Introduction to Software Synthesis

Mooly Sagiv
Idea and Slides taken from Sumit Gulwani, Armando Solar-Lezama, Eran Yahav, Emain Torlak,
## Recap

<table>
<thead>
<tr>
<th>Problem</th>
<th>Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Propositional SAT solving</td>
<td>MiniSat, Z3</td>
</tr>
<tr>
<td>First order solving with theories (SMT)</td>
<td>Z3, CVC3</td>
</tr>
<tr>
<td>Bounded Model Checking</td>
<td>CBMC, JBMC</td>
</tr>
<tr>
<td>Concolic Execution</td>
<td>DART, KLEE, SAGE, Cloud9, Mayhem</td>
</tr>
<tr>
<td>Static analysis</td>
<td>SLAM(SDV), Astrée, TVLA, CSSV</td>
</tr>
<tr>
<td>Testing</td>
<td>PITTEST, AFL</td>
</tr>
<tr>
<td>Program Synthesis</td>
<td>SKETCH(MIT), Rosettee(UWASH)</td>
</tr>
</tbody>
</table>
Verification vs. Synthesis

Solver

Is there a behavior of P that violates \( \varphi \)?

Program P

Desired Properties \( \varphi \)

Counterexample

Proof
Verification vs. Synthesis

Set of programs $\Pi$

Desired Properties $\varphi$

Solver

Is there a program $P \in \Pi$ that satisfies $\varphi$?

Counterexample

Efficient Program $P$
Potential Applications

• Low level programming
  – Configuration
  – Bit manipulation

• Programming for end-users
  – Excel
  – Spreadsheet
  – Spark
What is software synthesis?

Synthesis: Dreams $\Rightarrow$ Programs

ZOHAR MANNA AND RICHARD WALDINGER

INTRODUCTION

In recent years there has been increasing activity in the field of program verification. The goal of these efforts is to construct computer systems for determining whether a
Synthesis Methods

• Deductive Synthesis
  – Derive the low level implementation from high level specification

• Inductive Synthesis
  – Synthesize a program whose behavior satisfies a set of input/output examples
Why now?

- Computer programs are everywhere
- Small programs are tricky
- Maturity of underlying technology
  - Machine learning algorithms
  - SMT
  - Powerful hardware
Motivation

99% of computer users cannot program!
They struggle with simple repetitive tasks.
Spreadsheet help forums

ExcelForum
MS Office Application Help

ExcelExperts.com
Excel Consultancy, VBA Consultancy, Training and Tips Call:+442081234832

MrExcel.com
Your One Stop for Excel Tips & Solutions
Real world application of synthesis
FlashFill: a feature of Excel 2013 (Sumit Gulwani POPL’11)
FlashFill: a feature of Excel 2013

<table>
<thead>
<tr>
<th>Column1</th>
<th>Col 2</th>
<th>Col 3</th>
<th>Col 4</th>
<th>Col 5</th>
<th>Col 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ana Trujillo</td>
<td>357 21st Place SE, Redmond, WA, (757) 555-1634, 140-37-6064, 27171</td>
<td>757</td>
<td>555-1634</td>
<td>140-37-6064</td>
<td>27171</td>
</tr>
<tr>
<td>Antonio Moreno</td>
<td>515 93rd Ave NE, Renton, WA, (412) 555-2876, 682-87-3127, 28581</td>
<td>Renton WA</td>
<td>555-2876</td>
<td>682-87-3127</td>
<td>28581</td>
</tr>
<tr>
<td>Thomas Hardy</td>
<td>742 17th Street NE, Seattle, WA, (412) 555-5719, 921-29-4931, 24607</td>
<td>Seattle</td>
<td>555-5719</td>
<td>921-29-4931</td>
<td>24607</td>
</tr>
<tr>
<td>Christina Berglund</td>
<td>475 22nd Avenue NE, Redmond, WA, (443) 555-6774, 844-35-7674, 30146</td>
<td>Redmond WA</td>
<td>555-6774</td>
<td>844-35-7674</td>
<td>30146</td>
</tr>
<tr>
<td>Hanna Moos</td>
<td>785 45th Street NE, Puyallup, WA, (376) 555-2262, 515-68-1289, 29284</td>
<td>Puyallup</td>
<td>555-2262</td>
<td>515-68-1289</td>
<td>29284</td>
</tr>
<tr>
<td>Frédéric Citeaux</td>
<td>308 66th Place, Redmond, WA, (689) 555-2770, 552-23-2508, 21415</td>
<td>Redmond WA</td>
<td>555-2770</td>
<td>552-23-2508</td>
<td>21415</td>
</tr>
<tr>
<td>Martin Sommer</td>
<td>887 88th Place, Kent, WA, (715) 555-5450, 870-91-9824, 21536</td>
<td>Kent</td>
<td>555-5450</td>
<td>870-91-9824</td>
<td>21536</td>
</tr>
<tr>
<td>Elizabeth Lincoln</td>
<td>452 73rd Place NE, Renton, WA, (851) 555-4569, 425-97-6344, 22779</td>
<td>Renton WA</td>
<td>555-4569</td>
<td>425-97-6344</td>
<td>22779</td>
</tr>
<tr>
<td>Victoria Ashworth</td>
<td>463 16th Street, Renton, WA, (696) 555-6044, 690-29-7926, 22832</td>
<td>Renton WA</td>
<td>555-6044</td>
<td>690-29-7926</td>
<td>22832</td>
</tr>
<tr>
<td>Patrick Simpson</td>
<td>630 20th Street, Renton, WA, (179) 555-3265, 385-78-3236, 45252</td>
<td>Renton WA</td>
<td>555-3265</td>
<td>385-78-3236</td>
<td>45252</td>
</tr>
<tr>
<td>Francisco Chang</td>
<td>653 49th Lane, Seattle, WA, (272) 555-7434, 665-18-6435, 29453</td>
<td>Seattle</td>
<td>555-7434</td>
<td>665-18-6435</td>
<td>29453</td>
</tr>
<tr>
<td>Yang Wang</td>
<td>944 28th Lane, Redmond, WA, (151) 555-2272, 846-78-8452, 24388</td>
<td>Redmond WA</td>
<td>555-2272</td>
<td>846-78-8452</td>
<td>24388</td>
</tr>
<tr>
<td>Pedro Alfonso</td>
<td>411 70th Place, Kent, WA, (170) 555-2964, 774-35-2298, 29485</td>
<td>Kent WA</td>
<td>555-2964</td>
<td>774-35-2298</td>
<td>29485</td>
</tr>
<tr>
<td>Elizabeth Brown</td>
<td>412 20th Lane, Puyallup, WA, (373) 555-4134, 476-53-7164, 28417</td>
<td>Puyallup</td>
<td>555-4134</td>
<td>476-53-7164</td>
<td>28417</td>
</tr>
<tr>
<td>Sven Ottlieb</td>
<td>676 17th Lane NE, Redmond, WA, (828) 555-1593, 548-73-8633, 27440</td>
<td>Redmond WA</td>
<td>555-1593</td>
<td>548-73-8633</td>
<td>27440</td>
</tr>
<tr>
<td>Ann Devon</td>
<td>694 53rd Place, Kent, WA, (194) 555-8124, 659-40-2416, 22367</td>
<td>Kent</td>
<td>555-8124</td>
<td>659-40-2416</td>
<td>22367</td>
</tr>
<tr>
<td>Roland Mendel</td>
<td>581 12th Street NW, Kent, WA, (103) 555-2146, 303-79-1328, 20518</td>
<td>Kent</td>
<td>555-2146</td>
<td>303-79-1328</td>
<td>20518</td>
</tr>
<tr>
<td>Arla Cruz</td>
<td>594 85th Lane, Renton, WA, (431) 555-1376, 329-93-9922, 21498</td>
<td>Renton WA</td>
<td>555-1376</td>
<td>329-93-9922</td>
<td>21498</td>
</tr>
<tr>
<td>Diego Rosi</td>
<td>550 22nd Lane, Renton, WA, (639) 555-6238, 918-34-5172, 25931</td>
<td>Renton WA</td>
<td>555-6238</td>
<td>918-34-5172</td>
<td>25931</td>
</tr>
<tr>
<td>Martine Rancé</td>
<td>688 93rd Place NW, Kent, WA, (873) 555-3571, 695-94-3479, 22424</td>
<td>Kent</td>
<td>555-3571</td>
<td>695-94-3479</td>
<td>22424</td>
</tr>
</tbody>
</table>
Inductive Synthesis

Synthesize a program whose behavior satisfies a set of examples

Doesn’t machine learning do that?

Traditional Bayesian Machine Learning

- Learn a function from a set of examples
- Scalability is very important, algorithms must scale to millions of data points
- Data is assumed to be noisy;
  - need to avoid overfitting
- Space of possible functions is highly stylized
- Background knowledge incorporated as preprocessing and feature selection

Inductive Synthesis

- Learn a function from a set of examples
- Scalability is not so important, usually we are dealing with small numbers of examples
- Data is assumed to be clean
  - It’s annoying when user says $f(x) = y$ and the system assumes the user is wrong and decides that $f(x) = z$
- Space of possible functions can be arbitrary
- Background knowledge encoded in the description of the space and in the search itself
Goal: Synthesize a function that satisfies a specification

- How do we know the specification has been satisfied? Isn’t verification itself already quite hard?
- Can we leverage inductive synthesis machinery for this problem?
- What is the relevant space of functions?
- How do we explore this space efficiently?
The Sketch Synthesis System

Armando Solar-Lezama
bit.ly/iptutorial2015
Extend base language with one construct

Constant hole: ??

```c
int bar (int x)
{
    int t = x * ??;
    assert t == x + x;
    return t;
}
```

Synthesizer replaces ?? with a constant

High-level constructs defined in terms of ??

```c
int bar (int x)
{
    int t = x * 2;
    assert t == x + x;
    return t;
}
```
Expressions with ?? == sets of expressions

- linear expressions
  \[ x^{*??} + y^{*??} \]
- polynomials
  \[ x^{*x^{*??}} + x^{*??} + ?? \]
- sets of variables
  \[ ?? \ ? \ x : \ y \]
Example: Registerless Swap

Swap two words without an extra temporary

```c++
int W = 32;

void swap(ref bit[W] x, ref bit[W] y){
    if(??){ x = x ^ y;}else{ y = x ^ y; }
    if(??){ x = x ^ y;}else{ y = x ^ y; }
    if(??){ x = x ^ y;}else{ y = x ^ y; }
}

harness void main(bit[W] x, bit[W] y){
    bit[W] tx = x; bit[W] ty = y;
    swap(x, y);
    assert x==ty && y == tx;
}
```
From simple to complex holes

We need to compose ?? to form complex holes

Borrow ideas from generative programming

- Define generators to produce families of functions
- Use partial evaluation aggressively
Generators

Look like a function
- but are partially evaluated into their calling context

Key feature:
- Different invocations → Different code
- Can recursively define arbitrary families of programs
Example: Least Significant Zero Bit

- 0010 0101 \rightarrow 0000 0010

```c
int W = 32;

bit[W] isolate0 (bit[W] x) {
    // W: word size
    bit[W] ret = 0;
    for (int i = 0; i < W; i++)
        if (!x[i]) { ret[i] = 1; return ret; }
}
```

Trick:
- Adding 1 to a string of ones turns the next zero to a 1
- i.e. 000111 + 1 = 001000
/**
* Generate the set of all bit-vector expressions
* involving +, &, xor and bitwise negation (~).
* the bnd param limits the size of the generated expression.
*/

generator bit[W] gen(bit[W] x, int bnd){
    assert bnd > 0;
    if(??) return x;
    if(??) return ??;
    if(??) return ~gen(x, bnd-1);
    if(??){
        return { | gen(x, bnd-1) (+ | & | ^) gen(x, bnd-1) |};
    }
}

Example: Least Significant Zero Bit

```c

generator bit[W] gen(bit[W] x, int bnd){
    assert bnd > 0;
    if(??) return x;
    if(??) return ??;
    if(??) return ~gen(x, bnd-1);
    if(??){
        return { | gen(x, bnd-1) (+ | & | ^) gen(x, bnd-1) |};
    }
}

bit[W] isolate0sk (bit[W] x) implements isolate0 {
    return gen(x, 3);
}
```
HOW DOES IT WORK?
CEGIS Synthesis algorithm

Synthesize

\[ \exists c \text{ s.t. Correct}(P_c, \text{in}_i) \]

\{\text{in}_i\}

Check

\[ \exists \text{in} \text{ s.t. Correct}(P_c, \text{in}_i) \]

Insert your favorite checker here
CEGIS

\[
\begin{align*}
&Q(c, in) \\
\text{Synthesize} & \quad Q(c, in_0) \quad Q(c, in_1) \\
& \quad Q(c, in_2) \quad Q(c, in_3) \\
\text{Check} & \quad \neg Q(c, in_3)
\end{align*}
\]
A sketch as a constraint system

```c
int lin(int x){
    if(x > ??)
        return ??2*x + ??3;
    else
        return ??4*x;
}

void main(int x){
    int t1 = lin(x);
    int t2 = lin(x+1);
    if(x<4) assert t1 >= x*x;
    if(x>=4) assert t2-t1 == 1;
}
```
Ex: Population count. 0010 0110 → 3

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

\[ F(x) = \text{count} \]
```c
int popSketched (bit[W] x) implements pop {
  repeat(??) {
    x = (x & ??)
    + ((x >> ??) & ??);
  }
  return x;
}
```

```
S(x,0) = x[x]
```
Synthesizing Locks and Fences

Input:
- A concurrent program

Assertions

Output:
- Locks that guarantee that the assertions hold

\[
\begin{align*}
x &= 1; \\
x &= x + 1; \\
\text{assert } x &= 3 \\
x &= x + 1;
\end{align*}
\]

Veselin Raychev, Martin T. Vechev, Eran Yahav:
Automatic Synthesis of Deterministic Concurrency. SAS 2013: 283-303

Michael Kuperstein, Martin T. Vechev, Eran Yahav:
Synthesizing API calls

- Modern libraries provide a lot of functionality
- But hard to figure out the right sequence of calls
  - A lot of boilerplate code

```java
public void test1()
{
    Area a1 = new Area(new Rectangle(0, 0, 10, 2));
    Area a2 = new Area(new Rectangle(-2, 0, 2, 10));
    Point2D p = new Point2D.Double(0, 0);
    assertTrue(a2.equals(rotate(a1, p, Math.PI/2))); }

Area rotate(Area obj, Point2D pt, double angle)
{
    AffineTransform at = new AffineTransform();
    double x = pt.getX();
    double y = pt.getY();
    at.setToRotation(angle, x, y);
    Area obj2 = obj.createTransformedArea(at);
    return obj2;
}
```

Yu Feng, Ruben Martins, Yuepeng Wang, Isil Dillig, Thomas W. Reps:
Component-based synthesis for complex APIs. POPL 2017: 599-612
John K. Feser, Swarat Chaudhuri, Isil Dillig:
Synthesizing data structure transformations from input-output examples. PLDI 2015: 229-239
Synthesizing Network Policies

Kausik Subramanian, Loris D'Antoni, Aditya Akella:
Synthesizing Cloud Configurations

• Number of machines
• Network
• Maximize throughputs
• Minimize cost