Introduction to Program Synthesis

CS560: Reasoning About Programs

Roopsha Samanta

Partly based on slides by Armando Solar-Lezama and Xiaokang Qiu
Roadmap

Previously
- Course overview

Today
- Introduction to program synthesis
- Project description
What is program synthesis?
1950’s - 1990’s
Much of my work has come from being lazy. I didn't like writing programs, and so, when I was working on the IBM 701, writing programs for computing missile trajectories, I started work on a programming system to make it easier to write programs.

Essentially, compilation!

Backus et al., *The FORTRAN Automatic Coding System*, 1957
1950’s, 1960’s: Church’s Synthesis Problem

Ongoing/Reactive Programs

Alonzo Church

Programs represented as circuits/finite automata

Church, *Application of Recursive Arithmetic to the Problem of Circuit Synthesis*, 1957

Church, *Logic, Arithmetic and Automata*, 1962
The goal of reactive synthesis is to generate a reactive system whose behavior satisfies a temporal specification, in the presence of continuous interaction with an environment.
1960’s, 1970’s: Church’s Synthesis Problem Solved!

Buchi and Landwebber, *Solving Sequential Conditions by Finite-State Strategies*, 1967

Rabin, *Automata on infinite objects and Church’s Problem*, 1972

Extract program from finite-state winning strategy of an infinite two-player game
For every move of the adversary (every action of the environment), the synthesized program must make a counter-move that maintains correctness.

The game can be modeled as an automaton
1960’s, 1970’s: Deductive Synthesis

*Transformational/Functional Programs*

- Cordell Green
- Zohar Manna
- Richard Waldinger

Extract LISP-y program from proof of satisfiability of formal specification

Green, *Application of Theorem Proving to Problem Solving*, 1963

Manna and Waldinger, *Dreams ⇒ Programs*, 1979
Dreams → Programs
Complete Formal Specifications → Programs
∀x,y,z. x ≤ max(x,y,z) ∧ y ≤ max(x,y,z) ∧ z ≤ max(x,y,z) ∧ (max(x,y,z)=x ∨ max(x,y,z)=y ∨ max(x,y,z)=z)

```c
int max (int x, int y, int z)
int m = z;
if (z <= y) m = y;
if (m < x) m = x;
return m;
```
Easier!

(0, 10, 2) ➞ 10
(-1, 10, 20) ➞ 20
(-1, -2, -3) ➞ -1

```c
int max (int x, int y, int z)
int m = z;
if (z <= y) m = y;
if (m < x) m = x;
return m;
```
1970’s: Inductive Programming

Transformational Programs

Summers, *A Methodology for LISP Program Construction from Examples*, 1977

Biermann, *Inference of Regular LISP Programs from Examples*, 1978

Example 8: In a list of lists, obtain the first element of each list: \((A) (B)\) yields \((A \ B)\). First these lists are converted to S-expressions as described in Example 1.

<table>
<thead>
<tr>
<th>Example Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>((A \cdot D) \cdot (B \cdot E) \cdot C))</td>
<td>((A \cdot (B \cdot C)))</td>
</tr>
</tbody>
</table>

Program:

\[
(F_1 X) = (\text{COND}(\text{ATOM} \ X) \ X) \\
((\text{ATOM}(\text{CAR} \ X))(F_1(\text{CAR} \ X))) \\
(\text{COND}(\text{ATOM} \ X)(\text{CONS}(F_2 X)(F_3 X)))\) \\
(F_2 X) = (F_1(\text{CAR} \ X)) \\
(F_3 X) = (F_1(\text{CDR} \ X)).
\]

Time: 18 s.
1980’s: Synthesis of Reactive Programs

Extract program (model) from algorithmically-constructed witness to satisfiability of formal specification.

Clarke & Emerson, *Design and Synthesis of Synchronization Skeletons using Branching-Time Temporal Logic*, 1981
1980’s: Synthesis of Reactive Programs

Pnueli

Better algorithms than Buchi, Landwebber, Rabin

Still an active research area!

1980’s: Programmer’s Apprentice

- Codify expert knowledge on how to solve programming problems
- User guided synthesis

1990’s: Inductive Learning

*Transformational Programs*

Replaced ad-hoc approaches for PBE/PBD with techniques based on version space generalization and inductive logic programming

Post 2000: Modern Program Synthesis
Transformational program synthesis: A *search* problem

**Input-output examples**
- Logical specifications
- Equivalent programs
- Natural language

**Search space**
- Grammar
- DSL
- Partial program
- Components

**Programs**

Find a program in search space consistent with specification
Dimensions in modern program synthesis

User intent
How do you tell the system what you want?

Search strategy
How does the system find the program you actually want?

Search space
What is the space of programs to explore?

[Gulwani 2010]
Dimensions in modern program synthesis

User intent
How do you tell the system what you want?
- Specification formalism?
- Interaction model?
- Ambiguity?

Search strategy
How does the system find the program you actually want?
- How do you guide the system towards relevant programs?
- How does the system exploit the structure of the search space?

Search space
What is the space of programs to explore?
- How do you represent domain knowledge?
- Built-in or user-defined?
Dimensions in modern program synthesis

User intent
- Input-output examples
- Logical specifications
- Equivalent programs
- Natural language

Search strategy
- Enumerative search + pruning
- Constraint-based search
- Representation-based search
- Stochastic search
- ML-based
- ...

Search space
- Grammars/DSLs
- Generators
- Components
2006: Sketch

Armando-Solar Lezamma

User intent
- Input-output examples
- Logical specifications
- Equivalent programs

Search strategy
- Constraint-based search

Search space
- Generators

Solar-Lezamma et al., *Combinatorial Sketching for Finite Programs*, 2006
Goal: Output the least significant zero bit in a word/bitvector
Ex: 00100101 → 00000010

```c
int W = 32;

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

Adding 1 to a string of 1’s turns the next 0 to a 1!
000111 + 1 = 001000

A possible program
Not the most efficient

Trick for an efficient program
/* Generate the set of all bit-vector expressions
 * involving +, &, xor and bitwise negation (~).
 * the bound param limits the size of the generated expression.
 */

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

```c
int W = 32;
bit[W] isolate0 (bit[W] x) {
    bit[W] ret = 0;
    for (int i = 0; i < W; i++)
        if (!x[i]) { ret[i] = 1; return ret; } }

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

bit[W] isolate0fast (bit[W] x) implements isolate0 {
    return gen(x, 3); }
```

Naïve program

Program generator

Specification

“Program equivalence”
Sketch output

```plaintext
bit[W] isolate0fast (bit[W] x) implements isolate0 {
    return ~x & (x+1);
}
```

Desired efficient program
2011: FlashFill

Sumit Gulwani

User intent
Input-output examples

Search strategy
Representation-based search
(Version space algebras)

Search space
Grammars
DSLs

Gulwani, Automatic String Processing in Spreadsheets using Input-Output Examples, 2011
A feature of Excel 2013!
Example: FlashFill

Excel 2013’s coolest new feature that should have been available years ago

Excel Flash Fill Is A Brilliant Time Saver

Excel is now a lot easier for people who aren’t spreadsheet- and chart-making pros. The application’s new Flash Fill feature recognizes patterns, and will offer auto-complete options for your data. For example, if you have a column of first names and a column of last names, and want to create a new column of initials, you’ll only need to type in the first few boxes before Excel recognizes what you’re doing and lets you press Enter to complete the rest of the column.
FlashFill synthesis setup

Specification: “1 input-output example”

Other string inputs

FlashFill synthesizes program consistent with user example [Gulwani 2010]
### FlashFill output

String outputs generated by synthesized program

<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>Redmond WA</td>
<td>555-1634</td>
<td>140-37-6064</td>
<td>27171</td>
</tr>
<tr>
<td>Antonio Moreno</td>
<td>515 93rd Lane, Renton, WA, (411) 555-2786, 562-87-3127, 28581</td>
<td>Renton WA</td>
<td>555-2786</td>
<td>562-87-3127</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 WA</td>
<td>555-5719</td>
<td>921-29-4931</td>
<td>24607</td>
</tr>
<tr>
<td>Christina Berglund</td>
<td>475 22nd Lane, Redmond, WA, (413) 555-6774, 844-35-7646, 30146</td>
<td>Redmond WA</td>
<td>555-6774</td>
<td>844-35-7646</td>
<td>30146</td>
</tr>
<tr>
<td>Hanna Mosq</td>
<td>785 45th Street NE, Puyallup, WA, (376) 555-2462, 515-68-1285, 29284</td>
<td>Puyallup WA</td>
<td>555-2462</td>
<td>515-68-1285</td>
<td>29284</td>
</tr>
<tr>
<td>Frederique Cieux</td>
<td>306 56th Place, Redmond, WA, (689) 555-2770, 552-23-2908, 21415</td>
<td>Redmond WA</td>
<td>555-2770</td>
<td>552-23-2908</td>
<td>21415</td>
</tr>
<tr>
<td>Martin Sommer</td>
<td>887 86th Place, Kent, WA, (715) 555-5450, 870-91-8242, 21536</td>
<td>Kent WA</td>
<td>555-5450</td>
<td>870-91-8242</td>
<td>21536</td>
</tr>
<tr>
<td>Laurence Leibhan</td>
<td>944 13th Street NE, Redmond, WA, (620) 555-2361, 649-25-3312, 25252</td>
<td>Redmond WA</td>
<td>555-2361</td>
<td>649-25-3312</td>
<td>25252</td>
</tr>
<tr>
<td>Elizabeth Lincoln</td>
<td>452 73rd Lane NE, Renton, WA, (851) 555-4561, 425-97-6344, 22279</td>
<td>Renton WA</td>
<td>555-4561</td>
<td>425-97-6344</td>
<td>22279</td>
</tr>
<tr>
<td>Victoria Ashworth</td>
<td>453 16th Street, Renton, WA, (696) 555-6044, 690-29-7926, 23232</td>
<td>Renton WA</td>
<td>555-6044</td>
<td>690-29-7926</td>
<td>23232</td>
</tr>
<tr>
<td>Patricio Simpson</td>
<td>630 20th Street, Redmond, WA, (179) 555-3256, 389-78-3326, 24525</td>
<td>Redmond WA</td>
<td>555-3256</td>
<td>389-78-3326</td>
<td>24525</td>
</tr>
<tr>
<td>Francisco Chang</td>
<td>683 49th Lane SE, Seattle, WA, (272) 555-7434, 685-18-8435, 29453</td>
<td>Seattle WA</td>
<td>555-7434</td>
<td>685-18-8435</td>
<td>29453</td>
</tr>
<tr>
<td>Yang Wang</td>
<td>944 20th 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 Afonso</td>
<td>411 70th Place, Kent, WA, (170) 555-2964, 774-35-2998, 29485</td>
<td>Kent WA</td>
<td>555-2964</td>
<td>774-35-2998</td>
<td>29485</td>
</tr>
<tr>
<td>Elizabeth Brown</td>
<td>971 20th Lane, Puyallup, WA, (376) 555-4134, 476-53-1764, 26417</td>
<td>Puyallup WA</td>
<td>555-4134</td>
<td>476-53-1764</td>
<td>26417</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, 599-74-4016, 22367</td>
<td>Kent WA</td>
<td>555-8124</td>
<td>599-74-4016</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 WA</td>
<td>555-2146</td>
<td>303-79-1328</td>
<td>20518</td>
</tr>
<tr>
<td>Aria Cruz</td>
<td>594 85th Lane, Renton, WA, (431) 555-1376, 329-93-9992, 21498</td>
<td>Renton WA</td>
<td>555-1376</td>
<td>329-93-9992</td>
<td>21498</td>
</tr>
<tr>
<td>Diego Roel</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 93th Place NW, Kent, WA, (573) 555-3571, 695-94-3479, 22424</td>
<td>Kent WA</td>
<td>555-3571</td>
<td>695-94-3479</td>
<td>22424</td>
</tr>
</tbody>
</table>

Average: 27171, Count: 112, Sum: 27171

---

Note: The data is generated by FlashFill, a Microsoft Excel feature that fills out spreadsheet data with input patterns.
Many more interesting papers.....later!
2020: MANTIS

Semantics-guided Inductive Program Synthesis

User intent
Input-output examples

Search strategy
Semantics-guided search

Search space
Partial programs

An et al., Augmented Example-based Synthesis using Relational Perturbation Properties, 2020
Problems in inductive program synthesis

(0, 10, 2) → 10
(-1, 10, 20) → 20
(-1, -2, -3) → -1

Program/Search Space

int max (int x, int y, int z)
int m = z;
if (z <= y) m = y;
if (m < x) m = x;
return m;
Problems in inductive program synthesis

Program/Search Space

int max (int x, int y, int z)
int m = z;
if (z <= y) m = y;
if (m < x) m = x;
return m;

int max (int x, int y, int z)
int m = x;
if (y < z) m = z;
if (m < y) m = y;
return m;

Ambiguity!
Problems in inductive program synthesis

Program/Search Space

\[
\begin{align*}
\text{int max (int x, int y, int z)} \\
\text{int m = z; } \\
\text{if (z <= y) m = y; } \\
\text{if (m < x) m = x; } \\
\text{return m; }
\end{align*}
\]

\[
\begin{align*}
\text{int max (int x, int y, int z)} \\
\text{int m = x; } \\
\text{if (y < z) m = z; } \\
\text{if (m < y) m = y; } \\
\text{return m; }
\end{align*}
\]

(0, 10, 2) \mapsto 10

(-1, 10, 20) \mapsto 20

(-1, -2, -3) \mapsto -1

Overfitting!
Problems in inductive program synthesis

Program/Search Space

```
int max (int x, int y, int z)
int m = x;
if (y < z) m = z;
if (m < y) m = y;
return m;
```

Overfitting!
Problems in inductive program synthesis

Brittleness!

```plaintext
int max (int x, int y, int z)
int m = x;
if (y < z) m = z;
if (m < y) m = y;
return m;
```
Occam’s Razor
Structured DSL
Ranking Function

int max (int x, int y, int z)
int m = z;
if (z <= y) m = y;
if (m < x) m = x;
return m;

int max (int x, int y, int z)
int m = x;
if (y < z) m = z;
if (m < y) m = y;
return m;

Syntactic Bias

(0, 10, 2) ↦ 10
(-1, 10, 20) ↦ 20
(-1, -2, -3) ↦ -1

Syntactic bias can also be inadequate!
int max (int x, int y, int z)
int m = z;
if (z <= y) m = y;
if (m < x) m = x;
return m;

(0, 10, 2) ↦ 10
(-1, 10, 20) ↦ 20
(-1, -2, -3) ↦ -1

Permutation Invariance

Program/Search Space

int max (int x, int y, int z)
int m = x;
if (y < z) m = z;
if (m < y) m = y;
return m;
Semantic Bias!

Program/Search Space

\[
\begin{align*}
(0, 10, 2) & \mapsto 10 \\
(-1, 10, 20) & \mapsto 20 \\
(-1, -2, -3) & \mapsto -1 \\
(2, 10, 0) & \mapsto 10 \\
(20, -1, 10) & \mapsto 20 \\
(-2, -1, -3) & \mapsto -1 \\
\vdots
\end{align*}
\]

Permutation Invariance

```
int max (int x, int y, int z)
int m = z;
if (z <= y) m = y;
if (m < x) m = x;
return m;
```

\(\checkmark\)
Key Strategy

1. Augment examples by applying properties

2. Use PBE engine to synthesize program from augmented examples
197 Benchmarks: **SKETCH, SYGUS**
Bitvectors, Integers, Integer arrays

**SKETCH**

Reference Implementations

**SKETCHAX**

Set of 8 properties
60% improvement!

Benchmarks with Properties

- **SKETCH**
- **SKETCHAX SELECTION**
- **SKETCHAX VALIDATION**
- **SKETCHAX INFERENCE**

Success Rate

0% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%

Benchmarks with Properties
Benchmarks with Properties

- SKETCH
- SKETCHAX SELECTION
- SKETCHAX VALIDATION
- SKETCHAX INFERENCE

Bitvector Benchmarks with Properties

- SKETCH
- SKETCHAX SELECTION
- SKETCHAX VALIDATION
- SKETCHAX INFERENCE

60% improvement!

191% improvement!
Success Rate

- Properties
- Bitvector
- Properties
- Integer
- No Properties
- All

Success Rates:
- Properties: +60%
- Bitvector: +191%
- Properties: +30%
- Integer: +1%
- No Properties: +22%
Your project – Apply MANTIS to your domain!
Example Domains

- Imperative programs over simple datatypes (integers, Booleans, arrays, matrices)
- Heap-manipulating (recursive) imperative programs
- Functional programs with algebraic datatypes
- Relational queries (SQL, Datalog)
- String-manipulating programs
- Table-manipulating programs
- Parser synthesis
- Map-reduce distributed programs
- Web programming
- Regular expressions
- ...
1. Identify domain

2. Identify applicable semantic properties

3. Use existing inductive synthesizer as black-box
   or
   Build your own search algorithm/synthesizer!
Summary

Today
- Introduction to program synthesis
- Project description

Next
- Propositional logic
- Normal Forms