Interval Counterexamples for Loop Invariant Learning

Rongchen Xu[1], Fei He[1], Bow-Yaw Wang[2]

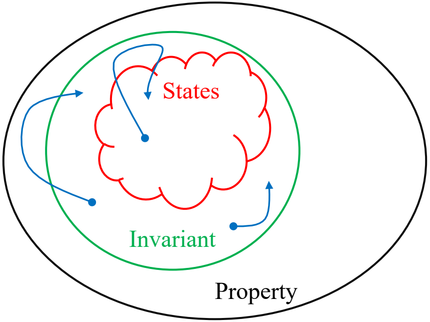
[1]School of Software, Tsinghua University.
[2]Academia Sinica.

November 8, 2020

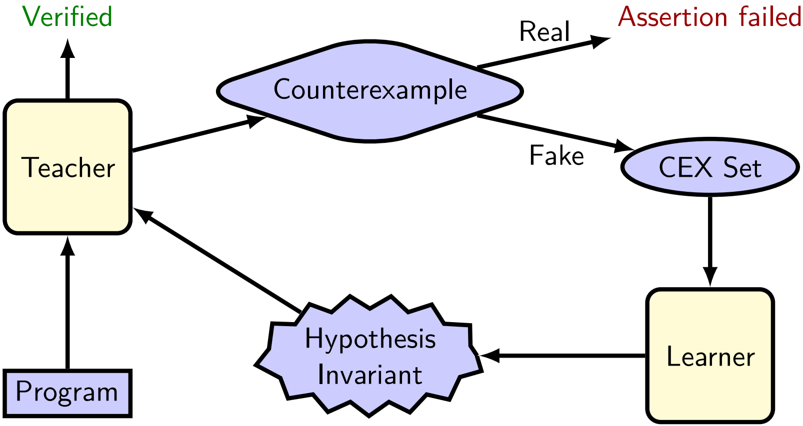
1. Background
2. Motivation
3. Approach
4. Evaluation
5. Conclusion

Loop invariants

- The loop invariant is a property that is always maintained when the program enters, iterates, and exits the loop.

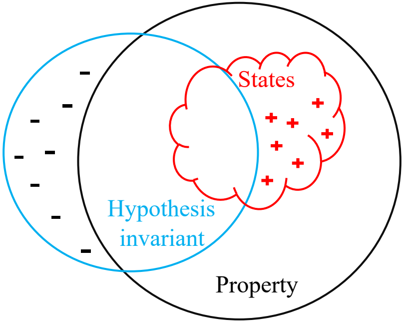


Black-box loop invariant inference



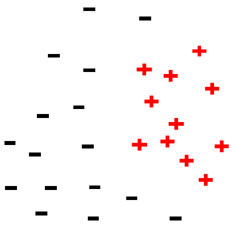
Positive and negative counterexamples

- Positive counterexample weakens the hypothesis invariant.
- Negative counterexample strengthens the hypothesis invariant.



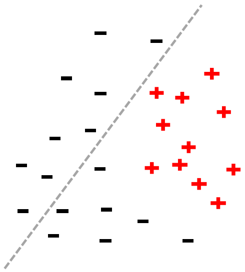
Decision tree learner

- We use decision tree to split all positive/negative examples and use the predicate of the positive as hypothesis invariant.



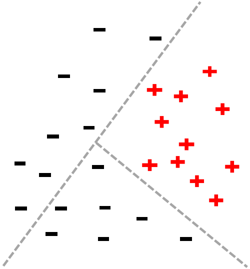
Decision tree learner

- We use decision tree to split all positive/negative examples and use the predicate of the positive as hypothesis invariant.



Decision tree learner

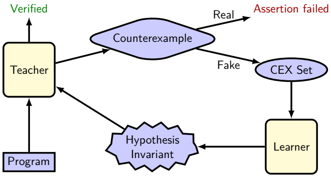
- We use decision tree to split all positive/negative examples and use the predicate of the positive as hypothesis invariant.



1. Background
2. **Motivation**
3. Approach
4. Evaluation
5. Conclusion

Motivation

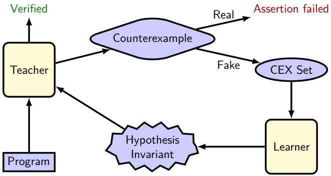
```
int x, y;
assume(x <= 2);
assume(y >= 0 && y <= 1);
while(*){
  if (x > 0) {
    x := x - 1;
    y := y + 1;
  }
}
assert(y >= 0 && y <= 3);
```



- Invariant hypothesis: *True*.

Motivation

```
int x, y;
assume(x <= 2);
assume(y >= 0 && y <= 1);
while(*){
  if (x > 0) {
    x := x - 1;
    y := y + 1;
  }
}
assert(y >= 0 && y <= 3);
```

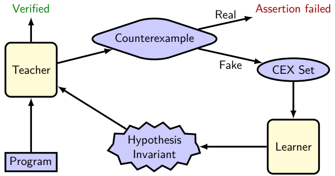


- Invariant hypothesis: *True*.
- Negative cex provided by teacher to strengthen the invariant:

$$\langle x = 0, y = 9 \rangle$$

Motivation

```
int x, y;
assume(x <= 2);
assume(y >= 0 && y <= 1);
while(*){
  if (x > 0) {
    x := x - 1;
    y := y + 1;
  }
}
assert(y >= 0 && y <= 3);
```



- Invariant hypothesis: *True*.
- Negative cex provided by teacher to strengthen the invariant:

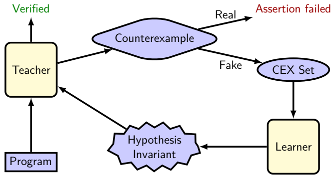
$$\langle x = 0, y = 9 \rangle$$

- Current cex set:

x	0
y	9
Label	Neg

Motivation

```
int x, y;
assume(x <= 2);
assume(y >= 0 && y <= 1);
while(*){
  if (x > 0) {
    x := x - 1;
    y := y + 1;
  }
}
assert(y >= 0 && y <= 3);
```



- Invariant hypothesis: *True*.
- Negative cex provided by teacher to strengthen the invariant:

$$\langle x = 0, y = 9 \rangle$$

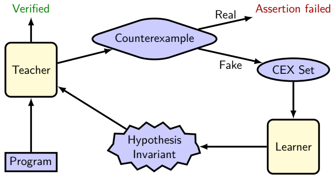
- Current cex set:

x	0
y	9
Label	Neg

- Modified hypothesis: *False*.

Motivation

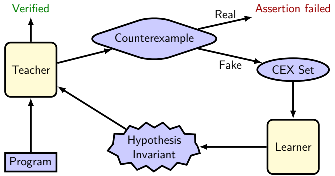
```
int x, y;
assume(x <= 2);
assume(y >= 0 && y <= 1);
while(*){
  if (x > 0) {
    x := x - 1;
    y := y + 1;
  }
}
assert(y >= 0 && y <= 3);
```



- Invariant hypothesis: *False*.

Motivation

```
int x, y;
assume(x <= 2);
assume(y >= 0 && y <= 1);
while(*){
  if (x > 0) {
    x := x - 1;
    y := y + 1;
  }
}
assert(y >= 0 && y <= 3);
```

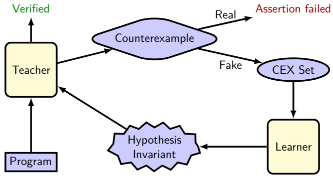


- Invariant hypothesis: *False*.
- Positive cex provided by teacher to weaken the invariant:

$$\langle x = 2, y = 1 \rangle$$

Motivation

```
int x, y;
assume(x <= 2);
assume(y >= 0 && y <= 1);
while(*){
  if (x > 0) {
    x := x - 1;
    y := y + 1;
  }
}
assert(y >= 0 && y <= 3);
```



- Invariant hypothesis: *False*.
- Positive cex provided by teacher to weaken the invariant:

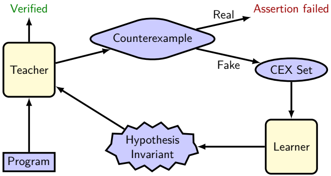
$$\langle x = 2, y = 1 \rangle$$

- Current cex set:

x	0	2
y	9	1
Label	Neg	Pos

Motivation

```
int x, y;
assume(x <= 2);
assume(y >= 0 && y <= 1);
while(*){
  if (x > 0) {
    x := x - 1;
    y := y + 1;
  }
}
assert(y >= 0 && y <= 3);
```



- Invariant hypothesis: *False*.
- Positive cex provided by teacher to weaken the invariant:

$$\langle x = 2, y = 1 \rangle$$

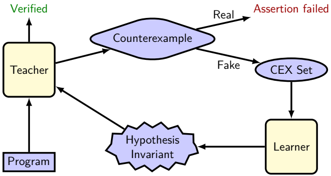
- Current cex set:

x	0	2
y	9	1
Label	Neg	Pos

- Modified hypothesis: $y \leq 8$.

Motivation

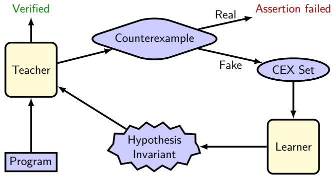
```
int x, y;
assume(x <= 2);
assume(y >= 0 && y <= 1);
while(*){
  if (x > 0) {
    x := x - 1;
    y := y + 1;
  }
}
assert(y >= 0 && y <= 3);
```



- Invariant hypothesis: $y \leq 8$.

Motivation

```
int x, y;
assume(x <= 2);
assume(y >= 0 && y <= 1);
while(*){
  if (x > 0) {
    x := x - 1;
    y := y + 1;
  }
}
assert(y >= 0 && y <= 3);
```

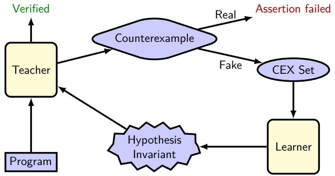


- Invariant hypothesis: $y \leq 8$.
- Negative cex provided by teacher to strengthen the invariant:

$$\langle x = 0, y = 8 \rangle$$

Motivation

```
int x, y;
assume(x <= 2);
assume(y >= 0 && y <= 1);
while(*){
  if (x > 0) {
    x := x - 1;
    y := y + 1;
  }
}
assert(y >= 0 && y <= 3);
```



- Invariant hypothesis: $y \leq 8$.
- Negative cex provided by teacher to strengthen the invariant:

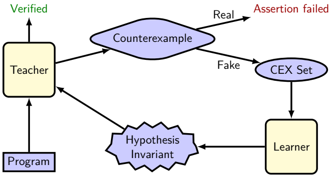
$$\langle x = 0, y = 8 \rangle$$

- Current cex set:

x	0	2	0
y	9	1	8
Label	Neg	Pos	Neg

Motivation

```
int x, y;
assume(x <= 2);
assume(y >= 0 && y <= 1);
while(*){
  if (x > 0) {
    x := x - 1;
    y := y + 1;
  }
}
assert(y >= 0 && y <= 3);
```



- Invariant hypothesis: $y \leq 8$.
- Negative cex provided by teacher to strengthen the invariant:

$$\langle x = 0, y = 8 \rangle$$

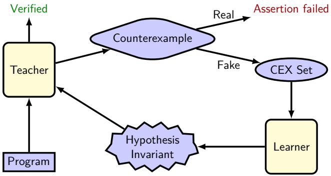
- Current cex set:

x	0	2	0
y	9	1	8
Label	Neg	Pos	Neg

- Modified hypothesis: $y \leq 7$.

Motivation

```
int x, y;
assume(x <= 2);
assume(y >= 0 && y <= 1);
while(*){
  if (x > 0) {
    x := x - 1;
    y := y + 1;
  }
}
assert(y >= 0 && y <= 3);
```



- Invariant hypothesis: $y \leq 7$
- Negative cex provided by teacher to strengthen the invariant:

$$\langle x = 0, y = 7 \rangle$$

- Current cex set:

x	0	2	0	0
y	9	1	8	7
Label	Neg	Pos	Neg	Neg

- Modified hypothesis: $y \leq 6$

Potential improvement

x	0	2	0	0	$\cdots$	0
y	9	1	8	7	$\cdots$	4
Label	Neg	Pos	Neg	Neg	$\cdots$	Neg

- A original counterexample is an *information-limited* single program state.

Potential improvement

x	0	2	0	0	$\cdots$	0
y	9	1	8	7	$\cdots$	4
Label	Neg	Pos	Neg	Neg	$\cdots$	Neg

- A original counterexample is an *information-limited* single program state.
- The generalized counterexamples we call them **interval counterexamples** look like:

$$\langle x \in (-\infty, \infty), y \in [4, \infty) \rangle$$

is a **set** of program states but not a single.

Core issues

- How to get a generalized interval counterexample?
- How to learn an invariant using interval counterexamples?

Generalization

- Suppose current hypothesis invariant by Learner: *True*.

```
int x, y;
assume(x <= 2);
assume(y >= 0 && y <= 1);
while(*){ // Inv: True
  if (x > 0) {
    x := x - 1;
    y := y + 1;
  }
}
assert(y >= 0 && y <= 3);
```

Generalization

- Suppose current hypothesis invariant by Learner: *True*.
- The verification condition to prove the assertion via the loop invariant: $True \rightarrow y \geq 0 \wedge y \leq 3$. (Simplify to $y \geq 0 \wedge y \leq 3$)

```
int x, y;
assume(x <= 2);
assume(y >= 0 && y <= 1);
while(*){ // Inv: True
  if (x > 0) {
    x := x - 1;
    y := y + 1;
  }
}
assert(y >= 0 && y <= 3);
```

Generalization by variable elimination

- Suppose the negative counterexample provided by Teacher:

$$x = 0 \wedge y = 9$$

Generalization by variable elimination

- Suppose the negative counterexample provided by Teacher:

$$x = 0 \land y = 9$$

- Consider the constraint:

$$\underbrace{y \geq 0 \land y \leq 3}_{\text{(VC)}} \land x = 0 \land y = 9$$

Generalization by variable elimination

- Suppose the negative counterexample provided by Teacher:

$$x = 0 \land y = 9$$

- Consider the constraint:

$$\underbrace{y \geq 0 \land y \leq 3}_{\text{(VC)}} \land x = 0 \land y = 9$$

- The UNSAT core is $\{y = 9\}$.

Generalization by variable elimination

- Suppose the negative counterexample provided by Teacher:

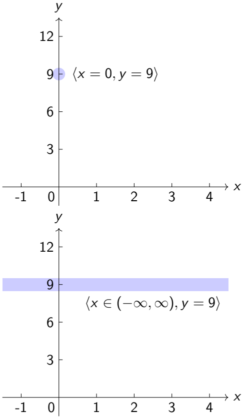
$$x = 0 \wedge y = 9$$

- Consider the constraint:

$$\underbrace{y \geq 0 \wedge y \leq 3}_{\text{(VC)}} \wedge x = 0 \wedge y = 9$$

- The UNSAT core is $\{y = 9\}$.
- We get a generalized counterexample:

$$\langle x \in (-\infty, \infty), y = 9 \rangle$$

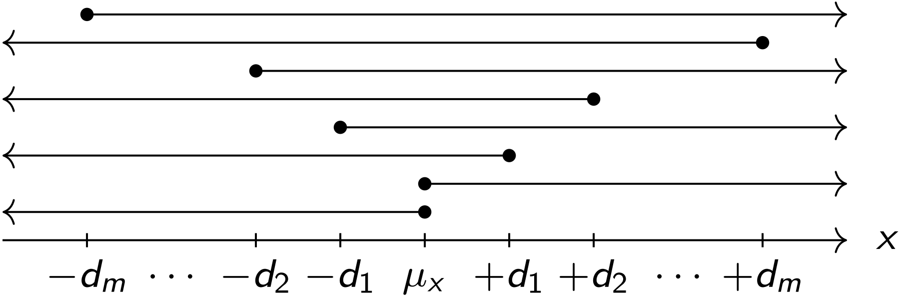


Generalization by boundary constraints

- Variable elimination: Rough and not widely general.

Generalization by boundary constraints

- Variable elimination: Rough and not widely general.
- Intuition: Find the generalization boundaries among some pre-defined distances.



Generalization by boundary constraints

- Consider distances (0,5) and new constraints to replace $x = 0$ and $y = 9$:

$$\underset{(VC)}{y \geq 0 \wedge y \leq 3} \wedge \bigwedge \begin{Bmatrix} x \geq -5 \\ x \geq 0 \\ x \leq 0 \\ x \leq 5 \end{Bmatrix} \wedge \bigwedge \begin{Bmatrix} y \geq 4 \\ y \geq 9 \\ y \leq 9 \\ y \leq 14 \end{Bmatrix}$$

Generalization by boundary constraints

- Consider distances (0,5) and new constraints to replace $x = 0$ and $y = 9$:

$$\underset{\text{(VC)}}{y \geq 0 \wedge y \leq 3} \wedge \bigwedge \left\{ \begin{matrix} x \geq -5 \\ x \geq 0 \\ x \leq 0 \\ x \leq 5 \end{matrix} \right\} \wedge \bigwedge \left\{ \begin{matrix} y \geq 4 \\ y \geq 9 \\ y \leq 9 \\ y \leq 14 \end{matrix} \right\}$$

- The UNSAT core may be: $y \geq 4$.

Generalization by boundary constraints

- Consider distances (0,5) and new constraints to replace $x = 0$ and $y = 9$:

$$\begin{array}{c} y \geq 0 \wedge y \leq 3 \\ \text{(VC)} \end{array} \wedge \bigwedge \left\{ \begin{array}{c} x \geq -5 \\ x \geq 0 \\ x \leq 0 \\ x \leq 5 \end{array} \right\} \wedge \bigwedge \left\{ \begin{array}{c} y \geq 4 \\ y \geq 9 \\ y \leq 9 \\ y \leq 14 \end{array} \right\}$$

- The UNSAT core may be: $y \geq 4$.
- More general counterexample:

$$\langle x \in (-\infty, \infty), y \in [4, \infty) \rangle$$

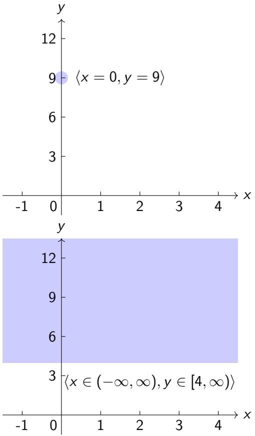
Generalization by boundary constraints

- Consider distances (0,5) and new constraints to replace $x = 0$ and $y = 9$:

$$\underset{\text{(VC)}}{y \geq 0 \wedge y \leq 3} \wedge \bigwedge \begin{Bmatrix} x \geq -5 \\ x \geq 0 \\ x \leq 0 \\ x \leq 5 \end{Bmatrix} \wedge \bigwedge \begin{Bmatrix} y \geq 4 \\ y \geq 9 \\ y \leq 9 \\ y \leq 14 \end{Bmatrix}$$

- The UNSAT core may be: $y \geq 4$.
- More general counterexample:

$$\langle x \in (-\infty, \infty), y \in [4, \infty) \rangle$$

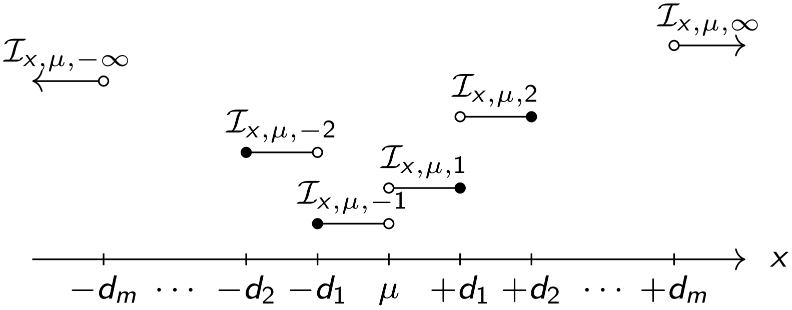


Generalization by interval digging

- UNSAT core may not the best range. ($y \geq 4$ or $y \geq 9$?)

Generalization by interval digging

- UNSAT core may not the best range. ($y \geq 4$ or $y \geq 9$?)
- Intuition: Dig out all intervals which contains program states can **not** be generalized.



Generalization by interval digging

- Rearrange the constraints:

$$\underset{\text{(VC)}}{y \geq 0 \land y \leq 3} \land \bigwedge \left\{ \begin{array}{c} \neg(x > 5) \\ \neg(x > 0 \land x \leq 5) \\ \neg(x < 0 \land x \geq -5) \\ \neg(x < -5) \end{array} \right\} \land \bigwedge \left\{ \begin{array}{c} \neg(y > 14) \\ \neg(y > 9 \land y \leq 14) \\ \neg(y < 9 \land y \geq 4) \\ \neg(y < 4) \end{array} \right\}$$

Generalization by interval digging

- Rearrange the constraints:

$$\underset{\text{(VC)}}{y \geq 0 \land y \leq 3} \land \bigwedge \left\{ \begin{array}{c} \neg(x > 5) \\ \neg(x > 0 \land x \leq 5) \\ \neg(x < 0 \land x \geq -5) \\ \neg(x < -5) \end{array} \right\} \land \bigwedge \left\{ \begin{array}{c} \neg(y > 14) \\ \neg(y > 9 \land y \leq 14) \\ \neg(y < 9 \land y \geq 4) \\ \neg(y < 4) \end{array} \right\}$$

- Now the UNSAT core is **unique**: $\neg(y < 4)$.

Generalization by interval digging

- Rearrange the constraints:

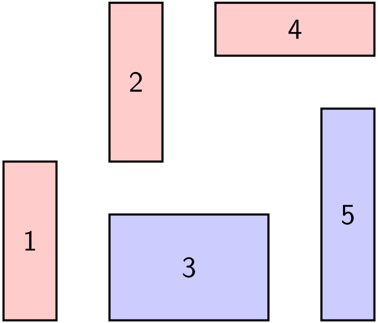
$$\underset{\text{(VC)}}{y \geq 0 \wedge y \leq 3} \wedge \bigwedge \left\{ \begin{array}{c} \neg(x > 5) \\ \neg(x > 0 \wedge x \leq 5) \\ \neg(x < 0 \wedge x \geq -5) \\ \neg(x < -5) \end{array} \right\} \wedge \bigwedge \left\{ \begin{array}{c} \neg(y > 14) \\ \neg(y > 9 \wedge y \leq 14) \\ \neg(y < 9 \wedge y \geq 4) \\ \neg(y < 4) \end{array} \right\}$$

- Now the UNSAT core is **unique**: $\neg(y < 4)$.
- The counterexample is $\langle x \in (-\infty, \infty), y \in [4, \infty) \rangle$.

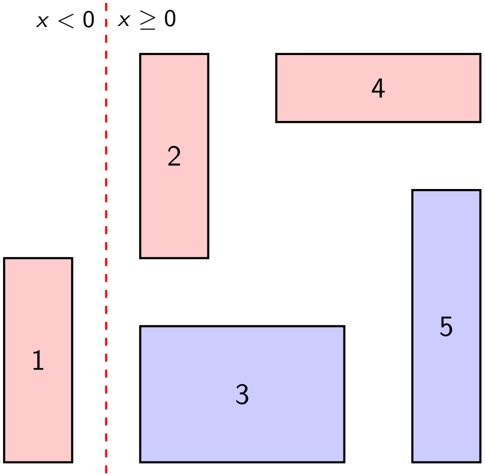
How to split interval examples?

- Avoid crossing examples first
- But also allow examples crossing

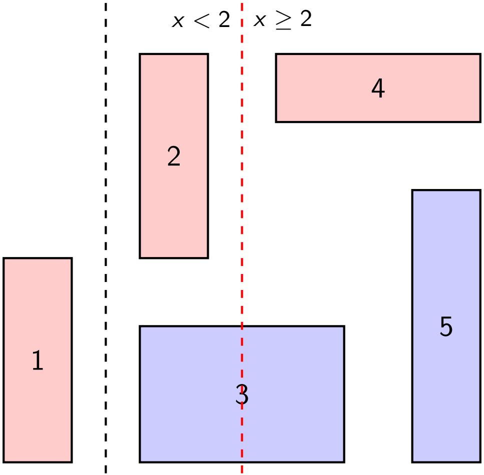
	x	y	Label
1	[-2,-1]	[-2, 1]	+
2	[0,1]	[1,4]	+
3	[0,3]	[-2,0]	-
4	[2,5]	[3,4]	+
5	[4,5]	[-2,2]	-



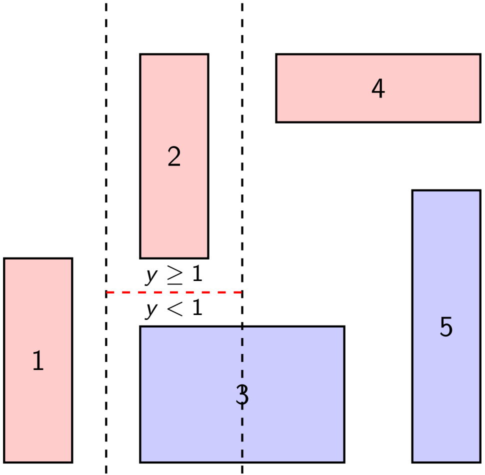
Interval decision tree



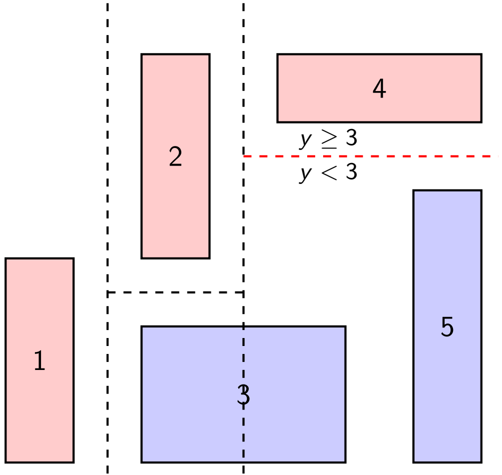
Interval decision tree



Interval decision tree

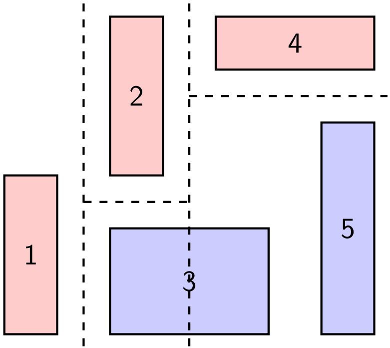


Interval decision tree



Interval decision tree

	x	y	Label
1	[-2,-1]	[-2, 1]	+
2	[0,1]	[1,4]	+
3	[0,3]	[-2,0]	-
4	[2,5]	[3,4]	+
5	[4,5]	[-2,2]	-



- Predicted invariant:

$$x < 0 \vee (x \geq 0 \wedge x < 2 \wedge y \geq 1) \vee (x \geq 0 \wedge x \geq 2 \wedge y \geq 3)$$

Experiment on generalization methods

- We implement prototype[1] of three generalization methods and also an interval counterexample supported decision tree.

[1] Available at: https://doi.org/10.5281/zenodo.3898483.

[2] Pranav Garg et al. Learning Invariants Using Decision Trees and Implication Counterexamples. POPL 2016

[3] Available at: https://github.com/sosy-lab/sv-benchmarks/

Experiment on generalization methods

- We implement prototype[1] of three generalization methods and also an interval counterexample supported decision tree.
- Total 94 benchmarks are from ICE-DT[2] and SV-COMP 2019[3].

[1] Available at: https://doi.org/10.5281/zenodo.3898483.

[2] Pranav Garg et al. Learning Invariants Using Decision Trees and Implication Counterexamples. POPL 2016

[3] Available at: https://github.com/sosy-lab/sv-benchmarks/

Experiment on generalization methods

- We implement prototype[1] of three generalization methods and also an interval counterexample supported decision tree.
- Total 94 benchmarks are from ICE-DT[2] and SV-COMP 2019[3].

	Solved	R	$T(s)$	$T_C(s)$	$T_L(s)$
ICE-DT	90	3007	135.79	20.95	97.51
Elim	91	1916(36.3%)	87.47(35.6%)	17.48(16.6%)	52.46(46.2%)
Boundary	91	1717(42.9%)	72.27(46.8%)	15.15(27.7%)	42.41(56.5%)
Digging	91	1712(43.1%)	73.41(45.9%)	15.87(24.2%)	42.55(56.4%)

Table 1: Summary results on interval generalization methods

[1] Available at: https://doi.org/10.5281/zenodo.3898483.

[2] Pranav Garg et al. Learning Invariants Using Decision Trees and Implication Counterexamples. POPL 2016

[3] Available at: https://github.com/sosy-lab/sv-benchmarks/

Experiment on learning methods

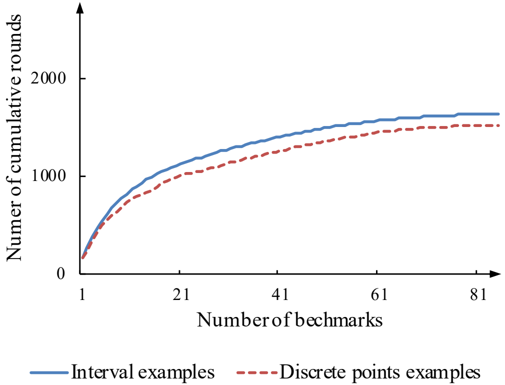
- How much efficiency improvement comes from the interval decision tree?

Experiment on learning methods

- How much efficiency improvement comes from the interval decision tree?
- Replace interval examples with representative state examples.

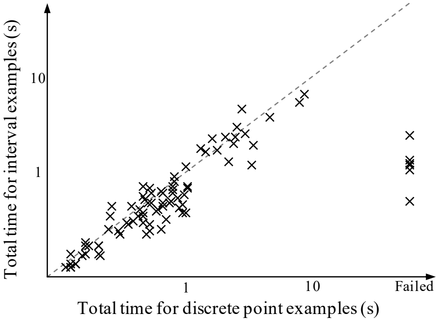
Experiment on learning methods

- How much efficiency improvement comes from the interval decision tree?
- Replace interval examples with representative state examples.

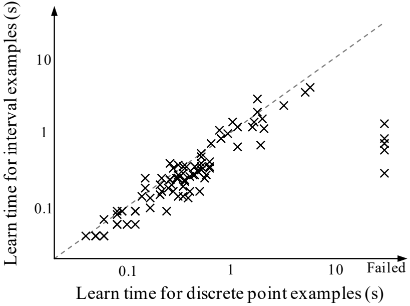


Experiment on learning methods

- Iteration rounds are similar, but time?



(a) Total time comparison



(b) Learn time comparison

1. Background
2. Motivation
3. Approach
4. Evaluation
5. Conclusion

Conclusion

- The introduction of interval counterexamples which represent a set of counterexamples from constraint solvers.

Conclusion

- The introduction of interval counterexamples which represent a set of counterexamples from constraint solvers.
- Three different generalization techniques to compute interval counterexamples.

Conclusion

- The introduction of interval counterexamples which represent a set of counterexamples from constraint solvers.
- Three different generalization techniques to compute interval counterexamples.
- An improved decision tree algorithm to adapt interval counterexamples.

Conclusion

- The introduction of interval counterexamples which represent a set of counterexamples from constraint solvers.
- Three different generalization techniques to compute interval counterexamples.
- An improved decision tree algorithm to adapt interval counterexamples.
- A prototype and over 40% improvement on learning rounds and verification time compared with existing methods.

Thanks!