Oracle-Guided Component-Based Program Synthesis

Reporter: Song Ao-qi

Authors

Susmit Jha

Technical Director, NuSCI Research Group,

Computer Science Laboratory, SRI International

Sumit Gulwani

Research Manager/Principal Researcher @ Microsoft, Redmond

Affiliate Faculty Member @ Univ. of Washington

Authors

Sanjit A. Seshia

Cadence Founders Chair Professor

Group in Logic and the Methodology of Science University of California, Berkeley

Ashish Tiwari

Principal Researcher @ Microsoft, Redmond

CONTENT

1. From program synthesis to search problem
    - Problem Definition
    - Encoding Programs
    - Constrains
    - Algorithm
2. Solve the search problem
3. Illustration on Running Example

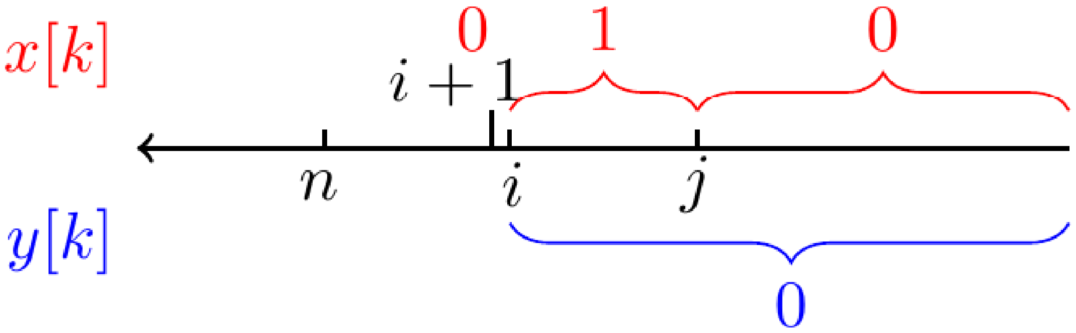
Two examples

Bit Manipulation

- **Given** a bit-vector x
- **Construct** a new bit-vector y that corresponds to x with the right most string of contiguous 1s turned off

$$
\begin{aligned}
\exists i, j.\ \{\ 0 \le i, j < n \wedge (\forall k.\, j \le k \le i \implies x[k] = 1)& \\
\wedge\ (\forall k.\, 0 \le k < j \implies x[k] = 0)& \\
\wedge\ (x[i+1] = 0 \vee i = n-1)& \\
\wedge\ (\forall k.\, i < k < n \implies x[k] = y[k])& \\
\wedge\ (\forall k.\, 0 \le k \le i \implies y[k] = 0)\ \}&
\end{aligned}
$$

specification

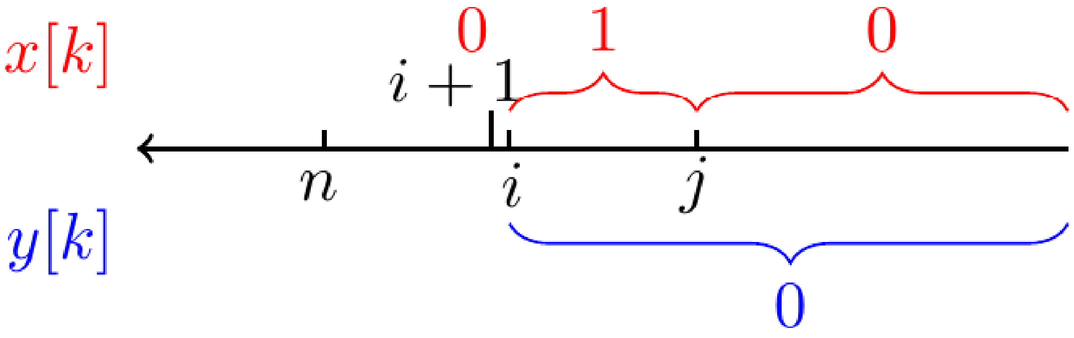


sketch

S. Jha, S. Gulwani, S. A. Seshia and A. Tiwari, "Oracle-guided component-based program synthesis," 2010 ACM/IEEE 32nd International Conference on Software Engineering, Cape Town, South Africa, 2010, pp. 215-224

Two examples

Bit Manipulation




```
turnOffRightMostOneBitString (x)
{ t1 = x-1; t2 = (x | t1); t3 = t2+1;
  t4 = (t3 & x); return t4; }
```

- Oracle-guide: input-output example
- Component-based： a given set of base components

S. Jha, S. Gulwani, S. A. Seshia and A. Tiwari, "Oracle-guided component-based program synthesis," 2010 ACM/IEEE 32nd International Conference on Software Engineering, Cape Town, South Africa, 2010, pp. 215-224

Two examples

Deobfuscation

```
genStringObs(int input)
{
  a1=1, a2=0; b1=1, b2=0; c1=0; c2=0;
  if (input == 0)
     { a1 = 0; a2 = 0; b1=0; b2 = 0; }
  else if (input == 1)
       { c1=0; c2 = 1; }
  else if (input == 2)
       { a1 =1; a2 = 0; c1=1; c2=1; }
  else if (input == 3)
       { b1 = 0; b2 = 0; c1=1; c2=1; }
  else return NULL;
  c = 2*c1 + c2;
  if(c == 1) { return rot13("EPCG GB, 7"); }
  else
  if (c == 2) {
     if (input * (input-1) % 2 == 0)
        return rot13("EPCG GB", 7);
     else
       return rot13("RUYB", 4);
  }
  else {
       if (b1 ⊕ b2) return rot13("ZNVY SEBZ",9)
       else if ((a1 ⊕ a2) = (b1 ⊕ b2))
            return rot13("RUYB", 4);
       else return rot13("QNGN", 4);
  }
}
rot13(char *buf, int sz)
{
  char *buf1 = malloc((sz+1) * sizeof(char));
  char a;
  while (a =˜ *buf)
  {
    *buf1 = (˜a-1/(˜((a | 32))/13*2-11)*13);
    buf++; buf1++;
  }
  return buf1;
}
```

```
genString(int input)
{ if(input == 0) return "EHLO";
  else if (input == 1) return "RCPT TO";
  else if (input == 2) return "MAIL FROM";
  else if (input == 3) return "DATA";
  else return NULL;
}
```

S. Jha, S. Gulwani, S. A. Seshia and A. Tiwari, "Oracle-guided component-based program synthesis," 2010 ACM/IEEE 32nd International Conference on Software Engineering, Cape Town, South Africa, 2010, pp. 215-224

Problem Definition

Base components (subprograms) → candidate program → desired program

Straight-line program (loop-free)

$P(\vec{I})$:

$\quad O_{\pi_1} := f_{\pi_1}(\vec{V}_{\pi_1}); \quad \ldots; \quad O_{\pi_N} := f_{\pi_N}(\vec{V}_{\pi_N});$

$\quad$ `return` O_{π_N};

Assumptions

- program P is using **all components** from the library
- program P is using each base component **only once**.

Given

Oracle:

- Validation oracle: validates the correctness of a candidate program
- I/O oracle : generates input/output examples

Specification for base components (library)

$$\{\langle \vec{I}_i, O_i, \phi_i(\vec{I}_i, O_i)\rangle \mid i = 1, \ldots, N\}$$

S. Jha, S. Gulwani, S. A. Seshia and A. Tiwari, "Oracle-guided component-based program synthesis," 2010 ACM/IEEE 32nd International Conference on Software Engineering, Cape Town, South Africa, 2010, pp. 215-224

Encoding Programs

Location coding

line number	
0	input1
1	input2
$\dots$	
$\lvert\vec{I}\rvert - 1$	input$\lvert\vec{I}\rvert - 1$
$\lvert\vec{I}\rvert$	the first line of program P
$\dots$	
$\lvert\vec{I}\rvert + N - 1$	the last line of program P

Coding of desired program/candidate program

Assumptions

- All components have exactly one output program
- All inputs and outputs have the same type

Location variables L

$$L := \{l_x \mid x \in \mathbf{P} \cup \mathbf{R}\}$$

$$\mathbf{P} := \bigcup_{i=1}^{N} \vec{I}_i \qquad \mathbf{R} := \bigcup_{i=1}^{N}\{O_i\} = \{O_1, \dots, O_N\}$$

S. Jha, S. Gulwani, S. A. Seshia and A. Tiwari, "Oracle-guided component-based program synthesis," 2010 ACM/IEEE 32nd International Conference on Software Engineering, Cape Town, South Africa, 2010, pp. 215-224

Well-formedness constraints

Syntactic constraints

$$\psi_{\texttt{wfp}}(L) \stackrel{\text{def}}{=} \bigwedge_{x \in \mathbf{P}} (0 \leq l_x < M) \land \bigwedge_{x \in \mathbf{R}} (|\vec{I}| \leq l_x < M)$$
$$\land\ \psi_{\texttt{cons}}(L) \land \psi_{\texttt{acyc}}(L)$$

Consistency constraints:

$$\psi_{\texttt{cons}} \stackrel{\text{def}}{=} \bigwedge_{x,y \in \mathbf{R}, x \not\equiv y} (l_x \neq l_y)$$

Acyclicity constraints:

$$\psi_{\texttt{acyc}} \stackrel{\text{def}}{=} \bigwedge_{i=1}^{N} \bigwedge_{x \in \vec{I}_i, y \equiv O_i} l_x < l_y$$

S. Jha, S. Gulwani, S. A. Seshia and A. Tiwari, "Oracle-guided component-based program synthesis," 2010 ACM/IEEE 32nd International Conference on Software Engineering, Cape Town, South Africa, 2010, pp. 215-224

Well-formedness constraints

semantic constraints

$$\phi_{\texttt{func}}(L, \vec{I}, O) \stackrel{\text{def}}{=} \exists \mathbf{P}, \mathbf{R}\ \psi_{\texttt{wfp}}(L) \ \wedge\ \phi_{\texttt{lib}}(\mathbf{P}, \mathbf{R}) \\ \wedge\ \psi_{\texttt{conn}}(L, \vec{I}, O, \mathbf{P}, \mathbf{R})$$

Semantics of the base components:

$$\phi_{\texttt{lib}}(\mathbf{P}, \mathbf{R}) \stackrel{\text{def}}{=} (\bigwedge_{i=1}^{N} \phi_i(\vec{I}_i, O_i))$$

Dataflow semantics:

$$\psi_{\texttt{conn}}(L, \vec{I}, O, \mathbf{P}, \mathbf{R}) \stackrel{\text{def}}{=} \bigwedge_{x,y \in \mathbf{P} \cup \mathbf{R} \cup \vec{I} \cup \{O\}} (l_x = l_y \ \Rightarrow\ x = y)$$

S. Jha, S. Gulwani, S. A. Seshia and A. Tiwari, "Oracle-guided component-based program synthesis," 2010 ACM/IEEE 32nd International Conference on Software Engineering, Cape Town, South Africa, 2010, pp. 215-224

Well-formedness constraints

Phase summary

Syntactic constrains:

- Ordering of location variables [acyclicity]
- Uniqueness of location variables [consistency]

Semantic constrains:

- Relation between inputs and outputs of each component (implies connections between different components)
- "boundary conditions"

Objective function:

$$\phi_{\mathtt{func}}(L, \vec{I}, O) \stackrel{\text{def}}{=} \exists \mathbf{P}, \mathbf{R}\ \psi_{\mathtt{wfp}}(L) \wedge \phi_{\mathtt{lib}}(\mathbf{P}, \mathbf{R}) \wedge \psi_{\mathtt{conn}}(L, \vec{I}, O, \mathbf{P}, \mathbf{R})$$

$$L := \{l_x \mid x \in \mathbf{P} \cup \mathbf{R}\}$$

$$\phi_{func}(L, (\vec{I}, O)) == loss(\theta, (x, y))$$

S. Jha, S. Gulwani, S. A. Seshia and A. Tiwari, "Oracle-guided component-based program synthesis," 2010 ACM/IEEE 32nd International Conference on Software Engineering, Cape Town, South Africa, 2010, pp. 215-224

I/O-behavioral Constraint & Distinguishing Constraint

I/O-behavioral Constraint

$$\mathtt{Behave}_E(L) \stackrel{\text{def}}{=} \bigwedge_{(\alpha_j,\beta_j)\in E} \phi_{\mathtt{func}}(L, \alpha_j, \beta_j)$$

where E is a set of input-output examples.

Distinguishing Constraint

target: find a unique solution

way: data augmentation, but generative ones

$$\mathtt{Distinct}_{E,L}(\vec{I}) \stackrel{\text{def}}{=} \exists L', O, O' \ \mathtt{Behave}_E(L') \wedge \phi_{\mathtt{func}}(L, \vec{I}, O) \wedge \phi_{\mathtt{func}}(L', \vec{I}, O') \wedge O \neq O'$$

~~**How do we know which of O and O' is the true output? Or both are false.**~~

~~**Can not halt**~~

S. Jha, S. Gulwani, S. A. Seshia and A. Tiwari, "Oracle-guided component-based program synthesis," 2010 ACM/IEEE 32nd International Conference on Software Engineering, Cape Town, South Africa, 2010, pp. 215-224

Algorithm

`IterativeSynthesis():`

`// Input: Set of base components used in`
`// construction of` $\texttt{Behave}_E$ `and` $\texttt{Distinct}_{E,L}$
`// Output: Candidate Program`
$E := \{(\alpha_0, \mathcal{I}(\alpha_0))\}$ `//` α_0 `is an arbitrary value for` $\vec{I}$
`while (1) {`
    $L :=$ `T-SAT`$(\texttt{Behave}_E(L))$`;`
    `if` $(L == \perp)$ `return "Components insufficient";`
    $\alpha :=$ `T-SAT`$(\texttt{Distinct}_{E,L}(\vec{I}))$`;`
    `if` $(\alpha == \perp)$ `{`
        $P :=$ `Lval2Prog`(L)`;`
        `if` $(\mathcal{V}(P))$ `return` P`;`
        `else return "Components insufficient"; }`
    $E := E \cup \{\alpha, \mathcal{I}(\alpha)\}$`; }`

- T-SAT
- Lval2Prog()
- I/O oracle
- Validation oracle

S. Jha, S. Gulwani, S. A. Seshia and A. Tiwari, "Oracle-guided component-based program synthesis," 2010 ACM/IEEE 32nd International Conference on Software Engineering, Cape Town, South Africa, 2010, pp. 215-224

Oracle & components

Bit manipulation

- Base component: standard library; extended library (user)
- I/O oracle: user
- Validation oracle: semantically unique candidate program to be correct program in practice; user

Deobfuscation

- Base component: standard library; extended library (user)
- I/O oracle: obfuscated program
- Validation oracle: a program equivalence checking tool; user

S. Jha, S. Gulwani, S. A. Seshia and A. Tiwari, "Oracle-guided component-based program synthesis," 2010 ACM/IEEE 32nd International Conference on Software Engineering, Cape Town, South Africa, 2010, pp. 215-224

Optimization

SMT solvers: z3(python)

Sampling:

- Sampling Uniformly at Random
- Sampling With Bias (a priori)

```
ConstrainedRandomInput:
// cnt is a global variable initialized to 0
// K is a parameter (number of rightmost bits to se
if (cnt < 2^K) {
    α := sample(𝓘nputs);
    α :=Set rightmost K bits of α to cnt;
    cnt := cnt + 1; }
else α := T-SAT(Distinct_{E,L}(I⃗));
```

S. Jha, S. Gulwani, S. A. Seshia and A. Tiwari, "Oracle-guided component-based program synthesis," 2010 ACM/IEEE 32nd International Conference on Software Engineering, Cape Town, South Africa, 2010, pp. 215-224

Illustration on Running Example

A given set of components: bit-wise logical operations; basic arithmetic operations

I/O oracle: the user

Iteration 1 Given the input/output pair (01011, 01000)

[Behave_E] Candidate program 1: (x + 1) & (x − 1)

[Distinct_E,L] Candidate program 2: (x + 1) & x

with distinguishing input 00000

[I/O oracle] (00000, 00000)

E = {(01011, 01000), (00000, 00000)}

Iteration 2

[Behave_E] Candidate program 1: −(¬x) & x

……

(((x − 1)|x) + 1)&x

```
IterativeSynthesis():
// Input:  Set of base components used in
// construction of Behave_E and Distinct_{E,L}
// Output:  Candidate Program
E := {(α_0, I(α_0))} // α_0 is an arbitrary value for I⃗
while (1) {
    L := T-SAT(Behave_E(L));
    if (L == ⊥) return "Components insufficient";
    α := T-SAT(Distinct_{E,L}(I⃗));
    if (α == ⊥) {
        P := Lval2Prog(L);
        if (V(P)) return P;
        else return "Components insufficient"; }
    E := E ∪ {α, I(α)}; }
```

S. Jha, S. Gulwani, S. A. Seshia and A. Tiwari, "Oracle-guided component-based program synthesis," 2010 ACM/IEEE 32nd International Conference on Software Engineering, Cape Town, South Africa, 2010, pp. 215-224

Conclusion & Discussion

Conclusion

1. Encoding program: Modeling Program Synthesis as a Search Problem
2. Program synthesis based on oracle-guided learning from examples and SMT solvers.

Discussion

1. Learning & reasoning
2. Modeling:
    - location coding, functional coding
    - Discover more constrains

S. Jha, S. Gulwani, S. A. Seshia and A. Tiwari, "Oracle-guided component-based program synthesis," 2010 ACM/IEEE 32nd International Conference on Software Engineering, Cape Town, South Africa, 2010, pp. 215-224