Spreadsheet Programming using Examples

Sumit Gulwani
Microsoft

Keynote at SEMS
July 2016
Motivation

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

Programming by examples (PBE)
can revolutionize this landscape!
Spreadsheet help forums

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

MRExcel.com
Your One Stop for Excel Tips & Solutions
Typical help-forum interaction

300_w30_aniSh_c1_b → w30
300_w5_aniSh_c1_b → w5

= MID(B1, 5, 2)

= MID(B1, FIND(“_”, $B:$B) + 1, FIND(“_”, REPLACE($B:$B, 1, FIND(“_”, $B:$B), “”)) ) - 1

A HUGE Thank you!!!!!! 😊😊😊😊😊😊
Flash Fill (Excel 2013 feature) demo

“Automating string processing in spreadsheets using input-output examples”; POPL 2011; Sumit Gulwani
Number Transformations

<table>
<thead>
<tr>
<th>Input</th>
<th>Output (Round to 2 decimal places)</th>
<th>Excel/C#:</th>
<th>Python/C#:</th>
<th>Java:</th>
</tr>
</thead>
<tbody>
<tr>
<td>123.4567</td>
<td>123.46</td>
<td>#.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>123.4</td>
<td>123.40</td>
<td></td>
<td>.2f</td>
<td></td>
</tr>
<tr>
<td>78.234</td>
<td>78.23</td>
<td></td>
<td></td>
<td>#.###</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input</th>
<th>Output (Nearest lower half hour)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0d 5h 26m</td>
<td>5:00</td>
</tr>
<tr>
<td>0d 4h 57m</td>
<td>4:30</td>
</tr>
<tr>
<td>0d 4h 27m</td>
<td>4:00</td>
</tr>
<tr>
<td>0d 3h 57m</td>
<td>3:30</td>
</tr>
</tbody>
</table>

Synthesizing Number Transformations from Input-Output Examples; CAV 2012; Singh, Gulwani
Semantic String Transformations

MarkupRec Table

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
<th>Markup</th>
</tr>
</thead>
<tbody>
<tr>
<td>S33</td>
<td>Stroller</td>
<td>30%</td>
</tr>
<tr>
<td>B56</td>
<td>Bib</td>
<td>45%</td>
</tr>
<tr>
<td>D32</td>
<td>Diapers</td>
<td>35%</td>
</tr>
<tr>
<td>W98</td>
<td>Wipes</td>
<td>40%</td>
</tr>
<tr>
<td>A46</td>
<td>Aspirator</td>
<td>30%</td>
</tr>
</tbody>
</table>

CostRec Table

<table>
<thead>
<tr>
<th>Id</th>
<th>Date</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>S33</td>
<td>12/2010</td>
<td>$145.67</td>
</tr>
<tr>
<td>S33</td>
<td>11/2010</td>
<td>$142.38</td>
</tr>
<tr>
<td>B56</td>
<td>12/2010</td>
<td>$3.56</td>
</tr>
<tr>
<td>D32</td>
<td>1/2011</td>
<td>$21.45</td>
</tr>
<tr>
<td>W98</td>
<td>4/2009</td>
<td>$5.12</td>
</tr>
</tbody>
</table>

Input $v_1$ | Input $v_2$ | Output (Price + Markup*Price)

- Stroller 10/12/2010 $145.67 + 0.30*145.67$
- Bib 23/12/2010 $3.56 + 0.45*3.56$
- Diapers 21/1/2011
- Wipes 2/4/2009
- Aspirator 23/2/2010

Learning Semantic String Transformations from Examples;
VLDB 2012; Singh, Gulwani
Data Science Class Assignment

To get Started!

cat superbowl.txt | awk '$1=$1' ORS=' ' | sed 's/-/\n/g' | grep -v "Championship"
Ships inside two Microsoft products:

- **Powershell**

  ConvertFrom-String cmdlet

- **Operations Management Suite**

  Custom Log, Custom Field

“FlashExtract: A Framework for data extraction by examples”; PLDI 2014; Vu Le, Sumit Gulwani
PBE allows creation of output table from couple of example tuples.

Flashrelate: extracting relational data from semi-structured spreadsheets using examples; PLDI 2014; Barowy, Gulwani, Hart, Zorn
Programming-by-Examples Architecture

Example-based Intent → Program Synthesizer → Program set (a sub-DSL of D) → DSL D
Balanced Expressiveness
- Expressive enough to cover wide range of tasks
- Restricted enough to enable efficient search
  - Restricted set of operators
    - those with small inverse sets
  - Restricted syntactic composition of those operators

Natural computation patterns
- Increased user understanding/confidence
- Enables selection between programs, editing
Flash Fill DSL (String Transformations)

*Tuple*(String $x_1, ..., String x_n$) → String

top-level expr $T := if\text{-}then\text{-}else(B, C, T)$
  \[ | \quad C \]

condition-free expr $C := Concatenate(A, C)$
  \[ | \quad A \]

atomic expression $A := SubStr(X, P, P)$
  \[ | \quad ConstantString \]

input string $X := x_1 \mid x_2 \mid ...$

position expression $P := K$
  \[ | \quad Pos(X, R_1, R_2, K) \]
  $K^{th}$ position in $X$ whose left/right side matches with $R_1/R_2$.

Boolean expression $B := ...$
Programming-by-Examples Architecture

Example-based Intent ➔ Program Synthesizer ➔ Program set (a sub-DSL of D) ➔ DSL D
**Goal:** Set of program expr of kind $e$ that satisfies spec $\phi$

[denoted $[e \models \phi]$ ]

e: DSL (top-level) expression

$\phi$: Conjunction of (input state $\sigma \xrightarrow{\cdot} \text{output value } v$)

**Methodology:** Based on divide-and-conquer style problem decomposition.

- $[e \models \phi]$ is reduced to simpler problems (over sub-expressions of $e$ or sub-constraints of $\phi$).
- Top-down (as opposed to bottom-up enumerative search).

"FlashMeta: A Framework for Inductive Program Synthesis"; OOPSLA 2015; Alex Polozov, Sumit Gulwani
Problem Reduction Rules

Let $e$ be a non-terminal defined as $e := e_1 \mid e_2$

$$[e \models \phi] = Union([e_1 \models \phi], [e_2 \models \phi])$$

An alternative strategy:

$$[e \models \phi_1 \land \phi_2] = Filter([e \models \phi_1], \phi_2)$$

$$[e \models \phi_1 \land \phi_2] = Intersect([e \models \phi_1], [e \models \phi_2])$$

Let $e$ be a non-terminal defined as $e := e_1 \mid e_2$

$$[e \models \phi] = Union([e_1 \models \phi], [e_2 \models \phi])$$
Problem Reduction Rules

\[
[F(e_1, \ldots, e_n) \models \sigma \Rightarrow v] = \\
\text{Union}(\{F([e_1 \models \sigma \Rightarrow u_1], \ldots, [e_n \models \sigma \Rightarrow u_n]) \mid (u_1, \ldots, u_n) \in F^{-1}(v)\})
\]

\(F(S_1, \ldots, S_n)\) denotes \(\{F(e_1, \ldots, e_n) \mid e_1 \in S_1, \ldots, e_n \in S_n\}\)

**Inverse Set:** Let \(F\) be an \(n\)-ary operator.

\[
F^{-1}(v) = \{(u_1, \ldots, u_n) \mid F(u_1, \ldots, u_n) = v\}
\]

\(\text{Concat}^{-1}("Abc") = \{("Abc","\epsilon"), ("Ab","c"), ("A","bc"), (\epsilon,"Abc")\}\)

\[
[\text{Concat}(X, Y) \models (\sigma \Rightarrow "Abc") = \text{Union}(\{
\text{Concat}([X \models (\sigma \Rightarrow "Abc")], [Y \models (\sigma \Rightarrow \epsilon)])),
\text{Concat}([X \models (\sigma \Rightarrow "Ab")], [Y \models (\sigma \Rightarrow "c")]),
\text{Concat}([X \models (\sigma \Rightarrow "A")], [Y \models (\sigma \Rightarrow "bc")]),
\text{Concat}([X \models (\sigma \Rightarrow \epsilon)], [Y \models (\sigma \Rightarrow "Abc")])\})
\]
Programming-by-Examples Architecture

Example-based Intent → Program Synthesizer

Ranking fn → Ranked Program set (a sub-DSL of D)

DSL D

Diagram: A flowchart illustrating the architecture of Programming-by-Examples. The flowchart starts with 'Example-based Intent' and moves through 'Program Synthesizer', 'Ranking fn', and 'DSL D' to 'Ranked Program set (a sub-DSL of D)'.
Prefer simpler programs
• Fewer constants.
• Smaller constants.

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rishabh Singh</td>
<td>Rishabh</td>
</tr>
<tr>
<td>Ben Zorn</td>
<td>Ben</td>
</tr>
</tbody>
</table>

• 1st Word
• If (input = “Rishabh Singh”) then “Rishabh” else “Ben”
• “Rishabh”

“Predicting a correct program in Programming by Example”; [CAV 2015] Rishabh Singh, Sumit Gulwani
Prefer simpler programs

• Fewer constants.
• Smaller constants.

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Missing page numbers, 1993</td>
<td>1993</td>
</tr>
<tr>
<td>64-67, 1995</td>
<td>1995</td>
</tr>
</tbody>
</table>

• 1\textsuperscript{st} Number from the beginning
• 1\textsuperscript{st} Number from the end

How to select between programs with same number of same-sized constants?

Prefer programs that generate more uniform output.
Outline

• Core Synthesis Architecture
  – Domain-specific Language
  – Search methodology
  – Ranking function

➢ Next generation Synthesis
  – Interactive
  – Predictive
  – Adaptive
Programming-by-Examples Architecture

Example based Intent

Program Synthesizer

Ranked Program set (a sub-DSL of D)

Refined Intent

Debugging

Intended Program in D

Translator

Intended Program in R/Python/C#/C++/…

Test inputs

Incrementality

Ranking fn DSL D
Interactive Debugging

- Sampling inputs
- Asking questions
- Visual explanations of the synthesized program
• Intended programs can sometimes be synthesized from just the input.
  – Tabular data extraction, Sort, Join

• Can save large amount of user effort.
  – User need not provide examples for each of tens of columns.
Programming-by-Examples Architecture

Example based Intent → Program Synthesizer

Ranked Program set (a sub-DSL of D) → Incrementality

Refined Intent → Debugging

Intended Program in D → Translator

Test inputs → Intended Program in R/Python/C#/C++/…
• Learn from past interactions
  – of the same user (personalized experience).
  – of other users in the enterprise/cloud.

• The synthesis sessions now require less interaction.
Programming-by-Examples Architecture

Example based Intent -> Program Synthesizer

Refined Intent

Ranked Program set (a sub-DSL of D)

Debugging

Translator

Intended Program in R/Python/C#/C++/…

Test inputs

Incrementality

Interaction history

Interaction history

Ranking fn

DSL D

Learner

Ranking fn

Intended Program in D
Efficient implementation of the generic search methodology.
Provides a library of reduction rules.

Role of synthesis designer
- Implement a DSL and provide reduction rules for new operators.
- Provide ranking strategy.
- Can specify tactics to resolve non-determinism in search.
The PROSE Team

We are hiring interns/full-time!

We are hiring interns/full-time!
Future Directions

- Learn from usage data
- Probabilistic noise handling
- Programming using natural language
- Application to robotics
Conclusion

• PBE can enable easier & faster data wrangling.
  – 99% of computer users are non-programmers.
  – Data scientists spend 80% time cleaning data.

• Algorithmic search
  – Domain-specific language
  – Deductive methodology based on back-propagation

• Ambiguity resolution
  – Ranking
  – Interactivity

Reference: “Programming by Examples (and its applications in Data Wrangling)”, In Verification and Synthesis of Correct and Secure Systems; IOS Press; 2016 [based on Marktoberdorf Summer School 2015 Lecture Notes]