Microsoft PROSE SDK: A Framework for Inductive Program Synthesis

Oleksandr Polozov
University of Washington

Sumit Gulwani
Microsoft Research
Why do people create frameworks?

Industrialization (a.k.a. “Tech Transfer”)
Program Synthesis: “The Ultimate Dream” of CS
Industrialization Time?

Flash Fill (2010-2012)

Trifacta (2012-2015)

SPIRAL (2000-2015) +114 more
Microsoft Program Synthesis using Examples SDK

The Program Synthesis using Examples (PROSE) SDK includes a set of technologies for the automatic generation of programs from input-output examples. This repo includes samples and sample data for the Microsoft PROSE SDK.

> Latest Release
Want to see a quick summary of the release highlights? Visit the Updates section to learn more about the PROSE SDK and the feature set available in our latest release.

> Playground
The PROSE playground is an interface to experiment with domain-specific languages for extracting and transforming data. It is built on top of the PROSE SDK.

> Getting Started
Everything you need to know to get up and running with the PROSE SDK in a quick and easy tutorial. This section will walk you through all the steps required to download

> Documentation
Learn more about the PROSE SDK.
- Base SDK
- Extract Text
- Extract Web
- Flash Fill
- API Documentation

https://microsoft.github.io/prose
Shoulders of Giants

Deductive Synthesis

Syntax-Guided Synthesis

Domain-Specific Inductive Synthesis

PROSE
Shoulders of Giants

Deductive Synthesis

+ No invalid candidates $\implies$ fast
- [Usually] complete specs
- Domain axiomatization

Püschel et al. [IEEE '05]
Panchekha et al. [PLDI '15]
Manna, Waldinger [TOPLAS '80]
Shoulders of Giants

Syntax-Guided Synthesis

Alur et al. [FMCAD '13]

+ Shrinks the search space
+ Generic algorithms

− No domain-specific insights
− Limited to SMT-LIB
Shoulders of Giants

+ Arbitrarily complex DSLs
+ Input/output examples
  - 1-2 person-years (PhD)
  - One-off

Domain-Specific Inductive Synthesis

Lau et al. [ICML '00]
Gulwani [POPL '10] etc.
Feser et al. [PLDI '15]
Shoulers of Giants

“Divide & Conquer”
Deductive Synthesis

“Search over a DSL”
Syntax-Guided Synthesis

“Learn from examples”
Domain-Specific Inductive Synthesis

⇓
Search Algorithm

⇓
Programming Language

⇓
User Intent
PROSE

Meta-synthesizer framework

Synthesis Strategies

PROSE

DSL Definition

Synthesizer

App

Programs

I/O Specification

Input

Output

13
Domain-Specific Language
FlashFill (portion) as a PROSE DSL

string output(string[] inputs) :=
    | ConstantString(s)
    | let string x = std.list.Kth(inputs, k) in
        Substring(x, positionPair(x));

Tuple<int, int> positionPair(string s) :=
    std.Pair(positionIn(s), positionInIn(s));

int positionIn(string s) := AbsolutePosition(s, k)
    | RegexPosition(s, std.Pair(r, r), k);

const int k;        const RegularExpression r;    const string s;
DSL design = Art + *Lots* of iterations
Inductive Specification
## Input-Output Examples

<table>
<thead>
<tr>
<th>input state $\sigma$</th>
<th>$\implies$</th>
<th>output value $\text{out}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>“206-279-6261”</td>
<td>$\implies$</td>
<td>“(206) 279-6261”</td>
</tr>
<tr>
<td>“415.413.0703”</td>
<td>$\implies$</td>
<td>“(415) 413-0703”</td>
</tr>
<tr>
<td>“(646) 408 6649”</td>
<td>$\implies$</td>
<td>“(646) 408-6649”</td>
</tr>
</tbody>
</table>
When one example is too many
Inductive Specification

input state $\sigma$ $\implies$ output constraint $\varphi(out)$

$\implies$ out $\equiv$ ["2010", "2014", ...]
Inductive Specification

input state $\sigma \implies$ output constraint $\varphi(out)$

$\exists ["2010", "2014", ...] \lor \exists "Springer" \lor \exists "[11]"$
Examples are ambiguous!
From:
all lines ending with “Number ◦ Dot”
“Space ◦ Number ◦ Dot”
starting with “Word ◦ Space ◦ CamelCase”

Extract:
the first “Number” before a “Dot”
the last “Number” before a “Dot”
the last “Number” before a “Dot ◦ LineBreak”
the last “Number”
text between the last “Space” and the last “Dot”
the first “Comma ◦ Space” and the last “Dot ◦ LineBreak”

... and up to $10^{20}$ more candidates
One program is insufficient.

Program Set $\Rightarrow$ Ranking

(Version Space Algebra)

User interaction
Runtime correction

...
Synthesis Strategy
Observation 1: Inverse Semantics

\[ F(A, B) \models \phi ? \]

\[ A \models \phi_A ? \]

\[ B \models \phi_B ? \]
Concat\((F, E)\)

\[ \varphi: \begin{cases} 
\text{“Kathleen S. Fisher” } \Rightarrow \text{“Dr. Fisher”} \\
\text{“Bill Gates, Sr.” } \Rightarrow \text{“Dr. Gates”} 
\end{cases} \]

\(\exists E:\) Concat\((F, E)\) satisfies \(\varphi\) if and only if \(F\) satisfies ___________?

\[ \varphi_f: \begin{cases} 
\text{“Kathleen S. Fisher” } \Rightarrow \text{“D” } \lor \text{“Dr” } \lor \text{“Dr.” } \lor \text{“Dr. ” } \lor \text{“Dr. F” } \lor \ldots \\
\text{“Bill Gates, Sr.” } \Rightarrow \text{“D” } \lor \text{“Dr” } \lor \text{“Dr.” } \lor \text{“Dr. ” } \lor \text{“Dr. G” } \lor \ldots 
\end{cases} \]

\(\exists F:\) Concat\((F, E)\) satisfies \(\varphi\) if and only if \(E\) satisfies ___________?

\(\times\quad F\text{ and } E\text{ are not independent!}\)
Observation 2: Skolemization

\[ F(A, B) \models \phi? \]

Given \( A(\sigma) = a \)

\[ A \models \phi_A? \quad \leftrightarrow \quad B \models \phi_B? \]
Concat($F$, $E$)

$\varphi$: \begin{align*}
"Kathleen S. Fisher" & \Rightarrow "Dr. Fisher" \\
"Bill Gates, Sr." & \Rightarrow "Dr. Gates"
\end{align*}

$\exists E$: Concat($F$, $E$) satisfies $\varphi$ if and only if $F$ satisfies __________?

$\varphi_f$: \begin{align*}
"Kathleen S. Fisher" & \Rightarrow "D" \lor "Dr" \lor "Dr." \lor "Dr. " \lor "Dr. F" \lor \ldots \\
"Bill Gates, Sr." & \Rightarrow "D" \lor "Dr" \lor "Dr." \lor "Dr. " \lor "Dr. G" \lor \ldots
\end{align*}

Given an output of $F$, Concat($F$, $E$) satisfies $\varphi$ if and only if $E$ satisfies __________?

$F = \begin{align*}
"Kathleen S. Fisher" & \Rightarrow "Dr. " \\
"Bill Gates, Sr." & \Rightarrow "Dr."
\end{align*}$ \quad $\Rightarrow$ \quad $\varphi_E$: \begin{align*}
"Kathleen S. Fisher" & \Rightarrow "Fisher" \\
"Bill Gates, Sr." & \Rightarrow "Gates"
\end{align*}$
Inverse Semantics + Skolemization = Witness Function

**Witness function:** $\varphi \mapsto \varphi_F$

$\exists E: \text{Concat}(F, E) \text{satisfies } \varphi \text{ if and only if } F \text{satisfies } \ldots$

**Conditional witness function:** $(\varphi \mid F(\sigma) = f) \mapsto \varphi_E$

*Given an output of $F$, Concat($F, E$) satisfies $\varphi$ if and only if $E$ satisfies $\ldots$*

**Domain-Specific**  
**Modular**  
**No synthesis reasoning**  
**Enable efficient deduction**
Results
Unifies 10+ prior POPL/PLDI/… papers

- ...
Program Synthesis meets Software Engineering

<table>
<thead>
<tr>
<th>Project</th>
<th>Reference</th>
<th>Lines of Code</th>
<th>Development Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Original</td>
<td>PROSE</td>
</tr>
<tr>
<td>Flash Fill</td>
<td>POPL 2010</td>
<td>12K</td>
<td>3K</td>
</tr>
<tr>
<td>Text Extraction</td>
<td>PLDI 2014</td>
<td>7K</td>
<td>4K</td>
</tr>
<tr>
<td>Text Normalization</td>
<td>IJCAI 2015</td>
<td>17K</td>
<td>2K</td>
</tr>
<tr>
<td>Spreadsheet Layout</td>
<td>PLDI 2015</td>
<td>5K</td>
<td>2K</td>
</tr>
<tr>
<td>Web Extraction</td>
<td>—</td>
<td>—</td>
<td>2.5K</td>
</tr>
</tbody>
</table>
Performance: 0.5 – 3X Original

More general ⇒ Slower  Algorithmic advances ⇒ Faster

Example: FlashExtract

Learning time = 1.6 sec

2300 nodes in a VSA data structure ≈ log(# of programs)

3 examples till task completion
Performance: $0.5 - 3X$ Original

More general $\Rightarrow$ Slower  
Algorithmic advances $\Rightarrow$ Faster

Example: FlashExtract
Applications
Email Parsing in Cortana
ConvertFrom-String in PowerShell

PS C:\> $template = @'
1  {Time*:1} ms  <1 ms  <1 ms  cusred024ca901-tengige0-007-13.network.microsoft.com [[IP:10.31.196.2]]
10 {Time*:2} ms  1 ms  1 ms  {IP:104.44.81.80}
12  2 ms  2 ms  1 ms  a-0001.a-msedge.net [204.79.197.200]
'

PS C:\> tracert bing.com | ConvertFrom-String -TemplateContent $template

Time IP
----- --
1  10.31.196.2
1  10.37.1.174
1  10.37.66.201
1  10.37.44.94
1  10.37.67.230
1  10.37.45.69
1  131.107.202.162
2  131.107.200.18
1  207.46.36.105
2  104.44.81.80
1  10.201.196.145
1  204.79.197.200
Research: https://microsoft.github.io/prose
Play: https://microsoft.github.io/prose/demo
Contact: prose-contact@microsoft.com
See our demo @ MSR table:

Thank you!

Questions?