Future of Software Development?
Program Synthesis and its Application

Paul Zhu
School of Software,
Tsinghua University

May 27, 2019
History & Trends in Software Development

In the past:
- Write programs that can work
- Write programs that runs fast

Issues:
- Write it once, debug it everywhere
- No silver bullet
- Many programmers don’t welcome changes

Nowadays and in future:
- Write programs that are correct
- Write programs that generate programs
Program Synthesis is the task of **automatically** finding programs from the underlying programming language that satisfy **user intent** expressed in some form of constraints. (Gulwani, Polozov, and Singh 2017)

Features:
- automated programming
- correctness-by-construction

v.s.
- compilers
- code generator
Program Synthesis in Action

Programming by Examples
- Case Study: FlashFill

Sketch-based Synthesis
- Case Study: Rosette
Contents

1 Program Synthesis in Action

2 Programming by Examples
   - Case Study: FlashFill

3 Sketch-based Synthesis
   - Case Study: Rosette
FlashFill
Automated String Processing in Spreadsheets by Examples (Gulwani 2011)

<table>
<thead>
<tr>
<th>Name</th>
<th>Title</th>
<th>Email Alias</th>
<th>First Name</th>
<th>Last Name</th>
<th>Full Email Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Johanna Lorenz</td>
<td>Senior Engineer</td>
<td>JohannL</td>
<td>Johanna</td>
<td>Lorenz</td>
<td><a href="mailto:johannl@contoso.com">johannl@contoso.com</a></td>
</tr>
<tr>
<td>Irvin Sayers</td>
<td>Project Manager, Co-Project Lead</td>
<td>IrvinS</td>
<td>Irvin</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lidia Holloway</td>
<td>Product Manager, Co-Project Lead</td>
<td>LidiaH</td>
<td>Lidia</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Henrietta Mueller</td>
<td>Developer</td>
<td>HenriettaM</td>
<td>Henrietta</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ben Walters</td>
<td>Tester</td>
<td>BenW</td>
<td>Ben</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Megan Bowen</td>
<td>Marketing Manager</td>
<td>MeganB</td>
<td>Megan</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enrico Cattaneo</td>
<td>Attorney</td>
<td>EnricoC</td>
<td>Enrico</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pradeep Gupta</td>
<td>Accountant</td>
<td>PradeepG</td>
<td>Pradeep</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Adele Vance</td>
<td>Developer</td>
<td>AdeleV</td>
<td>Adele</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alex Wilber</td>
<td>Tester</td>
<td>AlexW</td>
<td>Alex</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Allen Deyoung</td>
<td>Developer</td>
<td>AllenD</td>
<td>Allen</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Brian Johnson</td>
<td>Engineer</td>
<td>BrianJ</td>
<td>Brian</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Emily Braun</td>
<td>Engineer</td>
<td>EmilyB</td>
<td>Emily</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lee Gu</td>
<td>Developer</td>
<td>LeeG</td>
<td>Lee</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Patti Fernandez</td>
<td>Tester</td>
<td>PattiF</td>
<td>Patti</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Miriam Graham</td>
<td>Developer</td>
<td>MiriamG</td>
<td>Miriam</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Title / Author</td>
<td>Cited by</td>
<td>Year</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------------------------------------------------------------------------</td>
<td>----------</td>
<td>------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Finding bugs with a constraint solver</strong></td>
<td>197</td>
<td>2000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D Jackson, M Vaziri</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACM SIGSOFT Software Engineering Notes 25 (5), 14-25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Associating synchronization constraints with data in an object-oriented language</strong></td>
<td>176</td>
<td>2006</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M Vaziri, F Tip, J Dolby</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACM SIGPLAN Notices 41 (1), 334-345</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Some Shortcomings of OCL, the Object Constraint Language of UML.</strong></td>
<td>81</td>
<td>2000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M Vaziri, D Jackson</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOOLS (34), 555-562</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Model checking software systems: A case study</strong></td>
<td>68</td>
<td>1995</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JM Wing, M Vaziri-Farahani</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACM SIGSOFT Software Engineering Notes 20 (4), 128-139</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Hades
Transformations on Hierarchically Structured Data (Yaghmazadeh et al. 2016)
Sketch-N-Sketch
Programmatic and Direct Manipulation Together (Chugh et al. 2016)

(A) Excerpt from prelude.little

(defrec range (λ(i j)
    (if (> i j) nil
        (cons i (range (+ 1 l_1 i) j)))))

(def zeroTo (λn (range 0 l_0 (- n 1))))

(B) sineWaveOfBoxes.little

(def [x0 y0 w h sep amp] [50 120 20 90 30 60])
(def n 12!{3-30})
(def boxi (λi
    (let xi (+ x0 (* i sep))
        (let yi (- y0 (* amp (sin (* i (/ twoPi n))))))
            (rect 'lightblue' xi yi w h)))

(svg (map boxi (zeroTo n)))

(C) Suppose the user clicks on the third box from the left (colored darker in red for emphasis) and drags it to a new position down and to the right (colored lighter in gray):

(D) Sketch-N-Sketch synthesizes four candidate updates to the program, which have the following effects:
An end user says: “I would like the time of your earliest flight in the morning from Philadelphia to Washington on American Airlines.”

Then it is translated to a domain-specific language program:

```
ColSelect(DEP_TIME, RowMin(DEP_TIME,
    RowPred(EqDepart(PHILADELPHIA, Time(MORNING)),
        EqArrive(WASHINGTON, Time(ANY)),
        EqAirline(AMERICAN))))
```
def computeDeriv(poly):
    length = int(len(poly)-1)
    i = length
    deriv = range(1,length)
    if len(poly) == 1:
        deriv = [0]
    else:
        while i >= 0:
            new = poly[i] * i
            i -= 1
            deriv[i] = new
    return deriv

The program requires 2 changes:

- In the expression `range(1, length)` in line 4, increment `length` by 1.
- In the comparison expression `(i >= 0)` in line 8, change operator `>=` to `!=.`
Scythe
Synthesizing Highly Expressive SQL Queries by Examples (Wang, Cheung, and Bodik 2017)

<table>
<thead>
<tr>
<th>( T_1 )</th>
<th>( T_2 )</th>
<th>( T_{out} )</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>oid</strong></td>
<td><strong>val</strong></td>
<td><strong>col</strong>&lt;br&gt;( c_0 )&lt;br&gt;( c_1 )&lt;br&gt;( c_2 )&lt;br&gt;( c_3 )&lt;br&gt;( c_4 )</td>
</tr>
<tr>
<td>1</td>
<td>30</td>
<td>12/25 &lt;br&gt;1 &lt;br&gt;10 &lt;br&gt;10 &lt;br&gt;50 &lt;br&gt;10</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>12/25 &lt;br&gt;1 &lt;br&gt;10 &lt;br&gt;10 &lt;br&gt;50 &lt;br&gt;10</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>12/24 &lt;br&gt;50 &lt;br&gt;50 &lt;br&gt;50 &lt;br&gt;50 &lt;br&gt;50</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>12/24 &lt;br&gt;50 &lt;br&gt;50 &lt;br&gt;50 &lt;br&gt;50 &lt;br&gt;50</td>
</tr>
</tbody>
</table>

```
Select *
From  (Select *
    From T1
    Where T1.date = 12/24
    Or T1.date = 12/25) T3
Join  (Select oid, Max(val)
    From  (Select *
        From T2
        Where T2.val < 50) T4
    Group By oid) T5
On    T3.uid = T5.oid
```
AutoMerge
Conflict Resolution for Structured Merge (Zhu and He 2018)

constructed VSA

includes

expected

```java
1   if (sourceExcerpt != null) {
2     ...
3     if (excerpt.equals(LINE) && 0 <= charno
4       && charno < sourceExcerpt.length()) {
5       +if (excerpt.equals(LINE) && 0 <= charno
6         && charno <= sourceExcerpt.length()) {
7         ...
8     }

Figure 1: A bug in Closure compiler, revision 1e070472. The bug is at lines 3–4. The developer fix is shown on lines 5–6; it turns a < to a <= in the second line of the if condition.

<table>
<thead>
<tr>
<th>Test</th>
<th>charno</th>
<th>excerpt.equals(LINE)</th>
<th>sourceExcerpt.length()</th>
<th>Desired Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>7</td>
<td>true</td>
<td>7</td>
<td>true</td>
</tr>
<tr>
<td>B</td>
<td>10</td>
<td>true</td>
<td>10</td>
<td>true</td>
</tr>
</tbody>
</table>

Figure 2: Input-output examples for both variables and conditions, extracted for the Closure compiler bug described in Figure 1. We use M1 and M2 to refer to the conditions in columns 3–4 in subsequent exposition. The last column represents the desired output of the overall branch decision.
```java
1 public class MethodeReturn : MonoBehaviour
2 {
3     static void LongHypo(float a, float b)
4     {
5         float SommeCar = a * a + b * b;
6         return SommeCar;
7     }
8 }
9
10     void Start()
11     {
12         float result = LongHypo(3, 4);
13         Debug.Log(result);
14     }
15 }
```

Synthesized repair: return type `void` is changed to `float` for `LongHypo`.
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;
}
Interactive Parser Synthesis by Example
(Leung, Sarracino, and Lerner 2015)

Figure 1. The Parsify user interface: (a) File View, (b) Legend, (c) Label Box, (d) Label Button, (e) Parse Tree Pane, (f) Resolution Pane, and (g) Negative Label
Bonsai
Synthesis-based Reasoning for Type Systems (Chandra and Bodik 2017)
Program Synthesis Helps!

Many people can benefit from it:

- end users
- developers
- educators & learners
- researchers

A lot of fields are studying it:

- programming languages
- software development
- program repair
- program analysis & verification
- machine learning
Contents

1 Program Synthesis in Action

2 Programming by Examples
   • Case Study: FlashFill

3 Sketch-based Synthesis
   • Case Study: Rosette
**An Excel Task**

### Example (Month Extraction)

<table>
<thead>
<tr>
<th>Input ( v_1 )</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>01/21/2001</td>
<td>01</td>
</tr>
<tr>
<td>22.02.2002</td>
<td>02</td>
</tr>
<tr>
<td>2003-23-03</td>
<td>03</td>
</tr>
<tr>
<td>06/01/2019</td>
<td>?</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Bob specifies what he wants to do by filling in the output cell for the first three rows. Now he wants Excel to finish the remaining rows.
Bob asks Alice, and Alice says: “You can use regular expressions!”

Now, Bob has two problems...
Programming by Examples

Features:

- input-output examples as specification
- enable non-programmers to create programs for automating repetitive tasks

Challenges:

- description of program space
- representation of program space
- disambiguation
1 Program Synthesis in Action

2 Programming by Examples
   - Case Study: FlashFill

3 Sketch-based Synthesis
   - Case Study: Rosette
Domain-specific Language (DSL) of FlashFill

Program $P$ ::= \(\text{Switch}((b_1, e_1), \ldots, (b_n, e_n))\)

\(\text{Bool } b \ ::= \ d_1 \lor d_2 \lor \cdots \lor d_n\)

\(\text{Conjunction } d \ ::= \ \pi_1 \land \pi_2 \land \cdots \land \pi_n\)

\(\text{Predicate } \pi \ ::= \ \text{Match}(v_i, r, k) \mid \neg \text{Match}(v_i, r, k)\)

\(\text{Trace expression } e \ ::= \ \text{Concat}(f, e) \mid f\)

\(\text{Atomic expression } f \ ::= \ \text{SubStr}(v_i, p_1, p_2) \mid \text{Const}(s) \mid \text{Loop}(\lambda w.e)\)

\(\text{Position } p \ ::= \ \text{CPos}(k) \mid \text{Pos}(r_1, r_2, c)\)

\(\text{Regular expression } r \ ::= \ t_1 t_2 \cdots t_n\)

\(\text{Token } t \ ::= \ C+ \mid [^C]+ \mid \text{Start} \mid \text{End} \mid . \mid - \mid \ast \mid / \mid \cdots\)

\(\text{Character set } C \ ::= \ [0-9] \mid [A-Z] \mid [a-z] \mid [A-Za-z] \mid \cdots\)

\(\text{Integer expression } c \ ::= \ k \mid k_1 w + k_2\)

where $k$ denotes an integer, $s$ denotes a string, $w$ binds to an integer, and $v_i$ refers to the $i$-th column of input.
Shriram Krishnamurthi once\textsuperscript{1} said: “These are the only choices in life for a configuration language: it grows up to a programming language, or it dies.”

- Gradle build scripts are written using a variant of Groovy/Kotlin.
- Sbt can execute a Scala script containing dependency declarations or other settings.

\textsuperscript{1}https://www.youtube.com/watch?v=3N__tvzMrzc
Example (Month Extraction)

<table>
<thead>
<tr>
<th>Input $v_1$</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>01/21/2001</td>
<td>01</td>
</tr>
<tr>
<td>22.02.2002</td>
<td>02</td>
</tr>
<tr>
<td>2003-23-03</td>
<td>03</td>
</tr>
</tbody>
</table>

Program: \( \text{Switch}((b_1, e_1), (b_2, e_2), (b_3, e_3)) \), where

- \( b_1 = \text{Match}(v_1, /) \)
- \( b_2 = \text{Match}(v_1, .) \)
- \( b_3 = \text{Match}(v_1, -) \)
- \( e_1 = \text{SubStr}(v_1, \text{Pos}(\text{StartToken}, \varepsilon, 1), \text{Pos}(\varepsilon, /, 1)) \)
- \( e_2 = \text{SubStr}(v_1, \text{Pos}(., \varepsilon, 1), \text{Pos}(\varepsilon, ., 2)) \)
- \( e_3 = \text{SubStr}(v_1, \text{Pos}(-, \varepsilon, 2), \text{Pos}(\text{EndToken}, \varepsilon, 1)) \)
... and No

Example (To Lower Case)

<table>
<thead>
<tr>
<th>Input $v_1$</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>HELLO</td>
<td>hello</td>
</tr>
<tr>
<td>World</td>
<td>world</td>
</tr>
</tbody>
</table>
Learning Traces: Motivation

\[ e ::= \text{Concat}(f, e) \mid f \]

Example

All trace expressions yielding “abc” could be any one of the following:

- all atomic expressions yielding “abc”
- concatenations of all atomic expressions yielding “ab” and all trace expressions yielding “c”
- concatenations of all atomic expressions yielding “a” and all trace expressions yielding “bc”
Learning Traces: Motivation

\[ e ::= \text{Concat}(f, e) \mid f \]

**Example**

All trace expressions yielding “abc” could be any one of the following:
- all atomic expressions yielding “abc”
- all concatenations of a possible split of “abc”
Version Space Algebra (VSA)

- First defined by Mitchell 1982, and then expanded upon program synthesis by Gulwani 2011; Polozov and Gulwani 2015.
- Succint representation by its memory-sharing mechanism.

\[
VSA \tilde{N} \ ::= \begin{cases} 
\{P_1, P_2, \ldots, P_k\} & \text{(explicit)} \\
\tilde{N}_1 \cup \tilde{N}_2 \cup \cdots \cup \tilde{N}_k & \text{(union)} \\
F \ltimes (\tilde{N}_1, \tilde{N}_2, \ldots, \tilde{N}_k) & \text{(join)}
\end{cases}
\]

where each VSA node represents a set of concrete programs:

\[
\begin{align*}
\llbracket \{P_1, \ldots, P_k\} \rrbracket &= \{P_1, \ldots, P_k\} \\
\llbracket \tilde{N}_1 \cup \cdots \cup \tilde{N}_k \rrbracket &= \llbracket \tilde{N}_1 \rrbracket \cup \cdots \cup \llbracket \tilde{N}_k \rrbracket \\
\llbracket F \ltimes (\tilde{N}_1, \ldots, \tilde{N}_k) \rrbracket &= \{F(P_1, \ldots, P_k) \mid P_1 \in \llbracket \tilde{N}_1 \rrbracket, \ldots, P_k \in \llbracket \tilde{N}_k \rrbracket\}
\end{align*}
\]
Learning Traces: VSA Representation

\[ e ::= \text{Concat}(f, e) \mid f \]

Let \( f_{123}, \tilde{f}_{12}, \) etc. be the VSA for atomic expressions yielding resp. “abc”, “ab”, etc. Then the VSA for trace expressions yielding “abc” is:
## Deductive Synthesis

\[ \tilde{P} \models \phi \]

<table>
<thead>
<tr>
<th>Programs are expressed by</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>a DSL terminal</td>
<td>( \frac{P_1 \models \phi, \ldots, P_k \models \phi}{{P_1, \ldots, P_k} \models \phi} )</td>
</tr>
<tr>
<td>a DSL nonterminal with ( k ) choices</td>
<td>( \frac{P_1 \models \phi, \ldots, P_k \models \phi}{\tilde{P}_1 \cup \cdots \cup \tilde{P}_k \models \phi} )</td>
</tr>
<tr>
<td>a ( k )-ary DSL operator ( F )</td>
<td>( \frac{F #(\tilde{P}_1, \ldots, \tilde{P}_k) \models \phi}{\tilde{P}_1 \models \phi_1, \ldots, \tilde{P}_k \models \phi_k} )</td>
</tr>
</tbody>
</table>

In the last case, we need a witness function for \( F \) to compute \( \phi_1, \ldots, \phi_k \) from \( \phi \).
Witness Functions give Reverse Semantics

- **Semantics**: given $\llbracket f \rrbracket_\sigma = s_1$ and $\llbracket e \rrbracket_\sigma = s_2$, compute the output for $\llbracket \text{Concat}(f, e) \rrbracket_\sigma$, i.e. $s_1$ concatenates with $s_2$.

- **Witness function**: given $\text{Concat}_\triangledown(\tilde{f}, \tilde{e}) \models \phi$, find $\phi_1$ and $\phi_2$ such that $\tilde{f} \models \phi_1$ and $\tilde{e} \models \phi_2$.

- **A specification $\phi$ is a set of pairs of an input state with an output string.**

- **Reverse semantics**: given $\llbracket \text{Concat}(f, e) \rrbracket_\sigma = s$, find all possible $\llbracket f \rrbracket_\sigma$ and $\llbracket e \rrbracket_\sigma$, i.e. all concatenations which yields $s$. 
Ranking

Based upon Occam's razor:

- A Concat constructor is simpler than another one if it contains smaller number of arguments or its arguments are pairwise simpler.
- SubStr-expressions are simpler than Const-expressions.
- Pos-expressions are simpler than CPos-expressions.
- Start and End are simpler than all other tokens.
- etc.

Alternatively, scoring functions (e.g. in PROSE) and even machine learning techniques (e.g. Ellis and Gulwani 2017) come to rescue here.
Artifacts

Microsoft PROgram Synthesis using Examples SDK
- Tutorial: https://microsoft.github.io/prose
- Samples: https://github.com/microsoft/prose
- Platform: .NET Framework, .NET Core
- Warning: the official document may be inconsistent with the implementation, and some SDK APIs may behave unexpectedly

My implementation: https://github.com/paulzfm/StringProcessing
Contents

1 Program Synthesis in Action

2 Programming by Examples
   - Case Study: FlashFill

3 Sketch-based Synthesis
   - Case Study: Rosette
Logical Programming

In Prolog\(^2\), one may specify a knowledge base:

\[
\text{likes(sam,Food) :- indian(Food), mild(Food).}
\]
\[
\text{likes(sam,Food) :- chinese(Food).}
\]
\[
\text{likes(sam,chips).}
\]

\[
\text{indian(curry).}
\]
\[
\text{indian(tandoori).}
\]
\[
\text{mild(tandoori).}
\]
\[
\text{chinese(chow_mein).}
\]
\[
\text{chinese(jiao_zi).}
\]

and execute a query

\[
?- \text{likes(sam, X).}
\]

whose solutions are

\[
X = \text{tandoori}; \ X = \text{chow_mein}; \ X = \text{jiao_zi}; \ X = \text{chips}.
\]

\(^2\)http://www.swi-prolog.org
Make Logical Programming Great Again?

Next-paradigm programming languages could be based on any of several potential technologies – e.g., perhaps on machine learning and statistical techniques, or on SMT solvers and symbolic reasoning. Regardless of the technology, however ... “principles”:

- Productivity and Performance Tied Together
- Need For Firm Mental Grounding
- Workflows Will Change

– Yannis Smaragdakis

Also, he invented a “theorem”:

Programs ≠ Algorithms + Data Structures

Compiler = Algorithms + Data Structures

---

We’ve Seen this

Given a DSL expressing bitwise operations:

\[
\text{program } P ::= \text{plus}(E_1, E_2) | \text{mul}(E_1, E_2) | \text{shl}(E, C) | \text{shr}(E, C)
\]

\[
\text{expression } E ::= x | C
\]

8-bit constant \( C ::= (00000000)_2 | (00000001)_2 | \cdots | (11111111)_2 \)

where the input variable \( x \) is bound to a 8-bit unsigned integer.

We hope to synthesize a program \( P \) satisfying the specification

\[
\phi = \forall x. P(x) \geq 0.
\]

In other words, find an instantiation for the sketch \( P =? \) wrt \( \phi \).
Contents

1. Program Synthesis in Action

2. Programming by Examples
   - Case Study: FlashFill

3. Sketch-based Synthesis
   - Case Study: Rosette
Demonstration

Rosette\textsuperscript{4} is a solver-aided programming language based on Racket, which combines verification and sketch-based synthesis.

The example comes from Torlak and Bodik 2014.

\footnote{https://docs.racket-lang.org/rosette-guide/index.html}
Will Program Synthesis Change the World?

Bad news:
- **Scalability** is a big issue
- Yet not very practical for **industry**

Good news:
- Many people are already **benefit** from PBE
- Solvers are becoming more and more **efficient** and **powerful**
- Program synthesis is an **active** and **interdisciplinary** field


