LoopInvGen:

Data-Driven Loop Invariant Inference using Learned Features

SyGuS-COMP 2017

Saswat Padhi
Todd Millstein
(University of California, Los Angeles)

Verification

C/C++ Code

```c
int n, x = 0, m = 0;

while (x < n) {
  if (rand()) m = x;
  x = x + 1;
}

if(n > 0)
  assert (0 <= m && m <= n );
```

Verification

C/C++ Code

```c
int n, x = 0, m = 0;

while (x < n) {
  if (rand()) m = x;
  x = x + 1;
}

if(n > 0)
  assert (0 <= m && m <= n );
```

SyGuS-INV Problem

```
(define-fun pre-f ((x Int) (n Int) (m Int)) Bool
    (and (= x 0) (= m 0)))

(define-fun trans-f ((x Int) (n Int) (m Int)
                     (x! Int) (n! Int) (m! Int)) Bool
    (or (and (and (and (< x n) (= x! (+ x 1)))
                  (= n! n)) (= m! m))
        (and (and (and (< x n) (= x! (+ x 1)))
                  (= n! n)) (= m! x))))

(define-fun post-f ((x Int) (n Int) (m Int)) Bool
    (not (and (and (>= x n) (> n 0))
              (or (<= n m) (< m 0)))))

(inv-constraint inv-f pre-f trans-f post-f)
```

Verification

C/C++ Code

```c
int n, x = 0, m = 0;

while (x < n) {
  if (rand()) m = x;
  x = x + 1;
}

if(n > 0)
  assert (0 <= m && m <= n );
```

```lisp
(define-fun inv-f ((x Int) (n Int) (m Int)) Bool
    (and (>= x m)
         (or (and (> n m) (> m 0))
             (= 0 m))))
```

SyGuS-INV Problem

```lisp
(define-fun pre-f ((x Int) (n Int) (m Int)) Bool
    (and (= x 0) (= m 0)))

(define-fun trans-f ((x Int) (n Int) (m Int)
                     (x! Int) (n! Int) (m! Int)) Bool
    (or (and (and (and (< x n) (= x! (+ x 1)))
              (= n! n)) (= m! m))
        (and (and (and (< x n) (= x! (+ x 1)))
              (= n! n)) (= m! x))))

(define-fun post-f ((x Int) (n Int) (m Int)) Bool
    (not (and (and (>= x n) (> n 0))
              (or (<= n m) (< m 0)))))

(inv-constraint inv-f pre-f trans-f post-f)
```

LoopInvGen

- Data-driven inference technique
  - Guided by program trace
  - Extensible predicate language

LoopInvGen

- Data-driven inference technique
  - Guided by program trace
  - Extensible predicate language
- Reduces invariant inference to precondition inference problems
  - Based on PIE[PLDI 2016]

LoopInvGen

- Data-driven inference technique
  - Guided by program trace
  - Extensible predicate language
- Reduces invariant inference to precondition inference problems
  - Based on PIE[PLDI 2016]
- Based on Escher[CAV 2013] program synthesizer
- Queries Z3[TACAS 2008] for verification

LoopInvGen

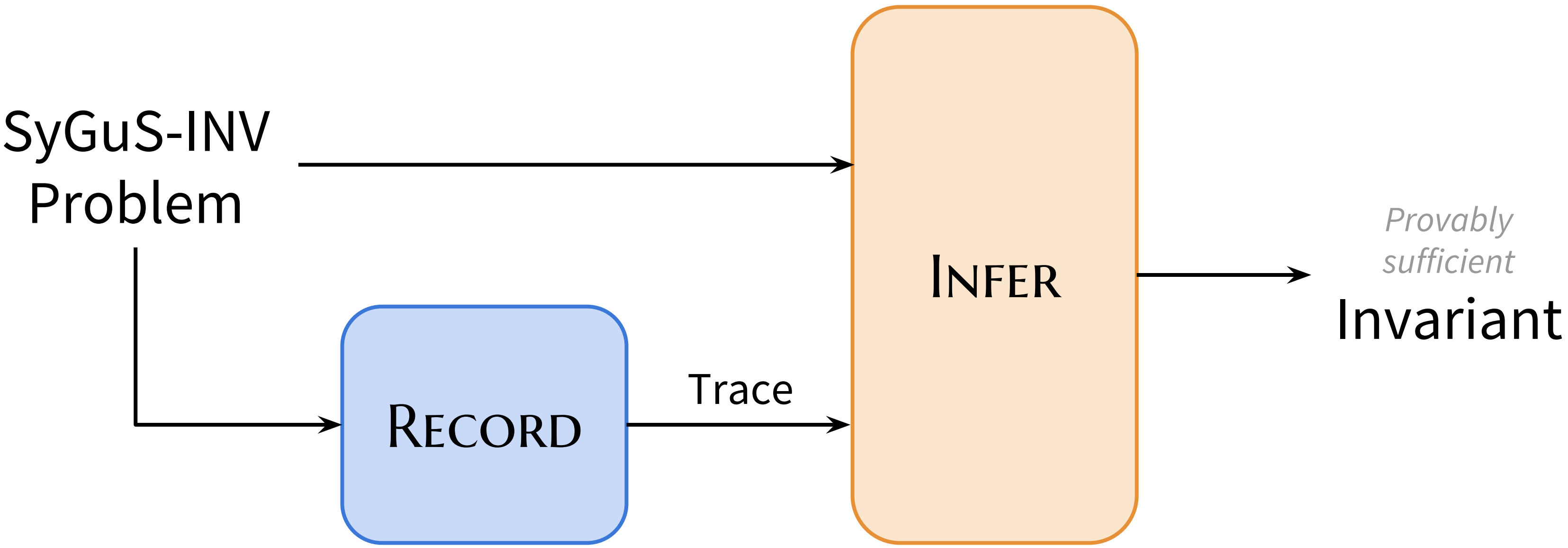
- Data-driven inference technique
  - Guided by program trace
  - Extensible predicate language
- Reduces invariant inference to precondition inference problems
  - Based on PIE[PLDI 2016]
- Based on Escher[CAV 2013] program synthesizer
- Queries Z3[TACAS 2008] for verification
- **Winner** of SyGuS-COMP 2017 (INV track)

LoopInvGen

- Data-driven inference technique
  - Guided by program trace
  - Extensible predicate language
- Reduces invariant inference to precondition inference problems
  - Based on PIE[PLDI 2016]
- Based on Escher[CAV 2013] program synthesizer
- Queries Z3[TACAS 2008] for verification
- **Winner** of SyGuS-COMP 2017 (INV track)

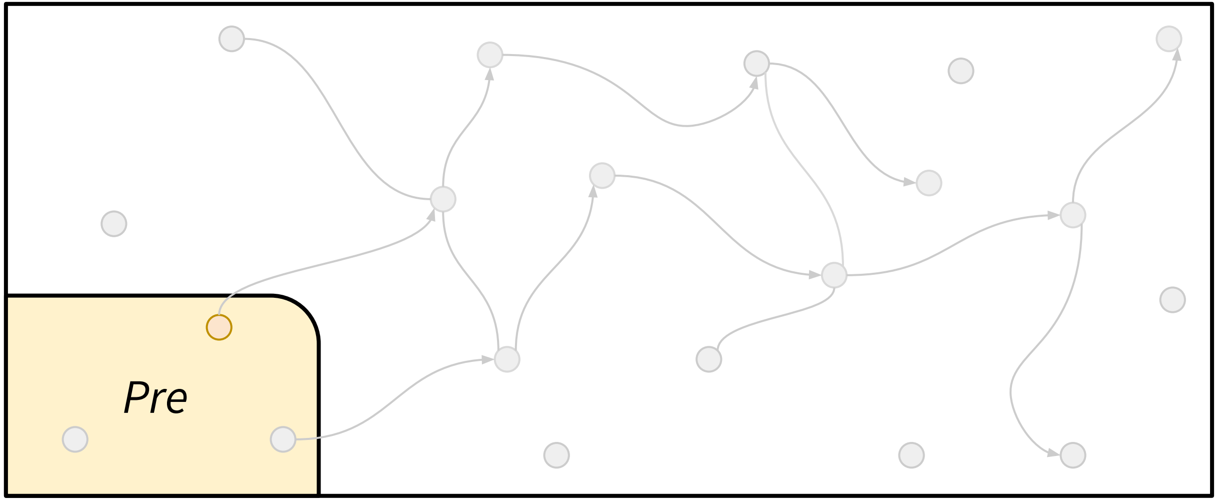
solver	solved	time (s)
EUSolver_new	40/74	2749.9
Euphony	58/74	11485.5
Alchemist CS	59/74	9697.3
ICE DT	60/74	637.5
DryadSynth	64/74	908.3
CVC4-061117-sygus-comp-2017	65/74	3676.1
LoopInvGen_3	65/74	54.1

Overview


SyGuS-INV
Problem
Record
Trace
Infer
Provably
sufficient
Invariant

RECORD-ing Reachable States

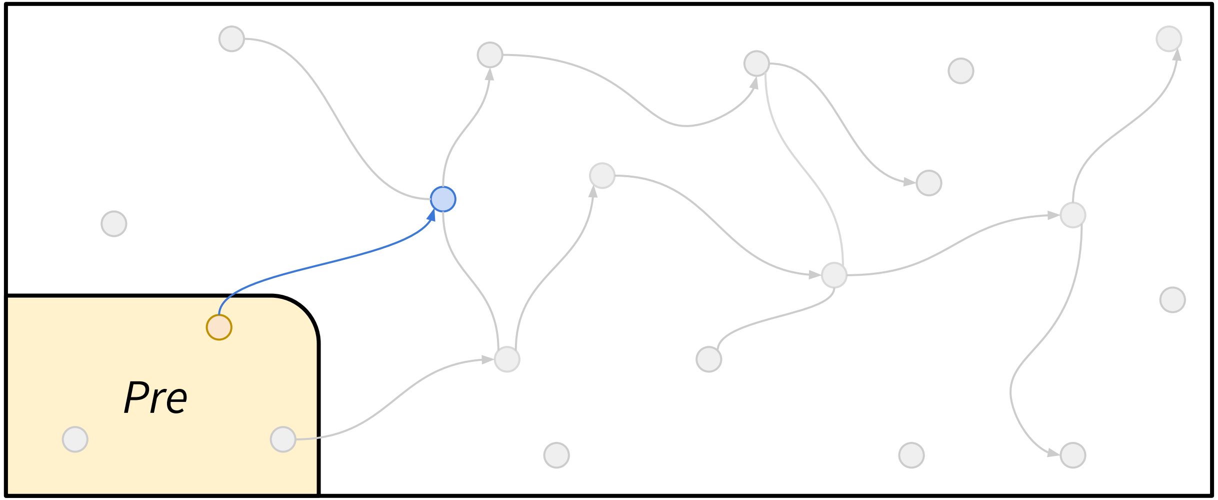
SyGuS Problem (Pre, Trans, Post) →List of variable assignments



1. Pick state s, s.t. Pre(s)

RECORD-ing Reachable States

SyGuS Problem (Pre, Trans, Post) →List of variable assignments

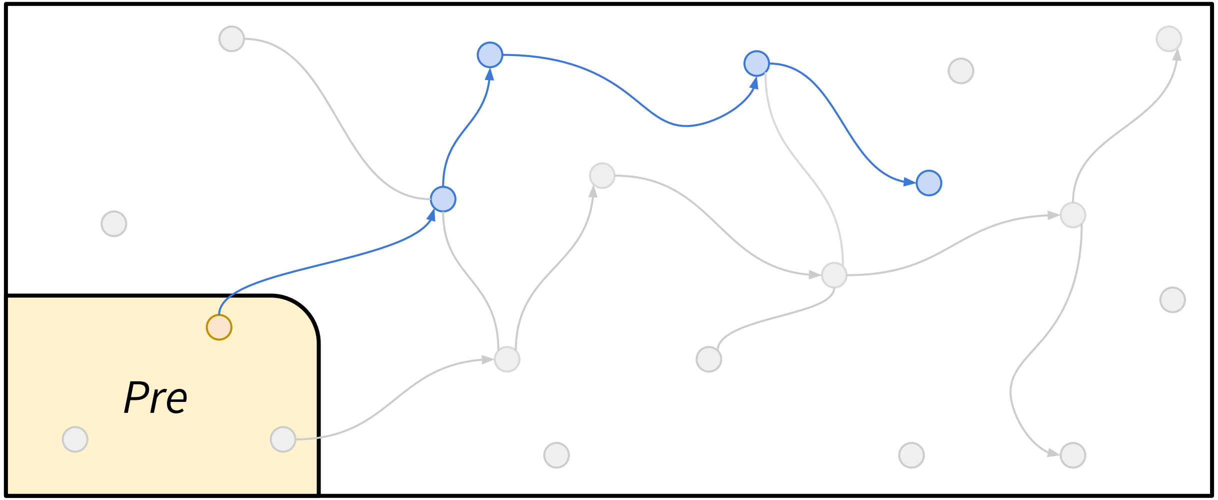


1. Pick state s, s.t. Pre(s)

2. Obtain state t, s.t. Trans(s,t)

RECORD-ing Reachable States

SyGuS Problem (Pre, Trans, Post) →List of variable assignments



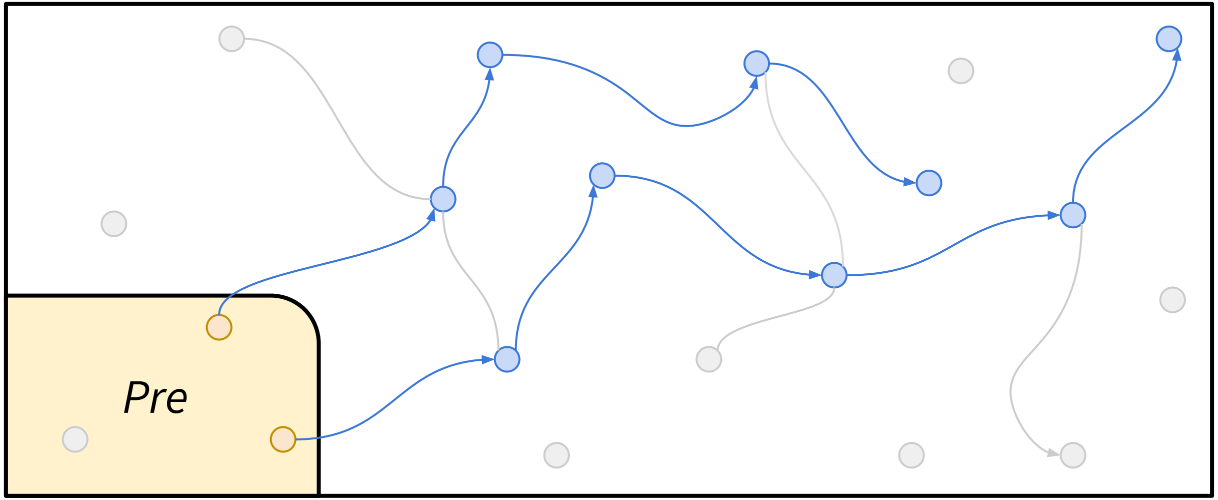
1. Pick state s, s.t. Pre(s)

2. Obtain state t, s.t. Trans(s,t)

3. Set s ← t and repeat (2)

RECORD-ing Reachable States

SyGuS Problem (Pre, Trans, Post) →List of variable assignments



1. Pick state s, s.t. Pre(s)   2. Obtain state t, s.t. Trans(s,t)   3. Set s ← t and repeat (2)

4. Repeat (1,2,3) till the desired number of states has been collected

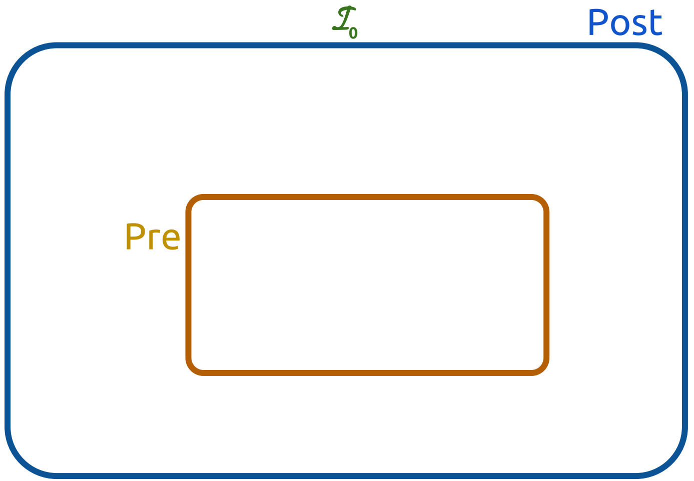
INFER-ing Sufficient Invariants

→ $\forall$ s: Pre(s) $\Rightarrow \mathcal{I}$(s)

→ $\forall$ s, t: $\mathcal{I}$(s) $\wedge$ Trans(s,t) $\Rightarrow \mathcal{I}$(t)

→ $\forall$ s: $\mathcal{I}$(s) $\Rightarrow$ Post(s)

1. Start with the weakest candidate



$\mathcal{I}_0$ = Post

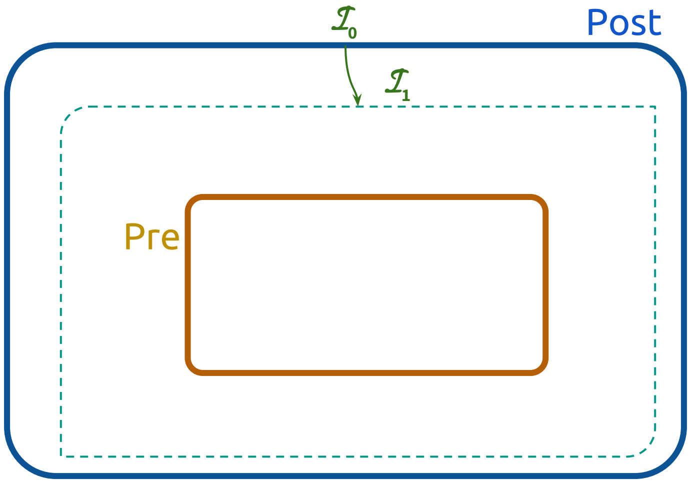
INFER-ing Sufficient Invariants

→ $\forall$ s: Pre(s) $\Rightarrow$ $\mathcal{I}$(s)

→ $\forall$ s, t: $\mathcal{I}$(s) $\wedge$ Trans(s,t) $\Rightarrow$ $\mathcal{I}$(t)

→ $\forall$ s: $\mathcal{I}$(s) $\Rightarrow$ Post(s)

1. Start with the weakest candidate
2. Iteratively strengthen for inductiveness (data-driven precondition inference)



$\mathcal{I}_0$ = Post

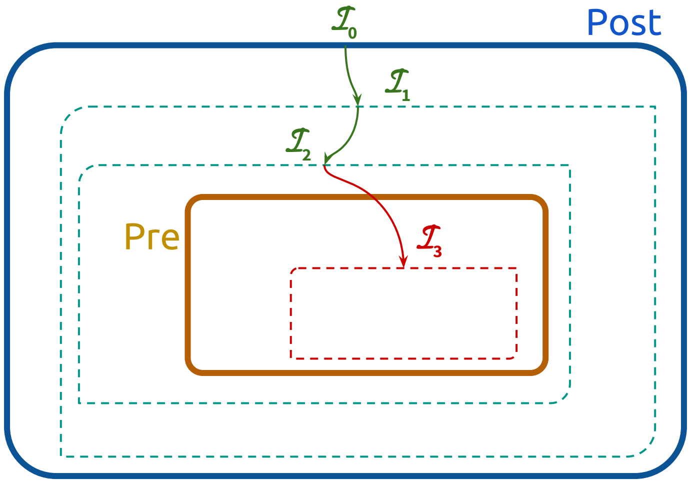
$\mathcal{I}_1 = \delta_0 \wedge \mathcal{I}_0$

$\delta_0 \Rightarrow (\mathcal{I}_0 \wedge \text{Trans} \Rightarrow \mathcal{I}_0')$

INFER-ing Sufficient Invariants

- $\forall\, s\colon \mathrm{Pre}(s) \Rightarrow \mathcal{I}(s)$
- $\forall\, s,t\colon \mathcal{I}(s) \wedge \mathrm{Trans}(s,t) \Rightarrow \mathcal{I}(t)$
- $\forall\, s\colon \mathcal{I}(s) \Rightarrow \mathrm{Post}(s)$

1. Start with the weakest candidate
2. Iteratively strengthen for inductiveness (data-driven precondition inference)



$\mathcal{I}_0 = \mathrm{Post}$

$\delta_0 \Rightarrow (\mathcal{I}_0 \wedge \mathrm{Trans} \Rightarrow \mathcal{I}_0')$

$\mathcal{I}_1 = \delta_0 \wedge \mathcal{I}_0$

$\delta_1 \Rightarrow (\mathcal{I}_1 \wedge \mathrm{Trans} \Rightarrow \mathcal{I}_1')$

$\vdots \quad \vdots \qquad\qquad \vdots \qquad \vdots$

$\mathcal{I}_n = \delta_{n-1} \wedge \mathcal{I}_{n-1} = \mathcal{I}_{n-1}$

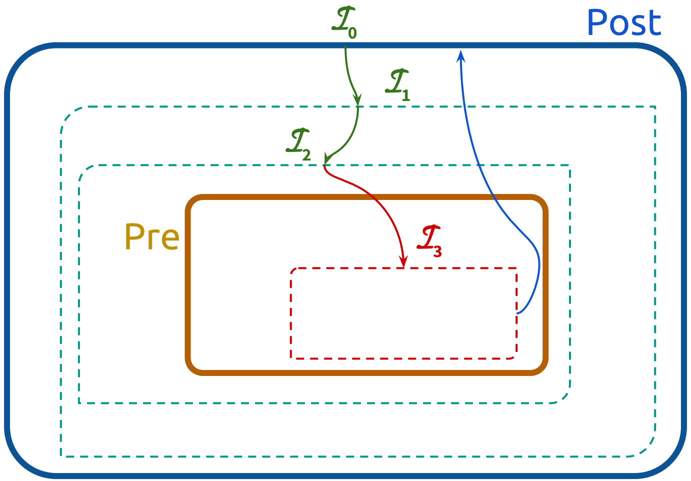
INFER-ing Sufficient Invariants

→ $\forall$ s: Pre(s) $\Rightarrow \mathcal{I}$(s)

→ $\forall$ s,t: $\mathcal{I}$(s) $\wedge$ Trans(s,t) $\Rightarrow \mathcal{I}$(t)

→ $\forall$ s: $\mathcal{I}$(s) $\Rightarrow$ Post(s)

1. Start with the weakest candidate
2. Iteratively strengthen for inductiveness (data-driven precondition inference)
3. If the invariant is too strong, restart from (1) after augmenting the recorded states with appropriate counterexamples



$\mathcal{I}_0$ = Post

$\delta_0 \Rightarrow (\mathcal{I}_0 \wedge \text{Trans} \Rightarrow \mathcal{I}_0')$

$\mathcal{I}_1 = \delta_0 \wedge \mathcal{I}_0$

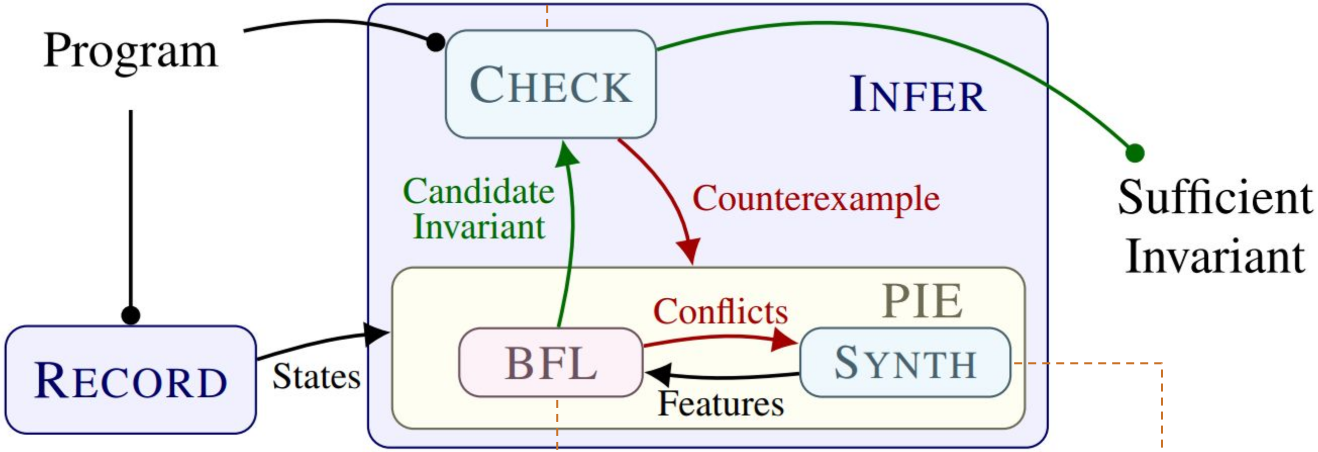
$\delta_1 \Rightarrow (\mathcal{I}_1 \wedge \text{Trans} \Rightarrow \mathcal{I}_1')$

$\vdots \quad \vdots \qquad \vdots \quad \vdots$

$\mathcal{I}_n = \delta_{n-1} \wedge \mathcal{I}_{n-1} = \mathcal{I}_{n-1}$

LoopInvGen Architecture

Z3 Theorem Prover [de Moura et al 2008]



Escher Synthesizer [Albarghouthi et al 2013]

Standard PAC algo for Learning k-CNF formulas

Thanks! ☺

Code + Benchmarks:

https://github.com/SaswatPadhi/LoopInvGen

Reach me at:

padhi @ cs . ucla . edu