Introduction to Sketching

IAP 2008

Armando Solar-Lezama

Experience with homework

Step 1: Turn holes into special inputs

- The ?? Operator is modeled as a special input
  - we call them control inputs

```
bit[W] isolSk(bit[W] x)
{
    return ~(x + ??) & (x + ??);
}
```

→

```
bit[W] isolSk(bit[W] x, bit[W] c1, c2)
{
    return ~(x + c1) & (x + c2);
}
```

- Bounded candidate spaces are important
  - bounded unrolling of **repeat** is important
  - bounded inlining of generators is important

Step 2: Constraining the set of controls

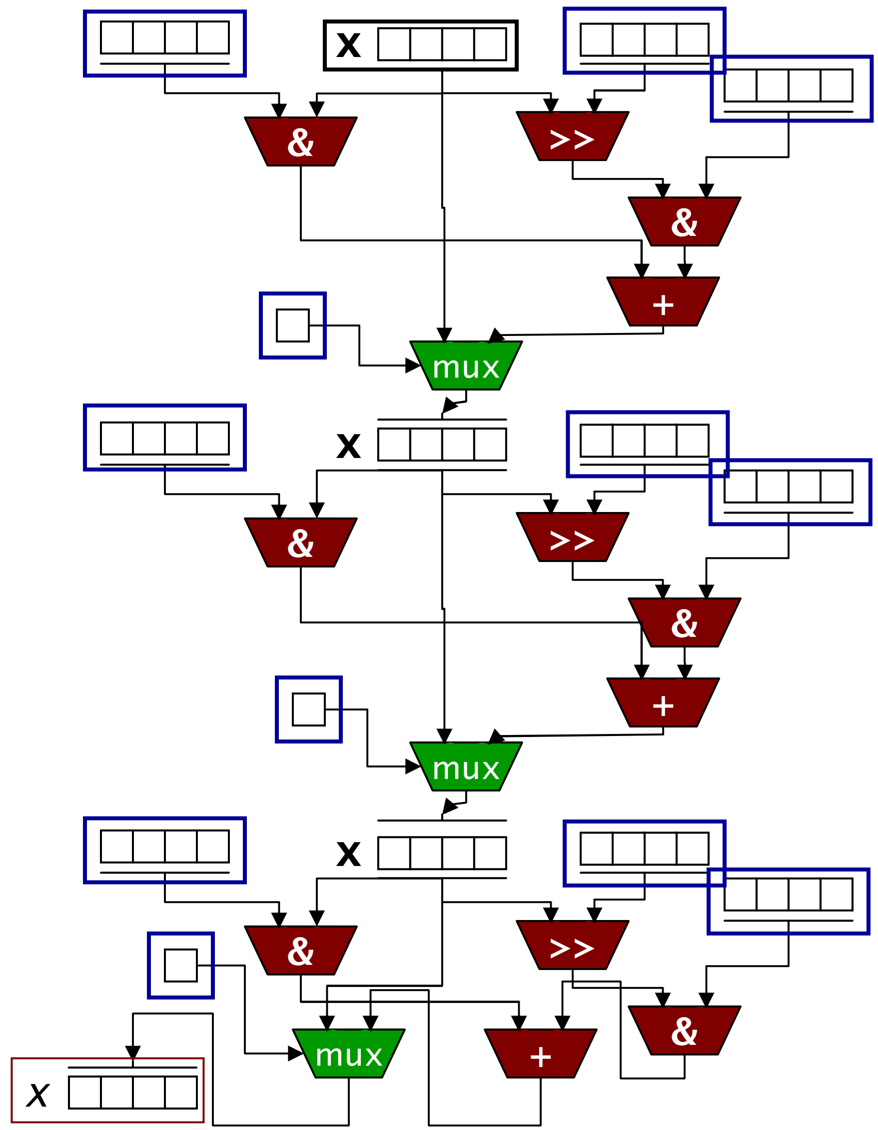
- Correct control
  - causes the spec & sketch to match for all inputs
  - causes all assertions to be satisfied for all inputs
- Constraints are collected into a predicate

```
Q(in, c)
```

- -showDAG will show you the constraints!

```
int popSketched (bit[W] x)
    implements pop {
    loop (??) {
⇒           x = (x & ??)
⇒             + ((x >> ??) & ??);
⇒   }
⇒   return x;
  }
```

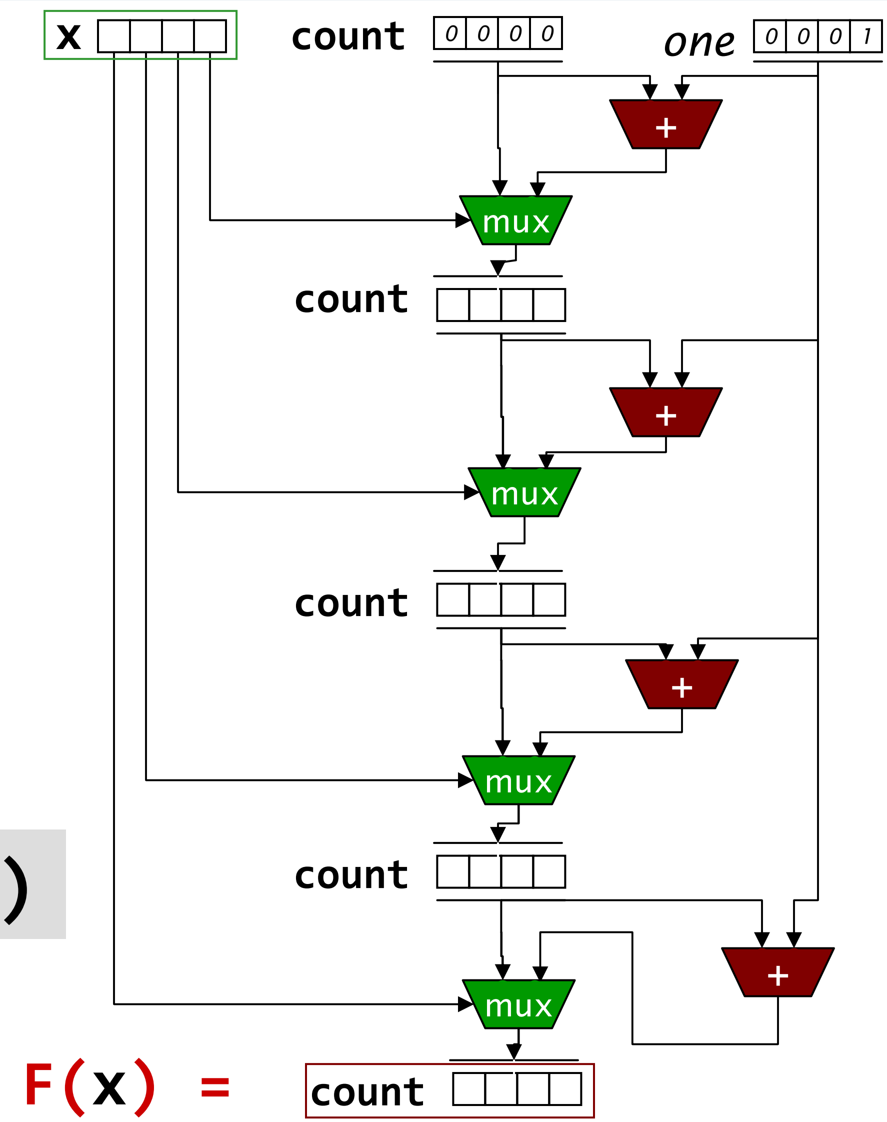
S(x,c) =



Ex : Population count.

```
int pop (bit[W] x)
{
   int count = 0;
   for (int i = 0; i < W;
   i++) {
      if (x[i]) count++;
   }
   return count;
}
```

Q(in, c) = S(x,c)==F(x)



A Sketch as a constraint system

Synthesis reduces to constraint satisfaction

$$\exists\, c.\ \forall\, x.\ Q(x, c)$$

Constraints are too hard for standard techniques

- Universal quantification over inputs
- Too many inputs
- Too many constraints
- Too many holes

Insight

Sketches are not arbitrary constraint systems

- They express the high level structure of a program

A small set of inputs can fully constrain the soln

- focus on corner cases

$$\exists\, c. \forall\, x \text{ in } E.\ Q(x, c)$$

$$\text{where } E = \{x_1, x_2, \ldots, x_k\}$$

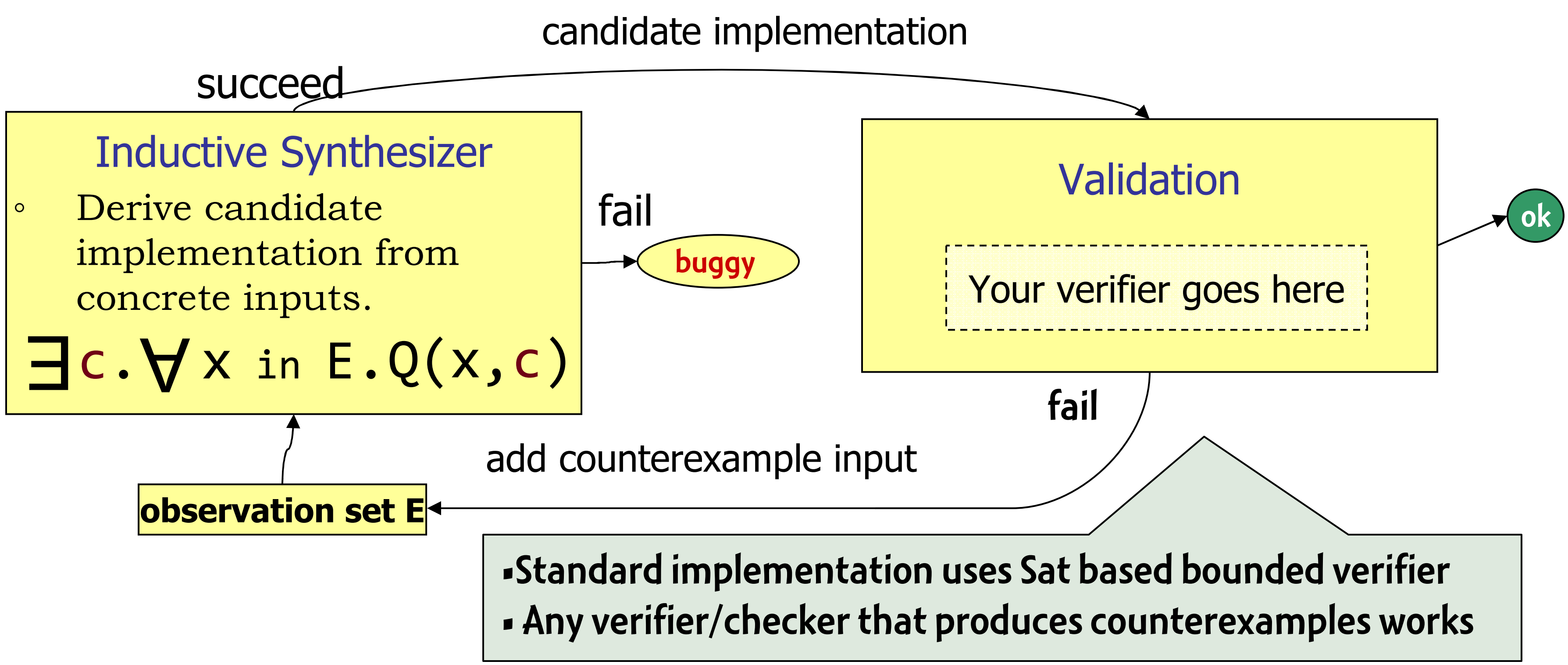
This is an inductive synthesis problem !

- but how do we find the set E?
- and how do we solve the inductive synthesis problem?

Step 3:
Counterexample Guided Inductive Synthesis

Idea: Couple Inductive synthesizer with a verifier

- Verifier is charged with detecting convergence



Inductive Synthesis

Deriving a candidate from a set of observations

$$\exists\, c. \quad \forall\, x \text{ in } E.\ Q(x, c)$$

where $E = \{x_1, x_2, \ldots, x_k\}$

Encode C as a bit-vector

- natural encoding given the integer holes

Encode $Q(x_i, c)$ as boolean constraints on the bit-vector

Solve constraints using SAT solver

- with lots of preprocessing in between

Controlling the SAT Solver

- Several options for the SAT Solver
  - --synth and --verif
    - ABC vs MINI (MiniSat)
  - --cbits and --inbits
  - --incremental

Using synthesizer feedback

- Results of different phases useful for debugging
  - counterexample inputs
  - number of iterations

- Some useful flags
  - --keeptmpfiles
  - --showInputs
  - --fakesolver
  - -checkpoint and -restore