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

Haniel Barbosa
Andrew Reynolds
Cesare Tinelli

Clark Barrett

The University of Iowa

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 $x, y, 0, 1, \leq, +, \text{if-then-else}$
Most common technique for SyGuS solving

Specification: \( x \leq f(x, y) \land y \leq f(x, y) \)

Expression search space:
- Combinations of \( x, y, 0, 1, \leq, +, \text{if-then-else} \)

Counter-examples = \{f(x=0,y=1)\}

Candidate \( f(x,y)=y \)

Counter-Exemple \( f(x=1,y=0) \)
Most common technique for SyGuS solving

Specification: $x \leq f(x, y) \land y \leq f(x, y)$

Expression search space:
- Combinations of $x, y, 0, 1, \leq, +, \text{if-then-else}$

Counter-examples = 
- $f(x=0, y=1)$
- $f(x=1, y=0)$
- $f(x=0, y=0)$
- $f(x=1, y=1)$

Candidate: $\text{ITE}(x \leq y, y, x)$

SUCCESS
Scalability issues

For this bit-vector grammar, enumerating

- Terms of size = 1 : 0.05 seconds
- Terms of size = 2 : 0.6 seconds
- Terms of size = 3 : 48 seconds
- Terms of size = 4 : 5.8 hours
- Terms of size = 5 : ??? (100+ days)
Divide-and-conquer

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

**Step 1:** Propose terms until all points covered

**Step 2:** Generate predicates

**Partial Solutions**

- 0
- 1
- x
- y

**Examples**

- (1, 1)
- (1, 2)
- (2, 1)
- ...

**Predicates**

- \(0 \geq 1\)
- \(1 \geq 1\)
- \(x \geq 1\)
- \(x \geq 2\)
- \(x \geq y\)

**Step 3:** Combine! if \((x \geq y)\) then \(x\) else \(y\)

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

Counter-examples =
- \( f(x=0, y=1) \)
- \( f(x=1, y=0) \)
- \( f(x=0, y=0) \)
- \( f(x=1, y=1) \)
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

Counter-examples =
- $f_1(x=0, y=1)$
- $f_2(x=1, y=0)$
- $f_3(x=0, y=0)$
- $f_4(x=1, y=1)$
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 $\log_2(\#\text{features})$

- **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: Result is the sum of the inputs
∀x, y : z = x + y
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:

\[
\begin{align*}
  z & = x \land i = 0 \land y > 0 \\
  Inv(x, y, z, i) \land i < y \land z' = z + 1 \land i' = i + 1 & \rightarrow \quad Inv(x, y, z', i') \\
  Inv(x, y, z, i) \land i \geq y & \rightarrow \quad z = x + y
\end{align*}
\]
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: Result is the sum of the inputs

\forall x, y : z = x + y

Verification:

\begin{align*}
z &= x \land i = 0 \land y > 0 & \rightarrow & Inv(x, y, z, i) \\
Inv(x, y, z, i) \land i < y \land z' = z + 1 \land i' = i + 1 & \rightarrow & Inv(x, y, z', i') \\
Inv(x, y, z, i) \land i \geq y & \rightarrow & z = x + y
\end{align*}
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
Combining data-driven and symbolic reasoning for Invariant Synthesis (WIP)
Combining data-driven and symbolic reasoning for Invariant Synthesis (WIP)

\[ c_{\text{unif}} + 38 \]
Combining data-driven and symbolic reasoning for Invariant Synthes

\[ + 63 \text{ / } -19 \]

\[ + 73 \text{ / } -42 \]
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
Combining data-driven and symbolic reasoning for Invariant Synthesis (WIP)
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”
  - $f_1(0, 0, 0, 1, 2, 1, 0) \odot f_2(1, 0, 0, 5, 2, 1, 3)$

  - Reducing noise: make points as similar as possible
    - $f'_1(1, 0, 0, 1, 2, 1, 0) \odot f'_2(1, 0, 0, 5, 2, 1, 0)$

  - Only consider relevant arguments when synthesizing features
    - Can drastically reduce search space

