#25: Sketch and Synquid

Sankha Narayan Guria

EECS 700: Introduction to Program Synthesis

Sketch: contributions

- Expressing structural and behavioral constraints as programs
  - the only primitive extension is an integer hole ??
  - why is it important to keep extensions minimal?
- Synthesis by translating to SAT
- CEGIS
  - became extremely popular!
- Handles imperative programs with loops
  - and proposes an encoding for those
- Can discover constants

Sketch: limitations

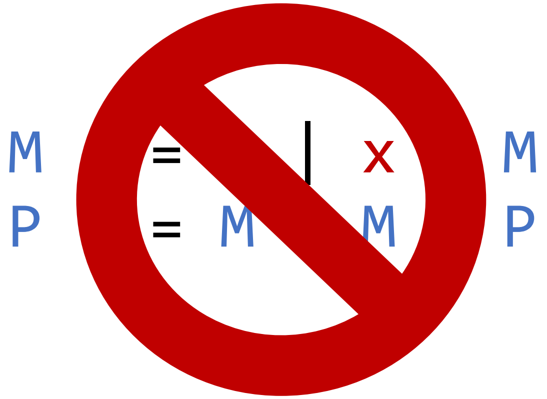
- Everything is bounded
  - loops are unrolled
  - integers are bounded
  - are any of the above easily fixable?
- Too much input from the programmer?
  - but: as search gets better, less user input is required
- CEGIS relies on the Bounded Observation Hypothesis
- Sketches hard to debug
- No bias, no non-functional constraints

Sketch: questions

- Behavioral constraints? structural constraints? search strategy?
  - assertions / reference implementation
  - sketches
  - constraint-based (CEGIS + SAT)
- Sketches vs CFGs? Brahma’s components?
  - A generator can encode a multiset of components (although it’s not very straightforward)
  - Can a generator encode a CFG?

Recursive generators

- What if monomial of every degree can occur at most once?




```
generator int mono(int x, int n) {
    if (n <= 0) {return ??;}
    else {return x * mono(x, n - 1);}
}

generator int poly(int x, int n) {
    if (n <= 0) {return mono(x,0);}
    else {return mono(x,n) + poly(x, n - 1);}
}
```

- Generators are more expressive than CFGs!
  - but unbounded generators cannot be encoded into constraints
  - need to bound unrolling depth
  - bounded generators less expressive than CFGs (but more convenient)

Semantics of abort

$$\mathcal{C}[\![\mathbf{abort}]\!]\langle \sigma, \psi \rangle = \langle \sigma, \bot \rangle$$

CEGIS: the worst case

Satisfiable constraint $\exists c. \forall x. Q(c, x)$ that violates the Bounded Observation Hypothesis

$\boxed{Q(c, x) \equiv x \neq c}$ unsatisfiable

$\boxed{Q(c, x) \equiv c[0] \wedge c[1] \wedge c[2]}$ solved in single iteration for ANY x

$\boxed{Q(c, x) \equiv x = (x \,\&\, c)}$ solved in max n iterations

$\boxed{Q(c, x) \equiv x \leq c}$ solved in one iteration with x = 111
(but will require 2^N iterations with worst-case counterexamples)

$\boxed{Q(c, x) \equiv x \neq c \vee c = 111}$ violates BOH:
no small set of counter-examples exists

Synquid: contributions

- Unbounded correctness guarantees
- Round-trip type system to reject incomplete programs
  - + GFP Horn Solver
- Refinement types can express complex properties in a simple way
  - handles recursive, HO functions
  - automatic verification for a large class of programs due to polymorphism (e.g. sorted list insert)

Synquid: limitations

- User interaction
  - refinement types can be large and hard to write
  - components need to be annotated (how to mitigate?)
- Expressiveness limitations
  - some specs are tricky or impossible to express
  - cannot synthesize recursive auxiliary functions
- Condition abduction is limited to liquid predicates
- Cannot generate arbitrary constants
- No ranking / quality metrics apart from correctness

Synquid: questions

- Behavioral constraints? Structural constraints? Search strategy?
  - Refinement types
  - Set of components + built-in language constraints
  - Top-down enumerative search with type-based pruning
- Typo in the example in Section 3.2
  - $\{B_0 \mid \bot\} \rightarrow \{B_1 \mid \bot\} \rightarrow \{\text{List Pos} \mid \text{len } v = \cancel{2}5\}$

Can RTTC reject these terms?

- `inc ?? :: {Int | v = 5}`
  - where `inc :: x:Int → {Int | v = x + 1}`
  - NO! don't know if we can find `?? :: {Int | v + 1 = 5}`
- `nats ?? :: List Pos`
  - where `nats :: n:Nat → {List Nat | len v = n}`
    `Nat = {Int | v >= 0}`, `Pos = {Int | v > 0}`
  - YES! `n:Nat → {List Nat | len v = n}` not a subtype of
    `_ → List Pos`
- `duplicate ?? :: {List Int | len v = 5}`
  - where `duplicate :: xs:List a → {List a | len v = 2*(len xs)}`
  - YES! using a consistency check (len v = 2*(len xs) ∧ len v = 5 → UNSAT)