CVC4 1.5 for Sygus Comp 2015

• CVC4 is an SMT solver
  • Fourth generation of Cooperating Validity Checker (CVC, CVC Lite, CVC3, CVC4)
  • Supports many ground theories:
    • Linear arithmetic, bitvectors, UF, datatypes, arrays, sets, strings, ...
  • Supports quantified formulas
  • Two new approaches for refutation-based synthesis [CAV 15]
    1. Single-invocation properties
    2. Syntax-guided synthesis (SyGuS) problems

• Submission for Sygus Comp 2015 was joint work between:
  • EPFL: Andrew Reynolds, Viktor Kuncak
  • University of Iowa: Cesare Tinelli
  • NYU: Clark Barrett, Morgan Deters
  • Verimag: Tim King
Refutation-Based Synthesis

\[ \exists f. \forall xy. (f(x, y) \geq x \land f(x, y) \geq y \land (f(x, y) = x \lor f(x, y) = y)) \]

- Example: find a function \( f \) that computes max of two integers
Refutation-Based Synthesis

$$\exists f. \forall xy. \text{isMax}(f(x,y), x, y)$$
Refutation-Based Synthesis

\[ \exists f. \forall x y. \text{isMax}(f(x, y), x, y) \]

Find model for \( f \) that satisfies this property
Refutation-Based Synthesis

\[ \exists f. \forall xy. \text{isMax}(f(x,y),x,y) \]

Negate

\[ \forall f. \exists xy. \neg \text{isMax}(f(x,y),x,y) \]

*Instead*, show negated formula is *unsatisfiable*.
Refutation-Based Synthesis

\[ \exists f. \forall xy. \text{isMax}(f(x,y), x, y) \]

Negate

\[ \forall f. \exists xy. \neg \text{isMax}(f(x,y), x, y) \]

- Eliminate second-order quantification over \( f \) in two ways
Refutation-Based Synthesis

\[ \exists f. \forall xy. \text{isMax}(f(x, y), x, y) \]

Negate

\[ \forall f. \exists xy. \neg \text{isMax}(f(x, y), x, y) \]

If *single invocation*, replace \( f \) with (first-order) variable \( g \)

\[ \exists xy. \forall g. \neg \text{isMax}(g, x, y) \]

\( \Rightarrow g \) represents the return value of \( f \)
Refutation-Based Synthesis

∃f. ∀xy.isMax(f(x,y), x, y)

Negate

∀f. ∃xy.¬isMax(f(x,y), x, y)

If single invocation, replace f with (first-order) variable g

∃xy.∀g.¬isMax(g, x, y)

Otherwise, replace f with datatype d, and operator ev

D := zero | one | plus(D1, D2) | ...
∀d.∃xy.¬isMax(ev(d, x, y), x, y)
∀dxy.ev(d, x, y) =...

⇒ D models the domain of possible solutions for f
Refutation-Based Synthesis

\[ \exists f. \forall x y. \text{isMax}(f(x,y), x, y) \]

Negate

\[ \forall f. \exists x y. \neg \text{isMax}(f(x,y), x, y) \]

If single invocation, replace \( f \) with (first-order) variable \( g \)

\[ \exists x y. \forall g. \neg \text{isMax}(g, x, y) \]

Otherwise, replace \( f \) with datatype \( d \), and operator \( \text{ev} \)

\[ D := \text{zero} | \text{one} | \text{plus}(D1, D2) | \ldots \]

\[ \forall d. \exists x y. \neg \text{isMax}(\text{ev}(d, x, y), x, y) \]

\[ \forall d xy. \text{ev}(d, x, y) = \ldots \]

Single invocation approach  Syntax-guided approach
Solving Synthesis Conjectures in an SMT Solver

\[ \exists f. \forall x y. \text{isMax}(f(x, y), x, y) \]
Solving Synthesis Conjectures in an SMT Solver

\[ \exists f. \forall xy. \text{isMax}(f(x,y), x, y) \]

1. Negate, convert to first order

\[ \forall g. \neg \text{isMax}(g, a, b) \]

SAT Solver +

Dec Procedures

SMT Solver

Quantifiers

Module
Solving Synthesis Conjectures in an SMT Solver

1. Negate, convert to first order

\[ \exists f. \forall xy. \text{isMax}(f(x,y), x, y) \]

2. Add instances until “unsat”, via counterexample-guided quantifier instantiation

\[ \neg \text{isMax}(a, a, b), \neg \text{isMax}(b, a, b), \]

[Diagram]

- SAT Solver + Dec Procedures
- Quantifiers Module
- SMT Solver

unsat
Solving Synthesis Conjectures in an SMT Solver

1. Negate, convert to first order
   \[ \exists f. \forall xy. \text{isMax}(f(x,y), x, y) \]

2. Add instances until "unsat", via counterexample-guided quantifier instantiation

   - \( \neg \text{isMax}(a,a,b) \)
   - \( \neg \text{isMax}(b,a,b) \)

3. Extract solution for \( f \) from unsat core

   \[ f := \lambda xy. \text{ite} \left( \text{isMax}(x,x,y), x, y \right) \]

   \[ \neg \text{isMax}(a,a,b), \neg \text{isMax}(b,a,b) \models \bot \]
CVC4 in Sygus Comp 2015

• Entered all three tracks (General, LIA, INV)
  • For general/LIA track:
    • Most benchmarks are \textit{single invocation}
    • Solution reconstruction methods to match syntactic restrictions, if necessary
  • For INV track:
    • All benchmarks are \textit{not single invocation}
      • Due to form of benchmarks, for transition relations $T$:
        $$\exists \text{inv. } \forall x. (\text{inv}(x) \land T(x, x')) \Rightarrow \text{inv}(x')$$
      \Rightarrow \text{Resorts to syntax-guided approach}