Combining data-driven and symbolic reasoning for Invariant Synthesis in SMT (Work in Progress)

Haniel Barbosa
Andrew Reynolds
Cesare Tinelli
The University of Iowa

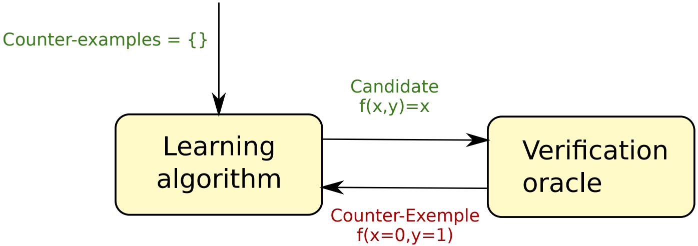
Clark Barrett
Stanford

MVD 2018

2018–09–29, Iowa City, IA, USA

SyGuS Solving

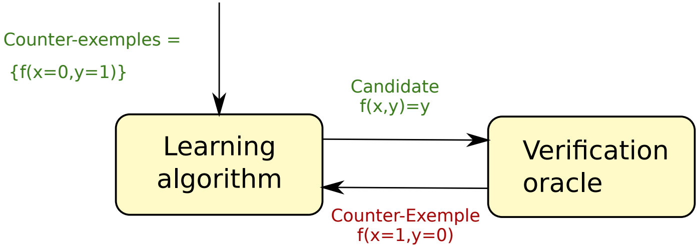
- Most common technique for SyGuS solving
- Specification: $x \leq f(x, y) \land y \leq f(x, y)$
- Expression search space:
  - Combinations of $x, y, 0, 1, \leq, +, \text{if-then-else}$



CEGIS

[Solar-Lezama et al. 2006]

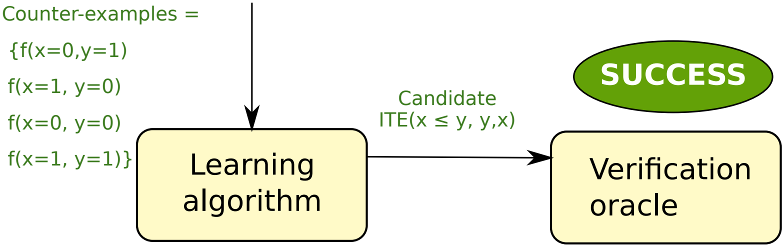
- ▷ Most common technique for SyGuS solving
- ▷ Specification: $x \leq f(x, y) \wedge y \leq f(x, y)$
- ▷ Expression search space:
  - ▶ Combinations of $x, y, 0, 1, \leq, +, \text{if-then-else}$



CEGIS

[Solar-Lezama et al. 2006]

- Most common technique for SyGuS solving
- Specification: $x \leq f(x,y) \wedge y \leq f(x,y)$
- Expression search space:
  - Combinations of $x, y, 0, 1, \leq, +, \text{if-then-else}$



Scalability issues

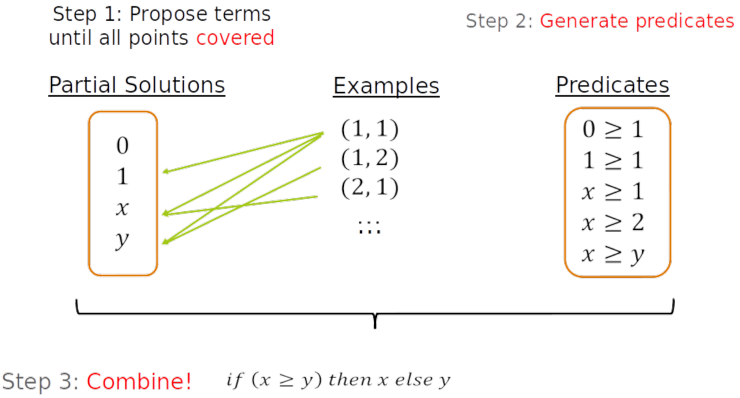
For this bit-vector grammar, enumerating

- Terms of size $= 1$: .05 seconds
- Terms of size $= 2$: .6 seconds
- Terms of size $= 3$: 48 seconds
- Terms of size $= 4$: 5.8 hours
- Terms of size $= 5$: ??? (100+ days)

```
(synth-fun f ((s (BitVec 4))
              (t (BitVec 4)))
 (BitVec 4) (
 (Start (BitVec 4) (
  s t #x0
  (bvneg  Start)
  (bvnot  Start)
  (bvadd  Start Start)
  (bvmul  Start Start)
  (bvand  Start Start)
  (bvlshr Start Start)
  (bvor   Start Start)
  (bvshl  Start Start)))))
```

Divide-and-conquer [Alur et al. 2017]

- Generate partial solutions correct on subset of input
- Combine using conditionals

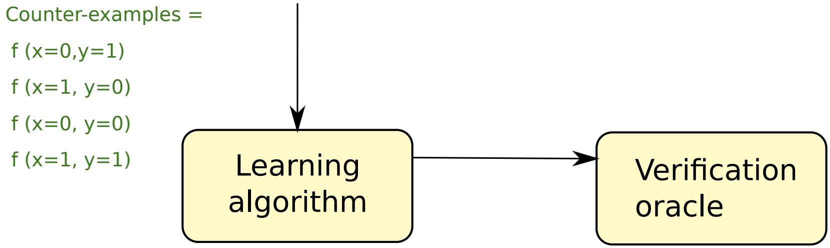


Only applicable for plainly separable specifications

A new framework for SyGuS solving

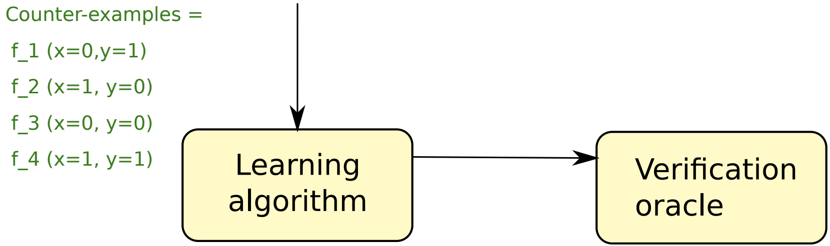
CegisUnif : combining CEGIS with unification

- Not limited to plainly separable specifications
- *Data-driven*: refinement lemmas generate data points
- *Divide-and-conquer*: each point yields a new function to synthesize
  - Terms assigned to functions must satisfy refinement lemmas
  - SMT solving provides term candidates through constraint solving



CegisUnif : combining CEGIS with unification

- Not limited to plainly separable specifications
- *Data-driven*: refinement lemmas generate data points
- *Divide-and-conquer*: each point yields a new function to synthesize
  - Terms assigned to functions must satisfy refinement lemmas
  - SMT solving provides term candidates through constraint solving



Feature synthesis

- *Symbolic approach* : derive minimal number of features that separate conflicting points (i.e. those that cannot be assigned the same term)
  - Optimal fairness criteria?
    - Currently: consider terms of size up to $\log_2(\#features)$

- *Heuristic approach* : accumulate "feature pool" and chose separating features based on information gain heuristic for decision tree learning
  - Select features that maximize information gain

Solving Invariant synthesis with CegisUnif

Invariant Synthesis

```
Add(Int x, y) {
  z := x; i := 0;
  assume(y > 0);
  while (i < y) {
    z := z + 1;
    i := i + 1;
  }
  return z;
}
```

Post-condition:
$\forall x, y: \ z = x + y$

Result is the sum of the inputs

Invariant Synthesis

```
Add(Int x, y) {
  z := x; i := 0;
  assume(y > 0);
  while (i < y) {
    z := z + 1;
    i := i + 1;
  }
  return z;
}
```

Invariant?

Post-condition:
$\forall x, y: \; z = x + y$

Result is the sum of the inputs

Verification:

$$
\begin{array}{lcl}
z = x \wedge i = 0 \wedge y > 0 & \rightarrow & Inv(x, y, z, i) \\
Inv(x, y, z, i) \wedge i < y \wedge z' = z + 1 \wedge i' = i + 1 & \rightarrow & Inv(x, y, z', i') \\
Inv(x, y, z, i) \wedge i \geq y & \rightarrow & z = x + y
\end{array}
$$

Invariant Synthesis

```
Add(Int x, y) {
  z := x; i := 0;
  assume(y > 0);
  while (i < y) {
    z := z + 1;
    i := i + 1;
  }
  return z;
}
```

$$Inv(x, y, z, i)$$
$$z = x + i$$
$$z \leq x + y$$

Post-condition:
$\forall x, y : z = x + y$

Result is the sum of the inputs

Verification:

$$\begin{array}{lcl} z = x \wedge i = 0 \wedge y > 0 & \rightarrow & Inv(x, y, z, i) \\ Inv(x, y, z, i) \wedge i < y \wedge z' = z + 1 \wedge i' = i + 1 & \rightarrow & Inv(x, y, z', i') \\ Inv(x, y, z, i) \wedge i \geq y & \rightarrow & z = x + y \end{array}$$

Invariant Synthesis in SyGuS

- State-of-the-art: LoopInvGen [Padhi and Millstein 2017]: *data-driven* loop invariant inference with automatic feature synthesis
  - Precondition inference from sets of “good” and “bad” states
    - Feature synthesis for solving conflicts
  - PAC (*probably approximately correct*) algorithm for building candidate invariants

- “Bad” states are dependent on model of initial condition (no guaranteed convergence)

- No support for implication counterexamples

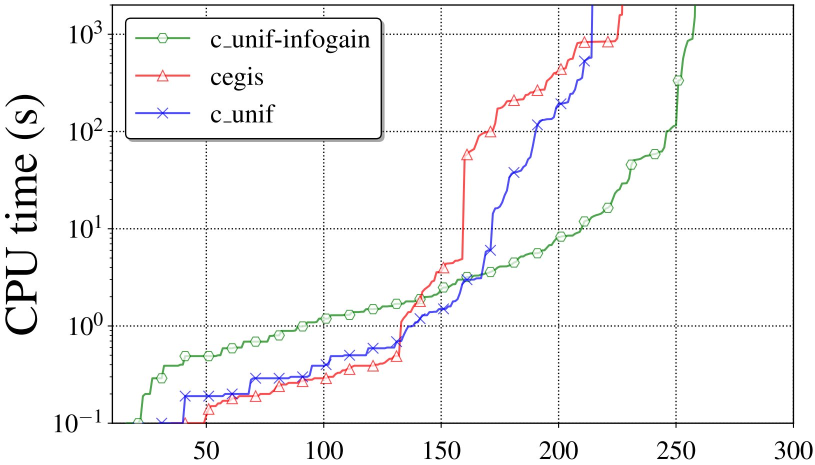
Invariant Synthesis with CegisUnif

- Refinement lemmas allows derivation of three kinds on data points:
  - "good points" (invariant must always hold)
  - "bad points" (invariant can never hold)
  - "implication points" (if invariant holds in first point it must hold in second)
- No need for restriction to one initial state
- Native support for implication counterexamples
- Straightforward usage of classic information gain heuristic to build candidate solutions with decision tree learning
  - SMT solver "resolves" implication counterexample points as "good" and "bad"
  - Out-of-the-box Shannon entropy

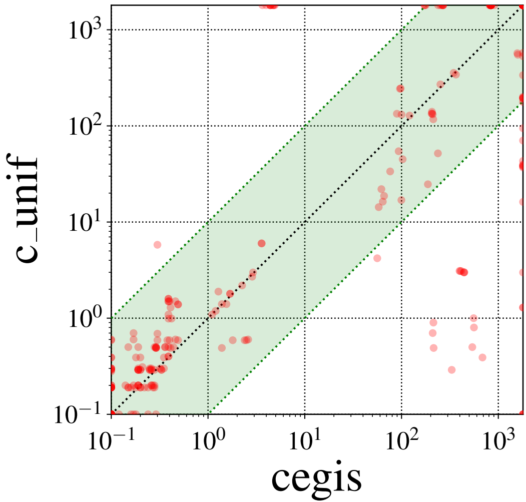
Preliminary results

Invariant generation for Lustre

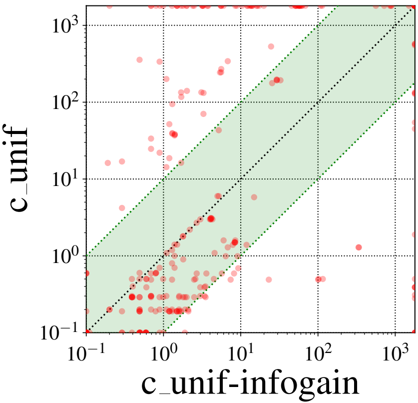
- Test suite with 487 invariant synthesis benchmarks generated by the Kind 2 model checker from Lustre models
- We evaluate three configurations of CVC4
  - **cegis** : regular CEGIS
  - **c_unif** : CegisUnif framework with symbolic solution building
  - **c_unif-infogain** : CegisUnif framework with solution building determined by information gain heuristic
- 1800s timeout

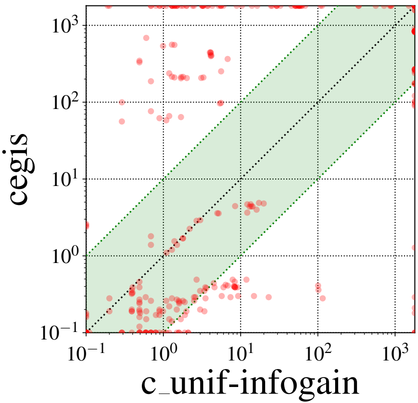


c_unif-infogain
cegis
c_unif
CPU time (s)
10^3
10^2
10^1
10^0
10^{-1}
50
100
150
200
250
300



▷ + 38 / - 13


c_unif
c_unif-infogain
10^{-1}
10^{0}
10^{1}
10^{2}
10^{3}

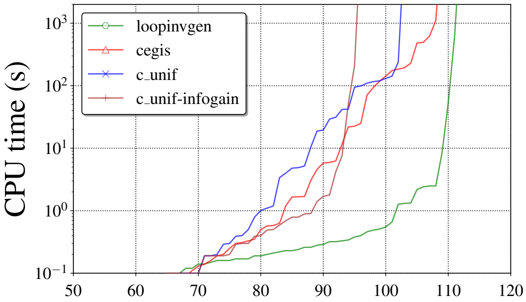

cegis
c_unif-infogain
10^{-1}
10^{0}
10^{1}
10^{2}
10^{3}

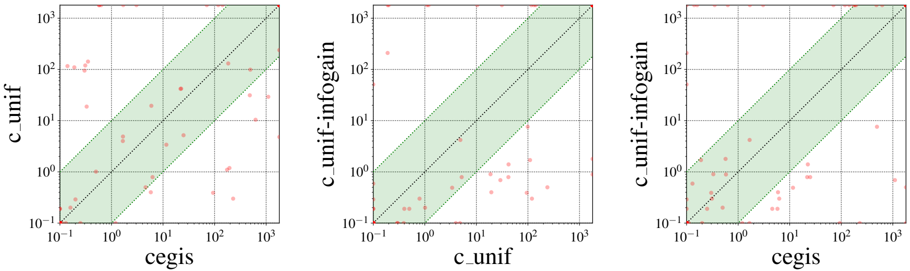
▷ + 63 / - 19

▷ + 73 / - 42

Invariants category from SyGuS-Comp 2018

- Test suite with 127 invariant synthesis benchmarks from numerous applications
- We evaluate three configurations of CVC4
  - **cegis** : regular CEGIS
  - **c_unif** : CegisUnif framework with symbolic solution building
  - **c_unif-infogain** : CegisUnif framework with solution building determined by information gain heuristic
- We also compare against LoopInvGen, the current winner of the invariants category in SyGuS-Comp
- 1800s timeout


loopinvgen
cegis
c_unif
c_unif-infogain
CPU time (s)
10^3
10^2
10^1
10^0
10^{-1}
50
60
70
80
90
100
110
120


c_unif
cegis
c_unif-infogain
c_unif
c_unif-infogain
cegis
10^{-1}
10^0
10^1
10^2
10^3

Future work

- Adapt ICE [Garg et al. 2016] information gain heuristics to our setting; derive new heuristics
- Extend heuristics to function synthesis [Alur et al. 2017]
- Use data to determine "relevant arguments"
  - $f_1(0, 0, 0, 1, 2, 1, 0) \diamond f_2(1, 0, 0, 5, 2, 1, 3)$
  - Reducing noise: make points as similar as possible

    $f_1'(1, 0, 0, 1, 2, 1, 0) \diamond f_2'(1, 0, 0, 5, 2, 1, 0)$
  - Only consider relevant arguments when synthesizing features
    - Can drastically reduce search space

References

Alur, Rajeev, Arjun Radhakrishna, and Abhishek Udupa (2017). "Scaling Enumerative Program Synthesis via Divide and Conquer". In: Tools and Algorithms for Construction and Analysis of Systems (TACAS). Ed. by Axel Legay and Tiziana Margaria. Vol. 10205. Lecture Notes in Computer Science, pp. 319–336.

Garg, Pranav et al. (2016). "Learning invariants using decision trees and implication counterexamples". In: Symposium on Principles of Programming Languages. Ed. by Rastislav Bodík and Rupak Majumdar. ACM, pp. 499–512.

Padhi, Saswat and Todd D. Millstein (2017). "Data-Driven Loop Invariant Inference with Automatic Feature Synthesis". In: CoRR abs/1707.02029. arXiv: 1707.02029.

Solar-Lezama, Armando et al. (2006). "Combinatorial sketching for finite programs". In: Architectural Support for Programming Languages and Operating Systems (ASPLOS). Ed. by John Paul Shen and Margaret Martonosi. ACM, pp. 404–415.