Data-Driven Invariant Learning for Probabilistic Programs

Published in CAV'22 (Recieved Distinguished Paper Award)

Jailu Bao [1] Nitesh Trivedi [2] Drashti Pathak [3]
Justin Hsu [1] Subhajit Roy [2]

[1]Cornell University, [2]Indian Institute of Technology (IIT) Kanpur, [3]Amazon

Foolish Game

Mr. Fool

Foolish Game

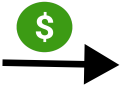
Mr. Fool

Me

Day 1

Mr. Fool

+



Me

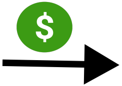
Mr. Annoying

Foolish Game

Day 2

Mr. Fool

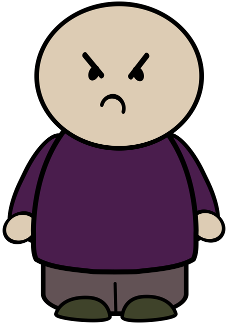
+



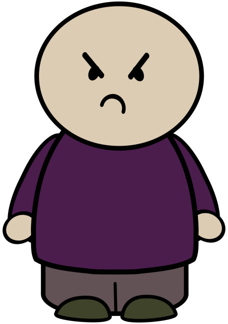
Me

Mr. Annoying

Game Continued...



```
main (p)
x = 0, n = 0
while(x == 0)
  x = bernoulli_dist(p)
  n += 1
```



p = 0.5

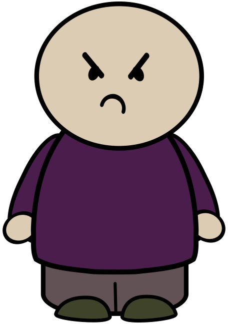
```
main (p)
x = 0, n = 0
while(x == 0)
  x = bernoulli_dist(p)
  n += 1
```

n = 3

n = 1

n = 0

Game Continued...



p = 0.3

```
main (p)
x = 0, n = 0
while(x == 0)
  x = bernoulli_dist(p)
  n += 1
```

n = 5   n = 2   n = 3

Probabilistic Programs

```
<1/p>
main (p)
x = 0, n = 0
while(x == 0)
  x = bernoulli_dist(p)
  n += 1
<n>
```

Pre Expectation
$< preE >$

Sampling Statement

Post Expectation
$< postE >$

Probabilistic Programs

```
<1/p>
main (p)
x = 0, n = 0
while(x == 0)
  x = bernoulli_dist(p)
  n += 1
<n>
```

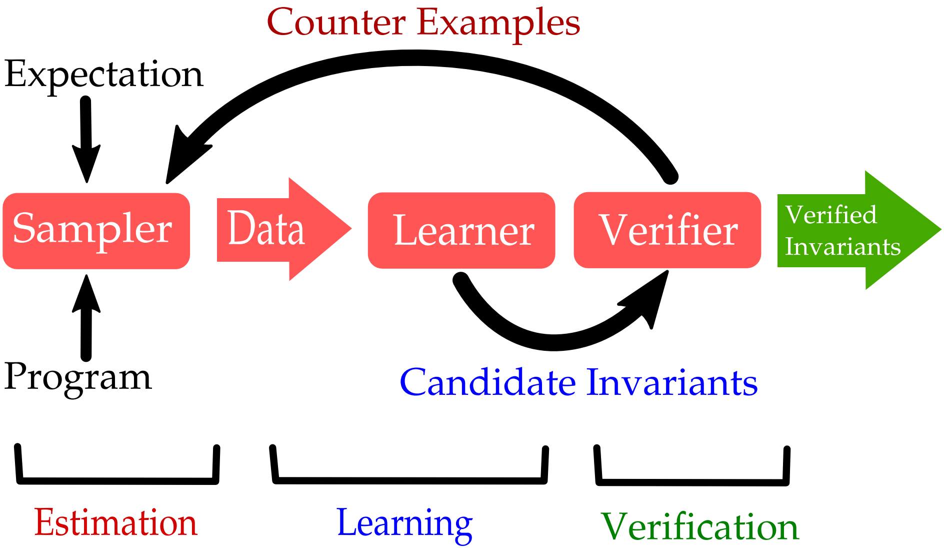
Pre Expectation
$< preE >$

Sampling Statement

Post Expectation
$< postE >$

$$\mathcal{I} = n + [x == 0] \cdot \frac{1}{p}$$

Given a loop *while* $G : P$ and an expectation *postE* as input, we aim to develop an algorithm to automatically synthesize an invariant $\mathcal{I}$.


Counter Examples
Expectation
Sampler
Data
Learner
Verifier
Verified Invariants
Program
Candidate Invariants
Estimation
Learning
Verification

Estimation

- Generate set of initial states

Initial States

x	p	n
0	0.1	0
0	0.5	2
1	0.5	4

Estimation

- Generate set of initial states
- Collect Traces on each initial state

Initial States

x	p	n
0	0.1	0
0	0.5	2
1	0.5	4

N_{runs}

x	p	Initial n	Final n
0	0.1	0	7
0	0.1	0	6
0	0.1	0	3
0	0.5	2	3
0	0.5	2	4
0	0.5	2	2
1	0.5	4	4

- Generate set of initial states
- Collect Traces on each initial state
- Estimate expectations: multiple runs from same initial state
- Generate dataset

Initial States

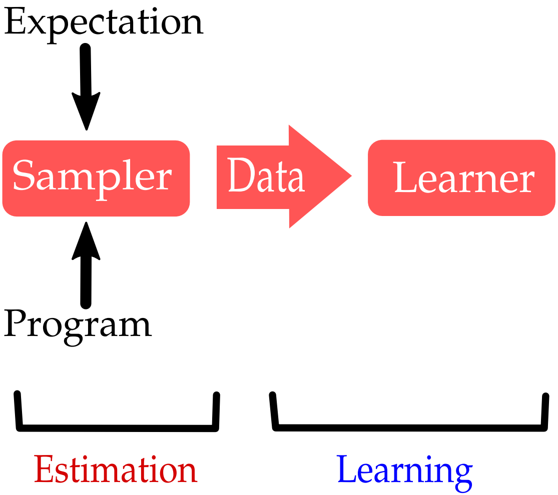
x	p	n
0	0.1	0
0	0.5	2
1	0.5	4

N_{runs}

x	p	Initial n	Final n
0	0.1	0	7
0	0.1	0	6
0	0.1	0	3
0	0.5	2	3
0	0.5	2	4
0	0.5	2	2
1	0.5	4	4

x	p	Initial n	Final n
0	0.1	0	5.33
0	0.5	2	3
1	0.5	4	4

Overview



Expectation
Sampler
Data
Learner
Program
Estimation
Learning

Learning

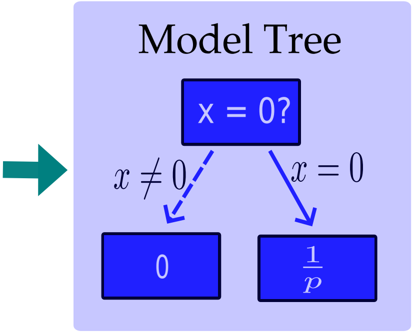
- Feed the dataset to the learner

x	p	Initial n	Final n
0	0.1	0	5.33
0	0.5	2	3
1	0.5	4	4

Learning

- Feed the dataset to the learner
- Learner learns a Model Tree

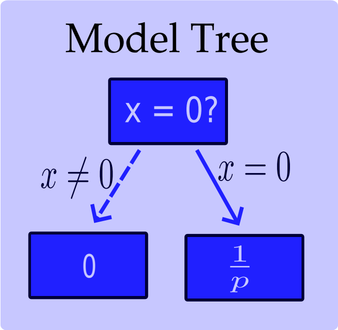
x	p	Initial n	Final n
0	0.1	0	5.33
0	0.5	2	3
1	0.5	4	4



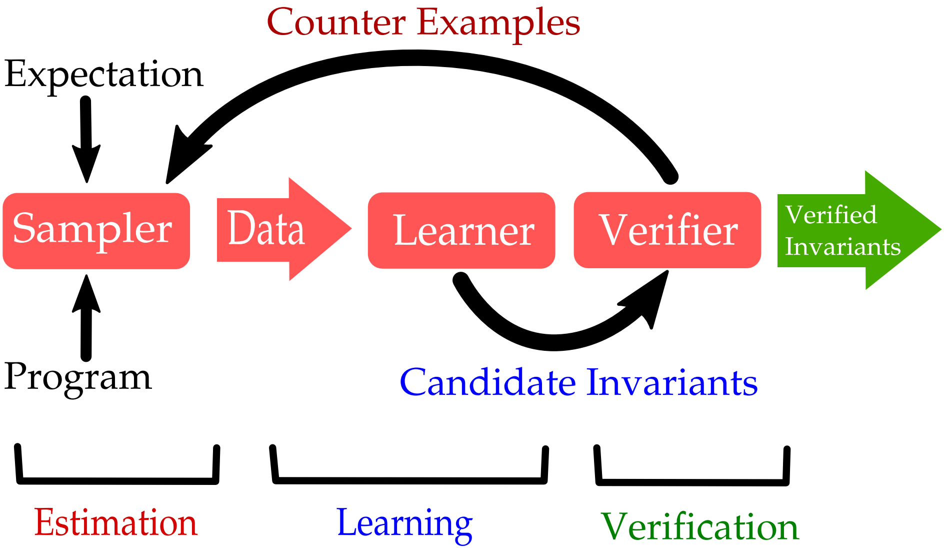
Learning

- Feed the dataset to the learner
- Learner learns a Model Tree
- Leaves encode invariant expression

x	p	Initial n	Final n
0	0.1	0	5.33
0	0.5	2	3
1	0.5	4	4



$$\mathcal{I} = n + [x == 0] \cdot \frac{1}{p}$$

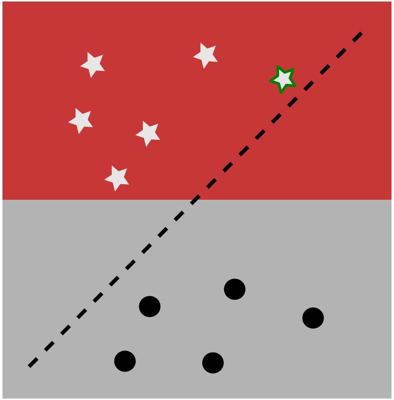

Counter Examples
Expectation
Sampler
Data
Learner
Verifier
Verified Invariants
Program
Candidate Invariants
Estimation
Learning
Verification

Verification

- Verify if synthesized invariant satisfies boundary and invariance conditions

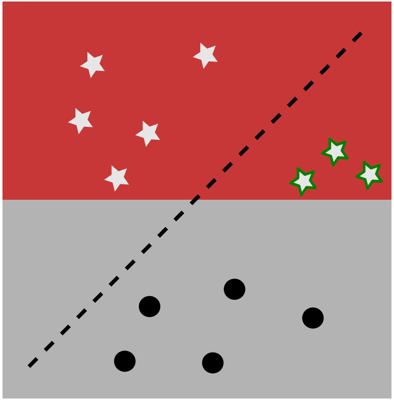
Verification

- Verify if synthesized invariant satisfies boundary and invariance conditions
- Solve an optimization to generate worst counter examples



Verification

- Verify if synthesized invariant satisfies boundary and invariance conditions
- Solve an optimization to generate worst counter examples
- Generate multiple counter examples



Experiments

- We implemented our apprach in a tool called EXIST (in Python)

Experiments

- We implemented our apprach in a tool called EXIST (in Python)
- Evaluated on 18 benchmarks collected from prior works

Experiments

- We implemented our apprach in a tool called EXIST (in Python)
- Evaluated on 18 benchmarks collected from prior works
- Successfully generates verified invariants for 14 benchmarks (taking between 1 to 237 seconds)

Experiments

- We implemented our apprach in a tool called EXIST (in Python)
- Evaluated on 18 benchmarks collected from prior works
- Successfully generates verified invariants for 14 benchmarks (taking between 1 to 237 seconds)
- Sampling phase dominates the total time

Experiments : Invariants Synthesized

<table>
<tr><th>Program</th><th>Invariant</th></tr>
<tr><td><pre>int z, bool flip, float p1
while (flip == 0):
  d = bernoulli_dist(p1)
  if d:
    flip = 1
  else:
    z = z + 1</pre></td><td>$z + [flip == 0] \cdot (1 - p_1)/p_1$</td></tr>
<tr><td><pre>int x , y , z , float p
while 0 < x and x < y :
d = bernoulli_dist(p1)
if d :
  x = x + 1
else :
  x = x - 1
  z = z + 1
rounds += 1</pre></td><td>$z + [x > 0] \cdot ([y > x] \cdot x \cdot (y - x))$</td></tr>
</table>

Conclusion

- We provided a general algorithm, EXIST (EXpectation Invariant SynThesis), for learning invariants for probabilistic programs

Kindly acknowledge the following sources for images used in the presentation: https://favpng.com, https://www.pngwing.com, https://www.pngkey.com, https://www.nicepng.com

Conclusion

- We provided a general algorithm, EXIST (EXpectation Invariant SynThesis), for learning invariants for probabilistic programs
  - Exact Invariants
  - Sub Invariants

Kindly acknowledge the following sources for images used in the presentation: https://favpng.com, https://www.pngwing.com, https://www.pngkey.com, https://www.nicepng.com

Conclusion

- We provided a general algorithm, EXIST (EXpectation Invariant SynThesis), for learning invariants for probabilistic programs
  - Exact Invariants
  - Sub Invariants
- We evaluated our implementation of EXIST on a diverse set of benchmarks

---

Kindly acknowledge the following sources for images used in the presentation: https://favpng.com, https://www.pngwing.com, https://www.pngkey.com, https://www.nicepng.com

Conclusion

- We provided a general algorithm, EXIST (EXpectation Invariant SynThesis), for learning invariants for probabilistic programs
  - Exact Invariants
  - Sub Invariants
- We evaluated our implementation of EXIST on a diverse set of benchmarks

https://github.com/JialuJialu/Exist

Kindly acknowledge the following sources for images used in the presentation: https://favpng.com, https://www.pngwing.com, https://www.pngkey.com, https://www.nicepng.com