Programming by Examples
Applications, Algorithms & Ambiguity Resolution

Sumit Gulwani
Microsoft
Stanford Lecture
March 2017
Motivation

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

ExcelForum.com
MS Office Application Help

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

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

300_w30_aniSh_c1_b → w30
300_w5_aniSh_c1_b → w5

=A HUGE Thank you!!!!!

=MID(B1,5,2)

=MID(B1,FIND(“_”,$B:$B)+1, FIND(“_”,REPLACE($B:$B, 1, FIND(“_”,$B:$B), “”))-1)
Flash Fill (Excel 2013 feature) demo

“Automating string processing in spreadsheets using input-output examples”; POPL 2011; Sumit Gulwani
Synthesizing Number Transformations from Input-Output Examples; CAV 2012; Singh, Gulwani
To get Started!

```
cat superbowl.txt | awk '{$1=$1}' ORS=' ' | sed 's/\- ||\n/g' | grep "^" | style="text-align: center;"" | grep -v "Championship"
```
FlashExtract Demo

Ships inside two Microsoft products:

- ConvertFrom-String cmdlet
- Custom Log, Custom Field

“FlashExtract: A Framework for data extraction by examples”; PLDI 2014; Vu Le, Sumit Gulwani
Trifacta facilitates the task through a series of steps, suggesting small transformations at each step:

1. **Split on “:” Delimiter**
   
<table>
<thead>
<tr>
<th>Bureau of L.A. Regional Director</th>
<th>Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Niles C.</td>
<td>Tel: (800)645-8397</td>
</tr>
<tr>
<td></td>
<td>Fax: (907)586-7252</td>
</tr>
<tr>
<td>Jean H.</td>
<td>Tel: (918)781-4600</td>
</tr>
<tr>
<td></td>
<td>Fax: (918)781-4604</td>
</tr>
<tr>
<td>Frank K.</td>
<td>Tel: (615)564-6500</td>
</tr>
<tr>
<td></td>
<td>Fax: (615)564-6701</td>
</tr>
</tbody>
</table>

2. **Delete Empty Rows**

3. **Fill Values Down**

4. **Pivot Number on Type**

<table>
<thead>
<tr>
<th>Bureau of L.A. Regional Director</th>
<th>Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Niles C.</td>
<td>Tel: (800)645-8397</td>
</tr>
<tr>
<td></td>
<td>Fax: (907)586-7252</td>
</tr>
<tr>
<td>Jean H.</td>
<td>Tel: (918)781-4600</td>
</tr>
<tr>
<td></td>
<td>Fax: (918)781-4604</td>
</tr>
<tr>
<td>Frank K.</td>
<td>Tel: (615)564-6500</td>
</tr>
<tr>
<td></td>
<td>Fax: (615)564-6701</td>
</tr>
</tbody>
</table>

From: Skills of the Agile Data Wrangler (tutorial by Hellerstein and Heer)
FlashRelate Demo

“FlashRelate: Extracting Relational Data from Semi-Structured Spreadsheets Using Examples”;
PLDI 2015; Barowy, Gulwani, Hart, Zorn
Killer Application: Data Wrangling

Data is the new oil.
Data scientists spend 80% time wrangling data.

Extraction
• FlashExtract: Tabular data from log files [PLDI 2014]
• Tabular data from spreadsheets [PLDI 2015]

Transformation
• FlashFill: Syntactic String Transformations [POPL 2011]
• Number Transformations [CAV 2013]
• Date Transformations [POPL 2016]
• Semantic String Transformations [VLDB 2012]
• Normalization Transformations [IJCAI 2015]

Formatting
• FlashRelate: Table layout transformations [PLDI 2011]
• Repetitive editing in Powerpoint [AAAI 2014]
Programming-by-Examples Architecture

Example-based Intent → Program Synthesizer → Program set (a sub-DSL of D) → DSL D
Domain-specific Language (DSL)

- **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)

\[ Tuple(String \ x_1, ..., String \ x_n) \rightarrow String \]

**top-level expr**
- if-then-else
  - | \( C \)

**condition-free expr**
- \( C := Concatenate(A,C) \)
  - | \( A \)

**atomic expression**
- \( A := SubStr(X,P,P) \)
  - | \( \text{ConstantString} \)

**input string**
- \( X := x_1 \mid x_2 \mid ... \)

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

**Boolean expression**
- \( B := ... \)
FlashExtract DSL (Sequence Extraction)

\[ String \ d \rightarrow List(\ String) \]

\[
\begin{align*}
\text{Seq expr } \ E & \ := \ \text{Map}(N, \ \lambda z: \ S[z]) \\
& \quad \mid \ \text{Merge}(E_1, \ E_2)
\end{align*}
\]

\[
\begin{align*}
\text{some lines } \ N & \ := \ \text{FilterByPosition}(L, \ \text{init}, \ \text{iter}) \\
& \quad \mid \ \text{Filter}(L, \ \lambda z: \ F[z]) \\
& \quad \mid \ \text{Filter}(L, \ \lambda y: \ F[\text{prevLine}(y)])
\end{align*}
\]

\[
\begin{align*}
\text{line filter function } \ F[z] & \ := \ \text{Contains}(z, r, K) \\
& \quad \mid \ \text{startsWith}(z, r)
\end{align*}
\]

\[
\begin{align*}
\text{all lines } \ L & \ := \ \text{Split}(d, \ "\backslash n")
\end{align*}
\]

\[
\begin{align*}
\text{substr expr } \ S[z] & \ :=
\end{align*}
\]

"FlashExtract: A Framework for data extraction by examples";
PLDI 2014; Vu Le, Sumit Gulwani
Programming by Examples Architecture

Example-based Intent → Program Synthesizer → Program set (a sub-DSL of D) → DSL D
Search Methodology

**Goal**: Set of expr of kind $e$ that satisfies spec $\phi$  
[denoted $[e\models\phi]$]

$e$: DSL (top-level) expression

$\phi$: Conjunction of (input state $\sigma$, output value $\nu$)  
[denoted $\sigma \leftrightarrow \nu$]

**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

\[ [e \models \phi \downarrow 1 \lor \phi \downarrow 2] = \text{Union}( [e \models \phi \downarrow 1 ] , [e \models \phi \downarrow 2 ] ) \]

\[ [e \models \phi \downarrow 1 \land \phi \downarrow 2] = \text{Intersect}( [e \models \phi \downarrow 1 ] , [e \models \phi \downarrow 2 ] ) \]

An alternative strategy:

\[ [e \models \phi \downarrow 1 \land \phi \downarrow 2] = [e' \models \phi \downarrow 2] , \text{ where } e' = \text{TopSymbol}( [e \models \phi \downarrow 1 ] ) \]

Let \( e \) be a non-terminal defined as \( e \coloneqq e \downarrow 1 \mid e \downarrow 2 \)

\[ [e \models \phi] = \text{Union}( [e \downarrow 1 \models \phi] , [e \downarrow 2 \models \phi] ) \]
Inverse Set: Let $F$ be an $n$-ary operator.

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

$Concat^{\uparrow-1}("Abc") = \{("Abc", \varepsilon), ("Ab", "c"), ("A", "bc"), (\varepsilon, "Abc")\}$

$$[F[e_1, \ldots, e_n] \models \sigma \mapsto v] =$$

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

$$[Concat(X,Y) \models (\sigma \mapsto "Abc") ] = \text{Union}\{$$

$$Concat([X \models (\sigma \mapsto "Abc")], [Y \models (\sigma \mapsto \varepsilon)]) ,$$

$$Concat([X \models (\sigma \mapsto "Ab")], [Y \models (\sigma \mapsto "c")]) ,$$

$$Concat([X \models (\sigma \mapsto "A")], [Y \models (\sigma \mapsto "bc")]) ,$$

$$Concat([X \models (\sigma \mapsto \varepsilon)], [Y \models (\sigma \mapsto "Abc")]) )$$

}
Problem Reduction Rules

Inverse Set: Let $F$ be an $n$-ary operator.

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

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

- IfThenElse$^{\uparrow -1}$ ($v$) $= \{ (true, v, T), (false, T, v) \}$

Consider the SubStr(x,i,j) operator

$$SubStr^{\uparrow -1} ("Ab" \mid x="Ab cd Ab") \neq \emptyset, 2), (6,8) \}$$

The notion of inverse sets (and corresponding reduction rules) can be generalized to the conditional setting.
Generalizing output values to output properties

User specification

Examples: Conjunction of (input state, output value)
Inductive Spec: Conjunction of (input state, output property)

Search methodology

(Conditional) Inverse sets: Back-propagation of values
(Conditional) Witness functions: Back-propagation of properties
Problem Reduction

FlashExtract DSL

list of strings $T := \text{Map}(L,S)$

substring fn $S := \lambda y: \ldots$

list of lines $L := \text{Filter}(\text{Split}(d,"\n"), B)$

boolean fn $B := \lambda y: \ldots$

Spec for $T$  $\rightarrow$  Spec for $L$  $\bowtie$  Spec for $S$

Ana Trujillo
357 21th Place SE
Redmond, WA
(757) 555-1634

Antonio Moreno
515 93th Lane
Renton, WA
(411) 555-2786

Thomas Hardy
742 17th Street NE

Ana Trujillo
357 21th Place SE
Redmond, WA
(757) 555-1634

Antonio Moreno
515 93th Lane
Renton, WA
(411) 555-2786

Thomas Hardy
742 17th Street NE

Redmond, WA

∧

Renton, WA
Problem Reduction

SubStr grammar

substring expr \( E := \text{SubStr}(y,P_1,P_2) \)

position expr \( P := K | \text{Pos}(y,R_1,R_2,K) \)

Spec for \( E \) \( \rightarrow \) Spec for \( P_1 \) \( \bowtie \) Spec for \( P_2 \)

Redmond, WA

Redmond, WA

Redmond, WA
Programming-by-Examples Architecture
Basic ranking scheme

Prefer programs with simpler Kolmogorov complexity
- Prefer fewer constants.
- Prefer 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”
Challenges with Basic ranking scheme

Prefer programs with simpler Kolmogorov complexity
- Prefer fewer constants.
- Prefer smaller constants.

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

- \(2^{\text{nd}}\) Word + “,” + \(1^{\text{st}}\) Word
- “Singh, Rishabh”

How to select between Fewer larger constants vs. More smaller constants?

**Idea:** Associate numeric weights with constants.
Challenges with Basic ranking scheme

Prefer programs with simpler Kolmogorov complexity

- Prefer fewer constants.
- Prefer 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
Same number of same-sized constants?

Idea: Examine data features (in addition to program features)
Machine learning based ranking scheme

Rank score of a program: Weighted combination of various features.
- Weights are learned using machine learning.

Program features
- Number of constants
- Size of constants

Features over user data: Similarity of generated output (or even intermediate values) over various user inputs
- IsYear, Numeric Deviation, Number of characters
- IsPersonName

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

Incrementality

Debugging

Translator

Intended Program in R/Python/C#/C++/…

Test inputs

Intended Program in D

Ranking fn

DSL D

Translation

Ranking func
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

- Ranking fn
- DSL D

Refined Intent

Ranked Program set (a sub-DSL of D)

Incrementality

Debugging

Test inputs

Translator

Intended Program in D

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

Ranked Program set (a sub-DSL of D)

Refined Intent

Incrementality

Test inputs

Intended Program in D

Translator

Intended Program in R/Python/C#/C++/…

Interaction history

Learner

Program Synthesizer

Ranking fn

DSL D
https://microsoft.github.io/prose
- Efficient implementation of the generic search methodology.
- Provides a library of reduction rules.

Role of synthesizer developer
- Implement a DSL with executable semantics for new operators.
- Implement reduction rules (inverse semantics) for some operators.
- Implement ranking strategy.
- Can also specify tactics to resolve non-determinism in search.
The PROSE Team

Sumit Gulwani  Ranvijay Kumar  Vu Le  Daniel Perelman  Mark Plesko

Please apply for intern/full-time positions!

Alex Polozov  Mohammad Raza  Danny Simmons  Abhishek Udupa
Future Directions

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

• Killer application: Data wrangling
  – 99% of end users are non-programmers.
  – Data scientists spend 80% time cleaning data.

• Algorithmic search
  – Domain-specific language
  – Divide-and-conquer methodology based on inverse functions

• 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]