Program synthesis

Tallinn University of Technology

The thesis of Armando Solar Lezama is used -
http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-177.html

Program synthesis

Program Synthesis is the task of discovering an executable program from user intent expressed in the form of some constraints

Challenge of synthesis

Establishing a proper synergy between the human and the synthesizer is fundamental to the success of synthesis

Domain specific synthesis

Domain specific systems take the human insight and build it directly into the synthesizer

- AutoBayes - *data analysis programs from statistical models*
- FFTW - *produces fast Fourier transforms optimized for specic architectures*

Domain specific synthesis

- Generate implementations that often out-perform hand-written code
- Very specific to a field and rely on domain specific knowledge

Deductive approach

- Synthesis systems which allow the user to provide insight directly into the synthesizer
- Program can be extracted from a constructive proof of the satisfiability of a specification
  - KIDS, NuPRL

Deductive approach

- In the hands of experts, these systems are extremely powerful (correct implementation)
- Demands a high level of expertise.

Sketching

A form of synthesis that uses partial programs as a communication device between the programmer and the synthesizer

– focus the synthesizer on low-level details, leaving control of the high-level strategy in the hands of the programmer

Program synthesis

Find a program P that meets a spec ϕ(input, output):

$$\exists P \forall x . \phi(x, P(x))$$

List example

```
list reverse(list l){
        if( isEmpty(l) ){
        return l;
    }else{
        node n = popHead(l);
        return append( reverse(l) , n );
    }
}
```

List example

```
list reverseEfficient(list l){
        list nl = new list();
        while(□) {□}
}
```

List example

The condition for the loop must be a pointer comparison involving some of the memory locations reachable from l and nl

These conditions are stated as expressions called generators

```
#define LOC {| (l | nl).(head | tail)(.next)?
| null |}
#define COMP {| LOC ( == | != ) LOC |}
```

List example

```
list reverseEfficient(list l){

#define LOC {| (l | nl).(head | tail)(.next)? | null |}
#define COMP {| LOC ( == | != ) LOC |}

        list nl = new list();
        while( COMP ){□ }
}
```

List example

- A sequence of assignments to some of the available pointers
- Guard assignments with some condition
- Temporary variable is required
- Use a different iteration condition for the first iteration

List example

```
#define LOC2 {| LOC | tmp |}
#define LHS {| (l | nl).(head)(.next)? | nl.tail | tmp |}

list reverseEfficient(list l){
        list nl = new list();
        node tmp = null;
        bit c = COMP;
        while(c){
                if( COMP ){ LHS = LOC2; }
                if( COMP ){ LHS = LOC2; }
                if( COMP ){ LHS = LOC2; }
                if( COMP ){ LHS = LOC2; }
                if( COMP ){ LHS = LOC2; }
                c = COMP;
        }
}
```

Program synthesis

Find a program P that meets a spec ϕ(input, output):

$$\exists P \forall x . \phi(x, P(x))$$

List example

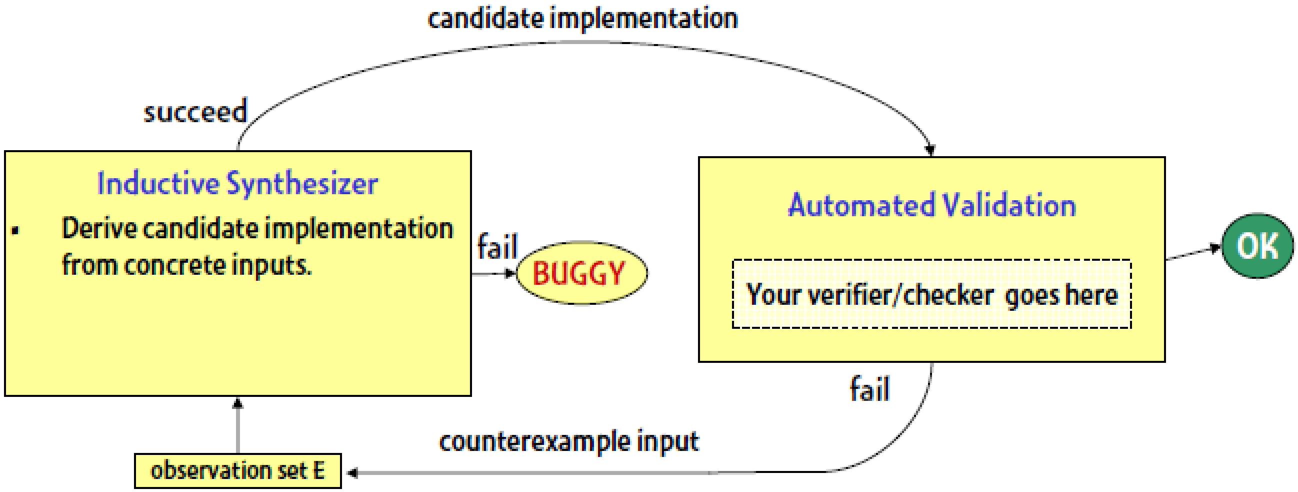
```
main(bit[N] elems, int n){
    if( n < N){
        list l1 = populate(elems, n);
        list l2 = populate(elems, n);
        l1 = reverse(l1);
        l2 = reverseEfficient(l2);
        assert compare( l1, l2) ;
    }
}
```

Counterexample Guided Inductive Synthesis (CEGIS)

In sketching, user insight is provided in the form of a partial program that needs to be completed

Synthesis problem is reduced to a search for constant values to assign to each hole in the sketch

Counterexample Guided Inductive Synthesis



candidate implementation

succeed

Inductive Synthesizer

- Derive candidate implementation from concrete inputs.

fail

BUGGY

Automated Validation

Your verifier/checker goes here

OK

fail

counterexample input

observation set E

Counterexample Guided Inductive Synthesis

The inductive synthesizer uses each new observation to refine its hypothesis about what the correct program should be until it converges to a solution.

Counterexample Guided Inductive Synthesis

Validation procedure checks the candidate implementation produced by the inductive synthesizer

Validation procedure is expected to produce a concrete input which exhibits the bug in the candidate program