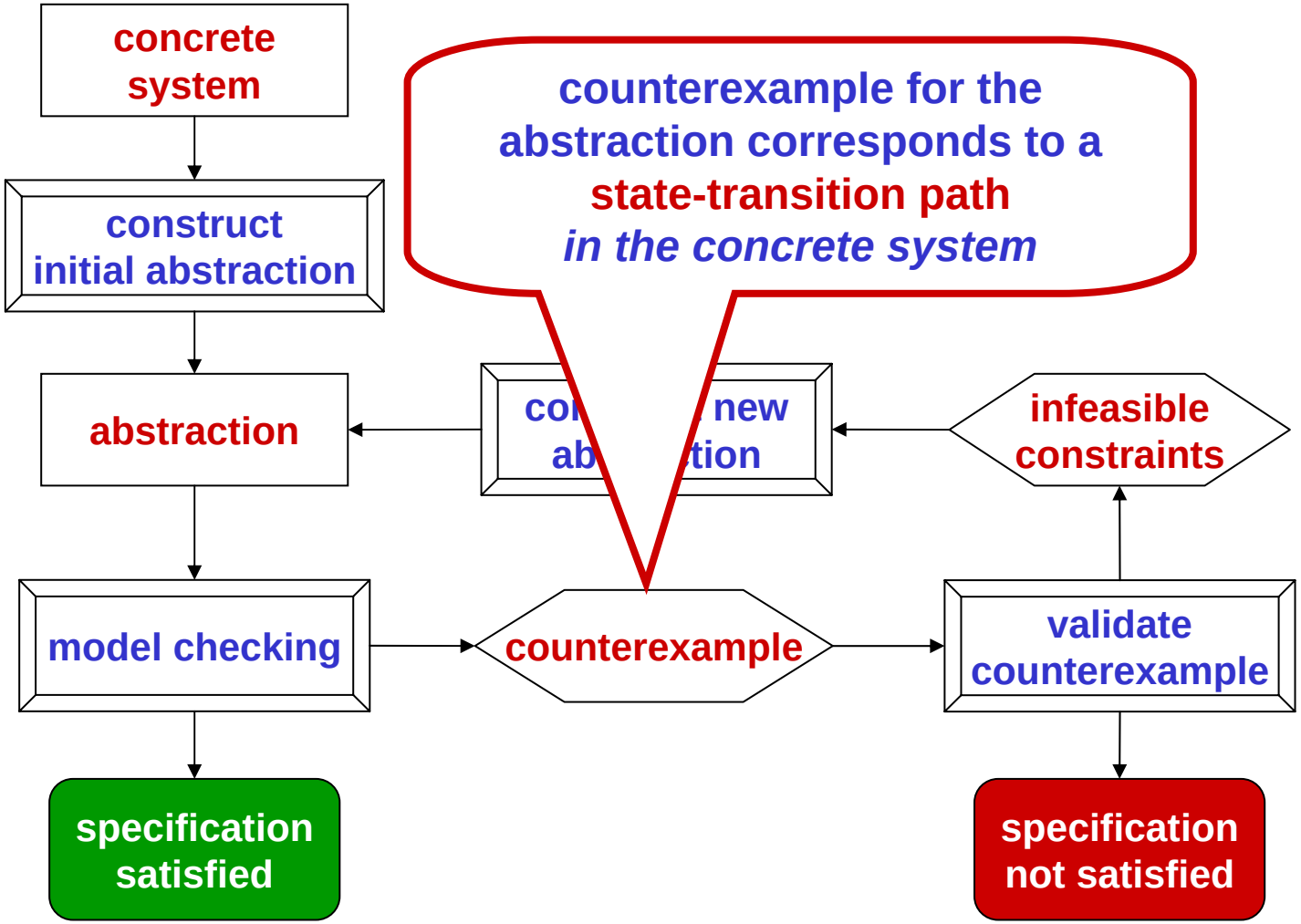
CEGAR


concrete system
construct initial abstraction
abstraction
specification
model checking
specification satisfied
model check the abstraction (faster than for the concrete system)
onstruct new abstraction
infeasible constraints
counterexample
validate counterexample
specification not satisfied

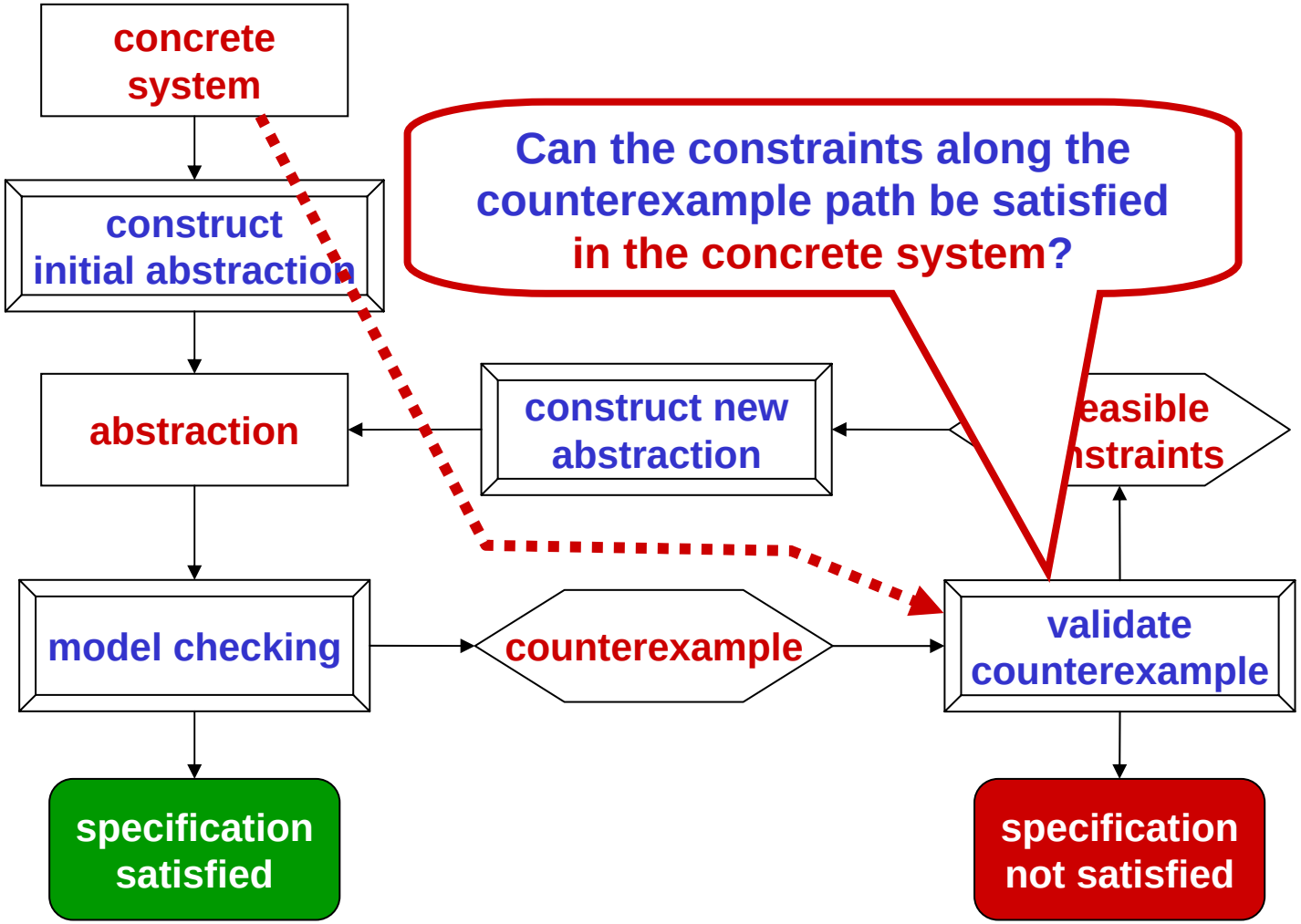
CEGAR


concrete system
construct initial abstraction
abstraction
model checking
specification satisfied
no counterexample ⇒ specification satisfied for the concrete system
uct new straction
infeasible constraints
counterexample
validate counterexample
specification not satisfied

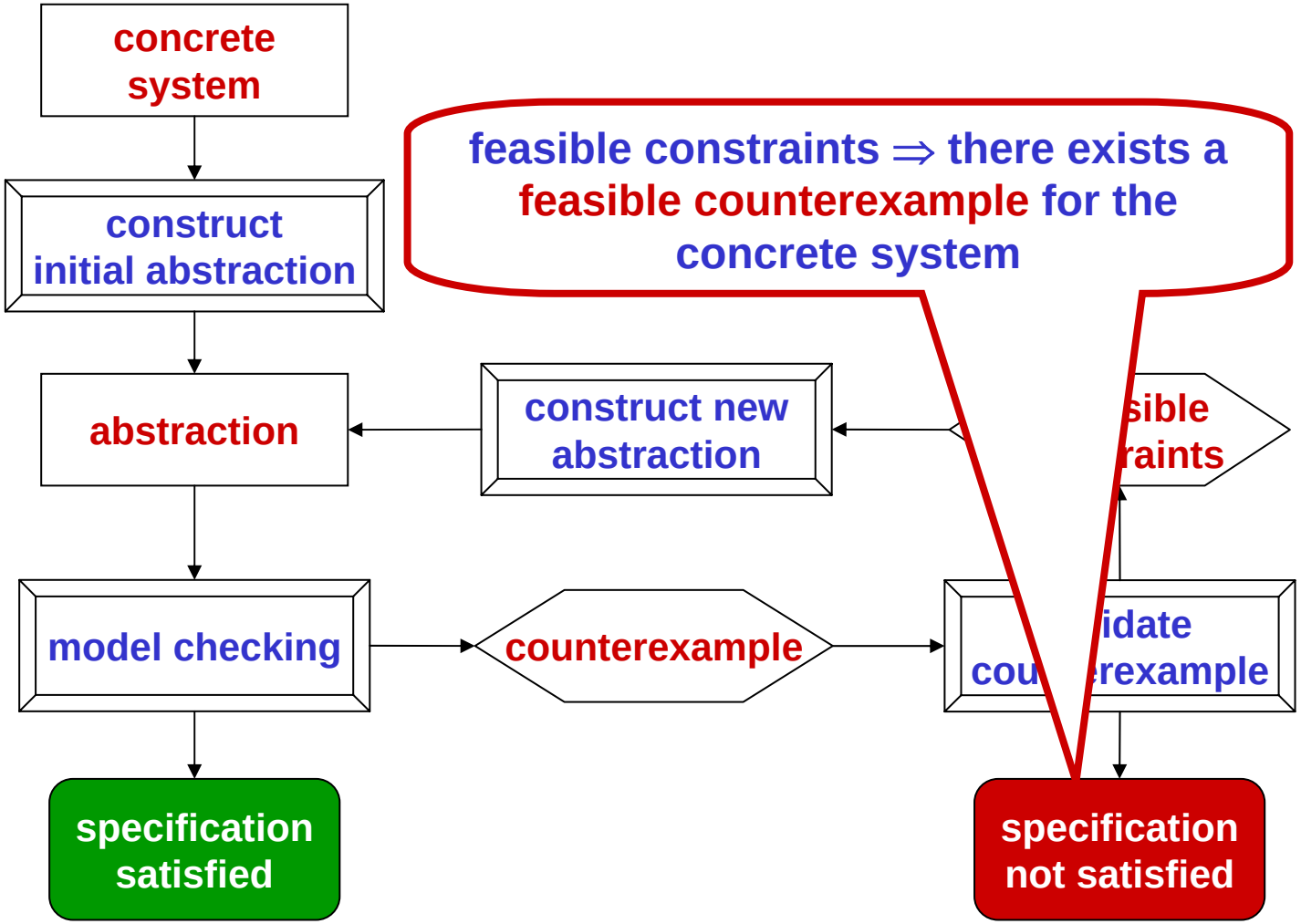
CEGAR


concrete system
construct initial abstraction
abstraction
model checking
specification satisfied
counterexample for the abstraction corresponds to a state-transition path in the concrete system
counterexample
infeasible constraints
validate counterexample
specification not satisfied

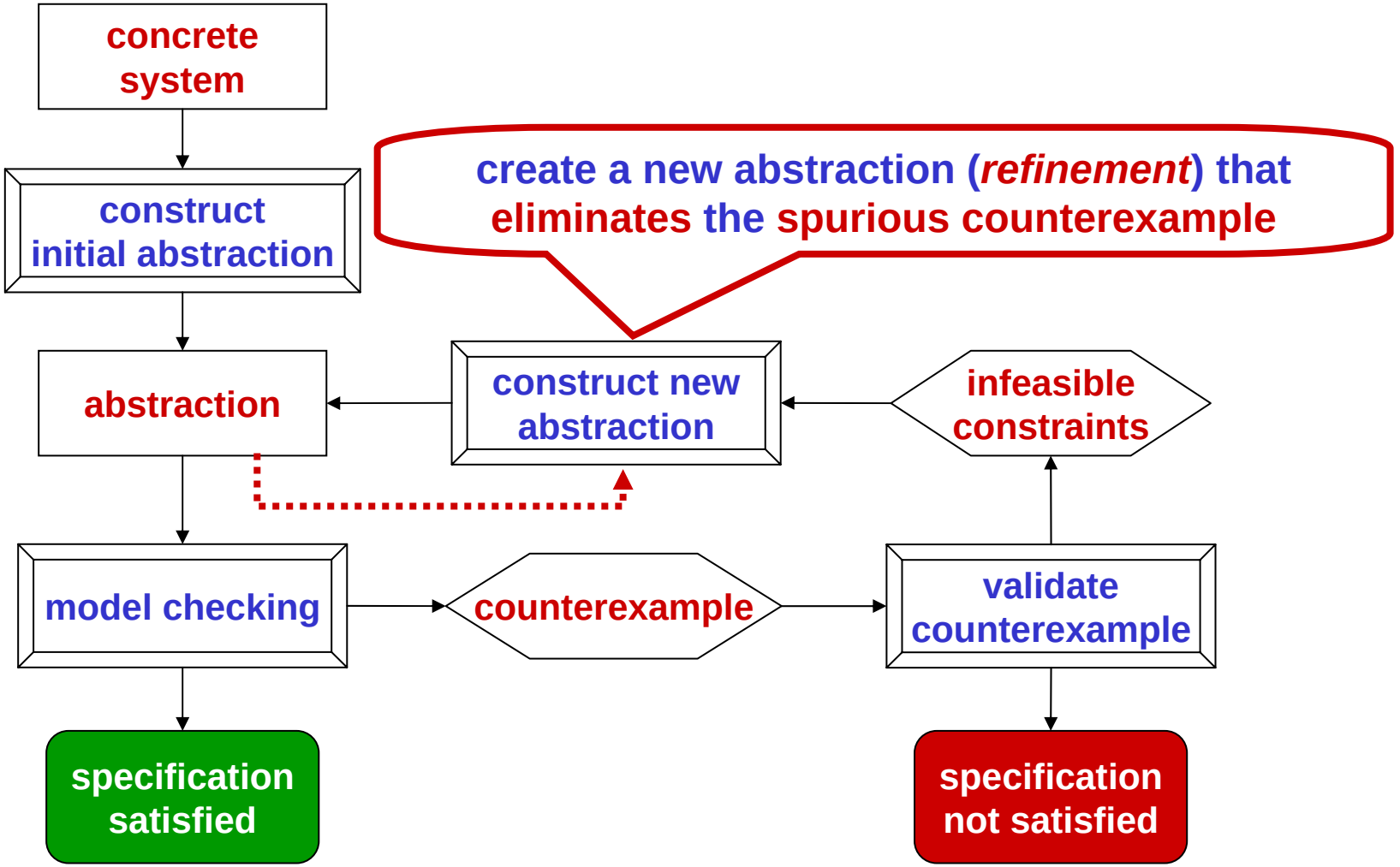
CEGAR


concrete system
construct initial abstraction
abstraction
model checking
specification satisfied
construct new abstraction
counterexample
Can the constraints along the counterexample path be satisfied in the concrete system?
easible nstraints
validate counterexample
specification not satisfied

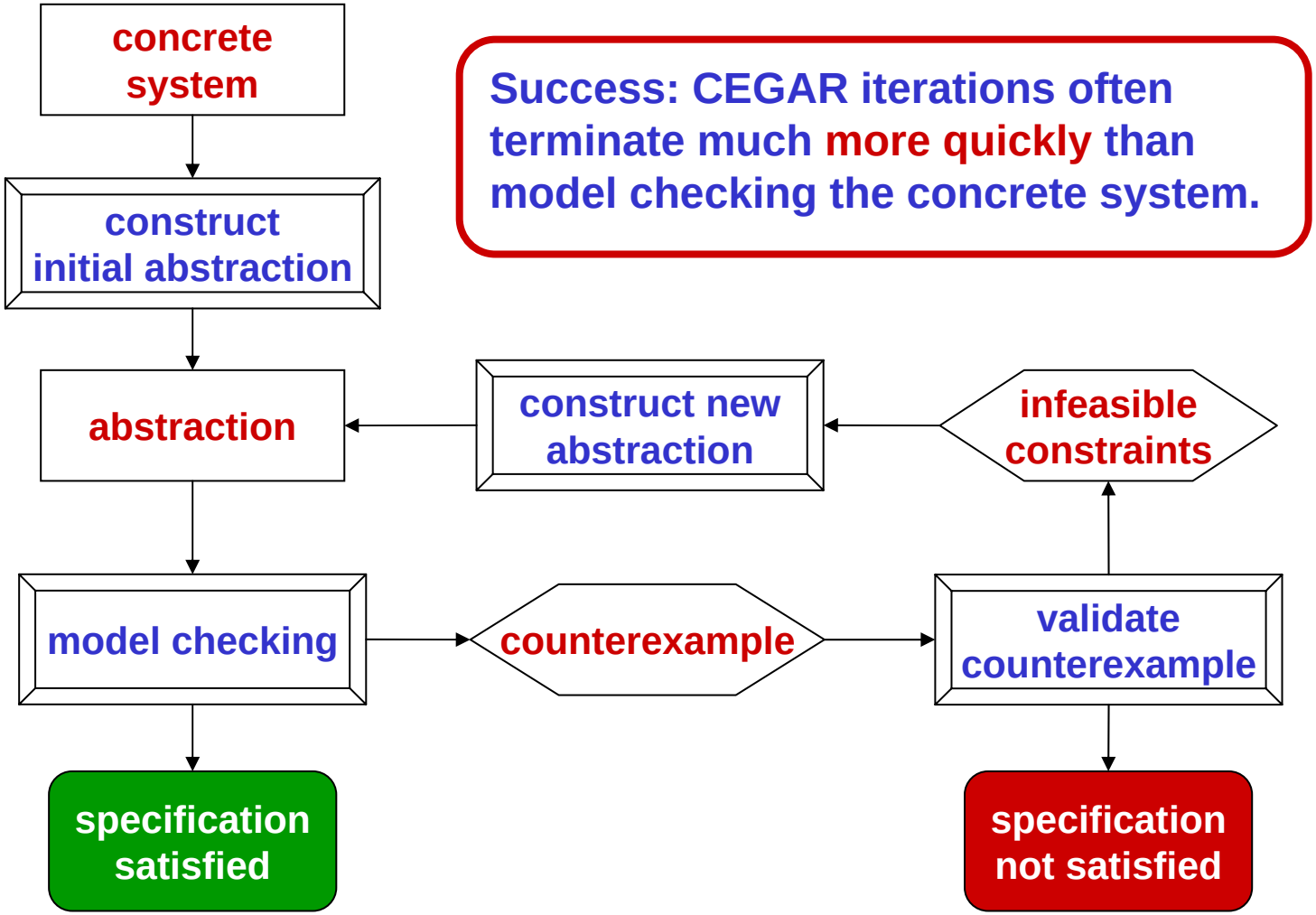
CEGAR


concrete system
construct initial abstraction
abstraction
model checking
specification satisfied
construct new abstraction
counterexample
feasible constraints ⇒ there exists a feasible counterexample for the concrete system
specification not satisfied

CEGAR


concrete system
construct initial abstraction
abstraction
model checking
specification satisfied
counterexample
validate counterexample
specification not satisfied
infeasible constraints
construct new abstraction
create a new abstraction (*refinement*) that eliminates the spurious counterexample

CEGAR


concrete system
construct initial abstraction
abstraction
model checking
specification satisfied
counterexample
construct new abstraction
infeasible constraints
validate counterexample
specification not satisfied
Success: CEGAR iterations often terminate much more quickly than model checking the concrete system.

CEGAR for Discrete Systems

- **Leverages**
  - Power of model checking on **simpler models**
  - Power of decision procedures / SAT solvers to **validate counterexamples**
- ***Empirically* a very powerful approach**
- **Many success stories**
  - **SLAM** : Verifying Device Drivers at Microsoft
    - Actually ships as a commercial product **Static Driver Verifier (SDV)**
  - Many software model checkers developed
    - MAGIC, BLAST, CBMC

CEGAR for Hybrid Systems

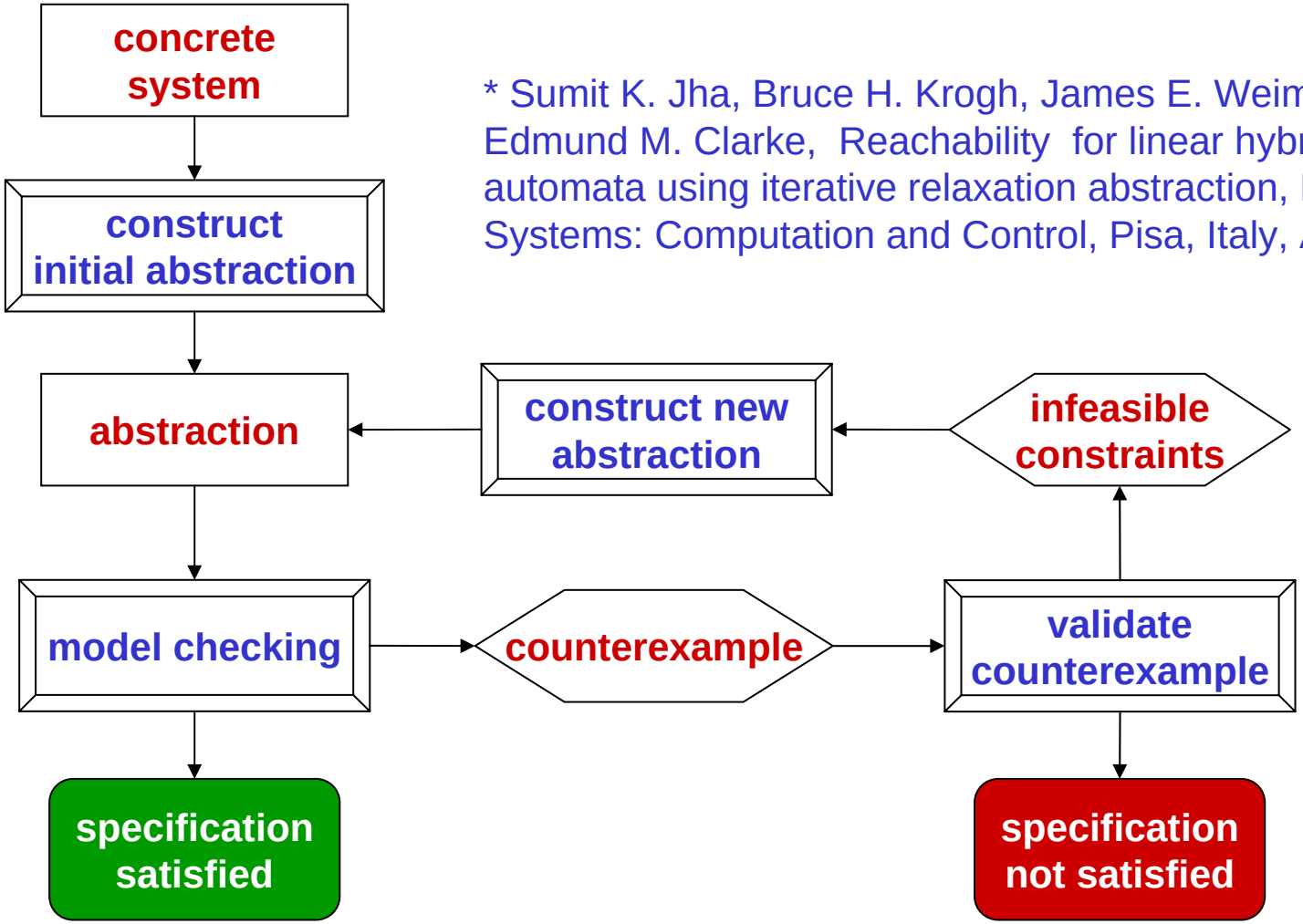
- **First attempt***
  - abstractions constructed by over approximating reachable sets in continuous state space
  - eliminated counterexamples by splitting locations
- **slow convergence: refinement eliminates only path at a time**
- **HS reachability limited to low dimensional systems (~5 continuous state variables)**

* E. Clarke, A. Fehnker, Z. Han, B. Krogh, J. Ouaknine, O. Stursberg, M. Theobald, Abstraction and counterexample-guided refinement in model checking of hybrid systems, *International Journal of Foundations of Computer Science*, Special Issue on Verification and Analysis of Infinite State Systems, August 2003, pp. 583-604.

Alternative to CEGAR for Hybrid Systems

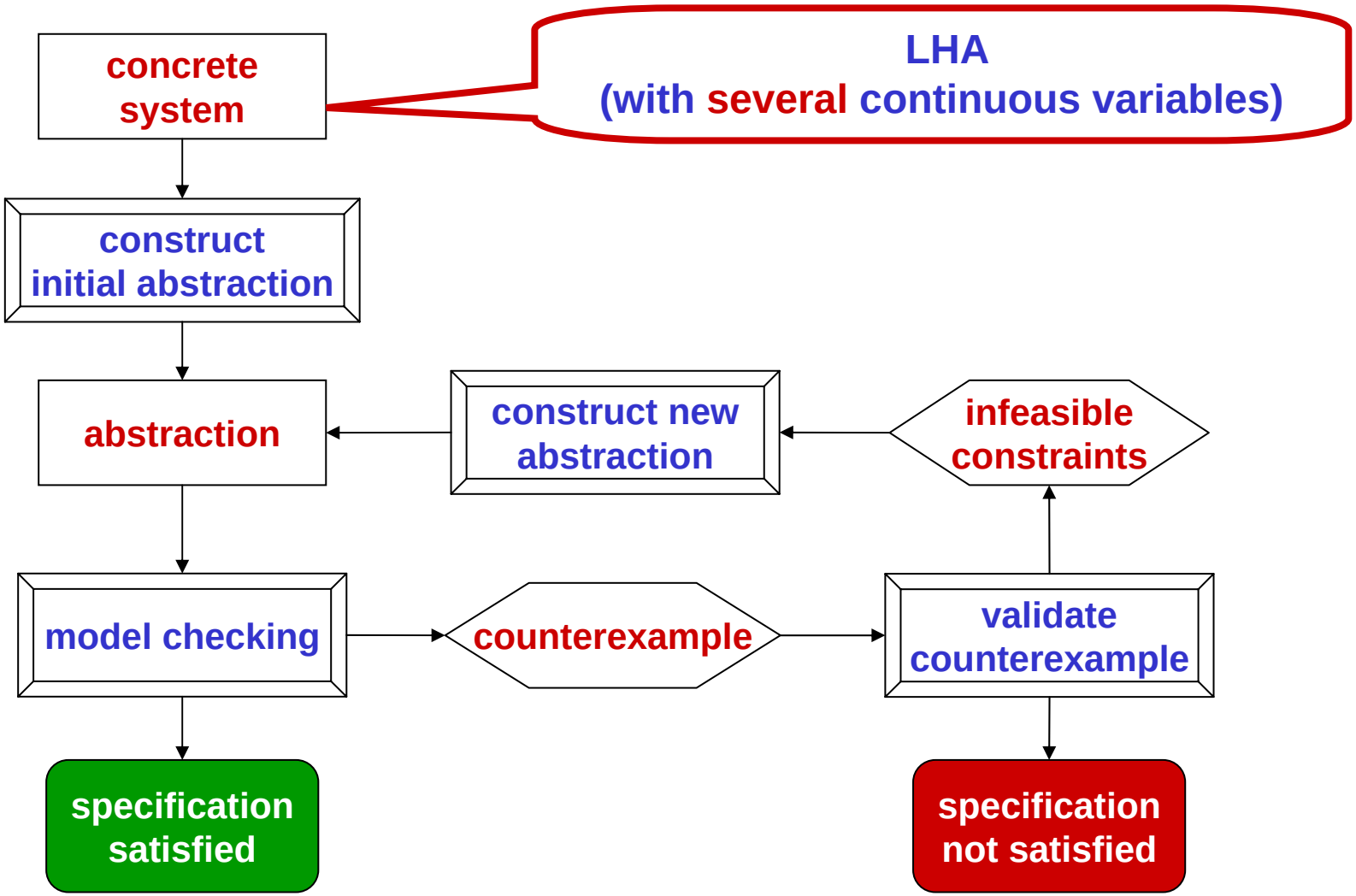
- **Construct abstractions by reducing the number of continuous state variables (*relaxations*)**
- **Retain the discrete state structure**
- **Rather than refine abstractions (which would add continuous state variables)**
  - construct a counterexample automaton representing *all* counterexamples for each abstraction
  - "intersect" the counterexample automaton with the intersection of all previous counterexample automata
- **Stop when**
  - a feasible counterexample is found, OR
  - there are no more counterexamples

Iterative Relaxation Abstraction (IRA) for Linear Hybrid Automata (LHA)*

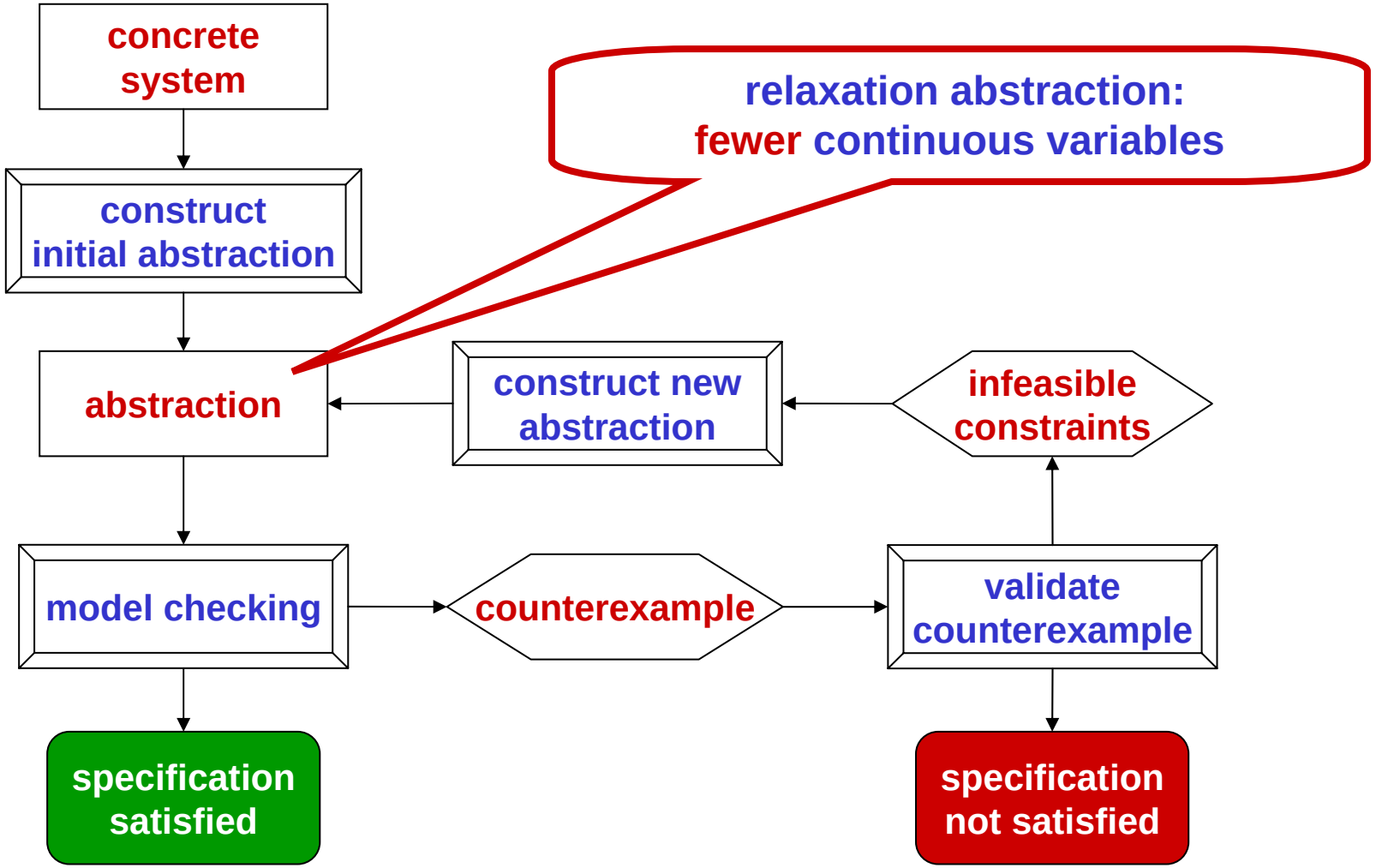


* Sumit K. Jha, Bruce H. Krogh, James E. Weimer, Edmund M. Clarke, Reachability for linear hybrid automata using iterative relaxation abstraction, Hybrid Systems: Computation and Control, Pisa, Italy, April 2007.

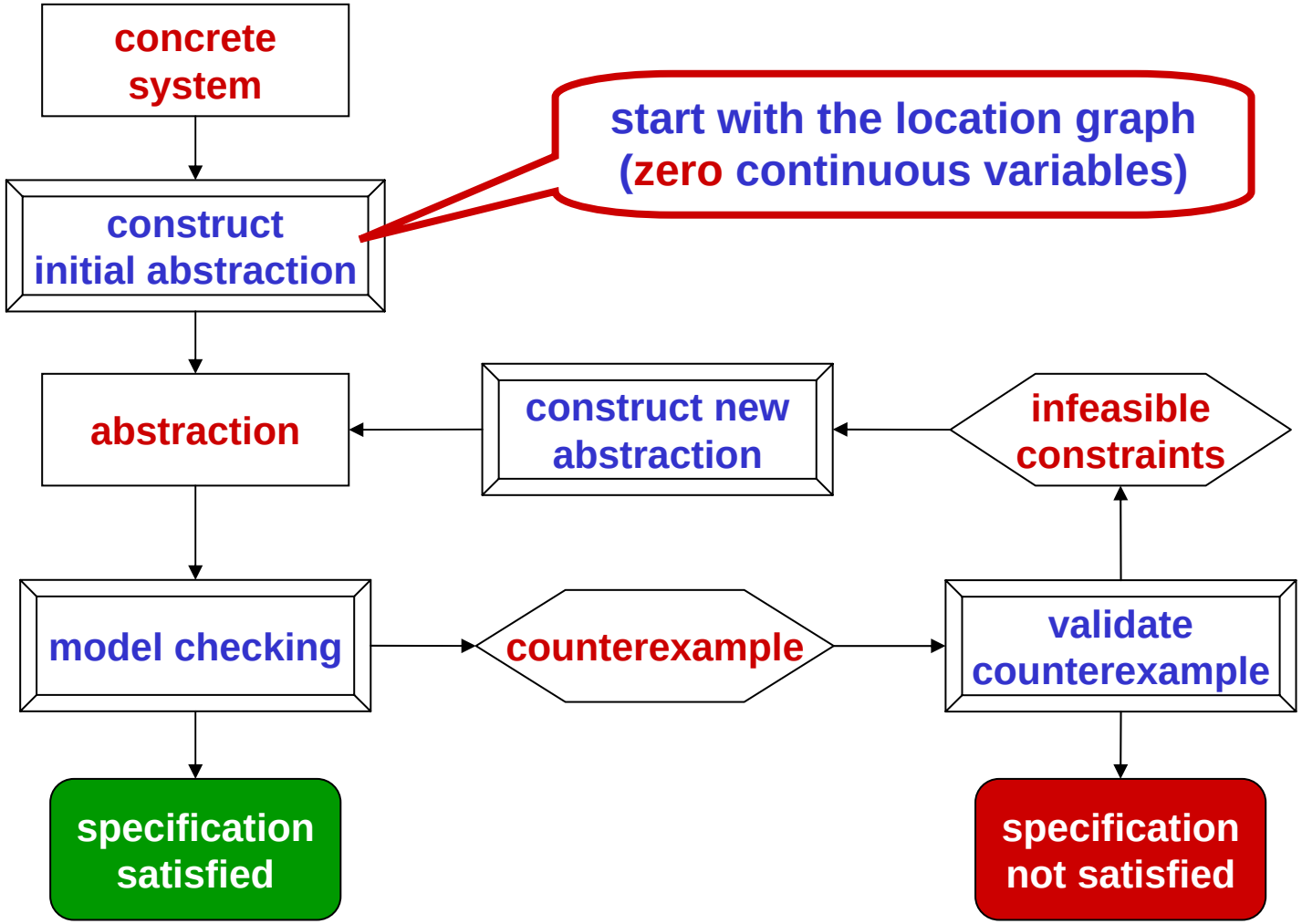
IRA for LHA


concrete system
LHA
(with several continuous variables)
construct initial abstraction
abstraction
construct new abstraction
infeasible constraints
model checking
counterexample
validate counterexample
specification satisfied
specification not satisfied

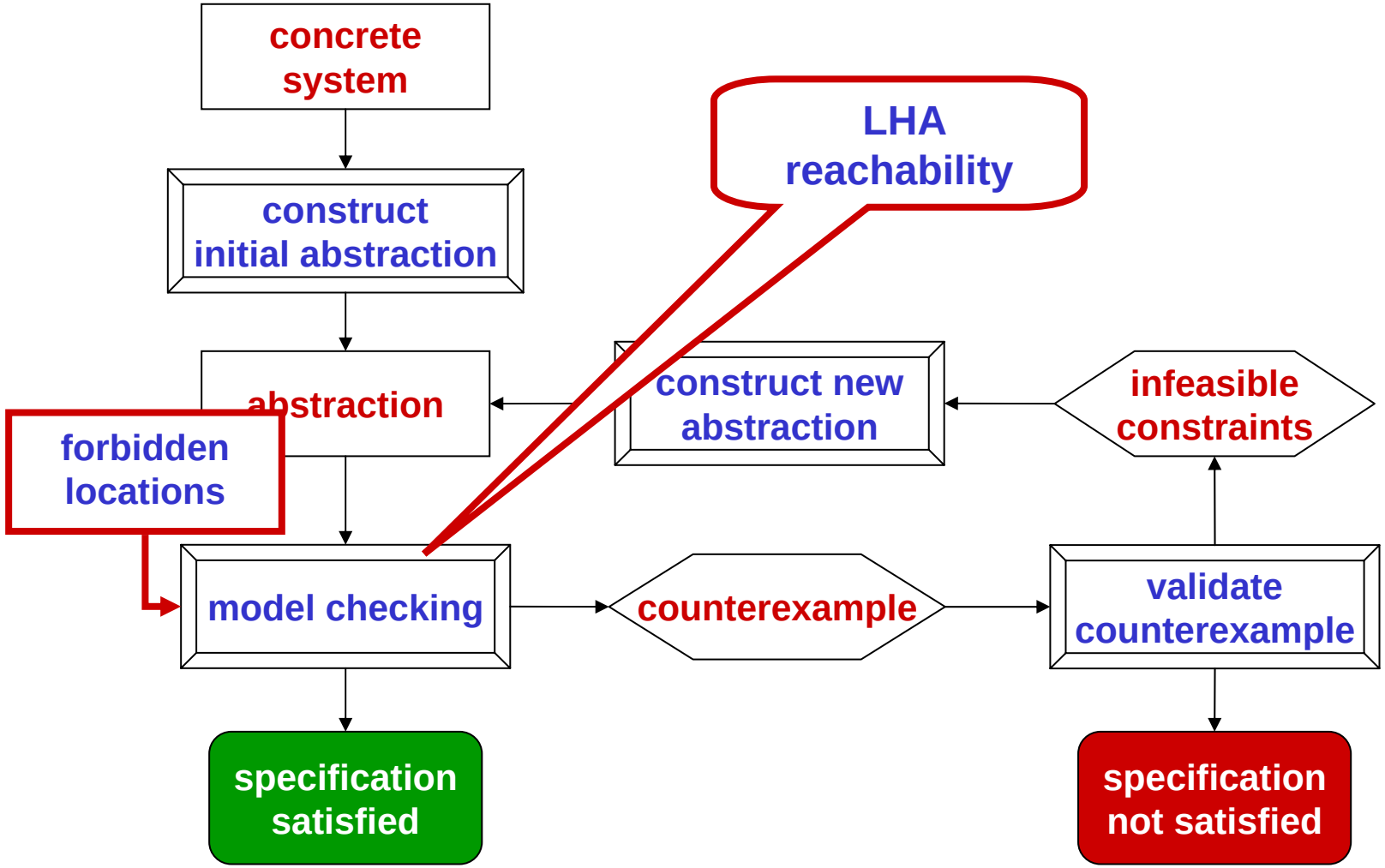
IRA for LHA


concrete system
relaxation abstraction: fewer continuous variables
construct initial abstraction
abstraction
construct new abstraction
infeasible constraints
model checking
counterexample
validate counterexample
specification satisfied
specification not satisfied

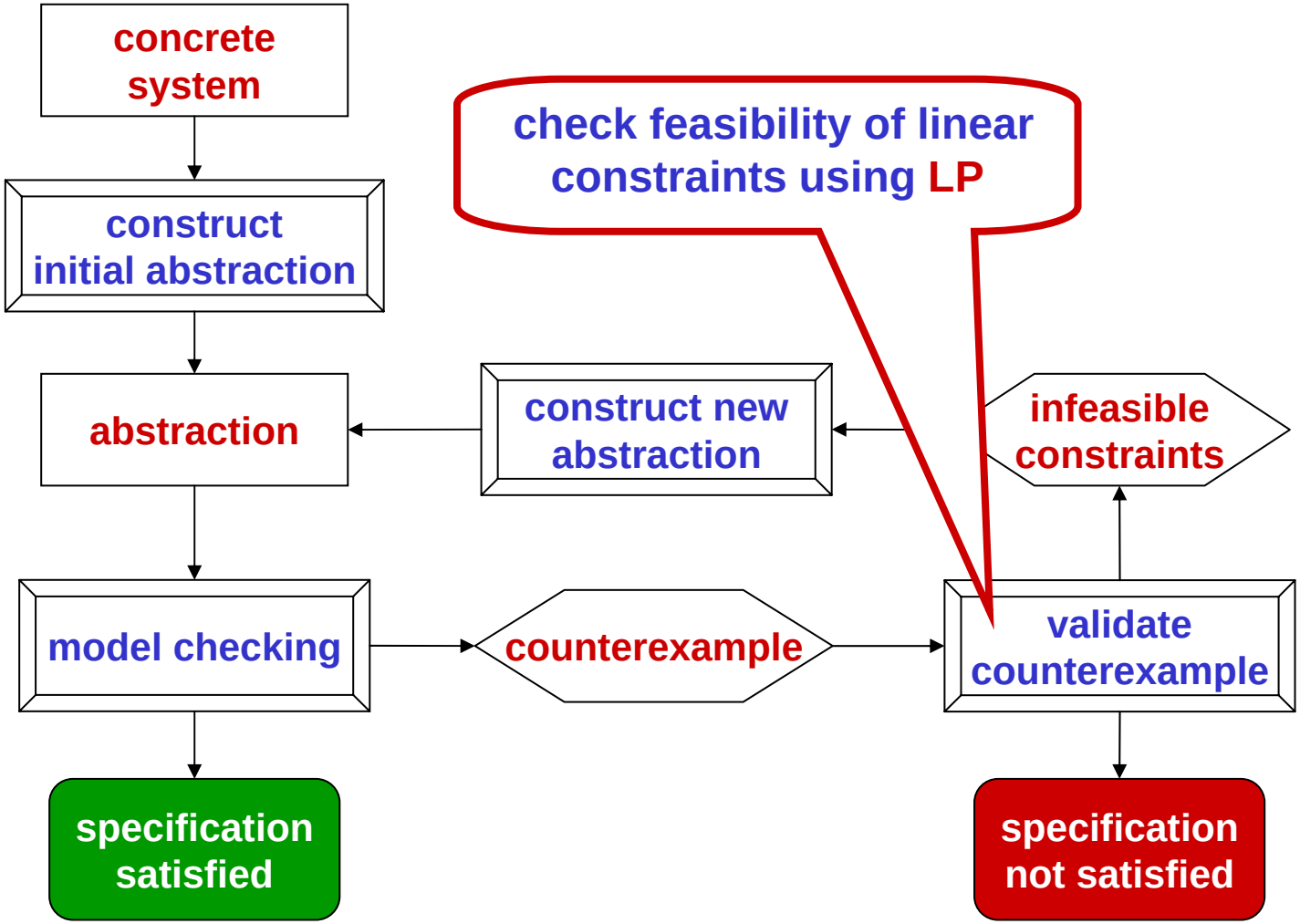
IRA for LHA


concrete system
start with the location graph (zero continuous variables)
construct initial abstraction
abstraction
construct new abstraction
infeasible constraints
model checking
counterexample
validate counterexample
specification satisfied
specification not satisfied

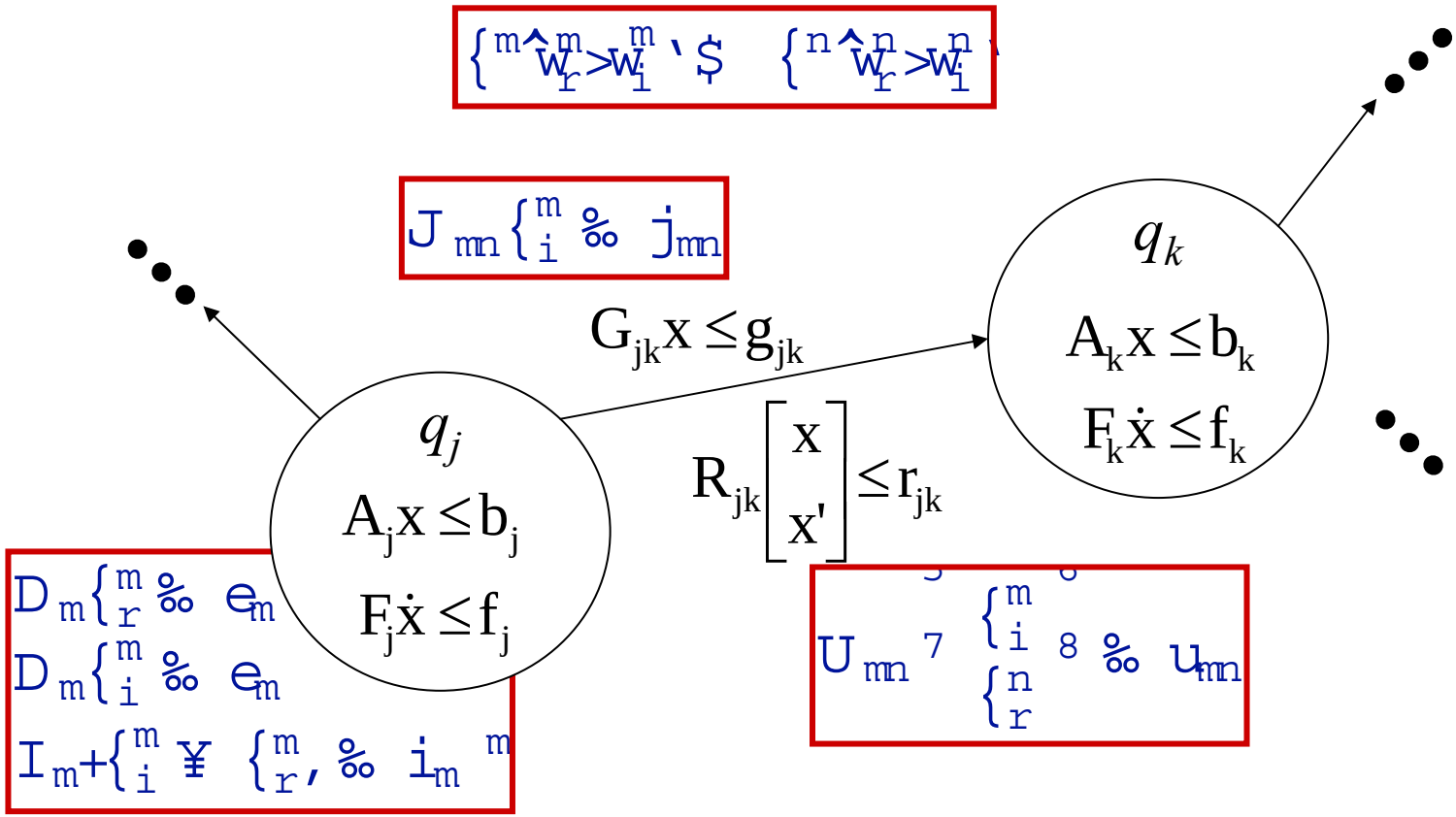
IRA for LHA


concrete system
construct initial abstraction
abstraction
forbidden locations
model checking
specification satisfied
counterexample
LHA reachability
construct new abstraction
infeasible constraints
validate counterexample
specification not satisfied

IRA for LHA

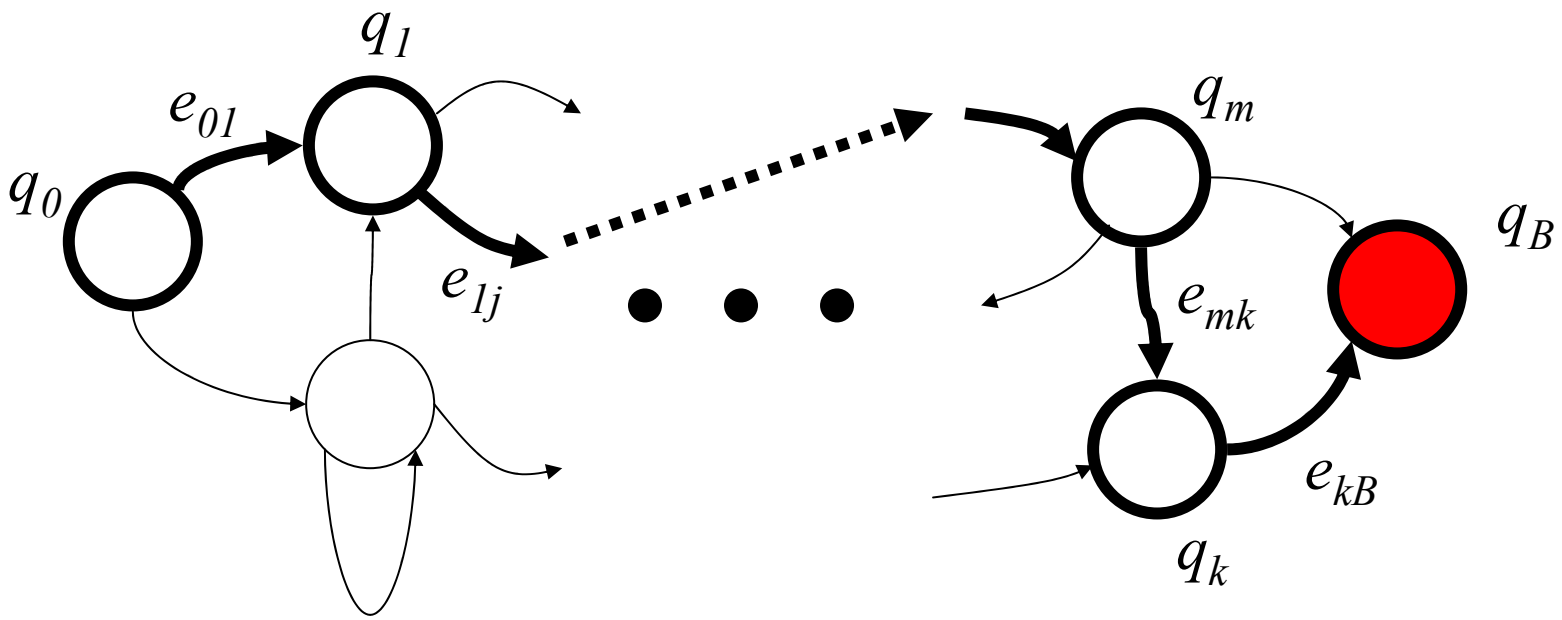

concrete system
construct initial abstraction
abstraction
model checking
specification satisfied
construct new abstraction
counterexample
check feasibility of linear constraints using LP
infeasible constraints
validate counterexample
specification not satisfied

LHA Path Constraints



Feasible Counterexamples*

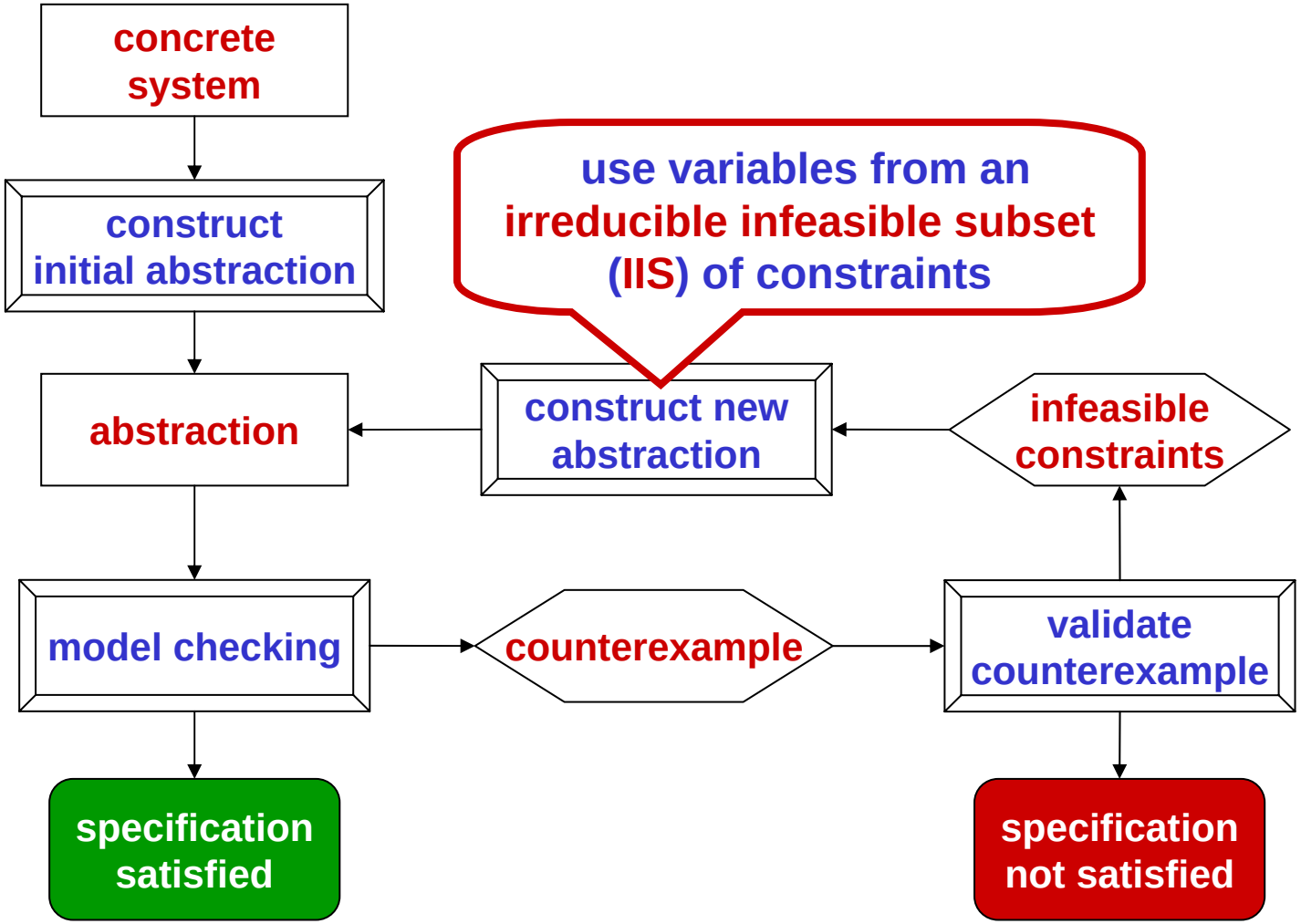
- $\pi = q_0\, e_{01}\, q_1\, e_{1j} \ldots e_{mk}\, q_k\, e_{kj}\, q_B$



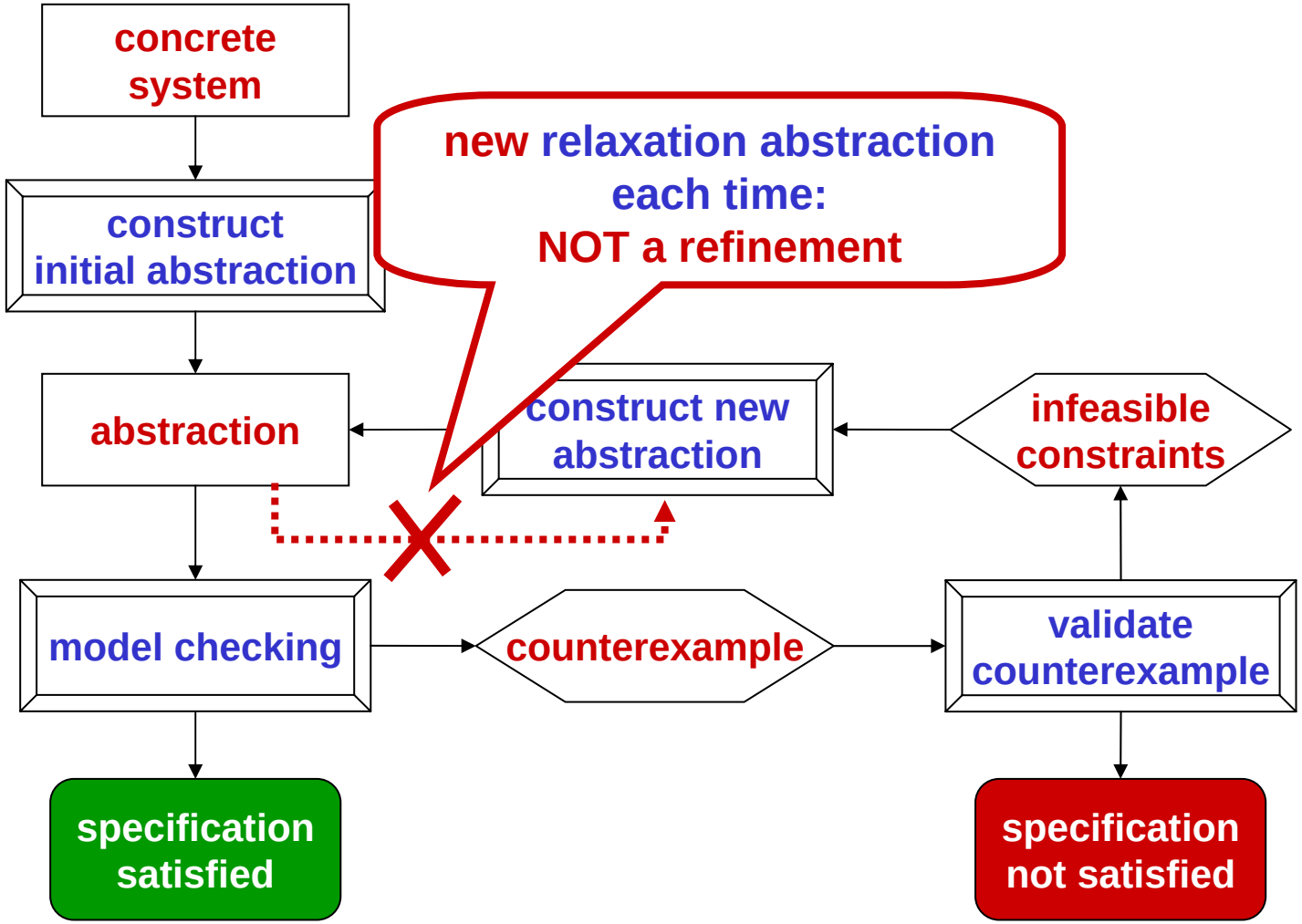
Path π is feasible if PathCon(π) is satisfiable, where PathCon(π) is the set of linear constraints over the variables:

$$\{^{r}_{r}\}\{^{r}_{i}\}\ ^{r}\}\{^{4}_{r}\}\{^{4}_{i}\}\ ^{4}\}==\!=\}\{^{n}_{r}\}\{^{n}_{i}\}\ ^{n}r\}\{^{4}_{E}$$

IRA for LHA


concrete system
construct initial abstraction
abstraction
model checking
specification satisfied
use variables from an irreducible infeasible subset (IIS) of constraints
construct new abstraction
infeasible constraints
counterexample
validate counterexample
specification not satisfied

IRA for LHA


concrete system
construct initial abstraction
abstraction
model checking
specification satisfied
new relaxation abstraction each time:
NOT a refinement
construct new abstraction
infeasible constraints
counterexample
validate counterexample
specification not satisfied

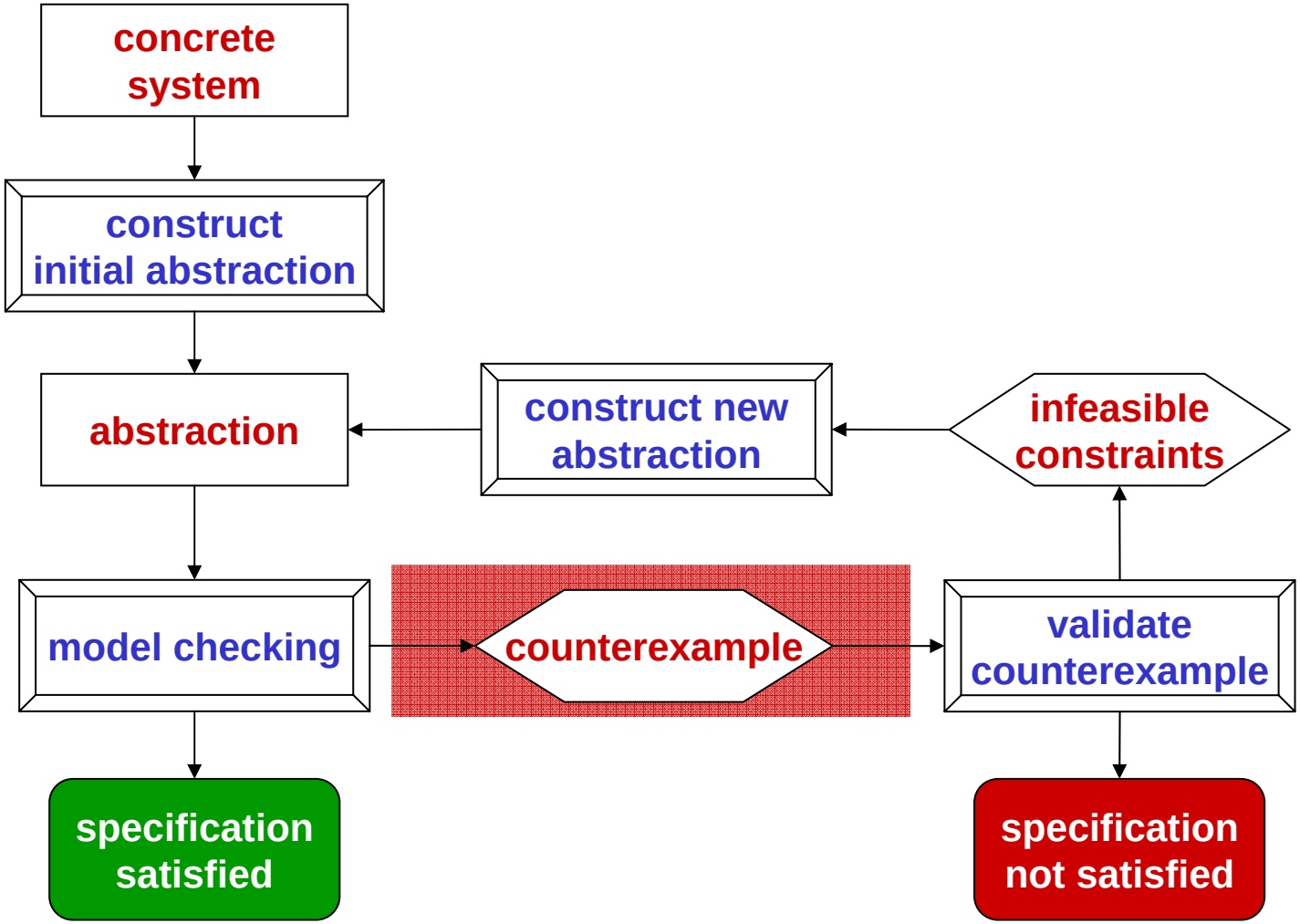
IRA for LHA – Leverages:

- Power of LHA reachability on low-order LHA models
- Power of LP to validate counterexamples involving huge number of continuous variables.
- Ability of a LP solver to identify an irreducible infeasible subset for an infeasible LP
- Inspired by CEGAR for discrete systems, but variables are not added to refine abstractions

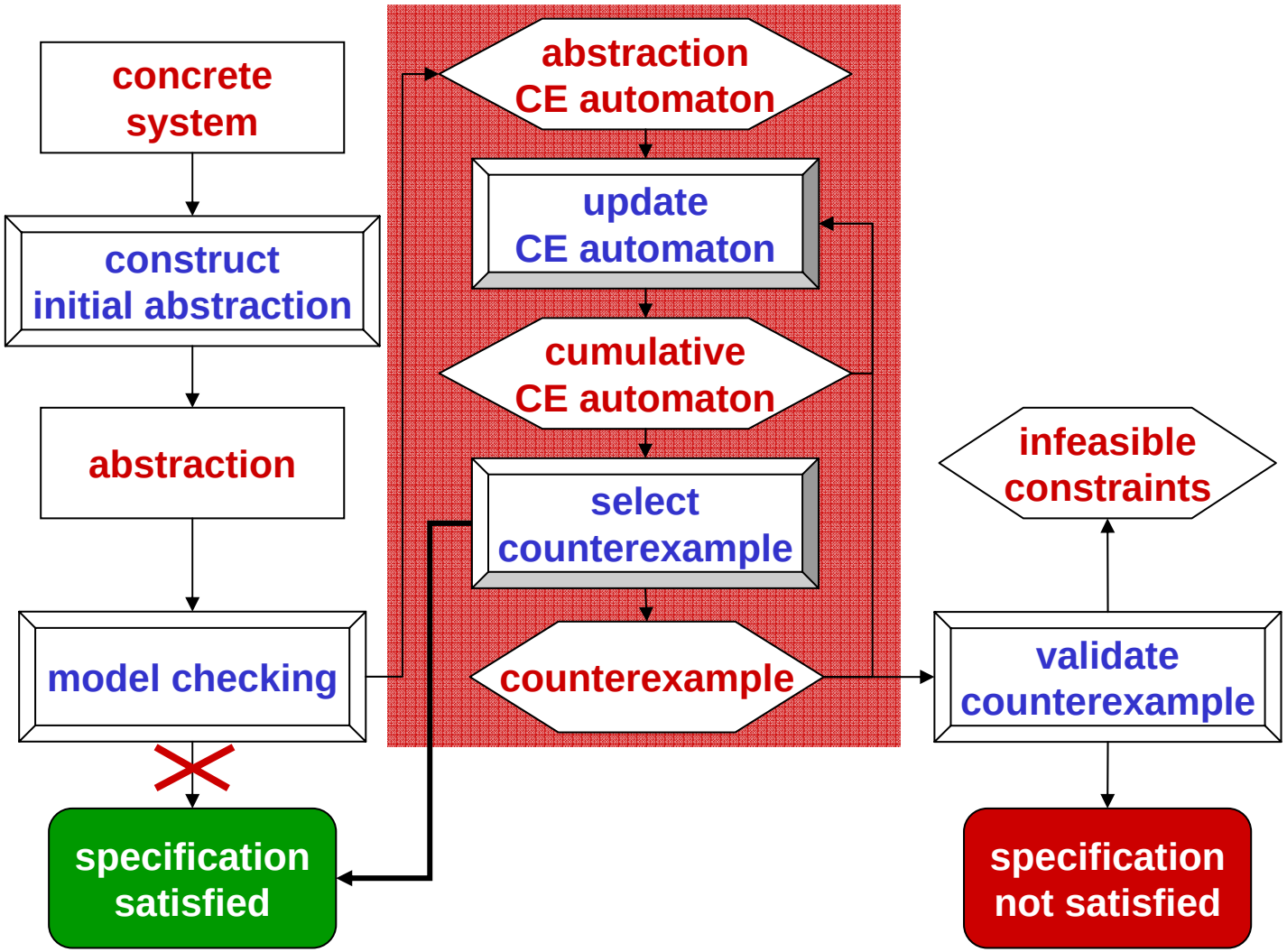
Relaxation Abstractions for LHS

- **Given a subset of continuous state variables V**
- **Replace linear constraints with relaxed constraints involving only variables in V**
- **Not unique – various relaxations**
  - Drop constraints involving variables not in V (**localization**)
  - Quantifier Elimination (**Fourier-Motzkin**)

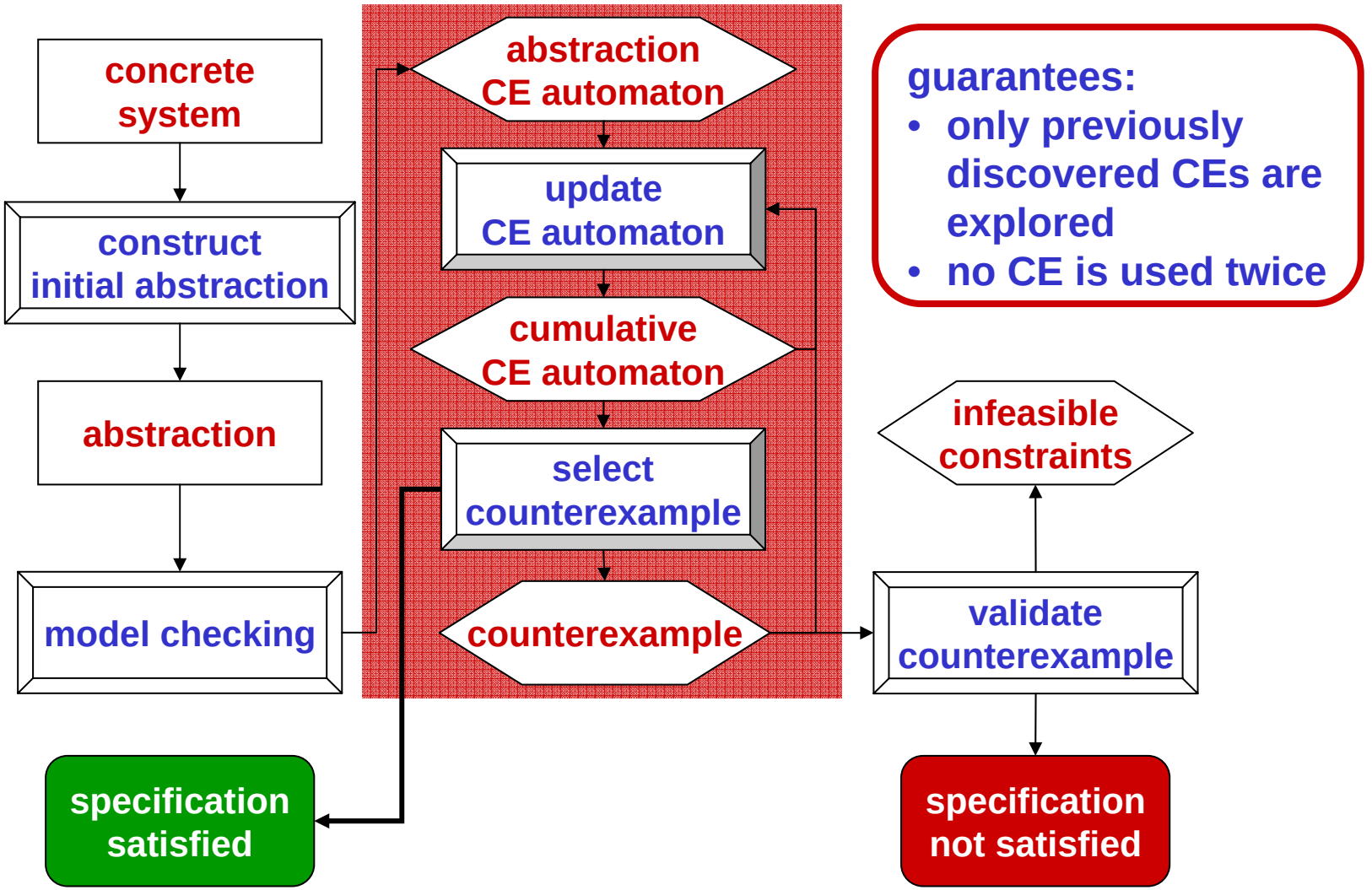
IRA for LHA
selecting counterexamples



IRA for LHA
selecting counterexamples



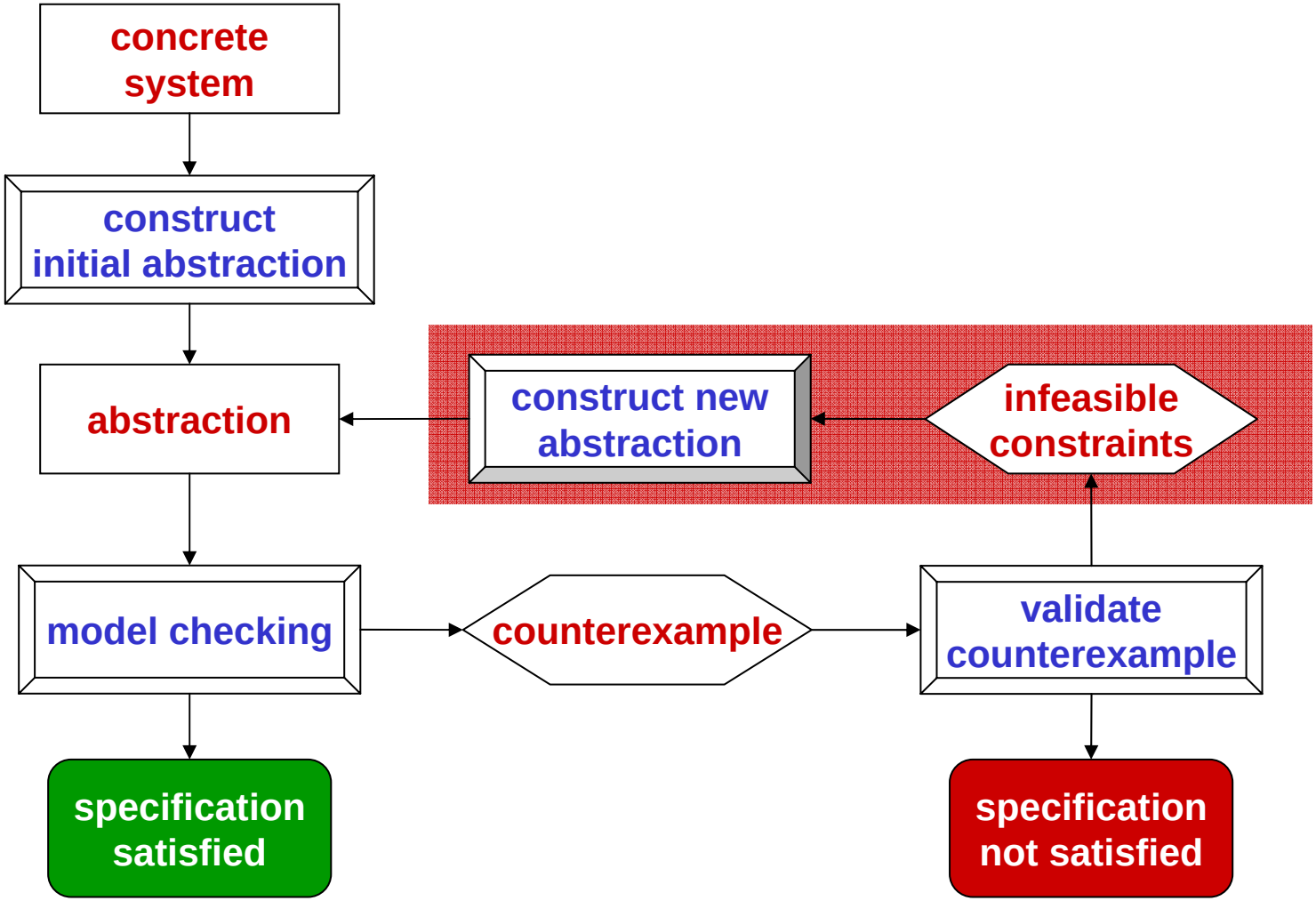
IRA for LHA
selecting counterexamples



guarantees:
- only previously discovered CEs are explored
- no CE is used twice

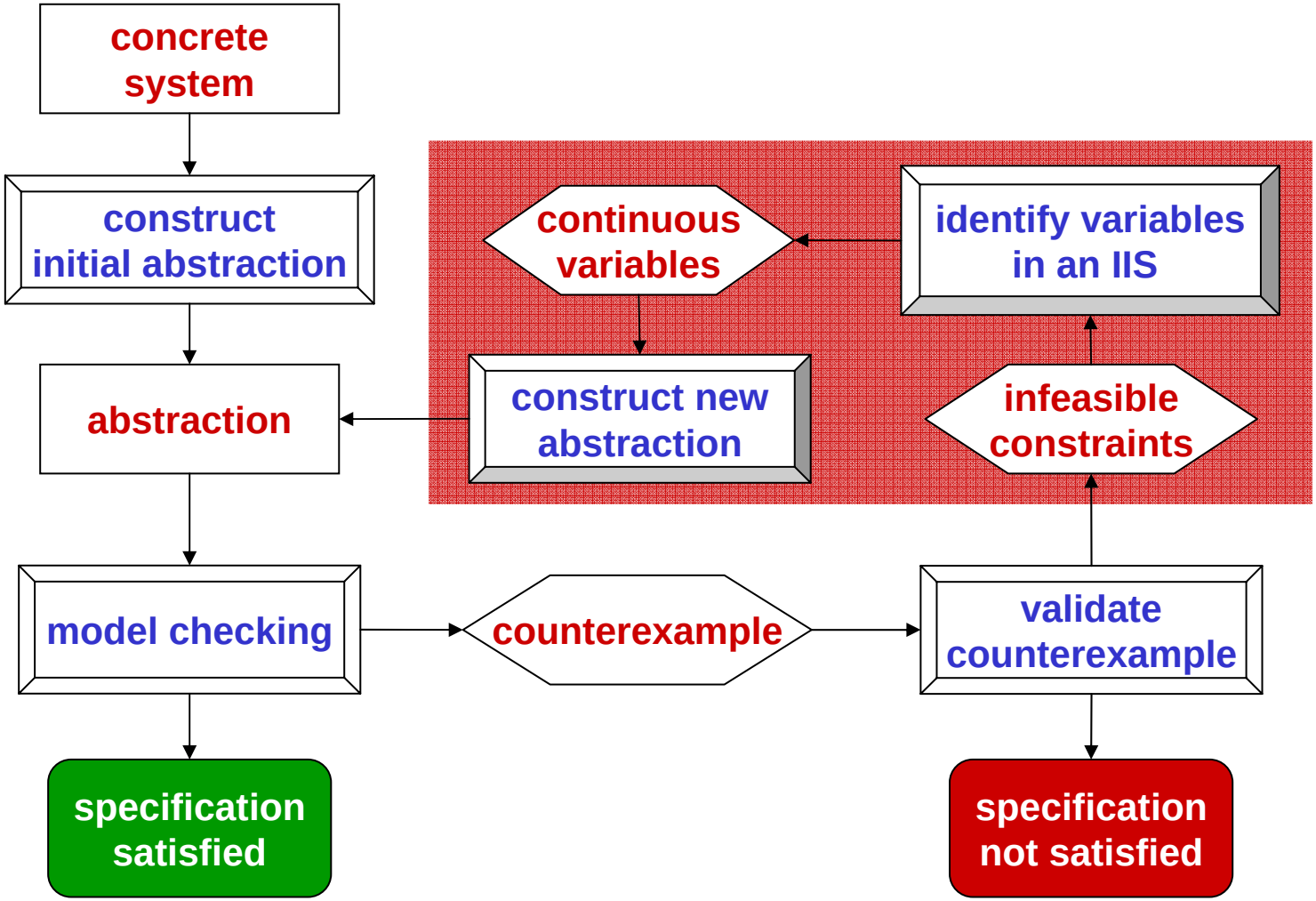
IRA for LHA

constructing new relaxation abstractions



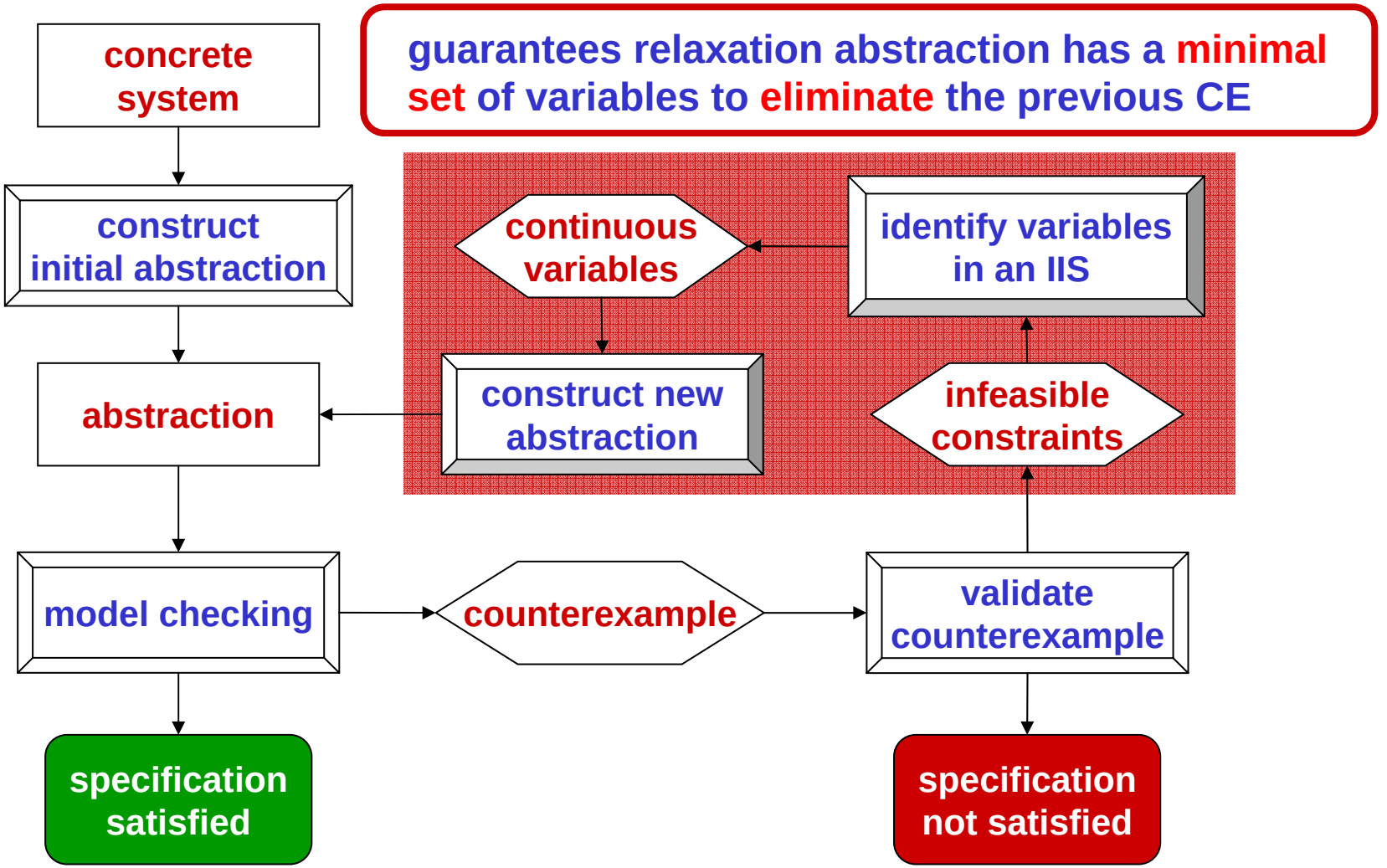
IRA for LHA

constructing new relaxation abstractions

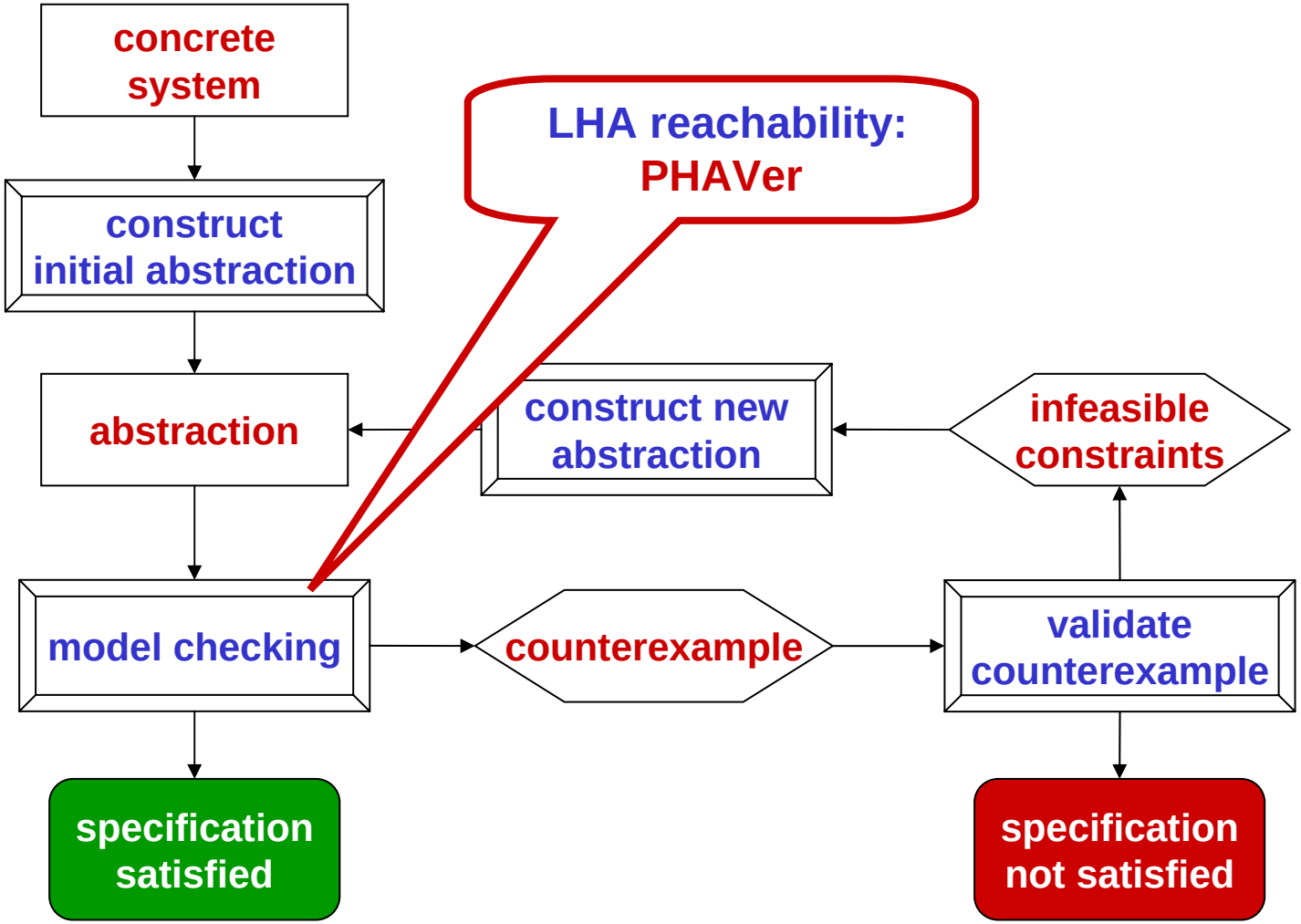


IRA for LHA

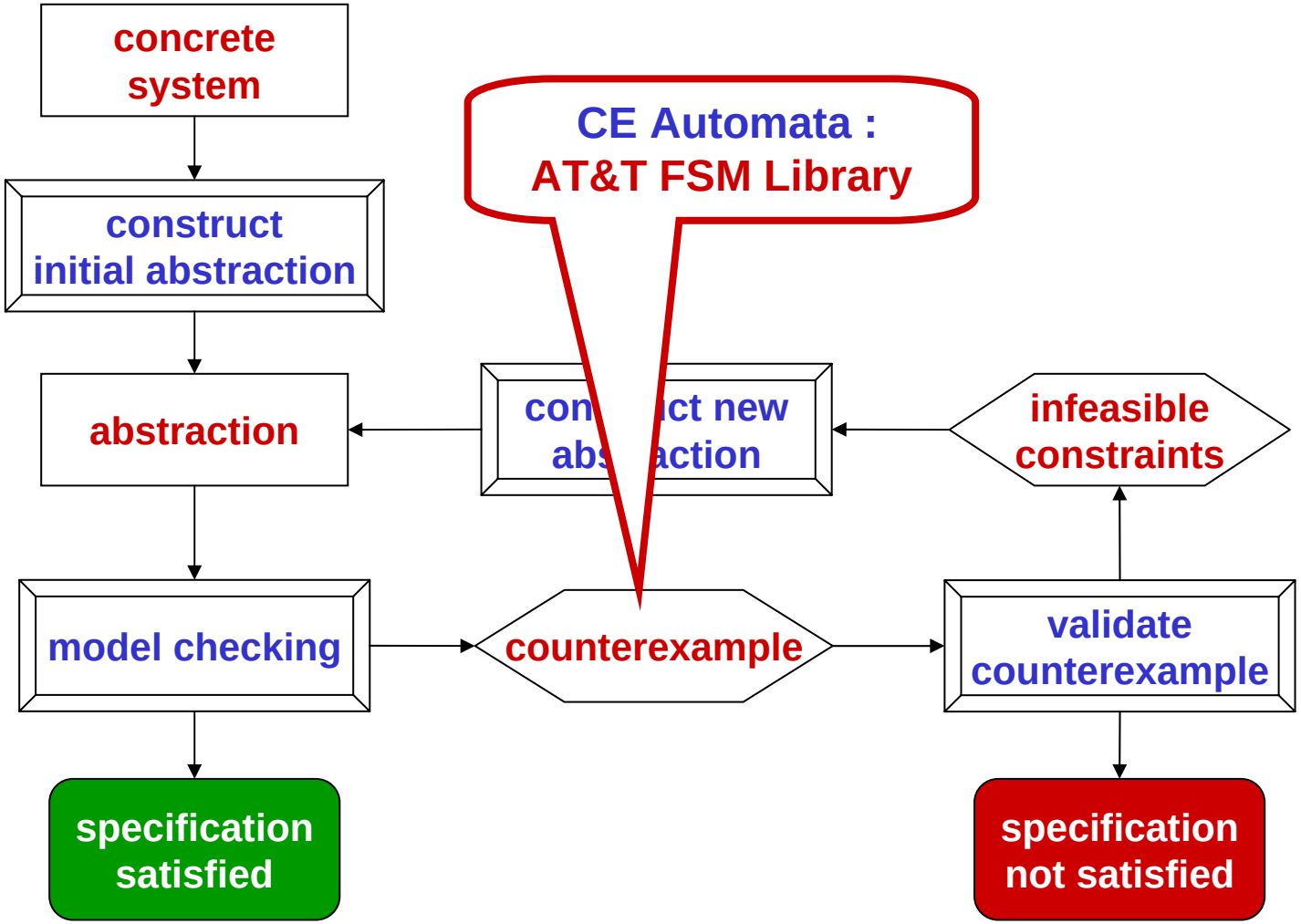
constructing new relaxation abstractions



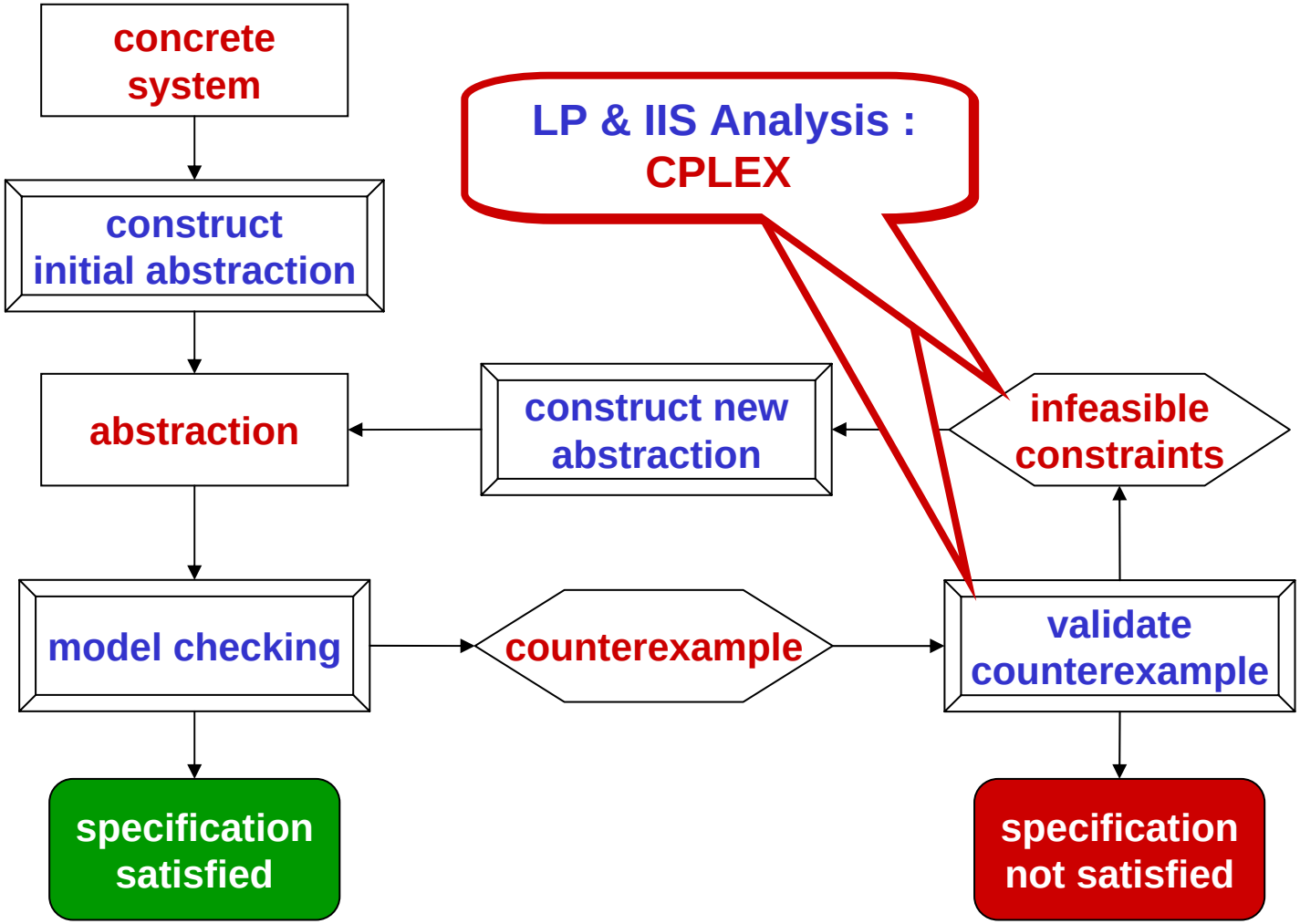
IRA for LHA implementation



IRA for LHA implementation



IRA for LHA
implementation



IRA vs. PHAVer for an Adaptive Cruise Control Example (time in sec)

No. of Variables	PHAVer	IRA – Localization	IRA Fourier-Motzkin
6	0.26	1.34	61.05
8	0.96	5.11	170.11
10	8.21	17.76	402.15
12	147.11	50.04	933.47
14	7007.51	123.73	1521.95
15	70090.06	181.74	2503.59
16	*did not complete*	*267.46*	*3519.51*

IRA vs. PHAVer for an Adaptive Cruise Control Example (time in sec)

No. of Vari	PHAVer	IRA – Localization	IRA Fourier-Motzkin
		1.34	61.05
8	0.[illegible]	5.11	170.11
10	8.21	17.76	402.15
12	147.11	50.04	933.47
14	7007.51	123.73	1521.95
15	70090.06	181.74	2503.59
16	*did not complete*	*267.46*	*3519.51*

IRA becomes faster for ≥ 12 variables

IRA vs. PHAVer for an Adaptive Cruise Control Example (time in sec)

No. of Variables	PHAVer	IRA – Localization	IRA Fourier-Motzkin
6			61.05
8			170.11
10	8.21		402.15
12	147.11	50.04	933.47
14	7007.51	123.73	1521.95
15	70090.06	181.74	2503.59
16	*did not complete*	*267.46*	*3519.51*

IRA-FM becomes faster for ≥ 14 variables

IRA vs. PHAVer for an Adaptive Cruise Control Example (time in sec)

No. of Variables	PHAVer	IRA – Localization	IRA Fourier-Motzkin
6	0.26	1.34	61.05
8	0.96	5.11	170.11
10	8.21	17.76	402.15
14	7007.[illegible]	123.73	1521.95
15	70090.06	181.74	2503.59
16	*did not complete*	*267.46*	*3519.51*

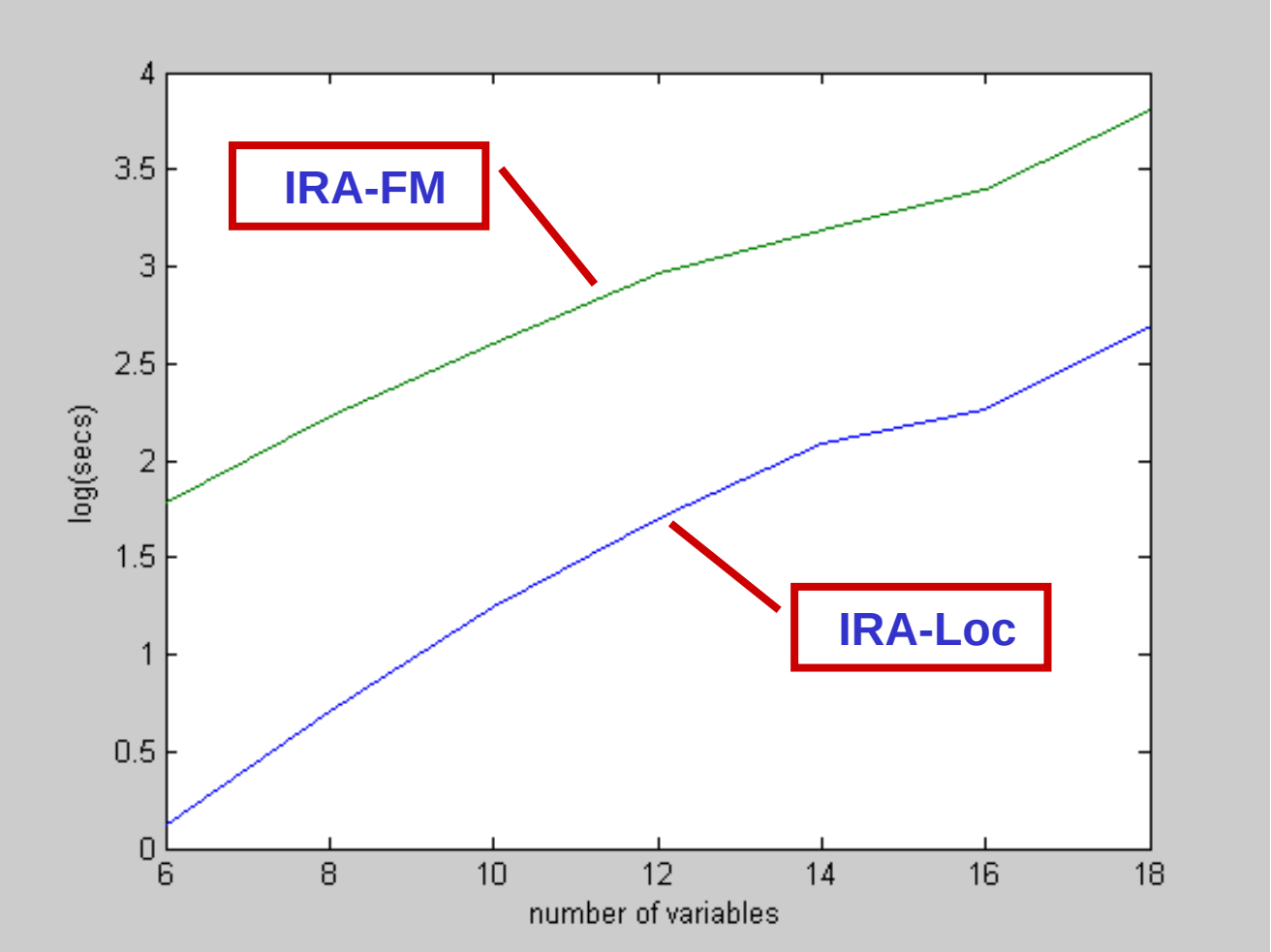
15 Vars: 19.5 hr. (PHAVer) vs. 3 min. (IRA-LOC)

IRA vs. PHAVer for an Adaptive Cruise Control Example (time in sec)

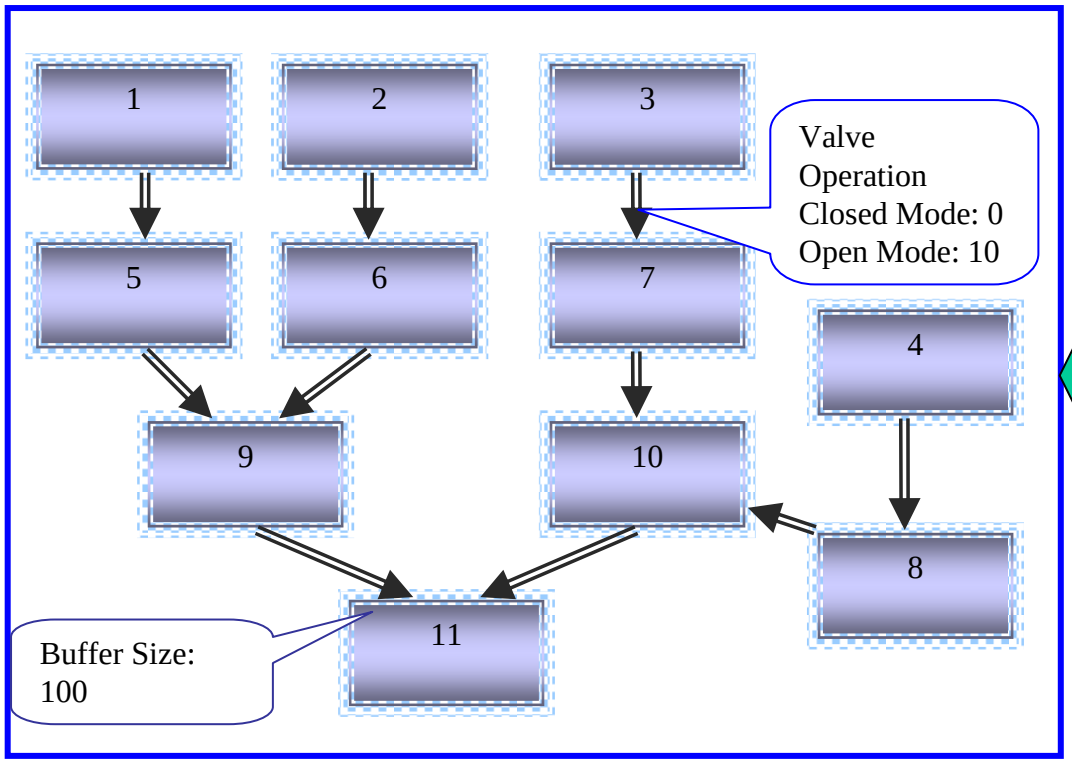
No. of Variables	PHAVer	IRA – Localization	IRA Fourier-Motzkin
6	0.26	1.34	61.05
8	0.96	5.11	170.11
10			402.15
12	147.1	50.04	933.47
14	7007.[illegible]	123.73	1521.95
15	70090.[illegible]6	181.74	2503.59
16	*did not complete*	*267.46*	*3519.51*

PHAVer fails to converge for 16 variables

IRA-Loc vs. IRA-FM



Switched Buffer Network*



*Frehse & Maler, HSCC '07

Controller

Hybrid automaton controlling the valves in the channels

- Buffers connected by pipes with valves.
- Valves have several modes
- Controller observes buffers and to switch valve modes
- Specification: No buffer overflow

Switched Buffer Network

- Implemented a simple controller with three locations and 11 continuous variables
- Design: sequence of actual counterexamples from IRA used to “tune” the control parameters
- One case led to a 101 location CE in 3 iterations of the abstraction refinement loop

Final design (verified):

- PHAVer took over 12 minutes
- IRA took 23.7 seconds

Nuclear Power Plant Control*

- **Temperature control**
  - rods immersed to cool the reactor, withdrawn to allow reaction
  - rods controlled temperature measurements and local timers.
  - each rod can stay inside only for a certain max time limit
- **Temperature should not rise beyond a critical threshold.**
- **Model**
  - 3 control rods
  - 11 continuous variables

* **Variation of the problem studied by Kapur and Shyamasundar (HART'97), R Alur et al (TCS'95), P. H. Ho 95 PhD thesis and others.**

Nuclear Power Plant Control

Several failed attempts

- First attempt:
  - simple counterexample of 3 locations
  - abstraction 3 continuous variables
  - all of variables related to control rod 1
  - clear that the rod was being inserted too late
  - changed the cutoff temperature
- Similar CEs for control rods 2 and 3

Final working design

- PHAVer verification: 16 hours
- IRA verification: 6 iterations, 30.04 seconds

Nuclear Power Plant Control

Several attempts

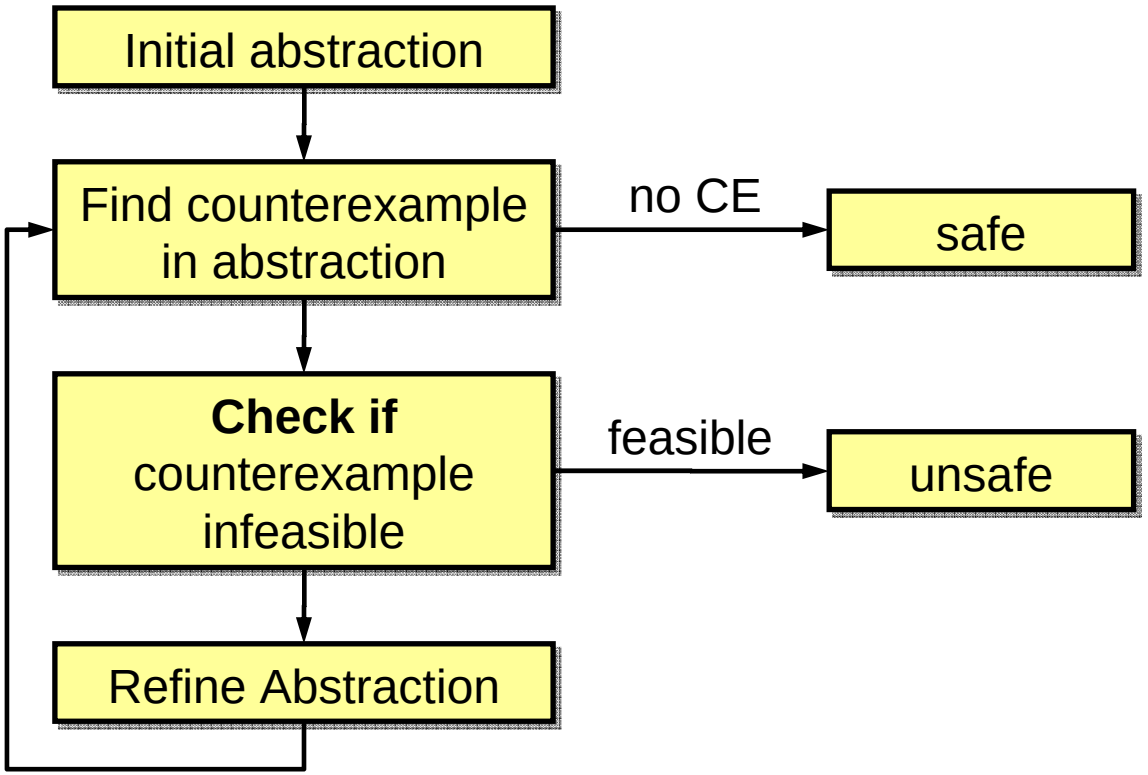
- First attempt:
  - simple counterexample of 3 locations
  - abstraction 3 continuous variables
  - all of variables related to control rod 1
  - clear that the rod was being inserted too late
  - changed the cutoff temperature
- Similar CEs for control rods 2 and 3

A *design procedure*!

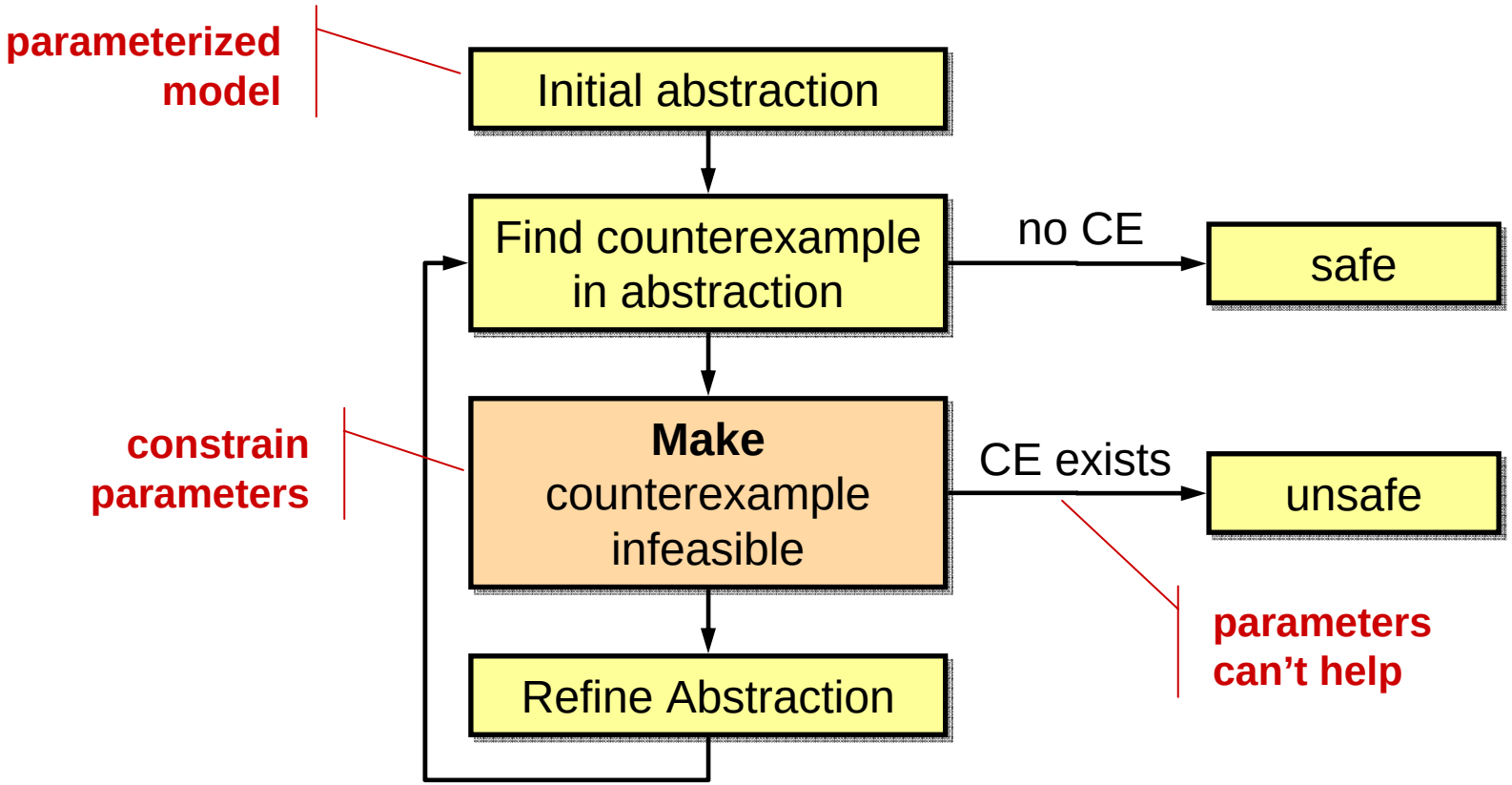
Final working design

- PHAVer verification: 16 hours
- IRA verification: 6 iterations, 30.04 seconds

Standard CEGAR Loop



Modified CEGAR Loop for Design*



* Goran Frehse, Sumit K. Jha, Bruce H. Krogh, A Counterexample-Guided Approach to Parameter Synthesis for Linear Hybrid Automata, Hybrid Systems: Computation and Control, St. Louis, MO, April 2008.

Linear Constraints with Parameters

$$D\{ . Hs\% e> H @ \hat{}h_4 fff h_p$$

- monotonic *positive/negative* parameters
  - $e_j \geq 0 \Rightarrow$ increasing p_j tightens constraints on x
  - $e_j \leq 0 \Rightarrow$ increasing p_j relaxes constraints on x

- feasible monotonic parameters

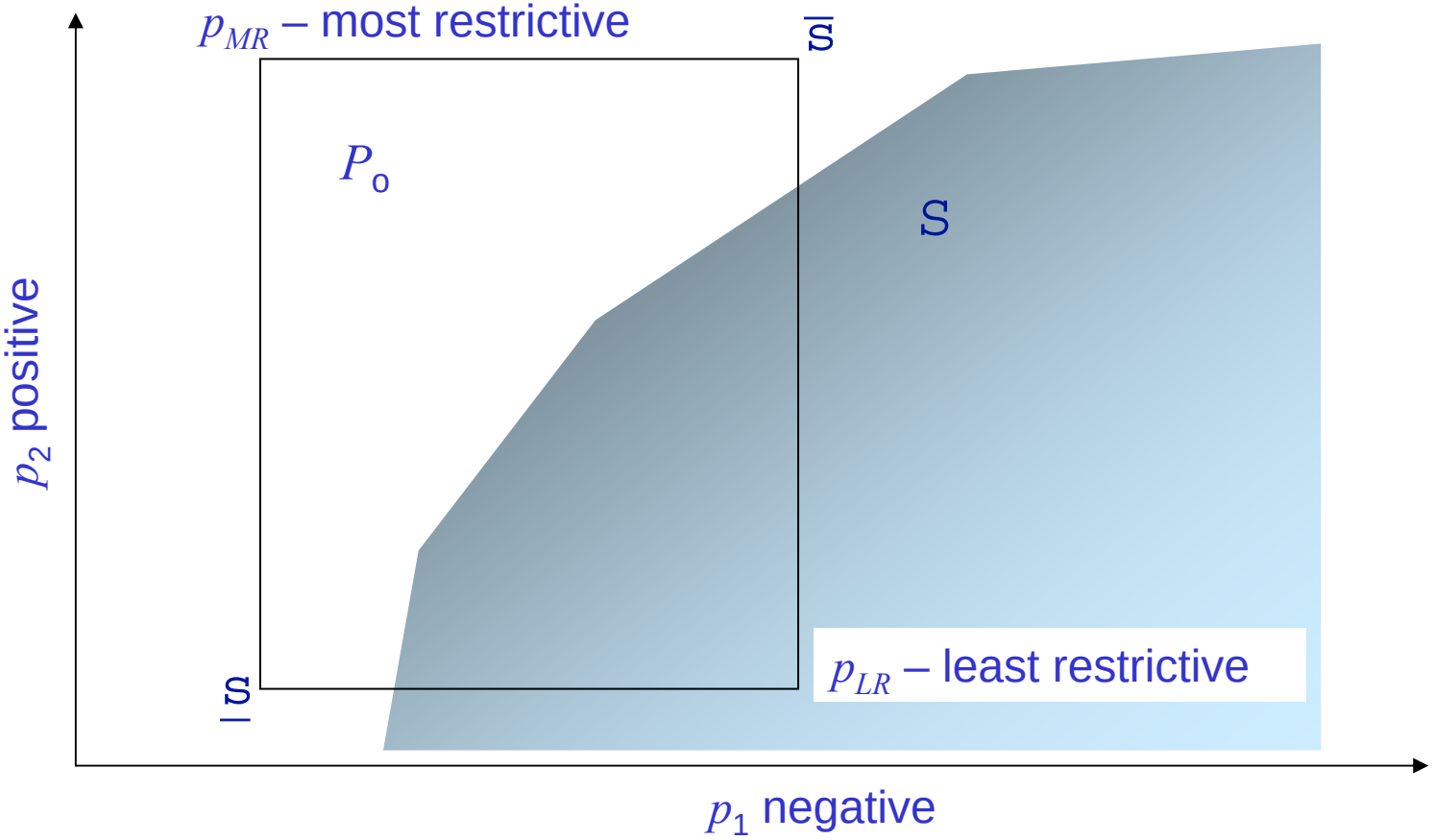
$$S_m @ \hat{}D\{ . Hs\% e'\&_{S_m}$$

  - positive $p_j' \in P_j \Rightarrow (-\infty, p_j'] \subseteq P_j$
  - negative $p_j' \in P_j \Rightarrow [p_j', \infty) \subseteq P_j$

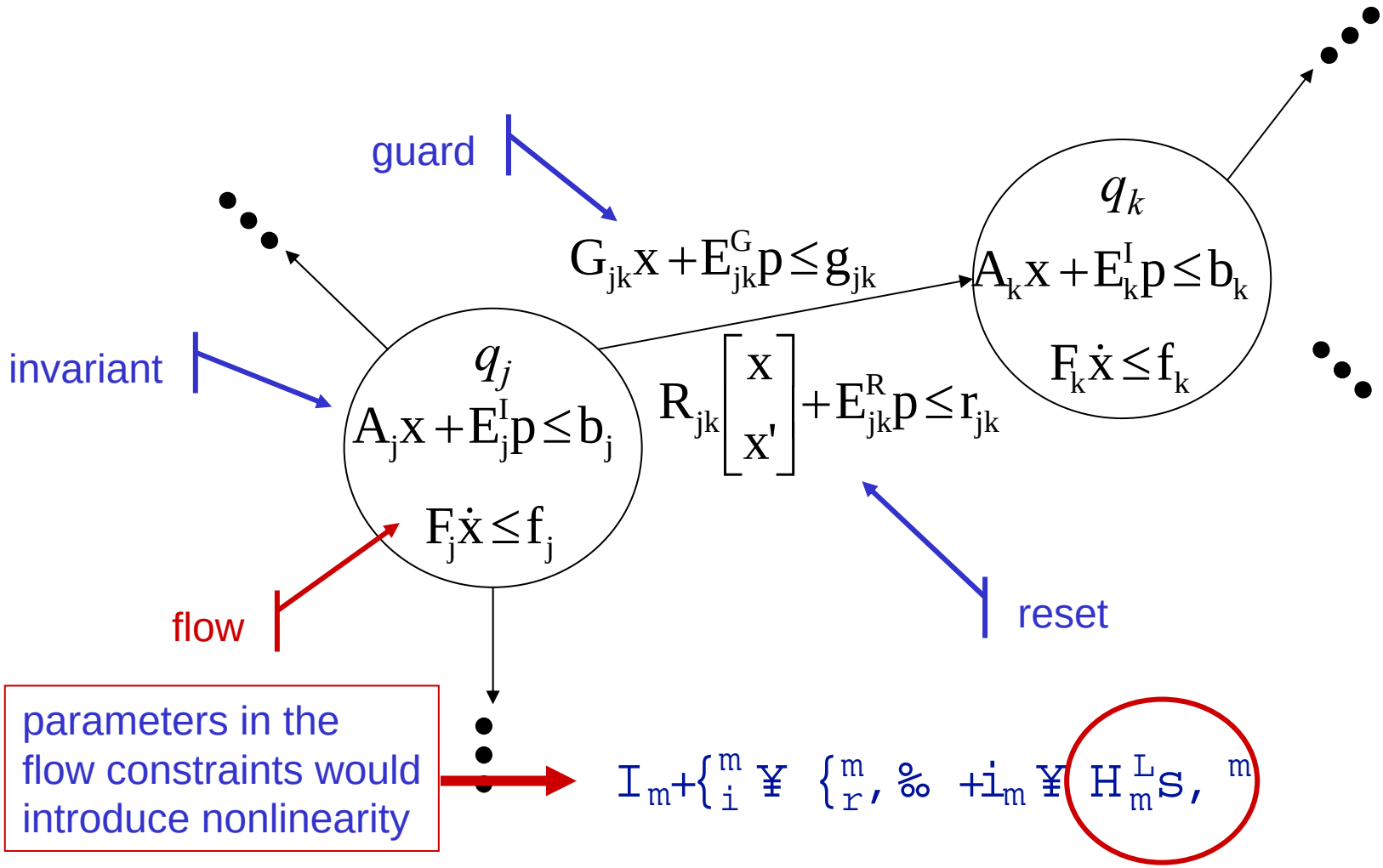
Monotonic Parameters

$$S = \{ \ldots \}$$

$$S_r = \underline{S} \cup S \cup \overline{S}$$

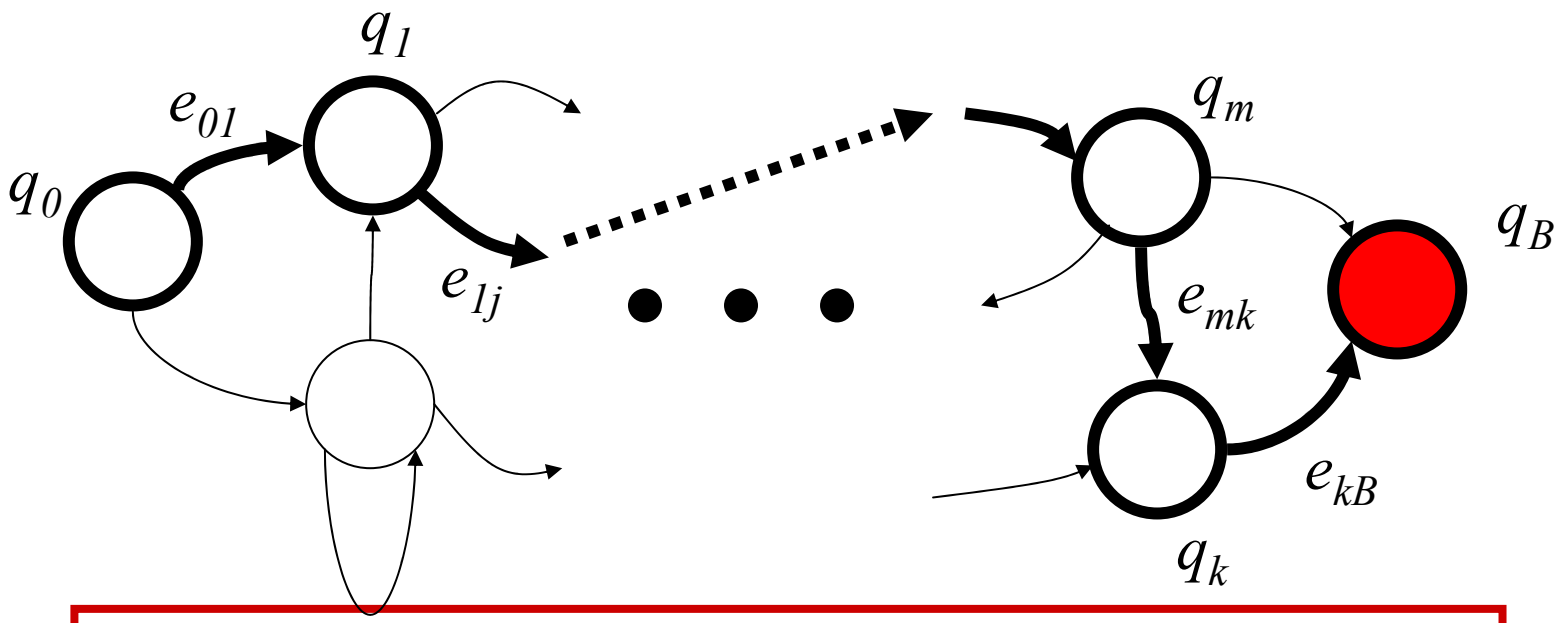


LHA with Parameters



Counterexample Paths with Parameters

$$\pi = q_0\, e_{01}\, q_1\, e_{1j} \ldots e_{mk}\, q_k\, e_{kj}\, q_B$$



***PathCon*(π)** : set of linear constraints over the variables:

$$\{^{r}_{r}\!>\{^{r}_{i}\!>\ ^{r}\!>\{^{4}_{r}\!>\{^{4}_{i}\!>\ ^{4}\!>\!=\!=\!=\!>\{^{n}_{r}\!>\{^{n}_{i}\!>\ ^{n}\mathtt{r}\!>\{^{E}_{r}\ \mathtt{dqg\ s1}$$

Consistent monotonicity of p in all LHA constraints

$\Rightarrow$ same monotonicity of p in ***PathCon*(π)**.

Good Parameters

- ***Design Problem***: **Choose p so that all counterexample paths are *infeasible*.**

$$S\,dwkF\,rq+\,,=D\;\{\;\;.\;H\;\;\%\;e$$

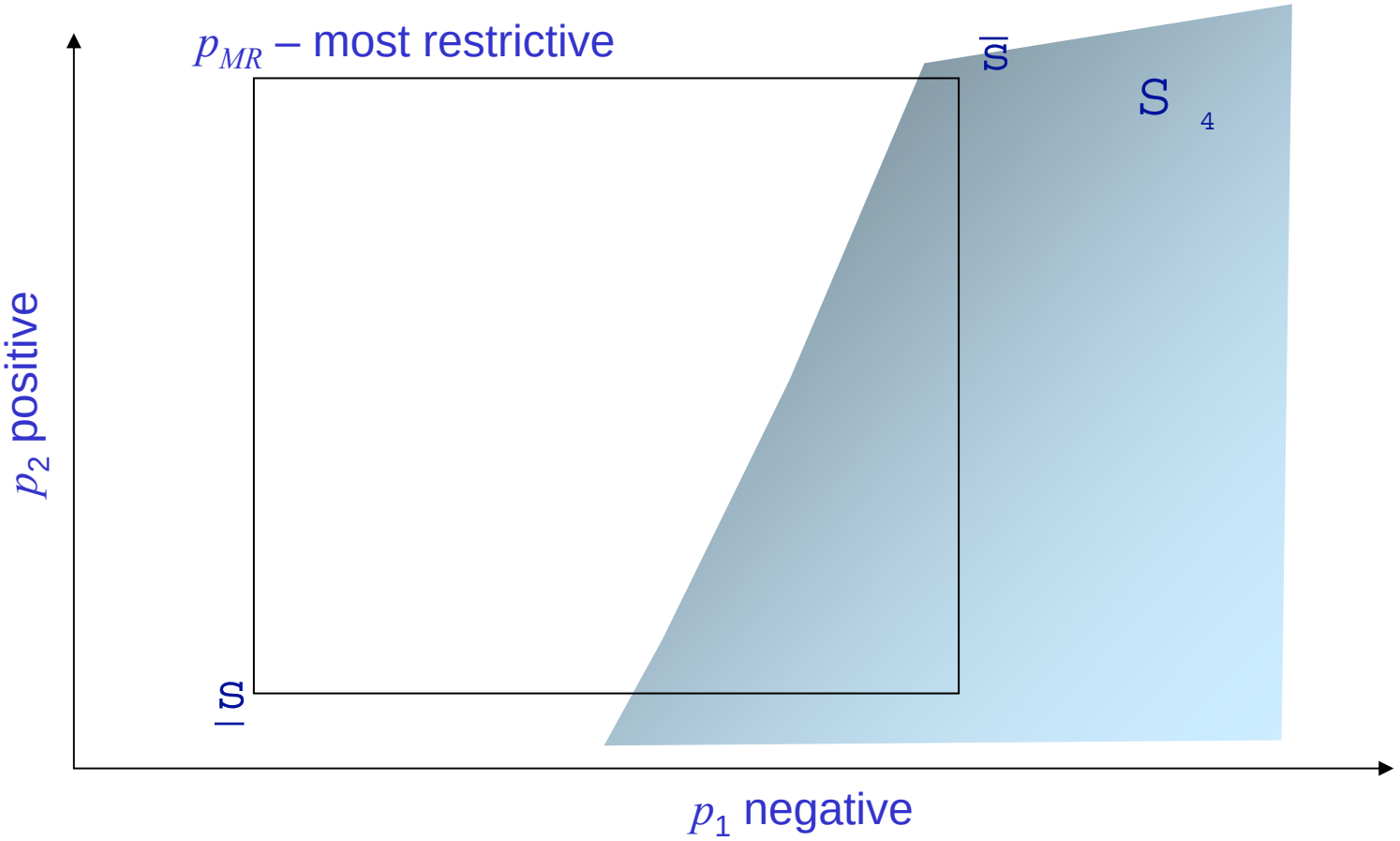
$$S_J\;@\;S_r\,¥^{\,v}\;{}_{5FH}\,S\;>\;\;S\;\;@\;\hat{}S\,dwkF\,rq+\,,`\&_s$$

- ***Optimal Design Problem***: $max \quad f(p)$

$$s.t. \quad p \in P_G$$

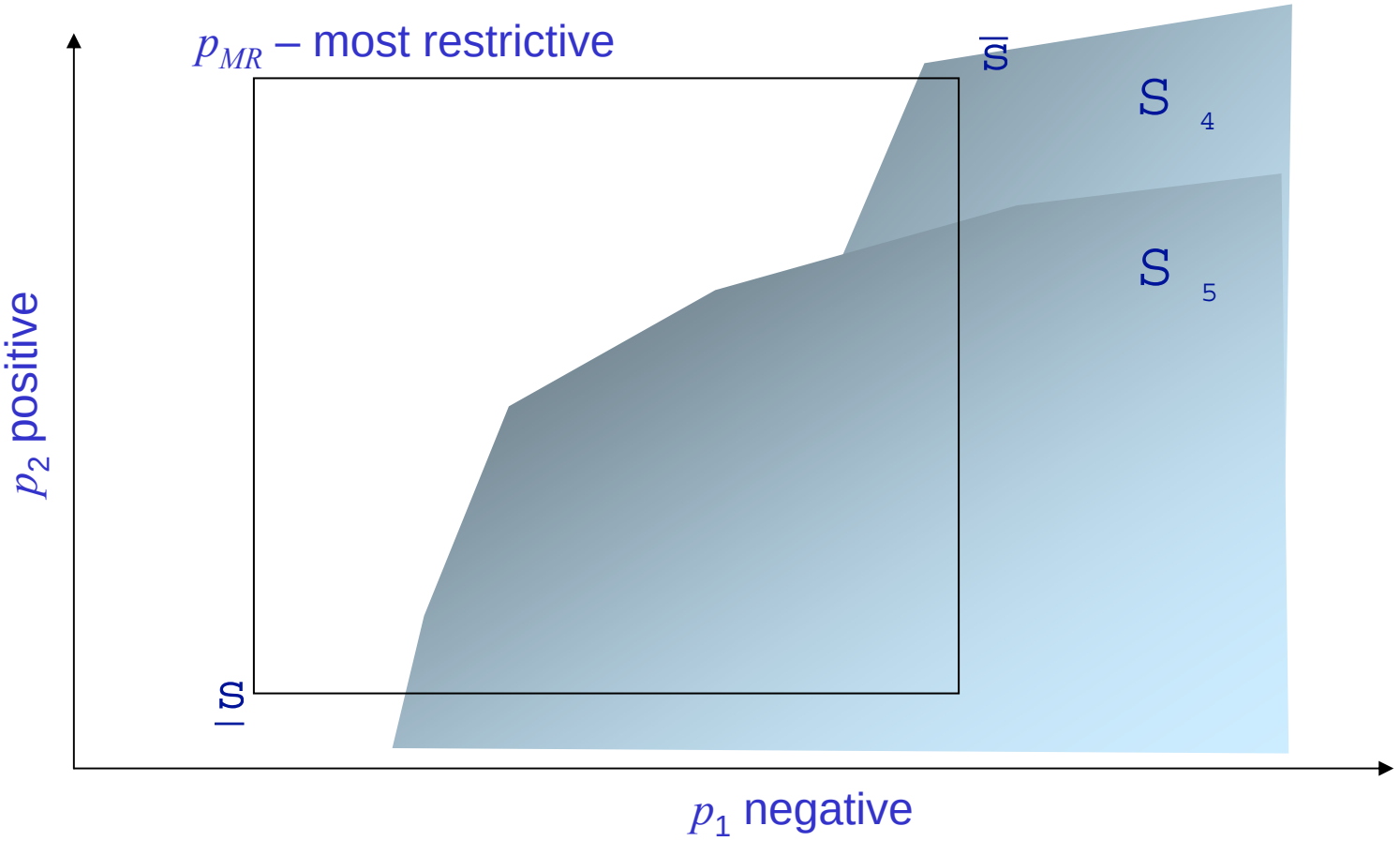
Good Parameters

S_J @ S_r ¥ ˅ 5FH S



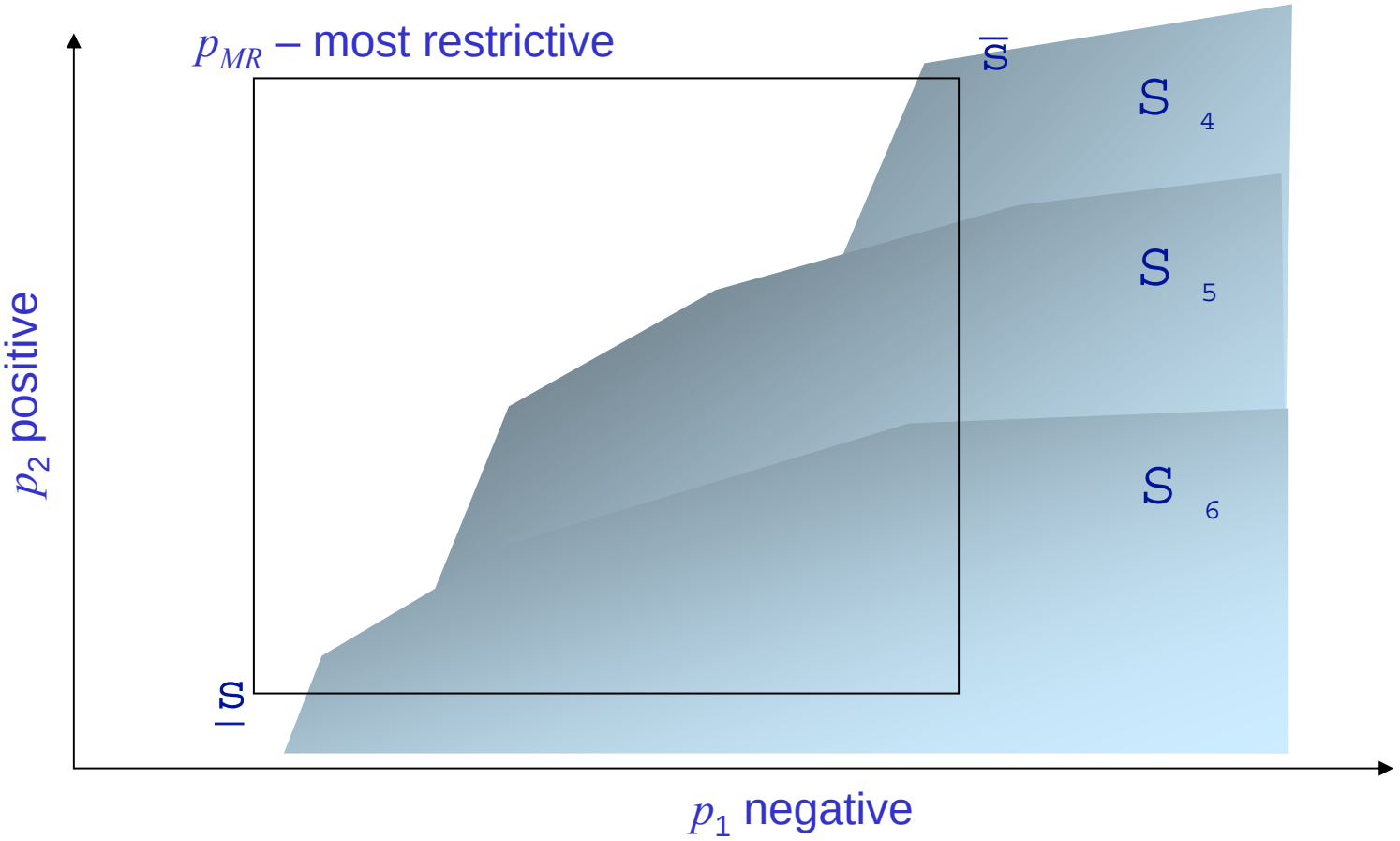
Good Parameters

$S_J @ S_r ¥ ^{v}$ 5FH S



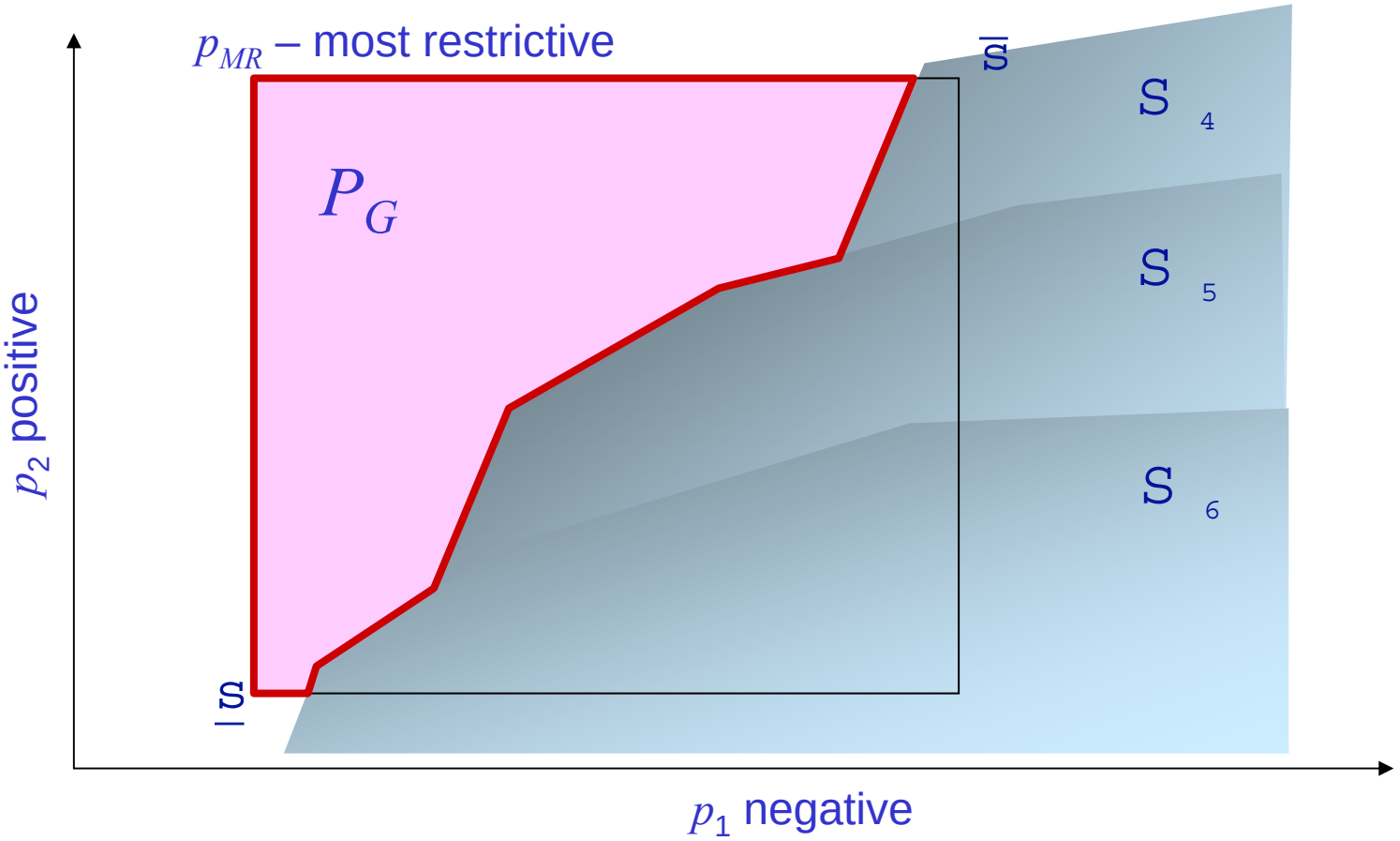
Good Parameters

$S_J @ S_r ¥ ^{v}$ 5FH S



Good Parameters

$S_J @ S_r ¥ ^{v}$ 5FH S

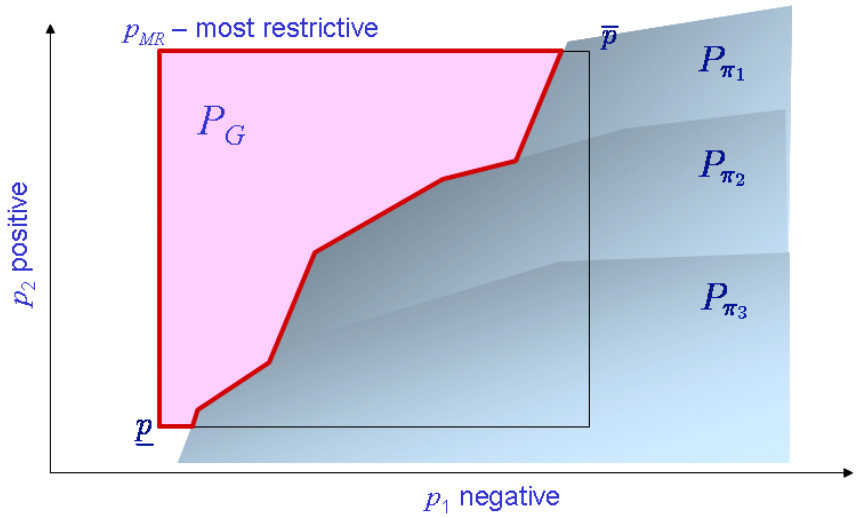


Good Parameters

$$P_G = P_o - \bigcup_{\pi \in \mathrm{CE}} [A_\pi x_\pi + E_\pi p \leq b_\pi]_{\downarrow p}$$

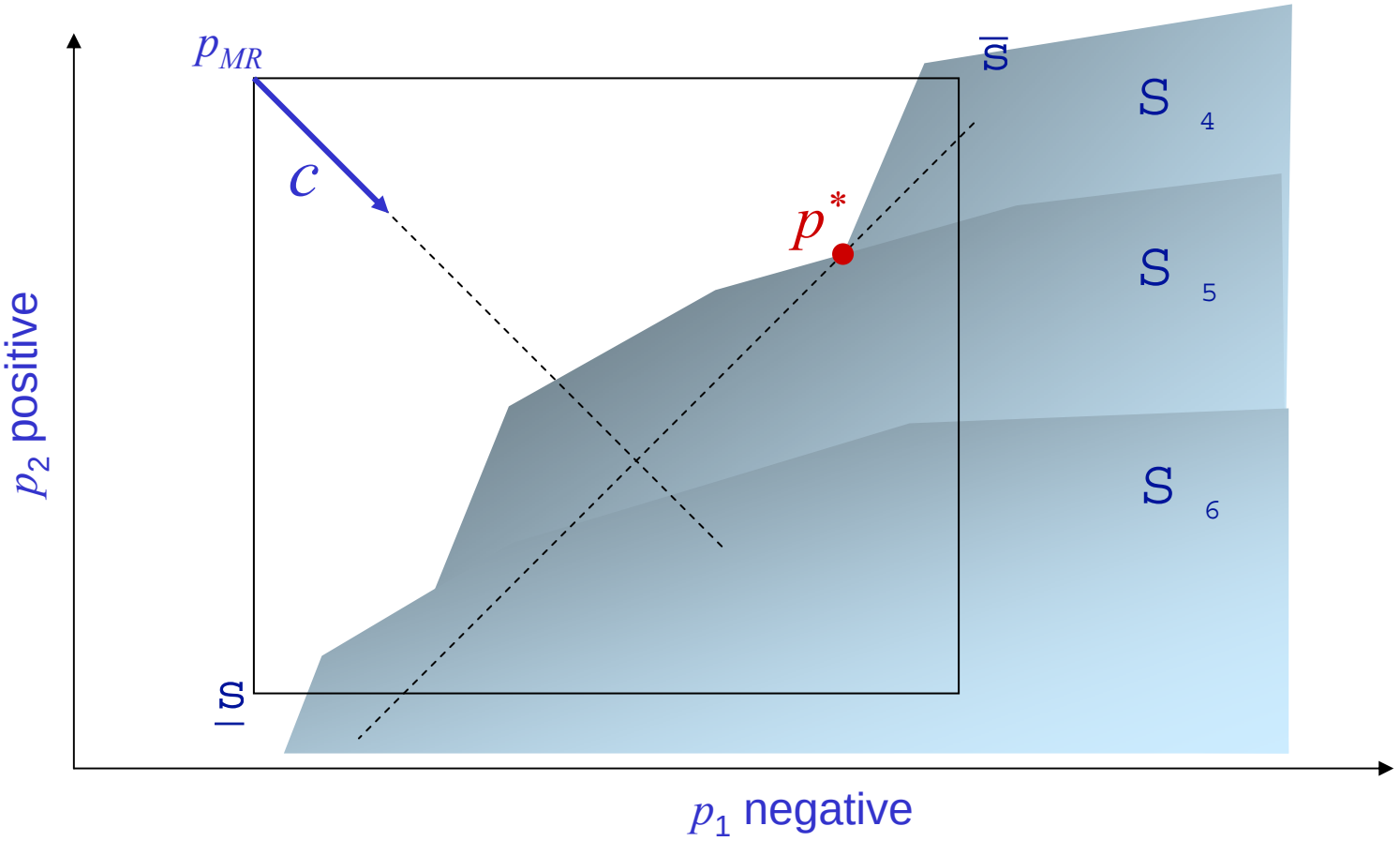
Observations

- P_G is nonempty $\Leftrightarrow$ p_{MR} is a good parameter
- projection is impractical (Fourier-Motzkin)
- CE may be infinite
- P_G is nonconvex and usually not closed



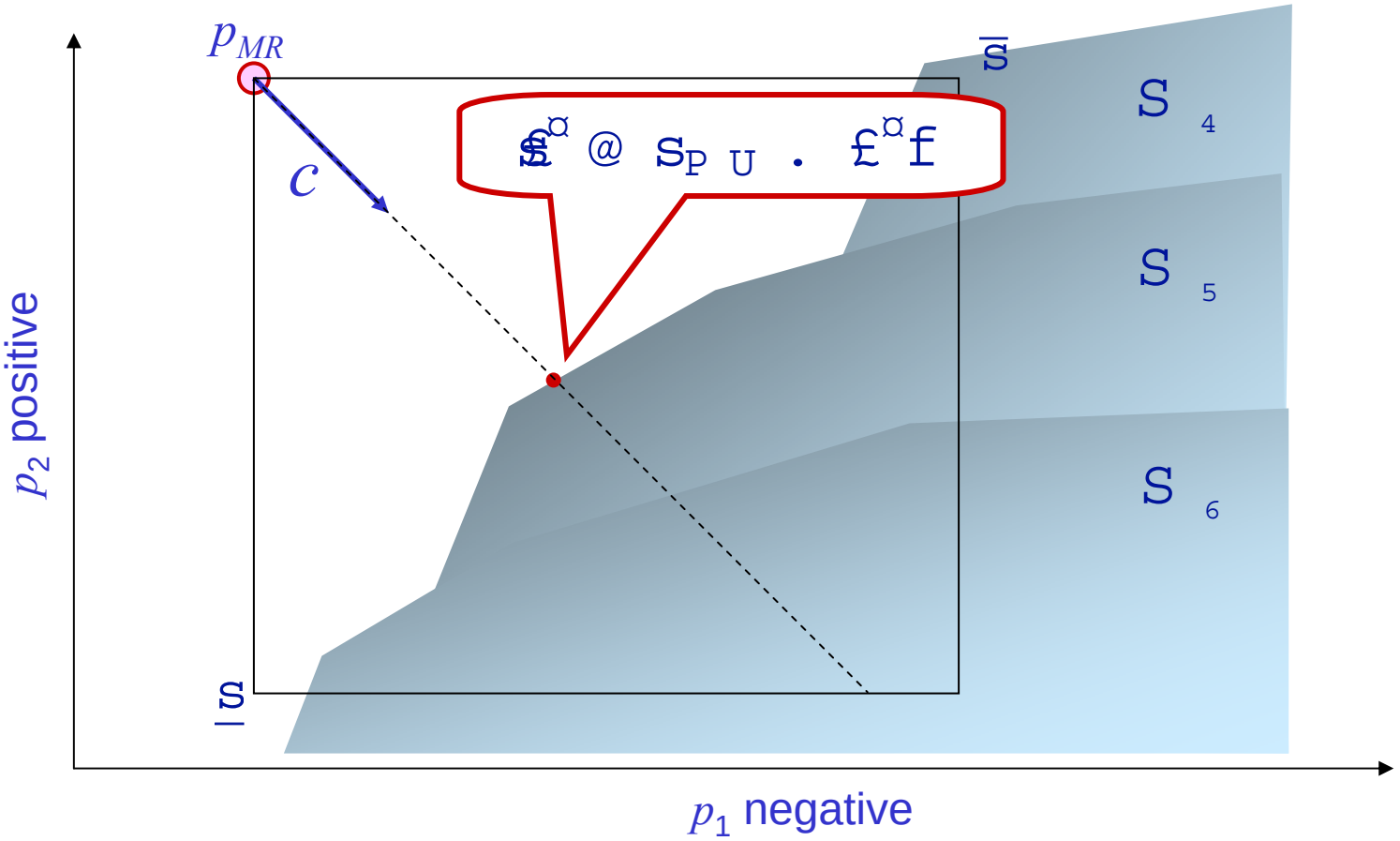
Optimal Design Problem

$$max \quad c^T p$$
$$s.t. \quad p \in P_G$$



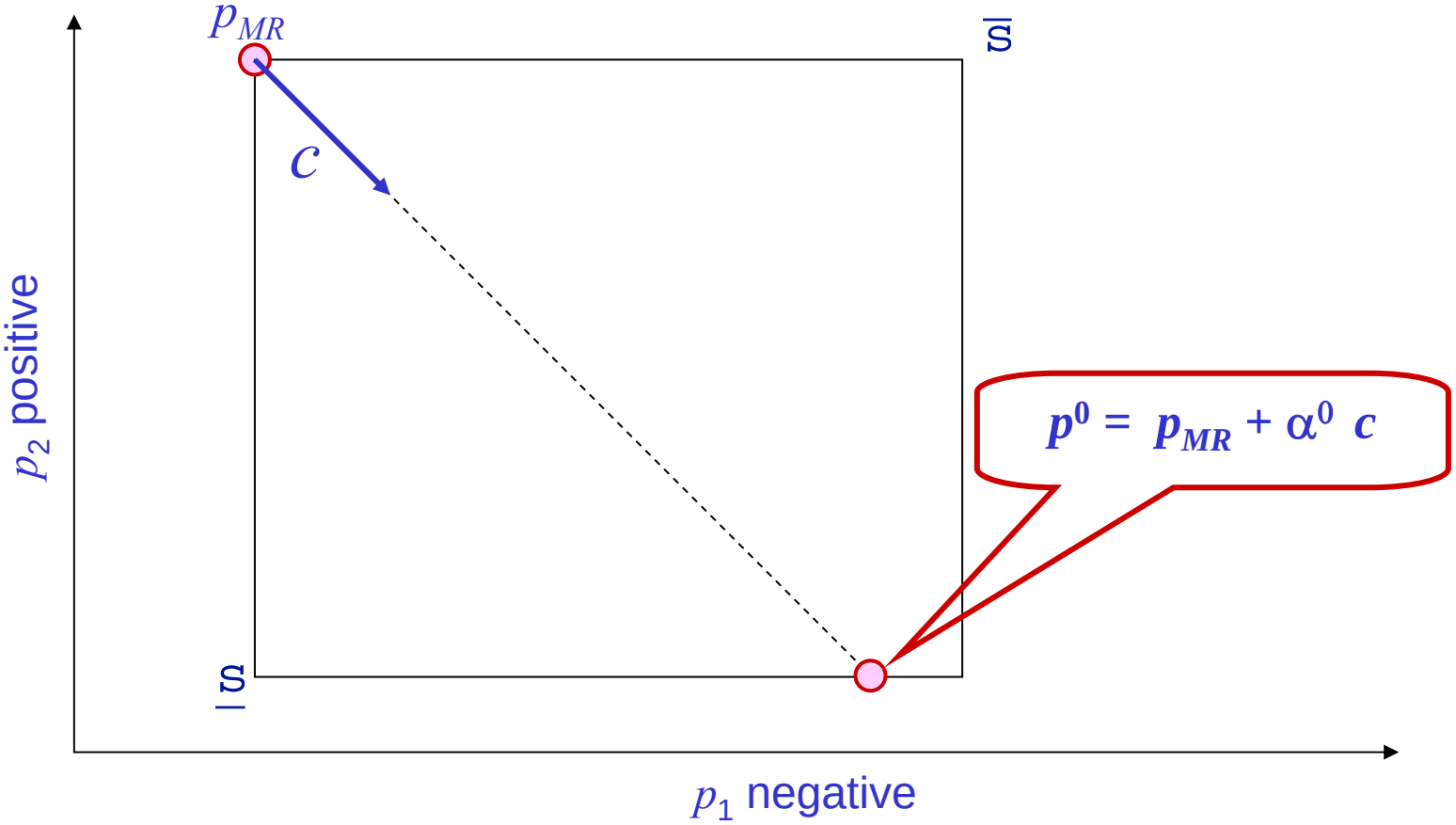
Heuristic 1: Line Search from $\mathbf{p} = \mathbf{p}_{MR}$

$$\alpha^{*} = arg:\ max_{\alpha}\ \alpha$$
$$s.t.\quad p_{MR} + \alpha\, c \in P_G$$

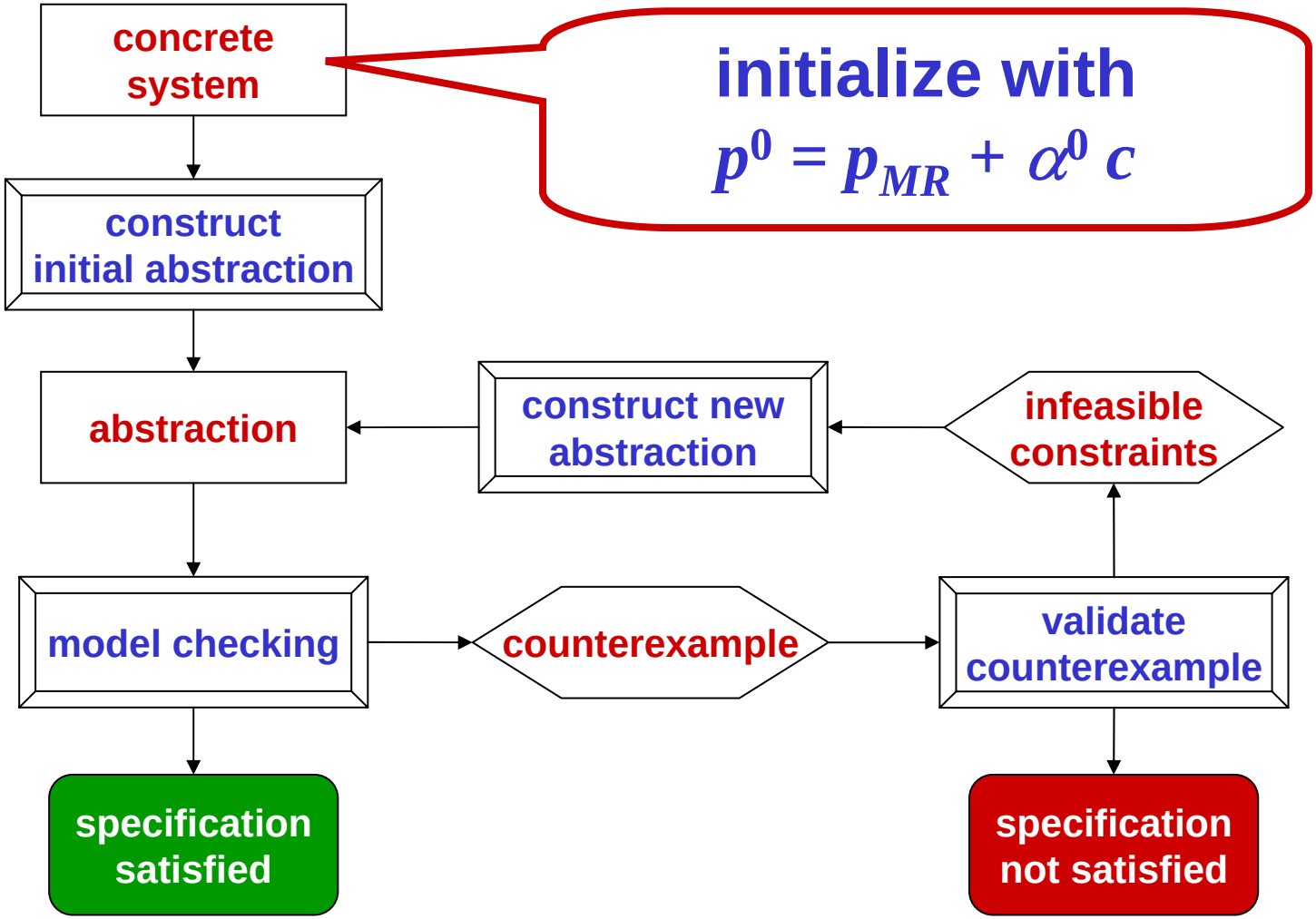


Initial α

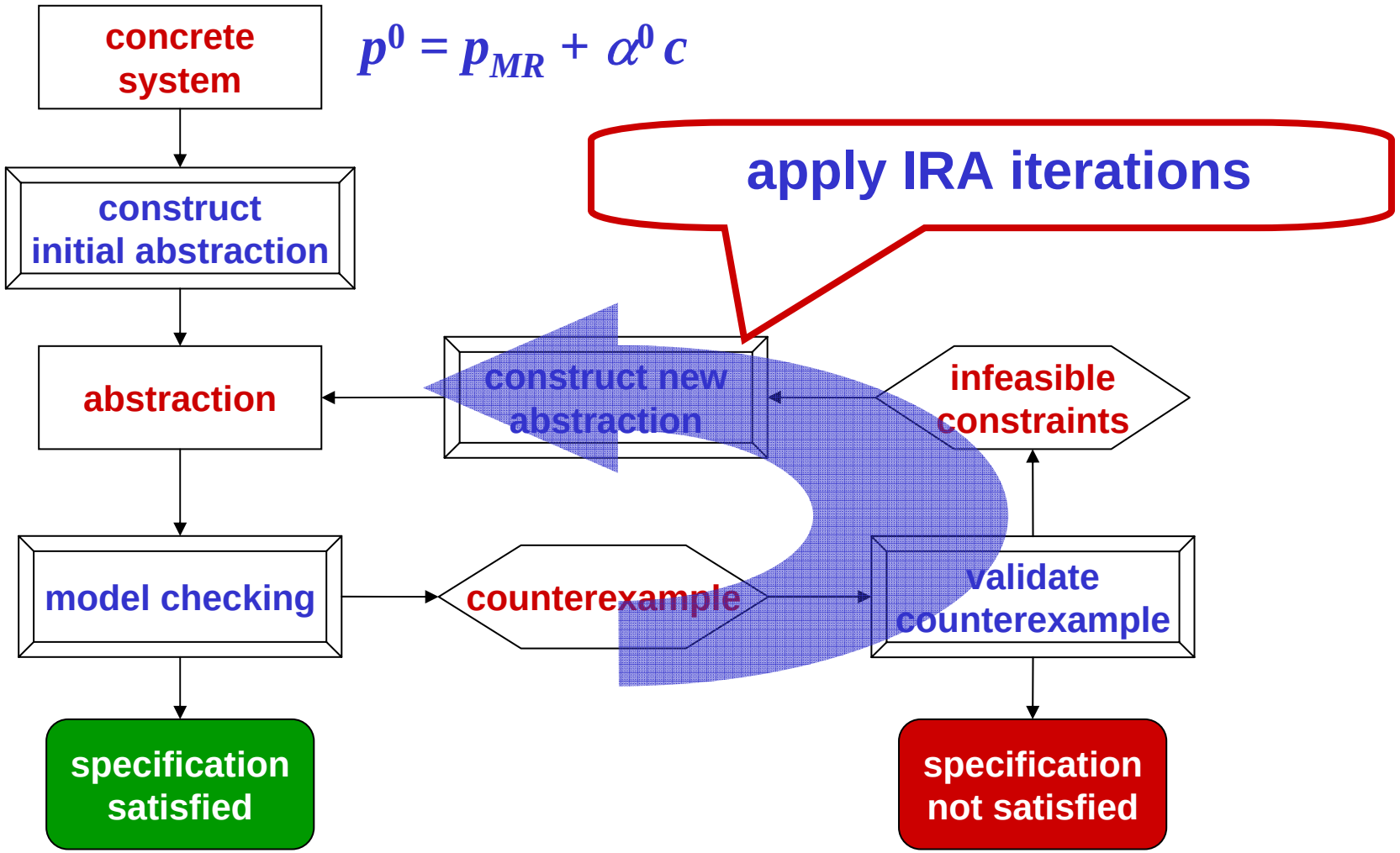
$$\alpha^0 = arg:\ max_\alpha \quad \alpha$$
$$s.t. \quad p_{MR} + \alpha\, c \in P_o$$



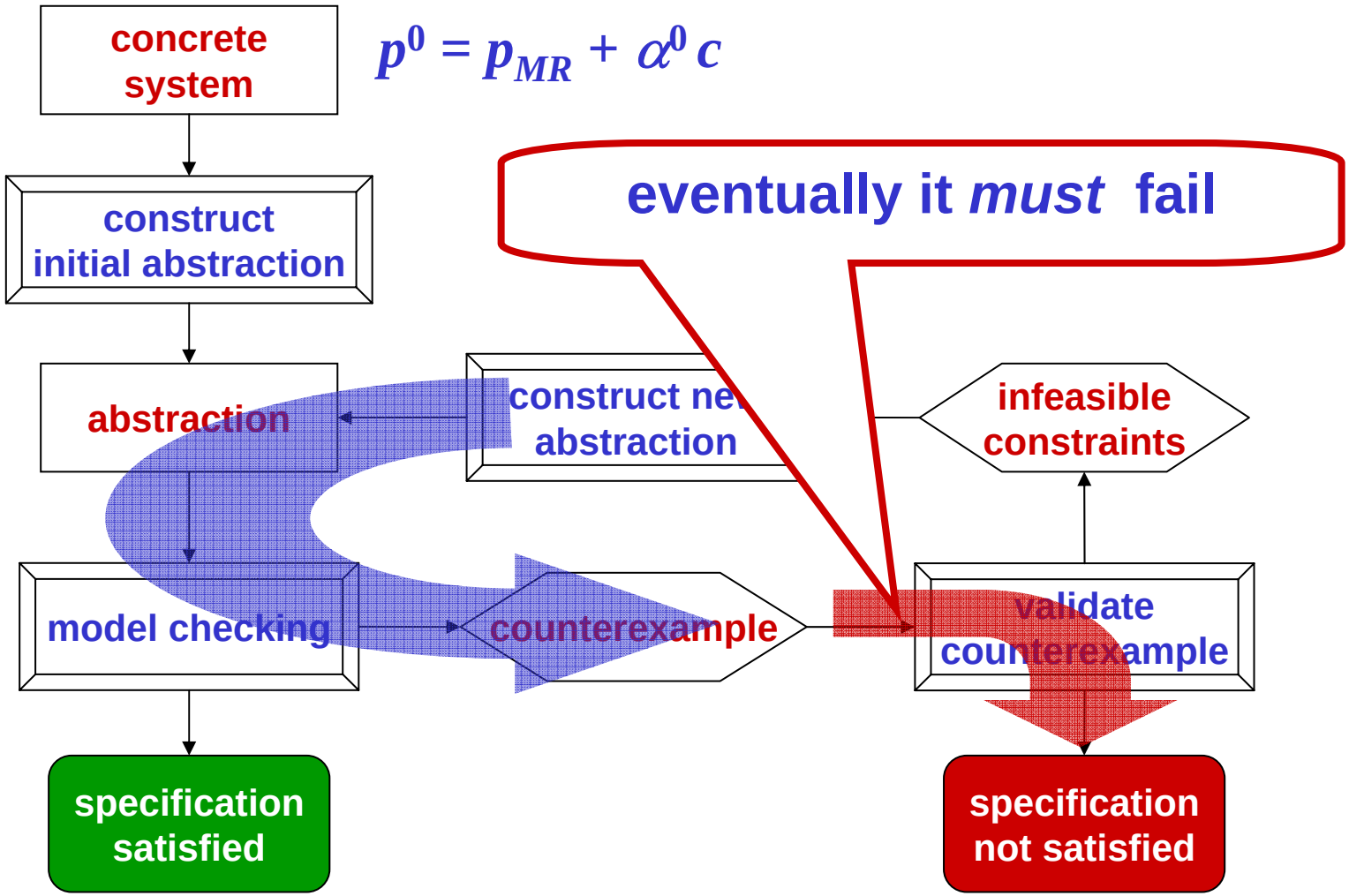
IRA-Based Line Search



IRA-Based Line Search

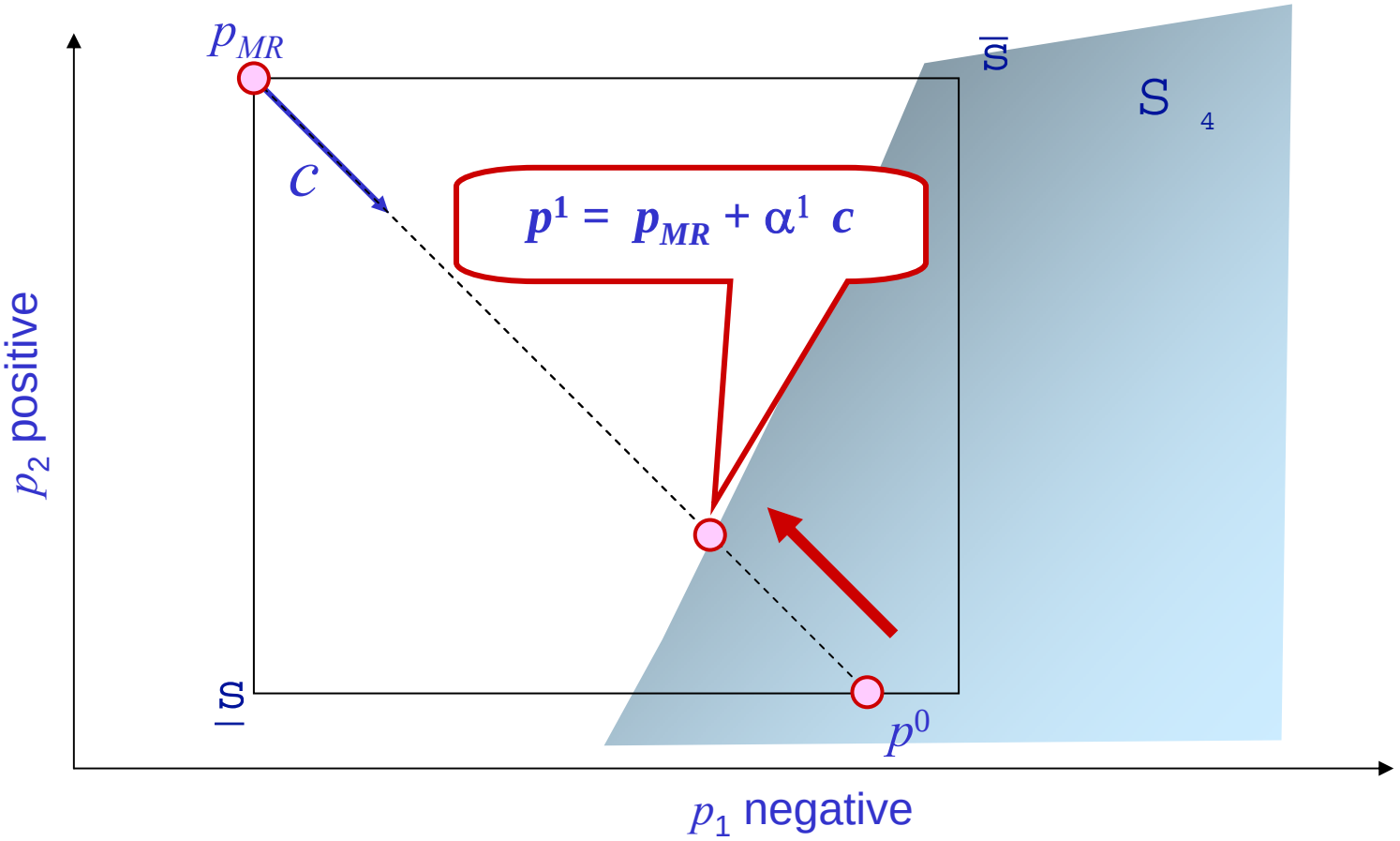


IRA-Based Line Search



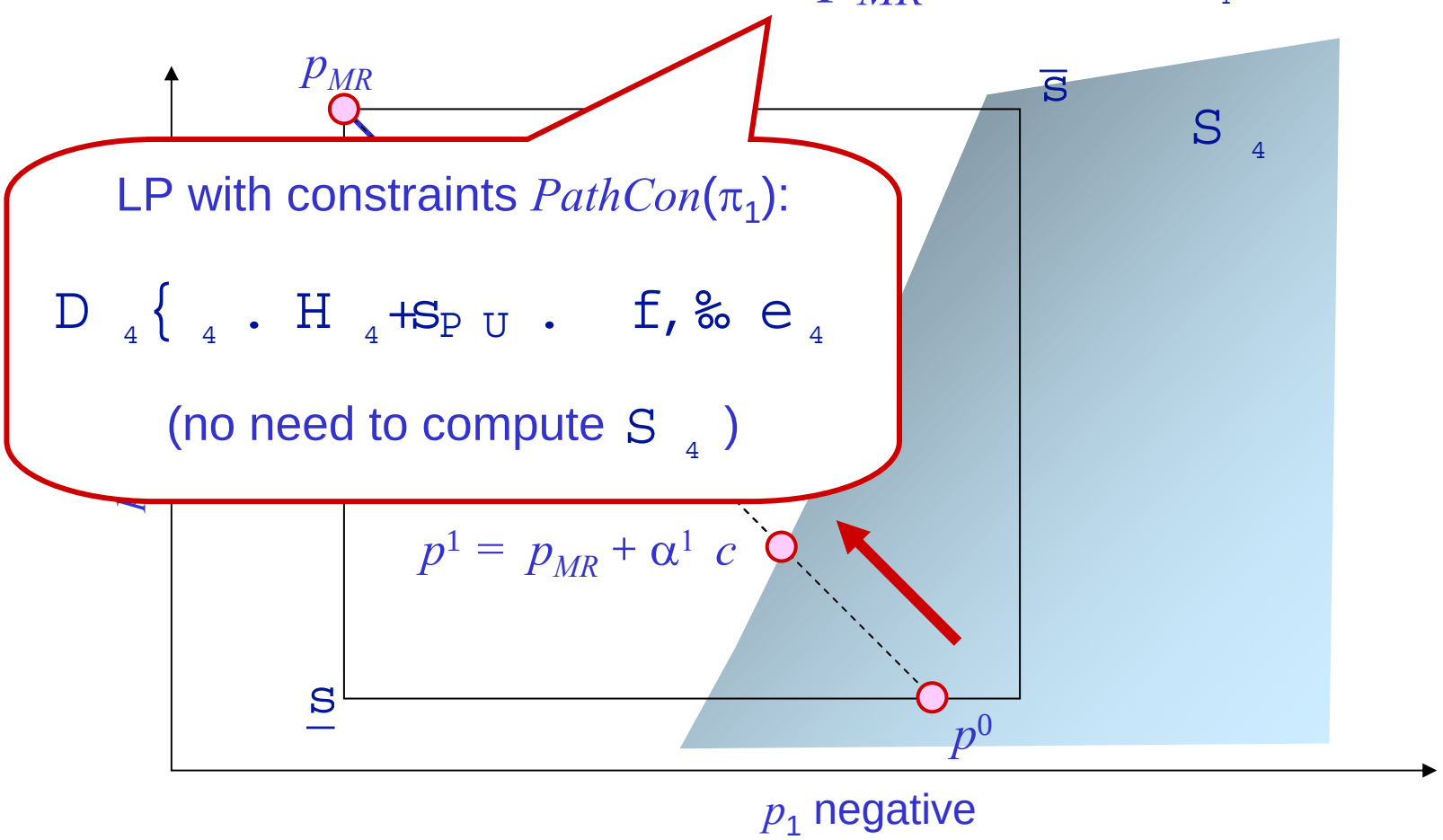
Line Search

$$\alpha^1 = arg:\ min_\alpha \quad \alpha$$
$$s.t. \quad p_{MR} + \alpha c \in S_4$$

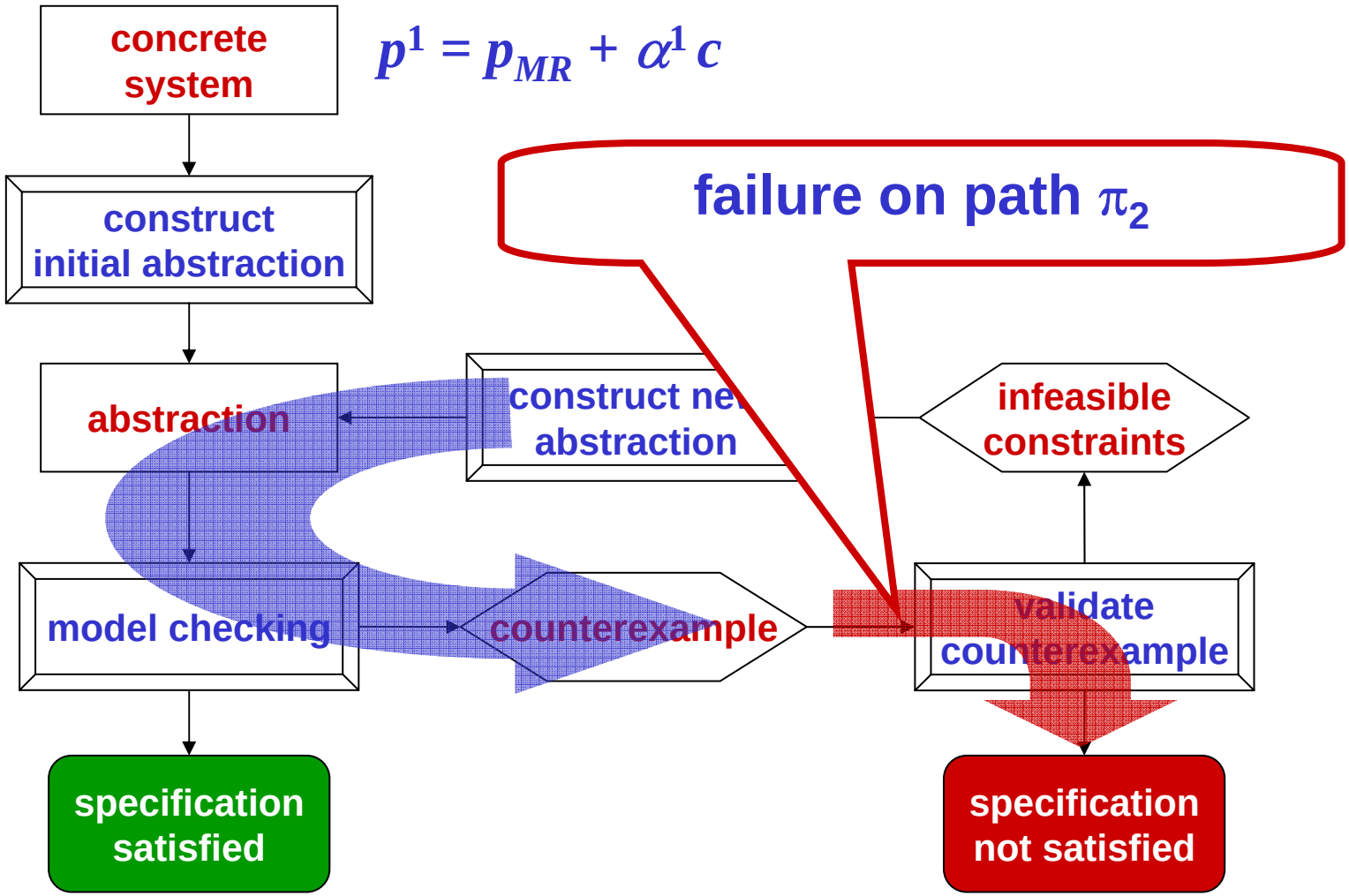


Line Search

$$\alpha^1 = arg:\ min_\alpha\ \alpha$$
$$s.t.\quad p_{MR} + \alpha c \in S_4$$

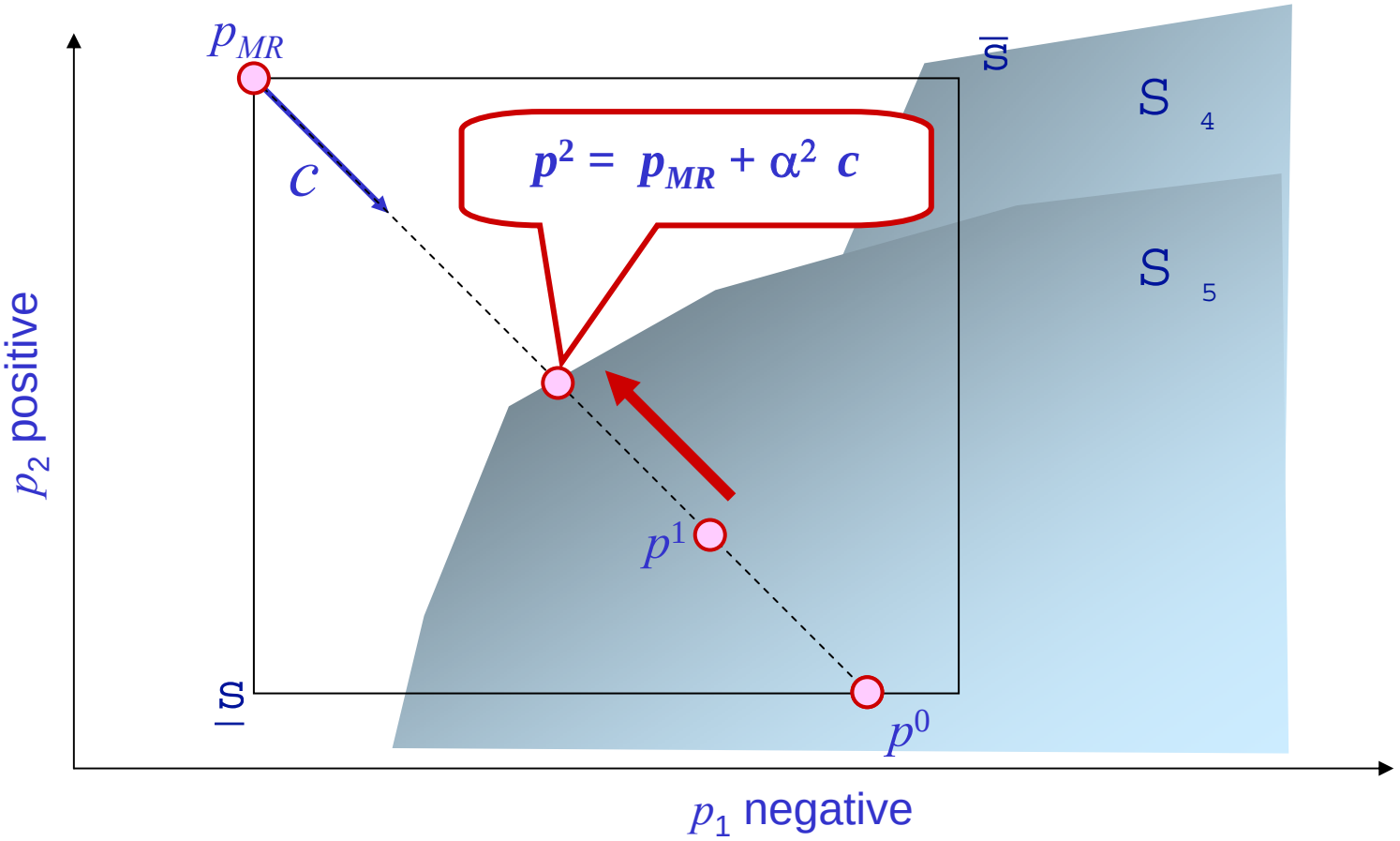


IRA Iteration with $p = p^1$

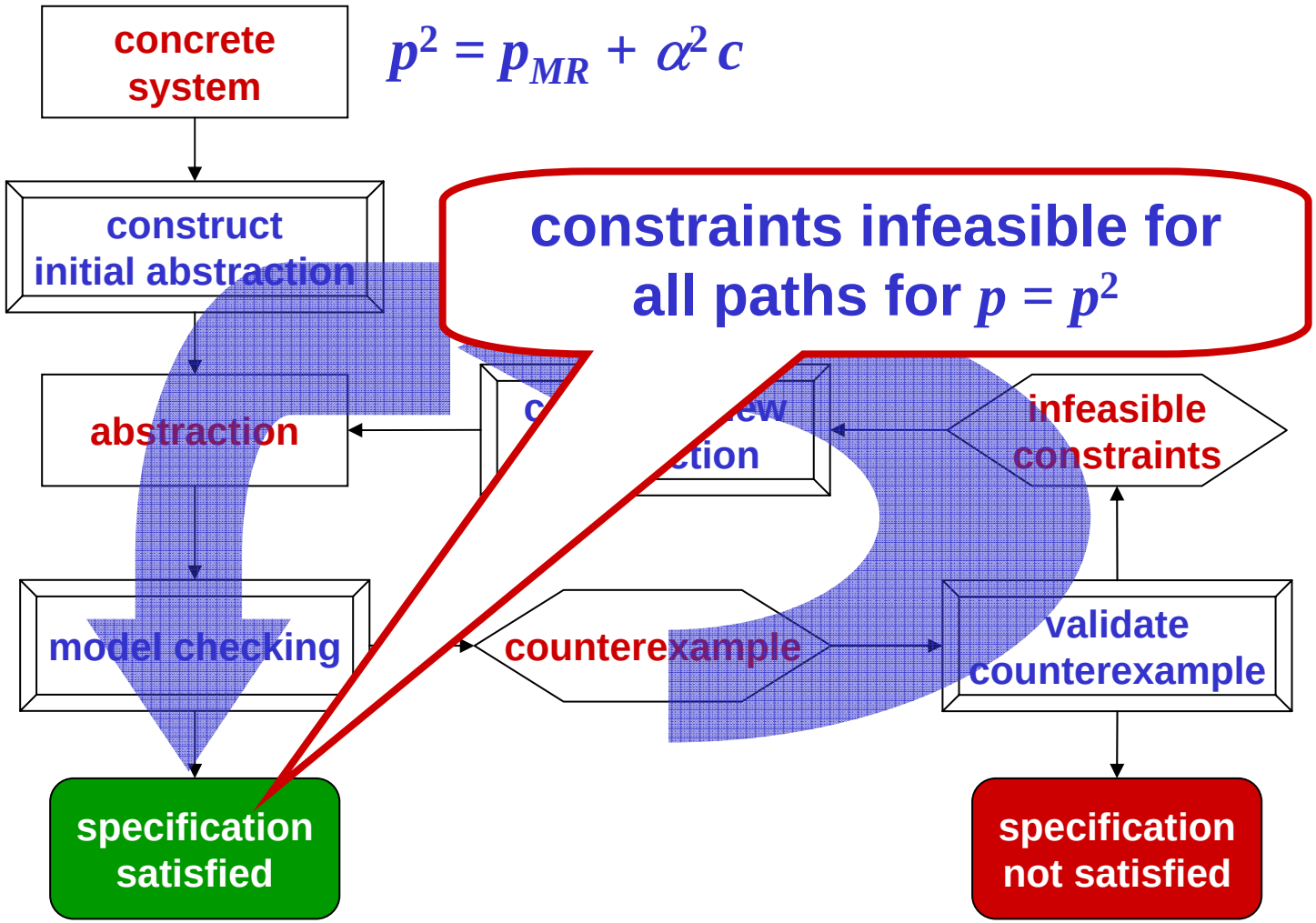


Line Search 2

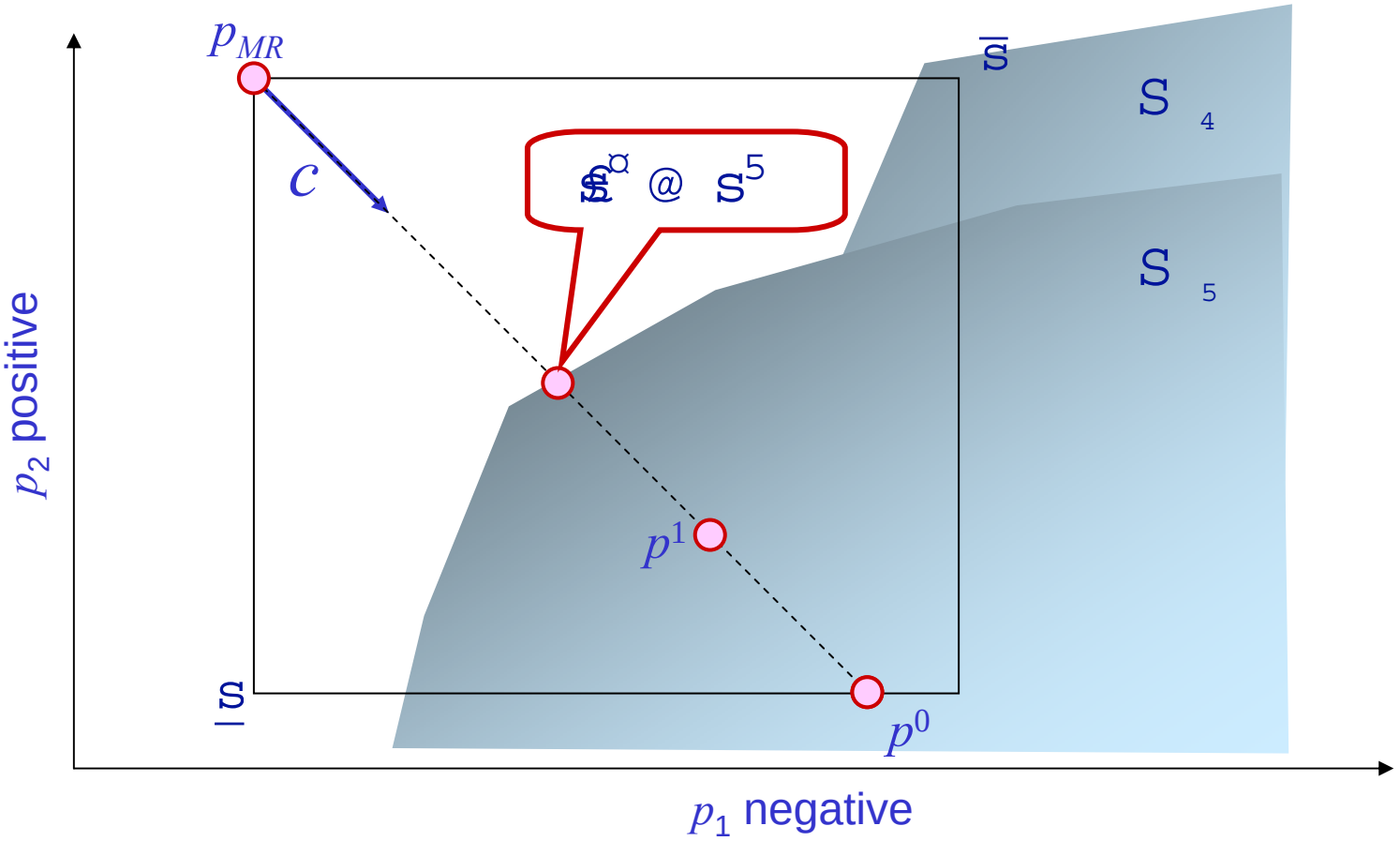
$$\alpha^2 = arg:\ min_\alpha\ \ \alpha$$
$$s.t.\quad p_{MR} + \alpha\, c \in S_5$$



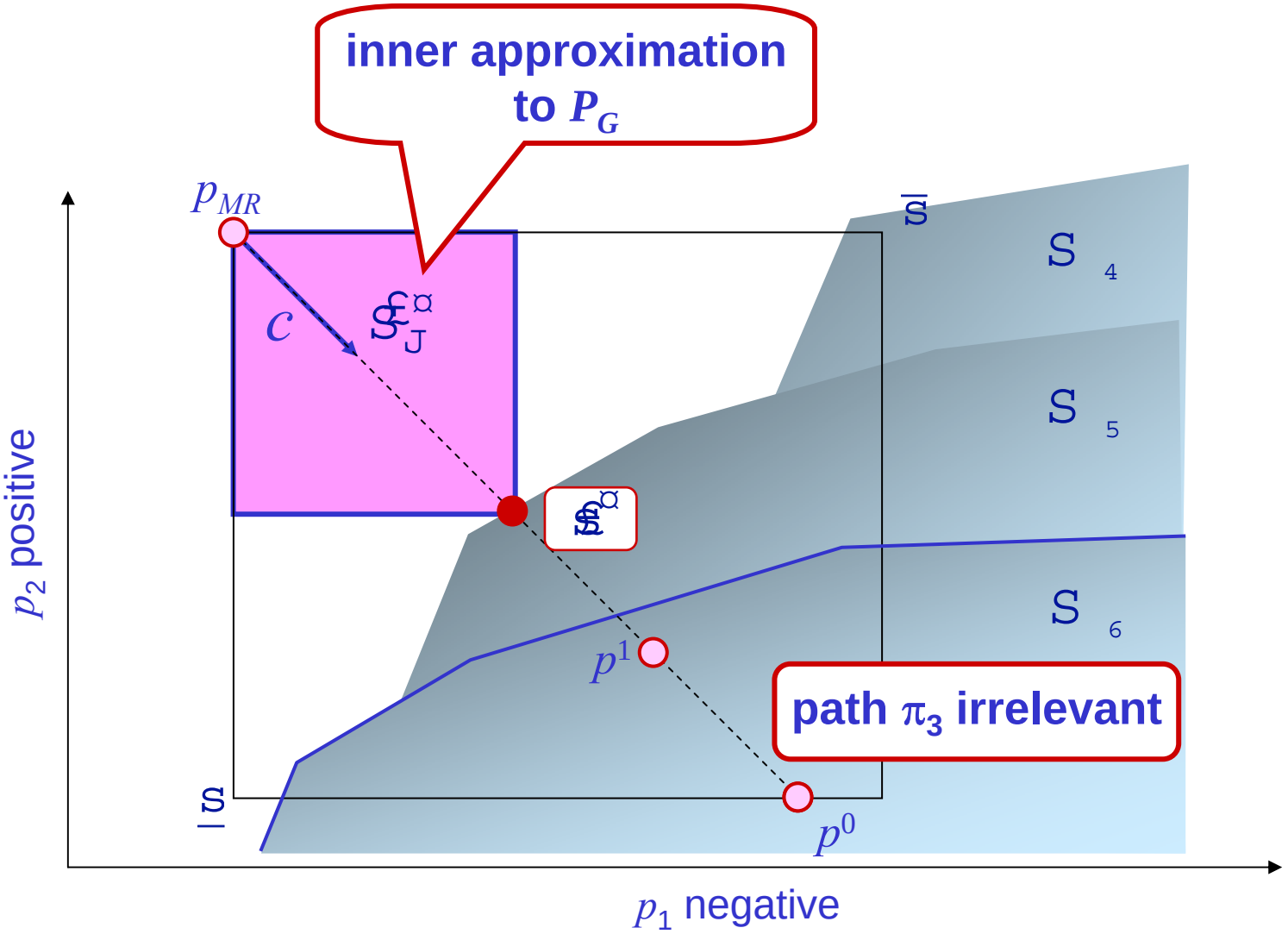
IRA Iteration with $p = p^2$



Termination at S @ p^{α}

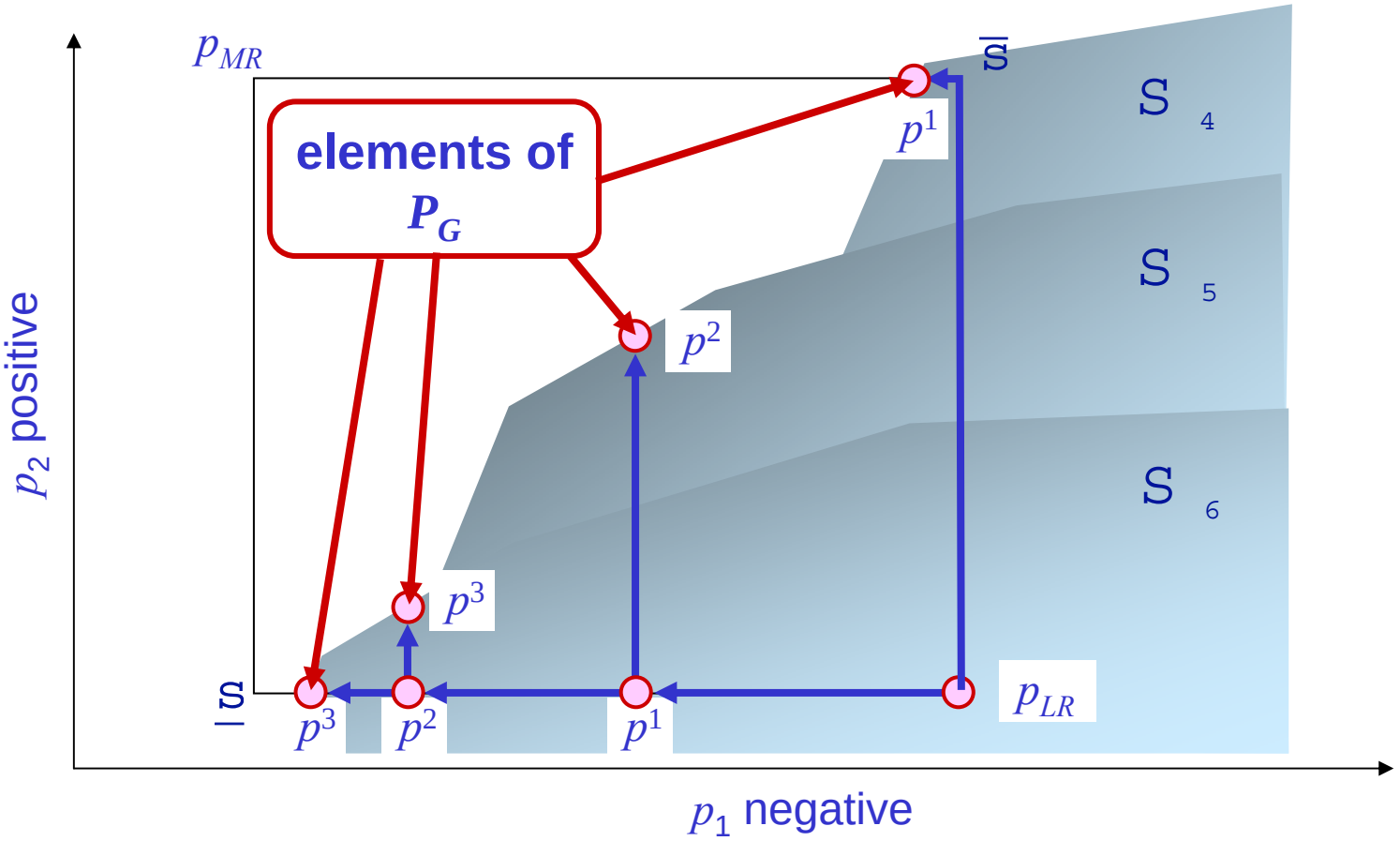


Termination at S @ $\hat{p}^{\alpha}$



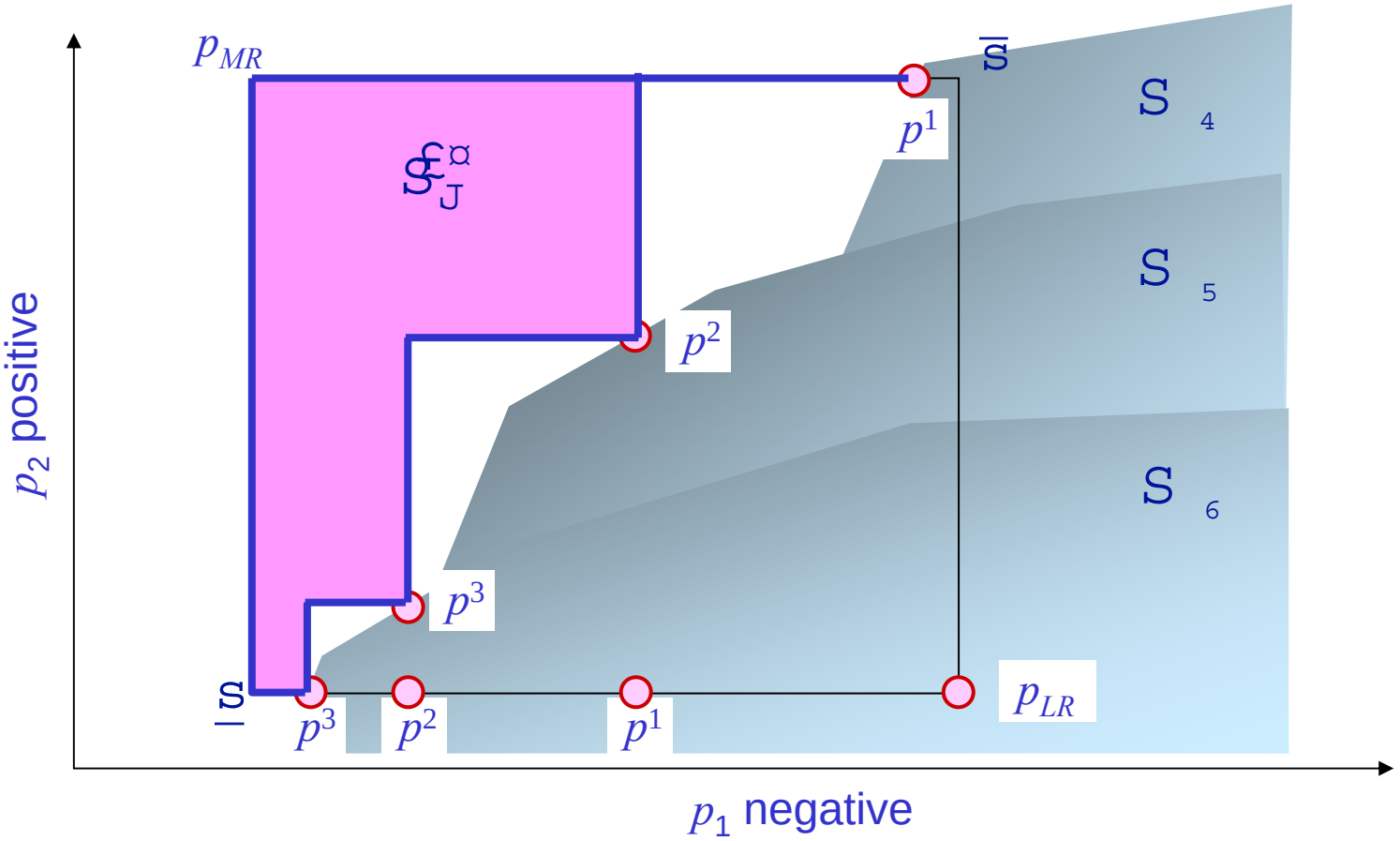
Heuristic 2: Single Parameter Iterations

- starting with $\boldsymbol{p} = \boldsymbol{p_{LR}}$ make feasible path constraints infeasible by moving single parameters



Heuristic 2: Single Parameter Iterations

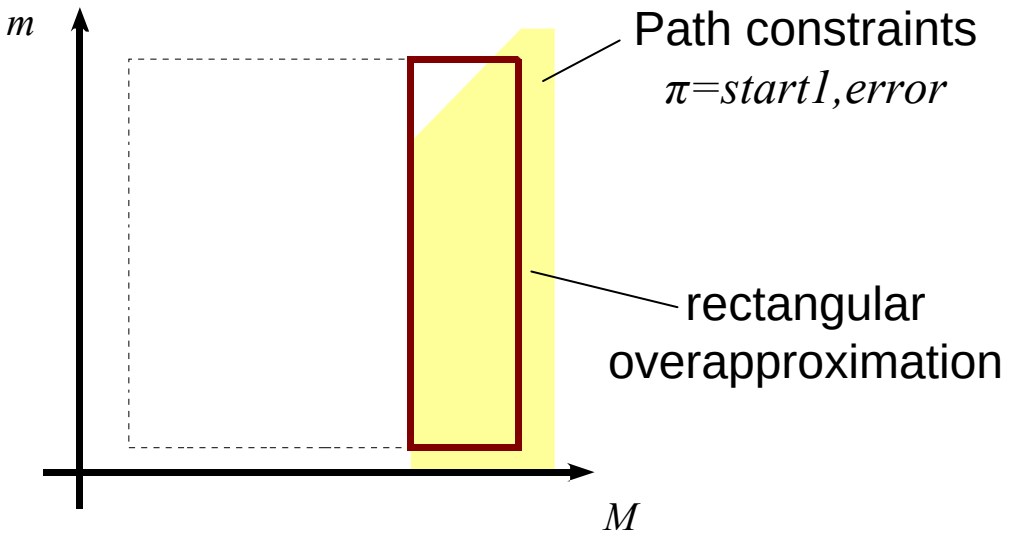
- starting with $\boldsymbol{p} = \boldsymbol{p_{LR}}$ make feasible path constraints infeasible by moving single parameters



Heuristic 3: Rectangular/Octagonal Overapproximations

Overapproximating paths individually:

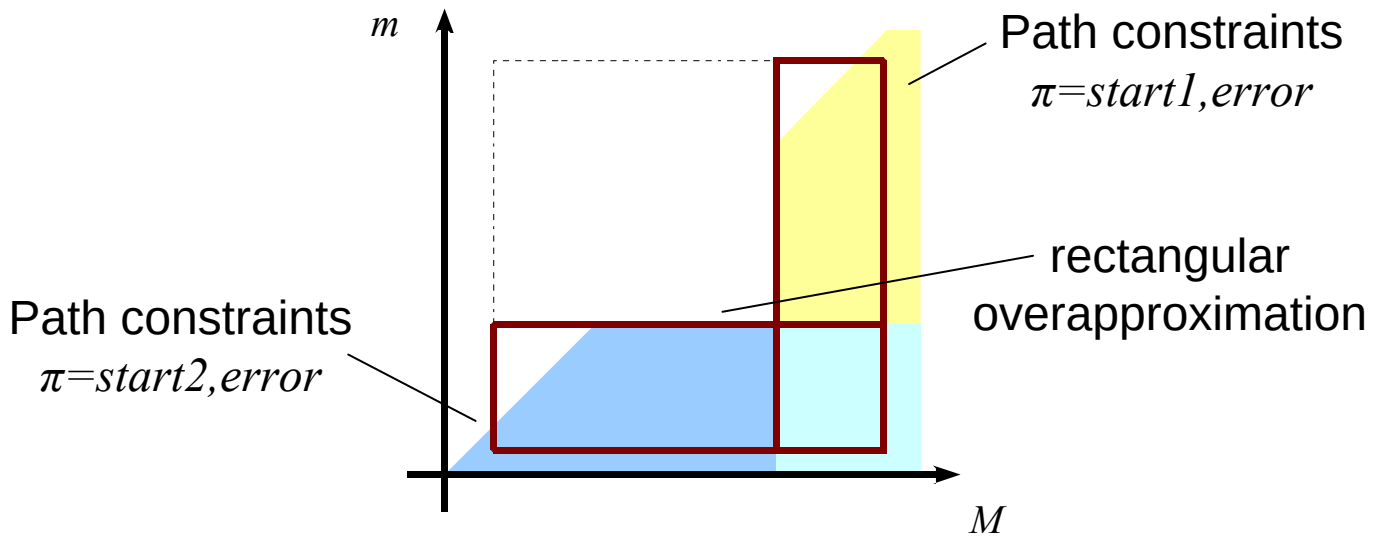
$$\hat{P}_G = P_0 \setminus \bigcup_{\pi \in CE(H,P_0)} OverAppr_{\mathtt{P}}([\![PathCon(\pi, p)]\!])$$



Heuristic 3: Rectangular/Octagonal Overapproximations

Overapproximating paths individually:

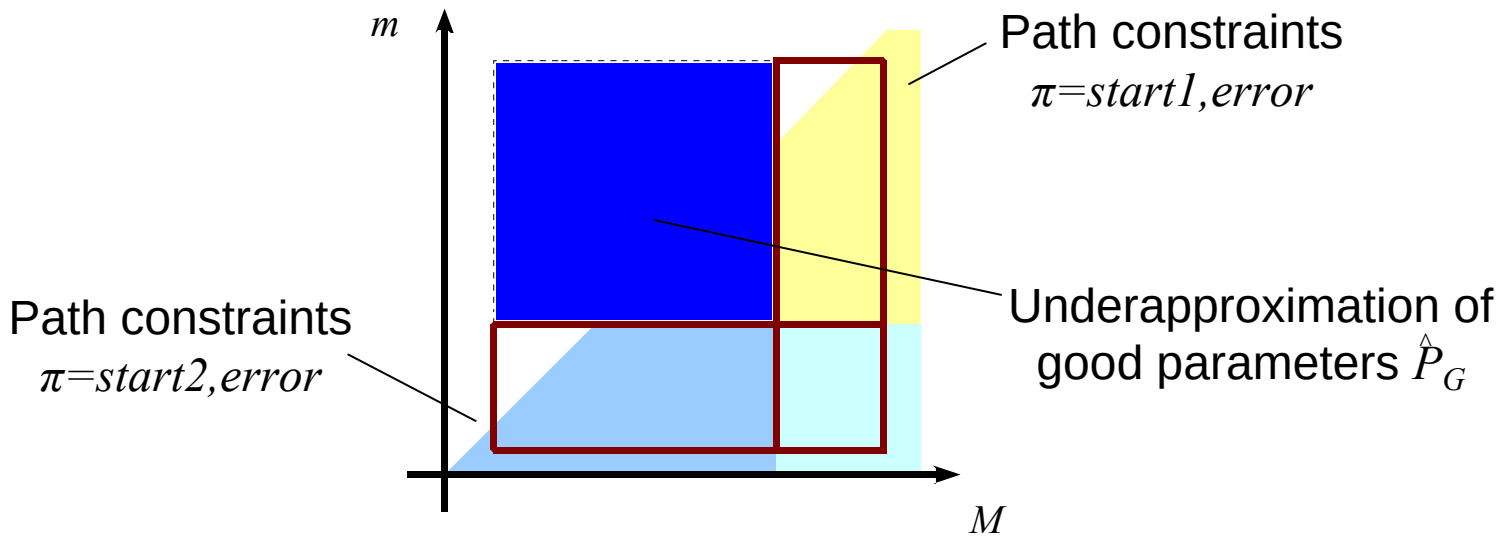
$$\hat{P}_G = P_0 \setminus \bigcup_{\pi \in CE(H,P_0)} OverAppr_{\mathtt{P}}([\![PathCon(\pi, p)]\!])$$



Heuristic 3: Rectangular/Octagonal Overapproximations

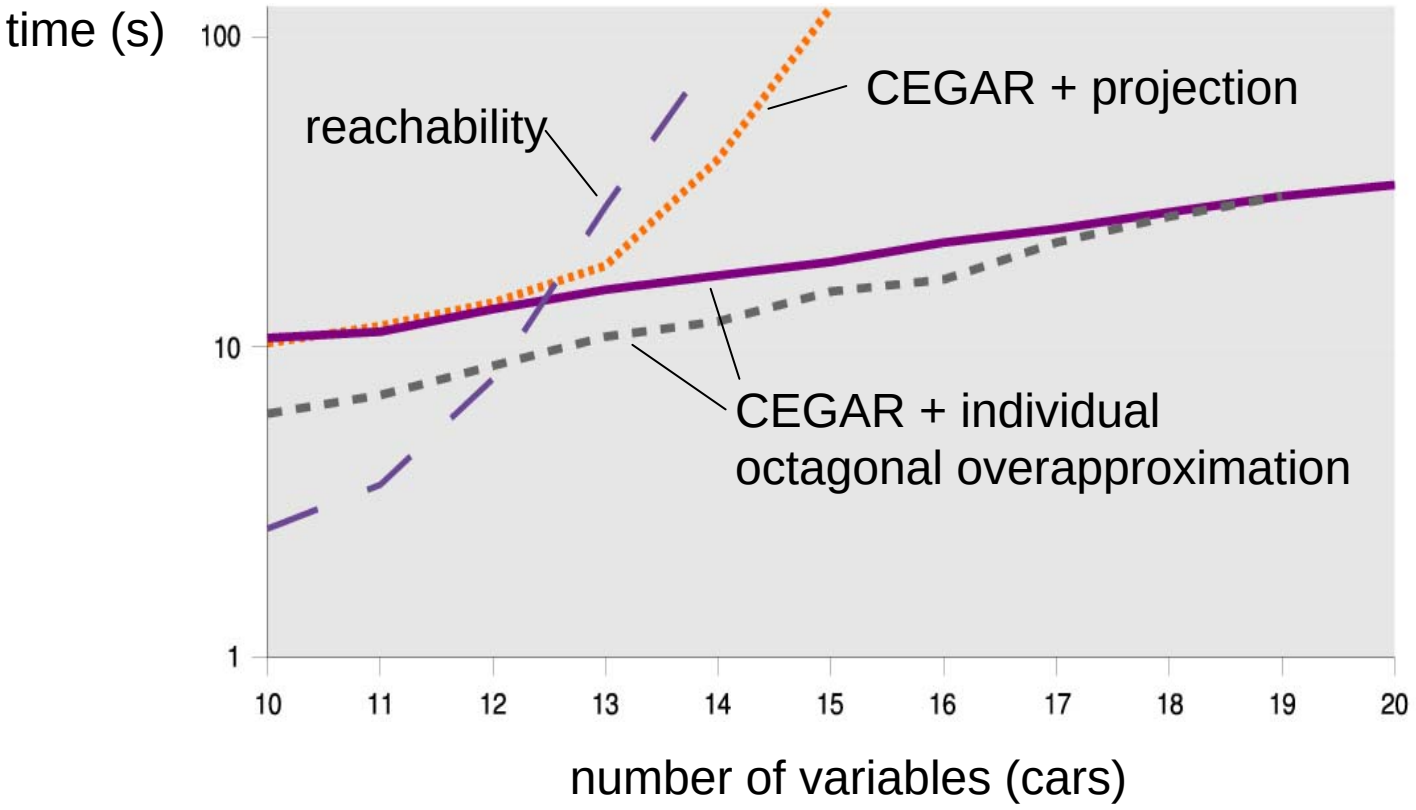
Overapproximating paths individually:

$$\hat{P}_G = P_0 \setminus \bigcup_{\pi \in CE(H,P_0)} OverAppr_{\mathbf{p}}([\![PathCon(\pi, p)]\!])$$



Experimental Results

Automated Highway Controller with 2 parameters



Summary

- **PhaVER – Computational tool for reachability analysis of hybrid systems with linear dynamics**
- **Iterative relaxation abstraction for verification of linear hybrid automaton**
  - reachability analysis applied to low-order abstractions
  - LP analysis applied to full-order counterexamples
- **Parameterized LHA for design**
  - monotonic parameters
  - heuristics to deal with non-convex feasible sets

Thanks to:

Edmund Clarke
Ansgar Fehnker
Goran Frehse (PHAVer)
Sumit Jha
Jim Kapinski
Flavio Lerda
Olaf Stursberg