SAND2019-0254C

EUForia: Complete Software Model Checking with Uninterpreted Functions

Denis Bueno and Karem A. Sakallah
University of Michigan
VMCAI 2019, Cascais, Portugal

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Control Properties

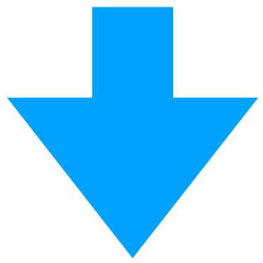
```
extern int CheckSecure();

int EstablishConnection() {
    int ret = 0;
    /* … checks */
        goto out;

    ret = CheckSecure();

out:
    return ret;
}
```

inadvertently added

- Abstract State Graphs with PVS (Graf & Saidi, ’97)
- SLAM (Ball et al., ’01)
- BLAST (Henzinger et al., ’02)
- Lazy Abstraction with Interpolants (McMillan, ‘06)
- Abstract Model Checking without Computing the Abstraction (Tonetta, ’09)
- IC3 + Implicit Abstraction (Cimatti, ’14)

Thesis

Most operations are irrelevant to proving control properties

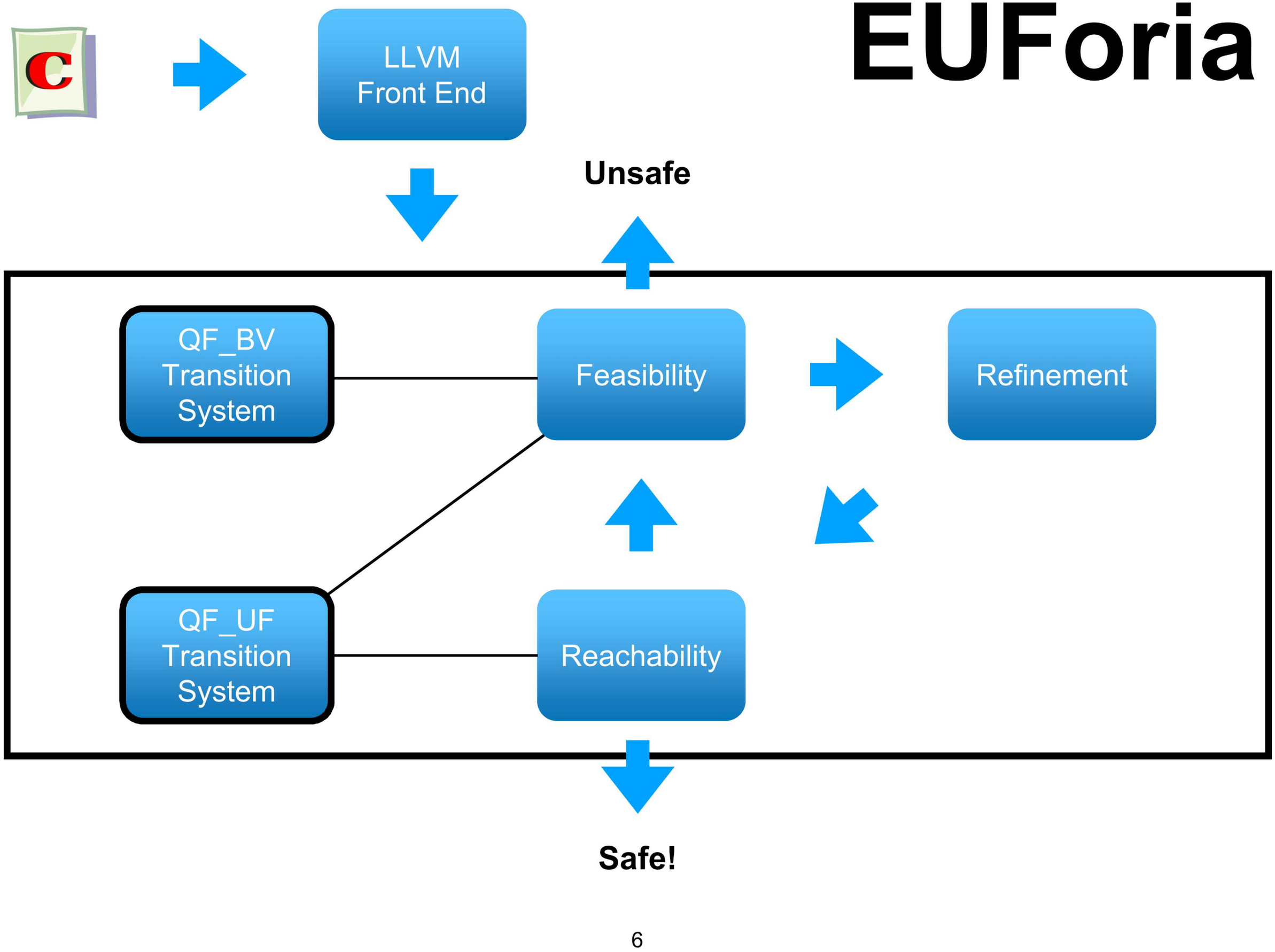


Abstract operations

Abstract predicates

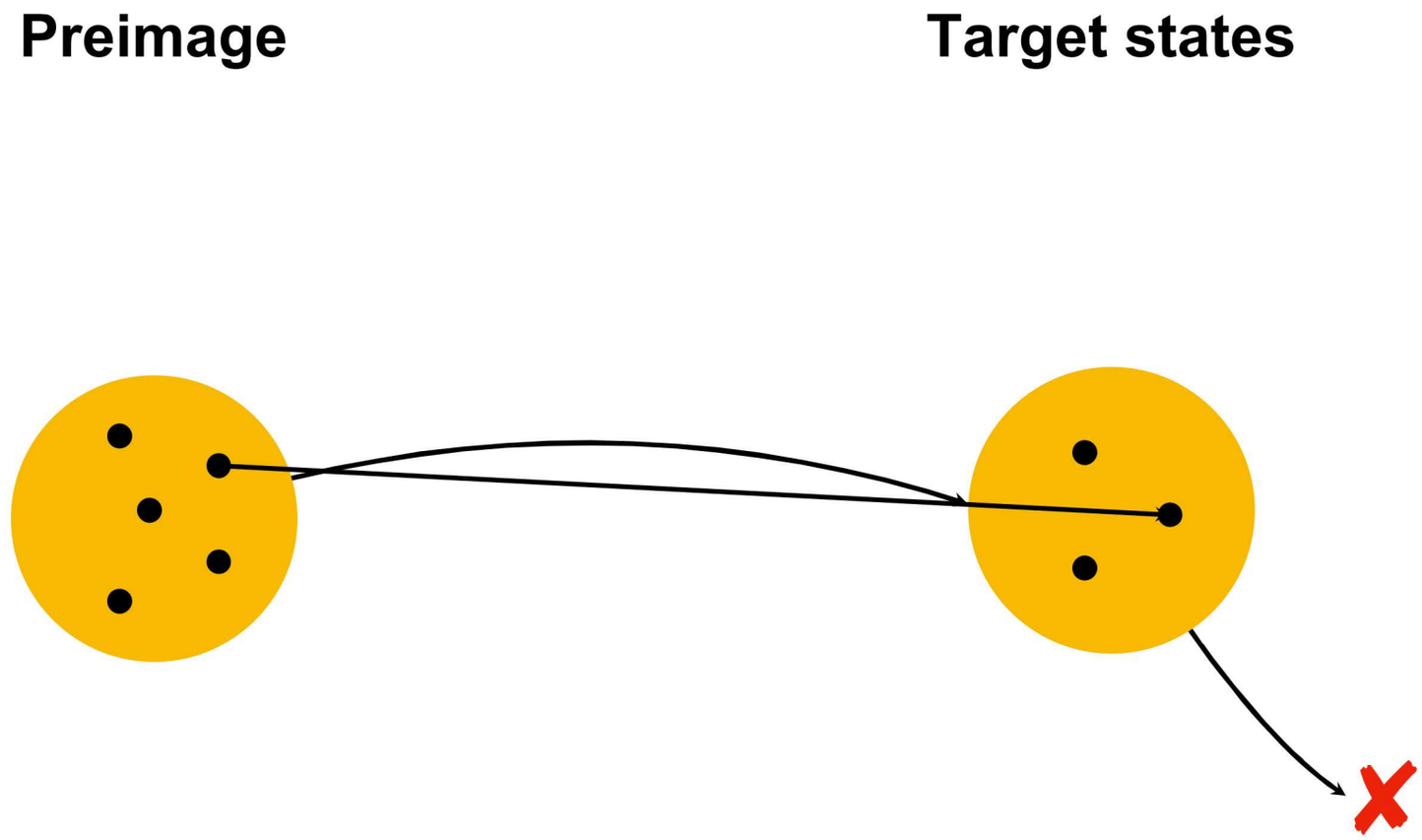
Preserve equality and functional consistency

EUForia


LLVM
Front End
Unsafe
QF_BV
Transition
System
Feasibility
Refinement
QF_UF
Transition
System
Reachability
Safe!

- Computing preimages in EUF
- Refinement lemma learning
- (Termination)
- Evaluation

Generating Preimages


Preimage
Target states

x’ = ITE(x = y, x+1, x-3)
y’ = x

x	y
0	0
1	0
-2	1
-5	-2

x’ = ITE(x = y, ADD(x,K1), SUB(x,K3))
y’ = x

x	y
K0	K0
ADD(X,K1)	K0
*	*
…	…

x’ = ITE(x = y, ADD(x,K1), SUB(x,K3))
y’ = x

Preimage state | **Model** | **Target state**

GT(x, y) & GT(x’, y’)

GT(x, y)
x = y
x ≠ K1
K1 = ADD(x, K1)
K1 ≠ SUB(x, K3)
K3 ≠ SUB(x, K3)

x, y, y’

K1, ADD(x, K1), x’

K3, SUB(x, K3)

t = GT(x’,y’)

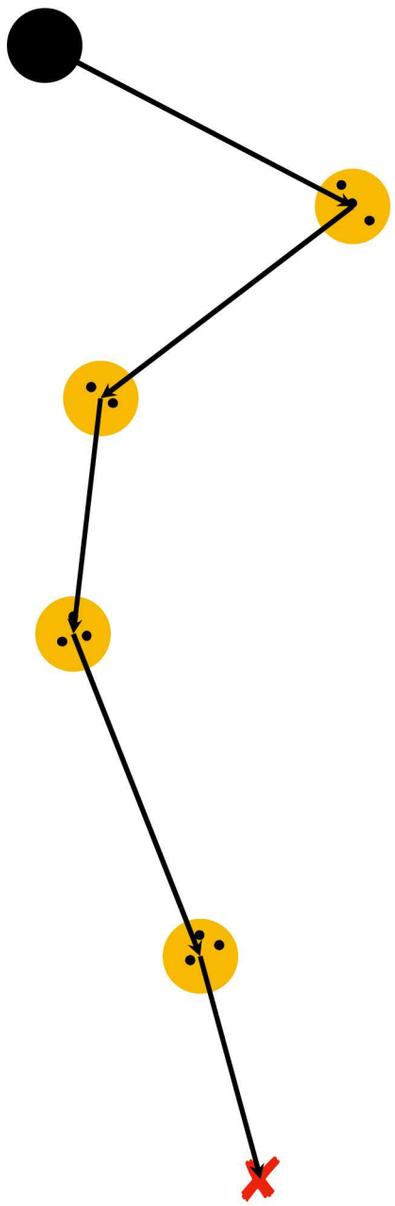
x = y K1 ADD(X, K1) x y

x’ = ITE(x = y, ADD(x,K1), SUB(x,K3))
x’ = x

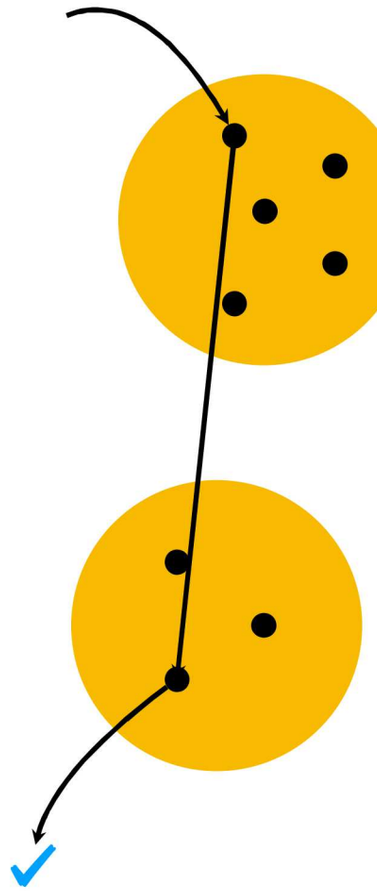
Preimage state	**Model**	**Target state**
x = y	GT(x, y) & GT(x’, y’)	
ADD(x, K1) = K1	x, y, y’	t = GT(x’, y’)
	K1, ADD(x, K1), x’	
x ≠ K1	K3, SUB(x, K3)	

Refinement

Abstract World (QF_UF)

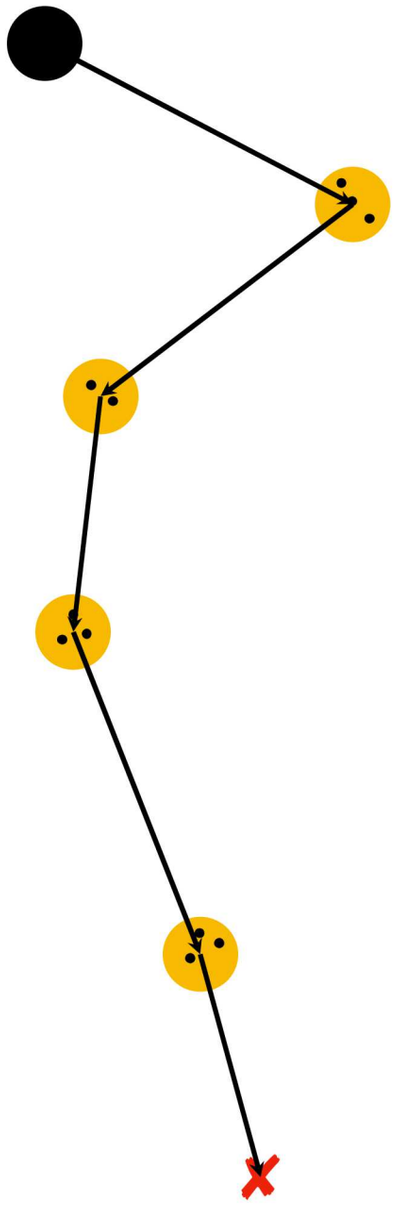


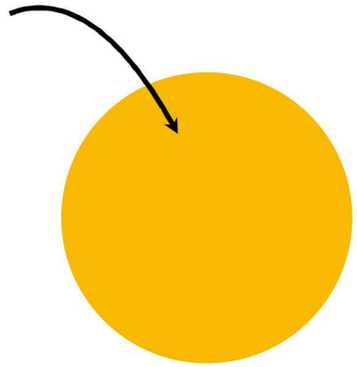
Concrete World (QF_BV)



Abstract World (QF_UF)

Concrete World (QF_BV)



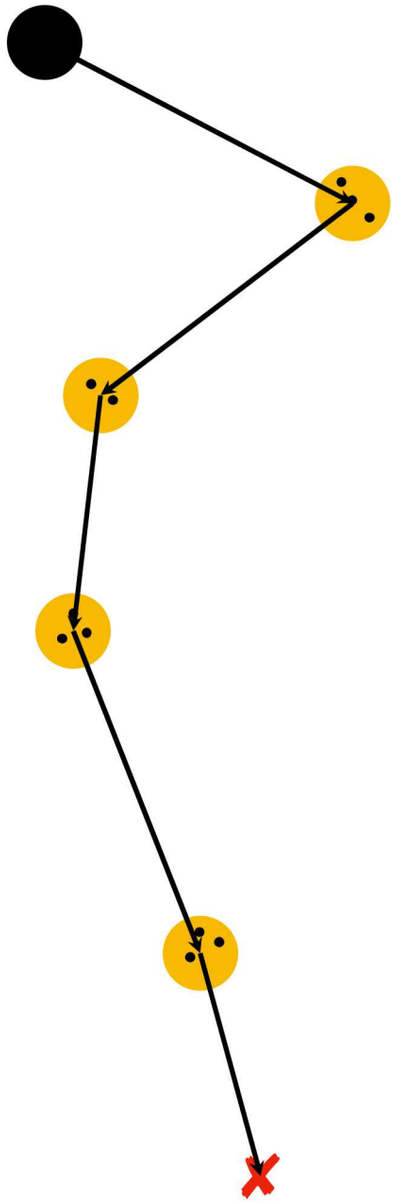


Infeasible reason #1

No concrete states

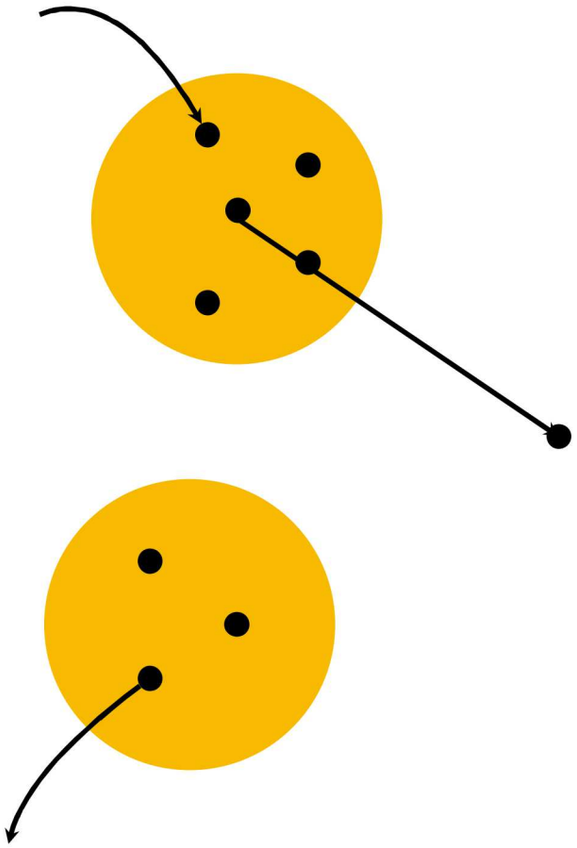
Abstract World (QF_UF)

Concrete World (QF_BV)



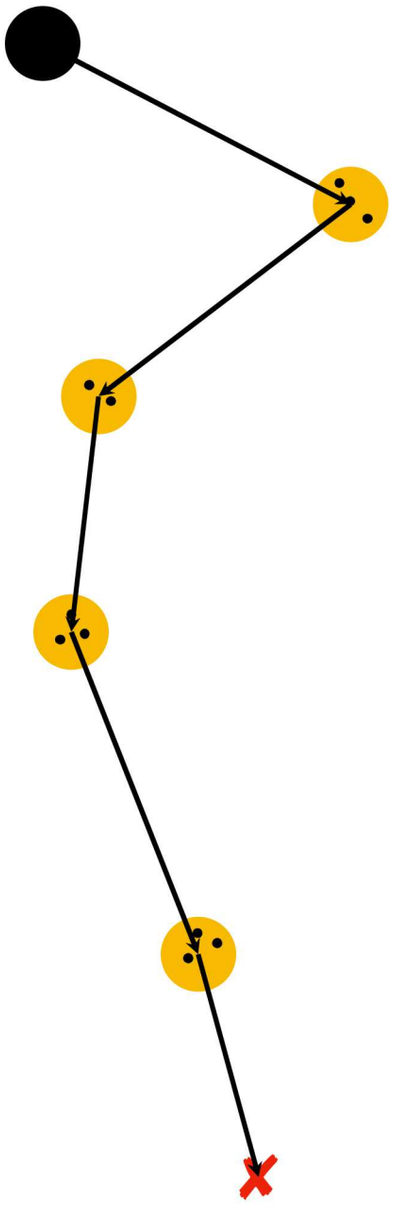
Infeasible reason #2

No concrete transition



Abstract World (QF_UF)

Concrete World (QF_BV)



Infeasible reason #3

Disconnected transitions

```
i = 0
do
  if (i > 10) then ERROR
  i = i+1
while (i < 5)
…
```

b: [i = 0]

```
b = 1
do
  if (!b) then ERROR
  b = b ? 0 : *;
while (1)
…
```

abstraction 1

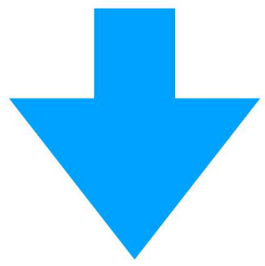
b: [i = 0]
c: [i < 5]

```
b, c = 1, 1
do
  if (!c) then ERROR
  b, c = b ? 0 : (c ? * : 0),
            c ? (b ? 1 : *) : 0;
while (c)
…
```

abstraction 2

How do we discover new predicates?

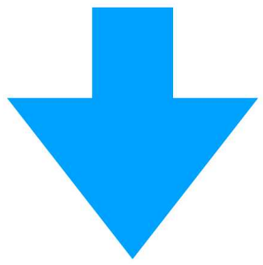
Discovering new predicates
doubles state space size



Exponential state space growth
as predicates are added

Infeasible transitions

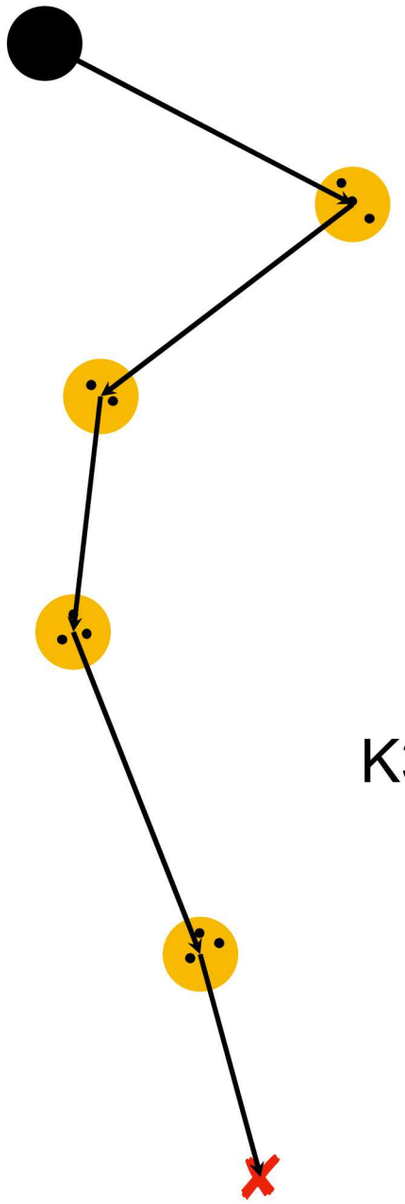
Infeasible states



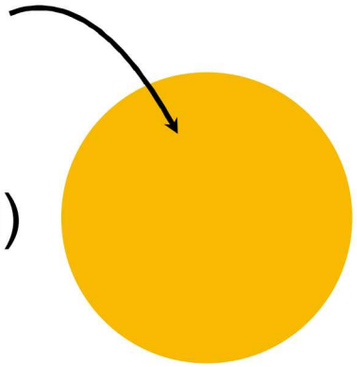
Lemmas do not increase size of state space

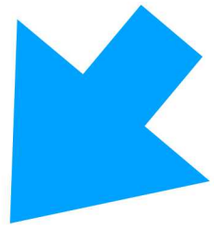
Abstract World (QF_UF)

Concrete World (QF_BV)



$(i = 0) \wedge (3 \neq 3 + i)$





K3 = ADD(K3, K0)

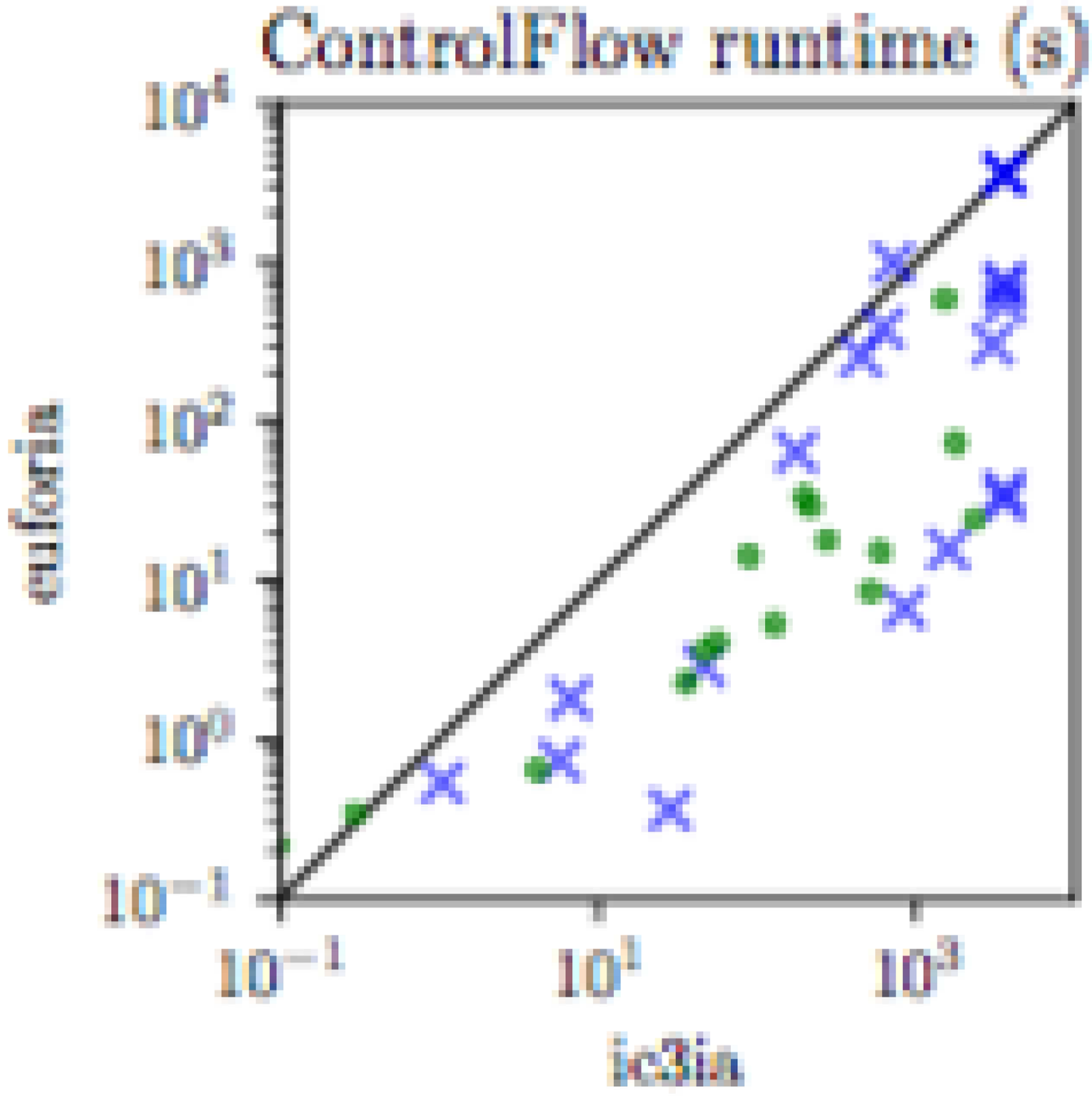
Out of 249 benchmarks that both EUForia and ic3ia solved

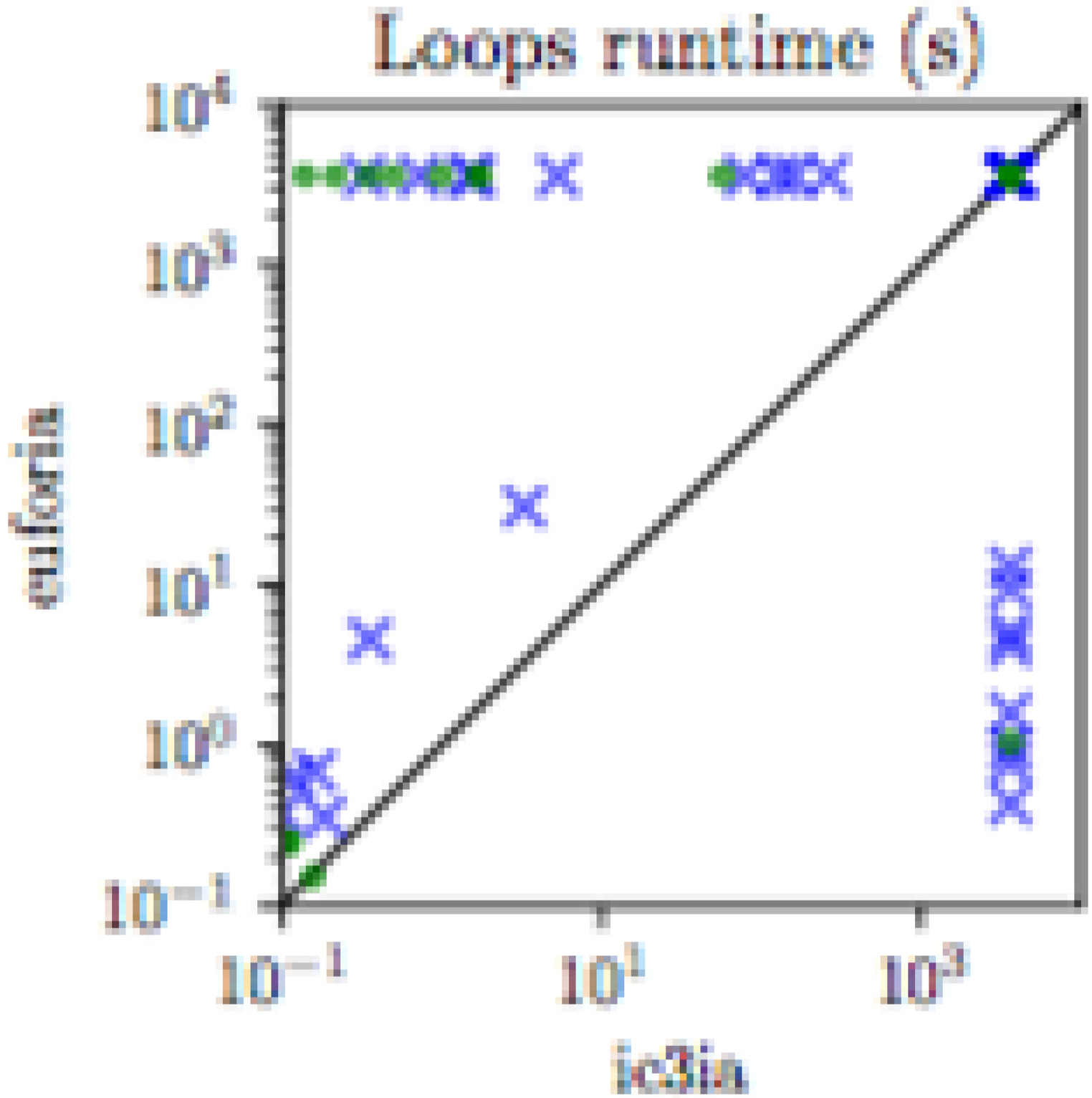
Nearly 200 are solved without refinement

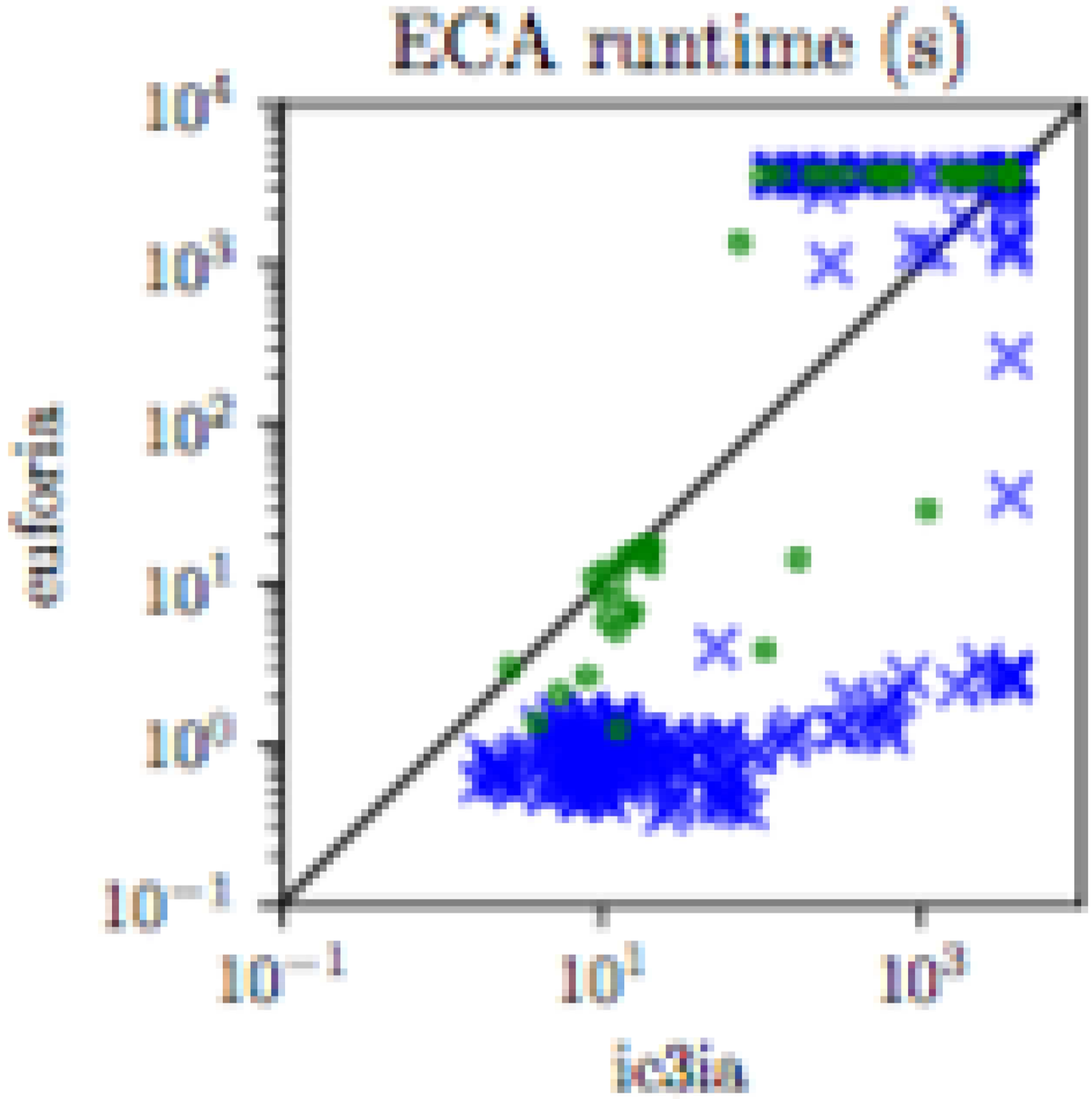
Majority of refinements do not increase state space size

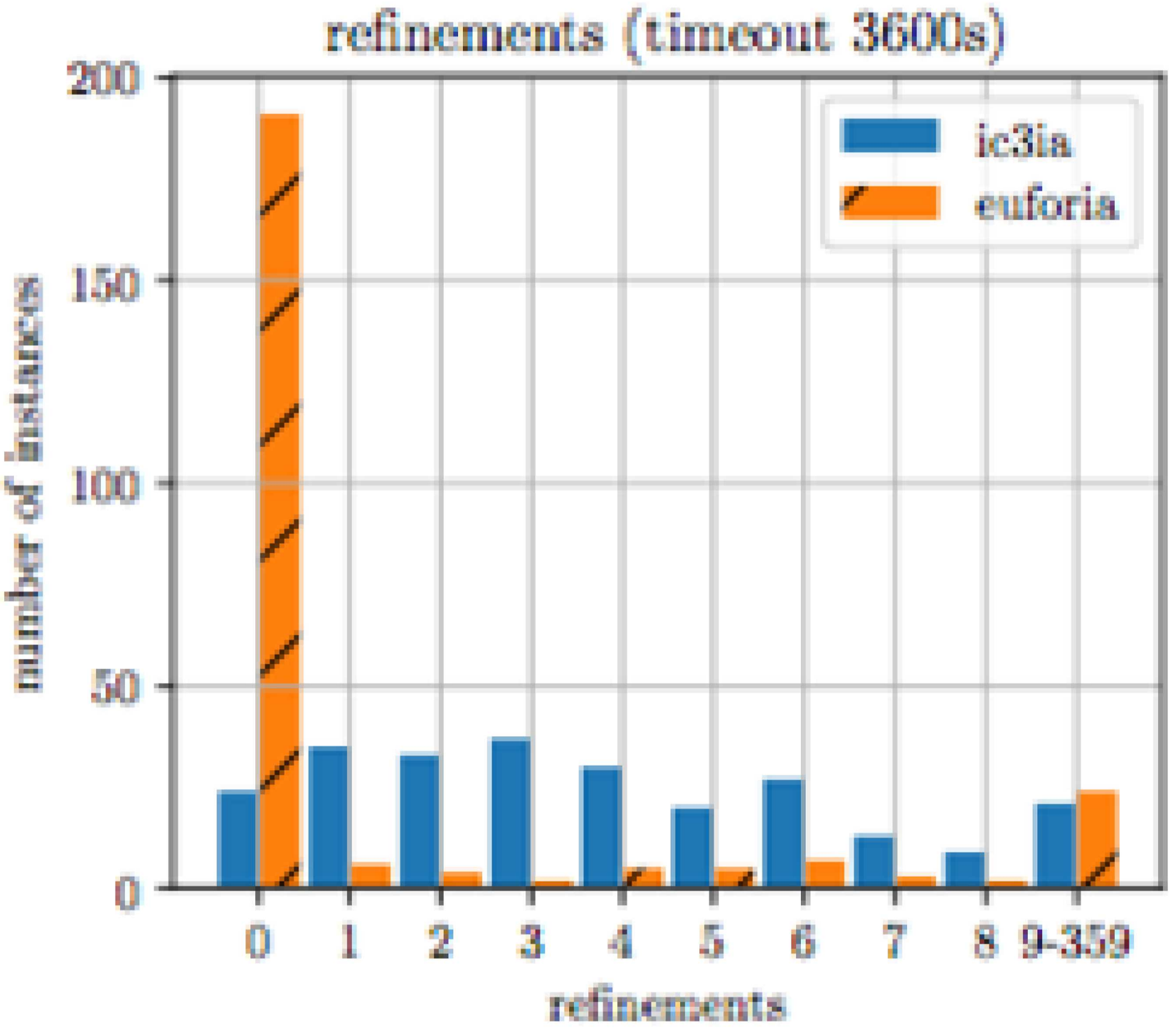
How well does it work?

- EUForia is ~14,000 lines of code
- 752 SV-COMP benchmarks: ControlFlow, Loops, ECA
- Compared with IC3IA (Cimatti et al., ’14)


ControlFlow runtime (s)
10^{4}
10^{3}
10^{2}
10^{1}
10^{0}
10^{-1}
10^{-1}
10^{1}
10^{3}
ic3in


Loops runtime (s)
10^4
10^3
10^2
10^1
10^0
10^{-1}
10^{-1}
10^1
10^3


ECA runtime (s)
10^4
10^3
10^2
10^1
10^0
10^{-1}
10^{-1}
10^1
10^3
ic3in


refinements (timeout 3600s)
200
150
100
50
0
number of instances
ic3ia
euforia
0
1
2
3
4
5
6
7
8
9-359
refinements

Future Work

- Support for arrays, recursion
- Partial concretization
- Refinement augmentation

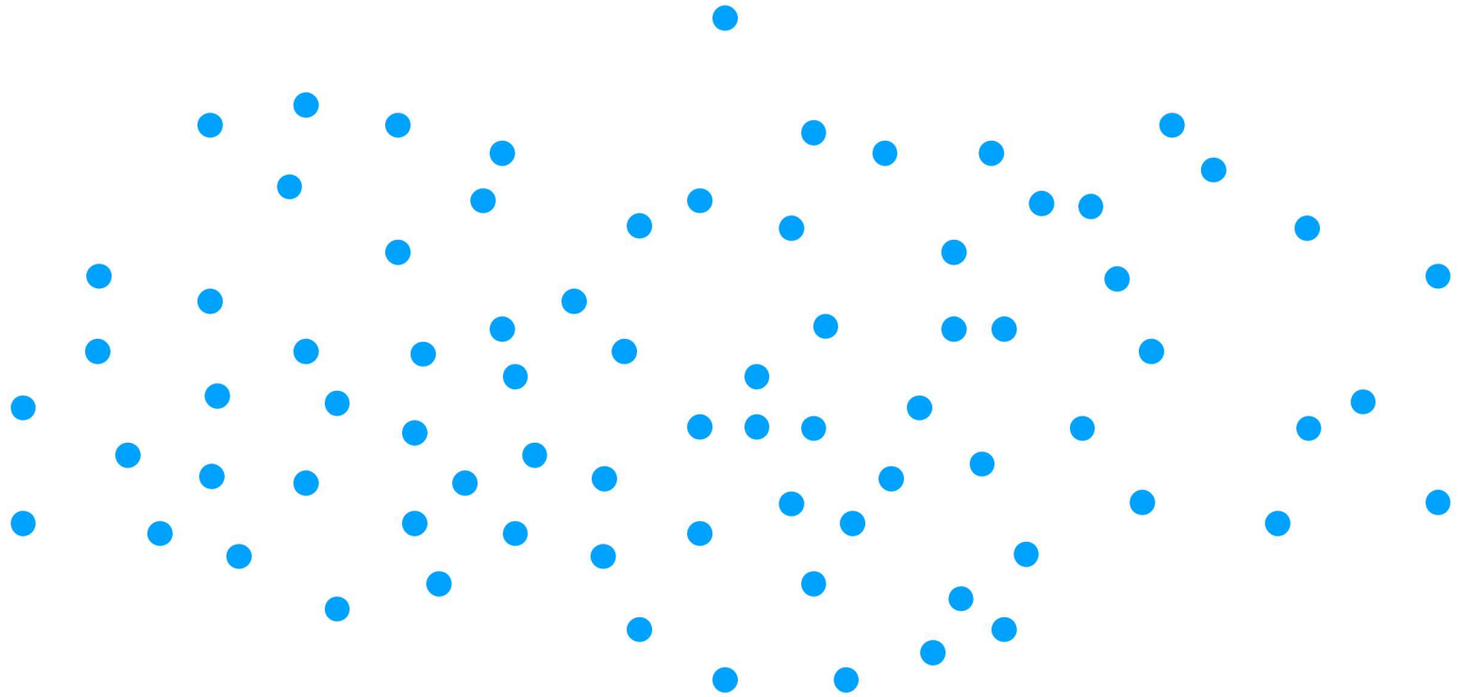
Backup Slides

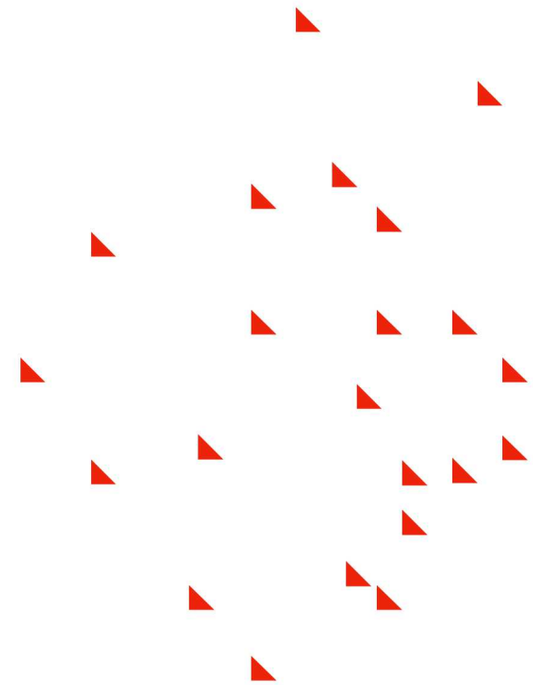
Safe states

P

!P

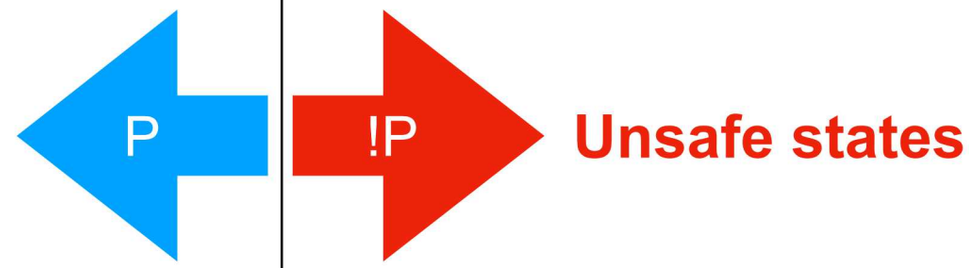
Unsafe states



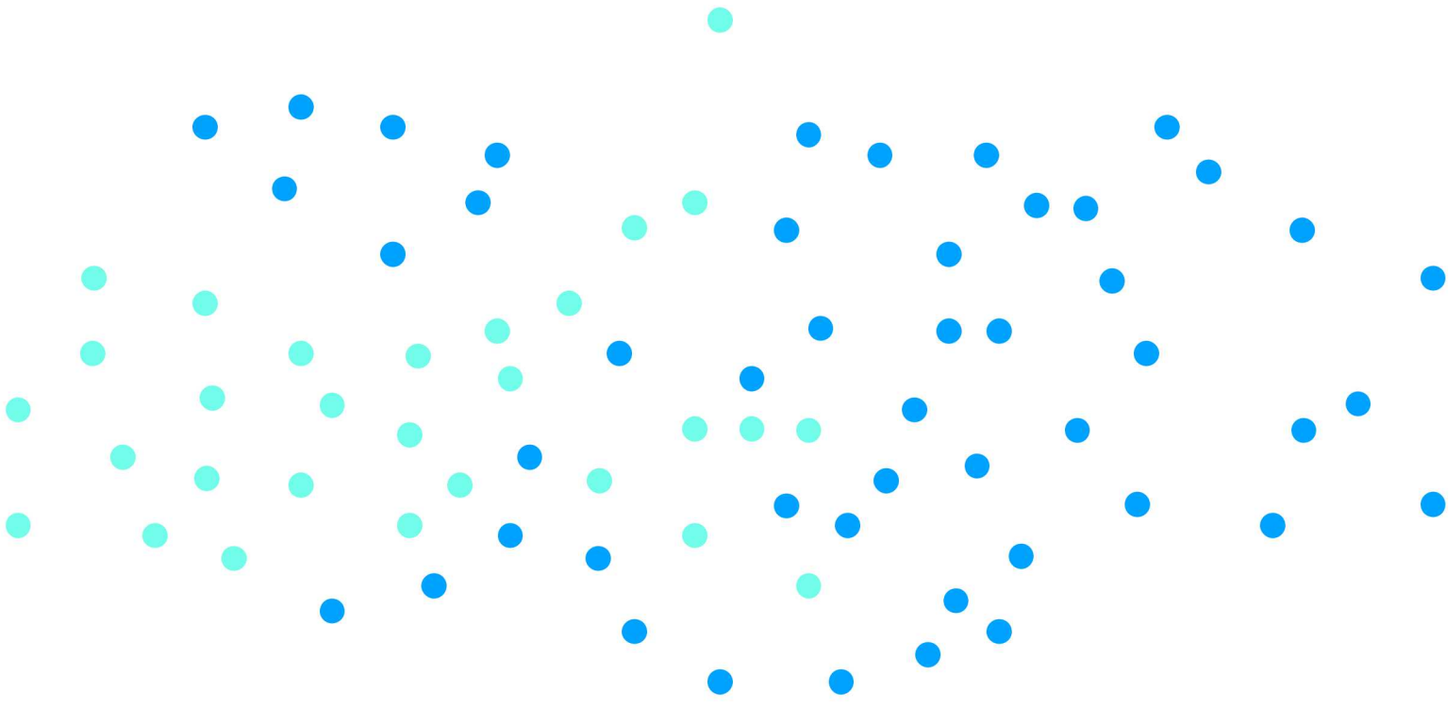


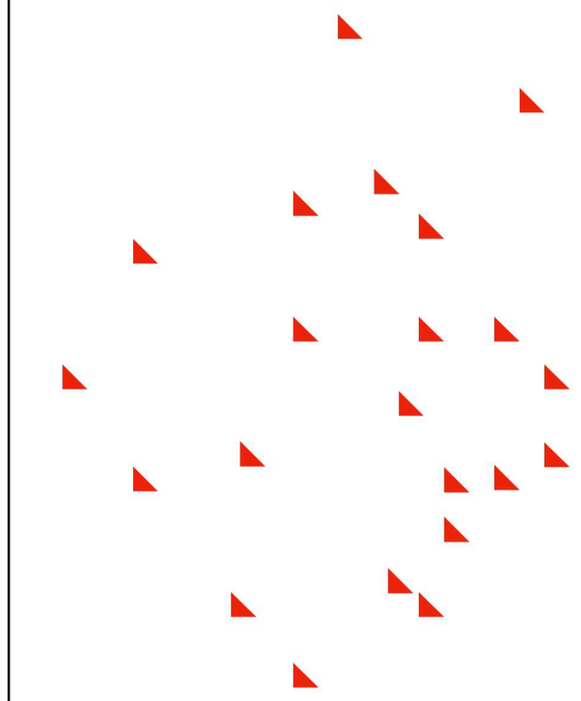
Safe states

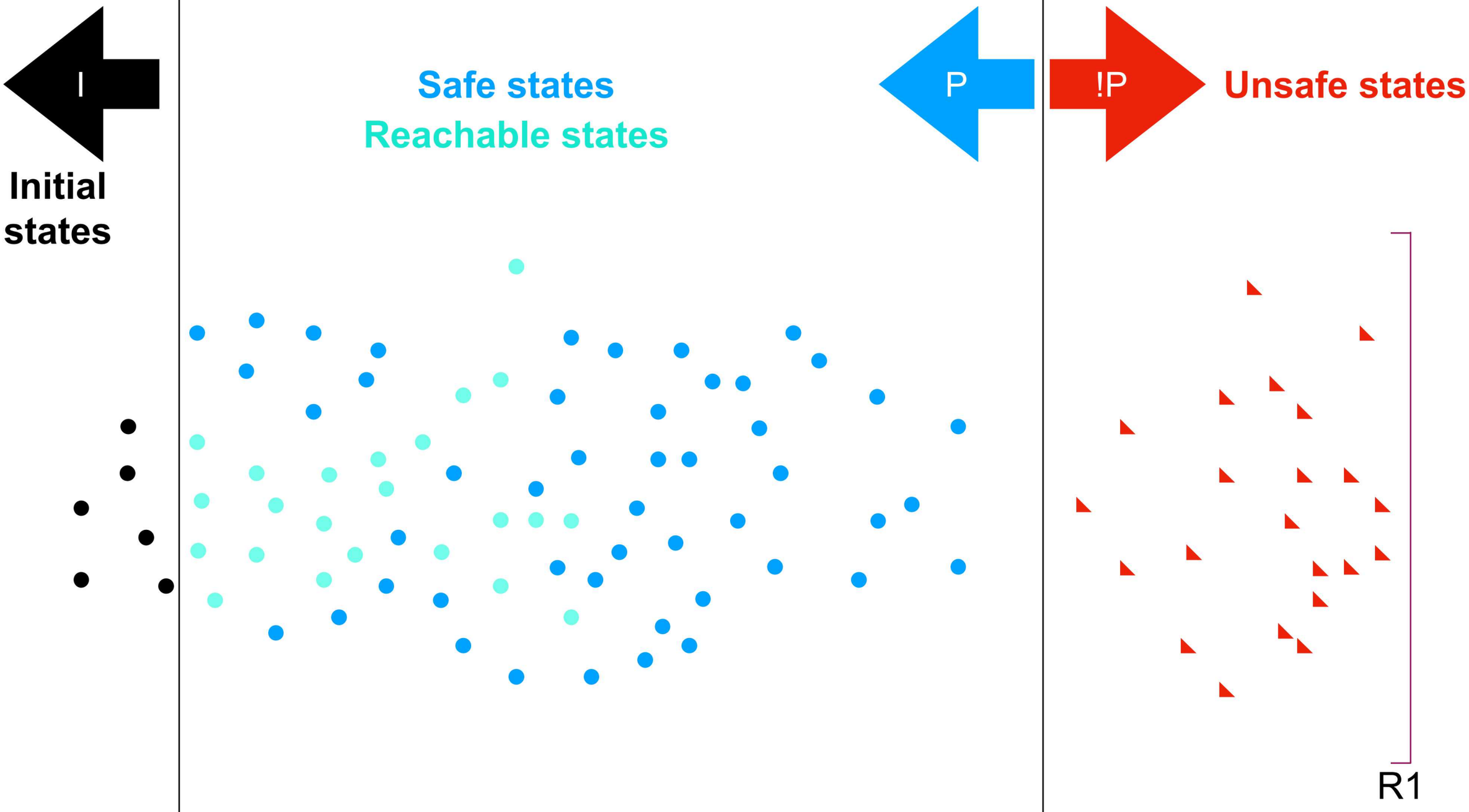
Reachable states



Unsafe states

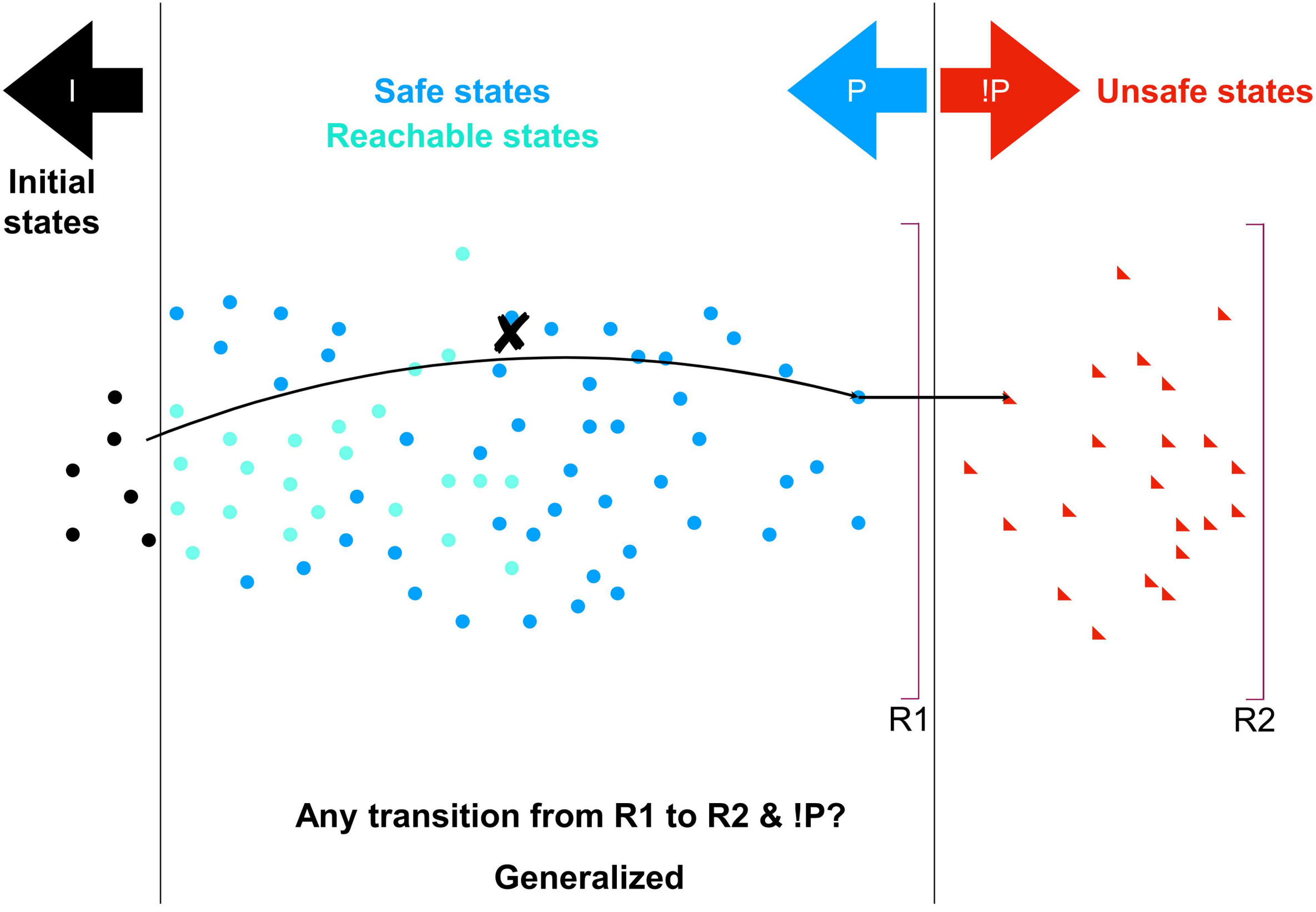






Any transition from I to R1 & !P?

(Let's say no)


I
Initial states
Safe states
Reachable states
P
!P
Unsafe states
R1
R2
Any transition from R1 to R2 & !P?
Generalized