CS5733 Program Synthesis

#1. Introduction and Overview

Ashish Mishra

Instructor

Ashish Mishra

- Asst. Professor at CSE
- Before: Postdoc, Purdue CS, Programming Languages Group.
- Even before: IISc, CSA, PhD
- Research Goal: Help programmers write correct and trustworthy software.
- Areas: Programming Languages, Program Verification, Synthesis.

Logistics

- Lecture:
  - When: Tuesdays 2:30 - 3:55 pm, Fridays 4:00 - 5:25 pm
  - Where: C-LH5
- Course Website: Please register on Google Classroom link.
  - https://aegis-iisc.github.io/cs5733/
- Office Hours:
  - After the class OR with appointment a few hours before.

Goals and Activities

1. Understand what program synthesis can do and how
2. Use existing synthesis tools
3. Contribute to synthesis techniques and tools towards a publication in an academic conference

Goal 1: lectures; read and discuss research papers

Goals 2–3: project

Evaluation

- Class Participation : 5%
  - Ask/answer questions in class
  - Participate in discussions on Classroom
- Paper Reviews : 25 %
  - 10 papers
- Midterm : 20 %
- Final Course Project : 50 %
  - Team formed by deadline: 5%
  - 1-page project proposal: 15%
  - Project presentation: 15%
  - Final report: 15%

Paper Reviews

- Due on Wed of weeks 2 onwards, by the end of the day
- First review due next week
    - Posted on the Reading List at least a week before due date
- Reviews submitted via a Google Form: see course page
    - Link posted on Reading List (add this to the page)
- Review content: see course page
- Discussion:
    - before due date: discuss on Google Classroom
    - after due date: discuss in class

Project

- Kinds of projects:
  - Re-implement techniques from a paper
  - Apply existing synthesis framework to a new domain
  - Extend/improve existing synthesis algorithm or tool
  - Develop a new synthesis algorithm or tool …
- Judged in terms of
  - Quality of execution
  - Originality
  - Scope

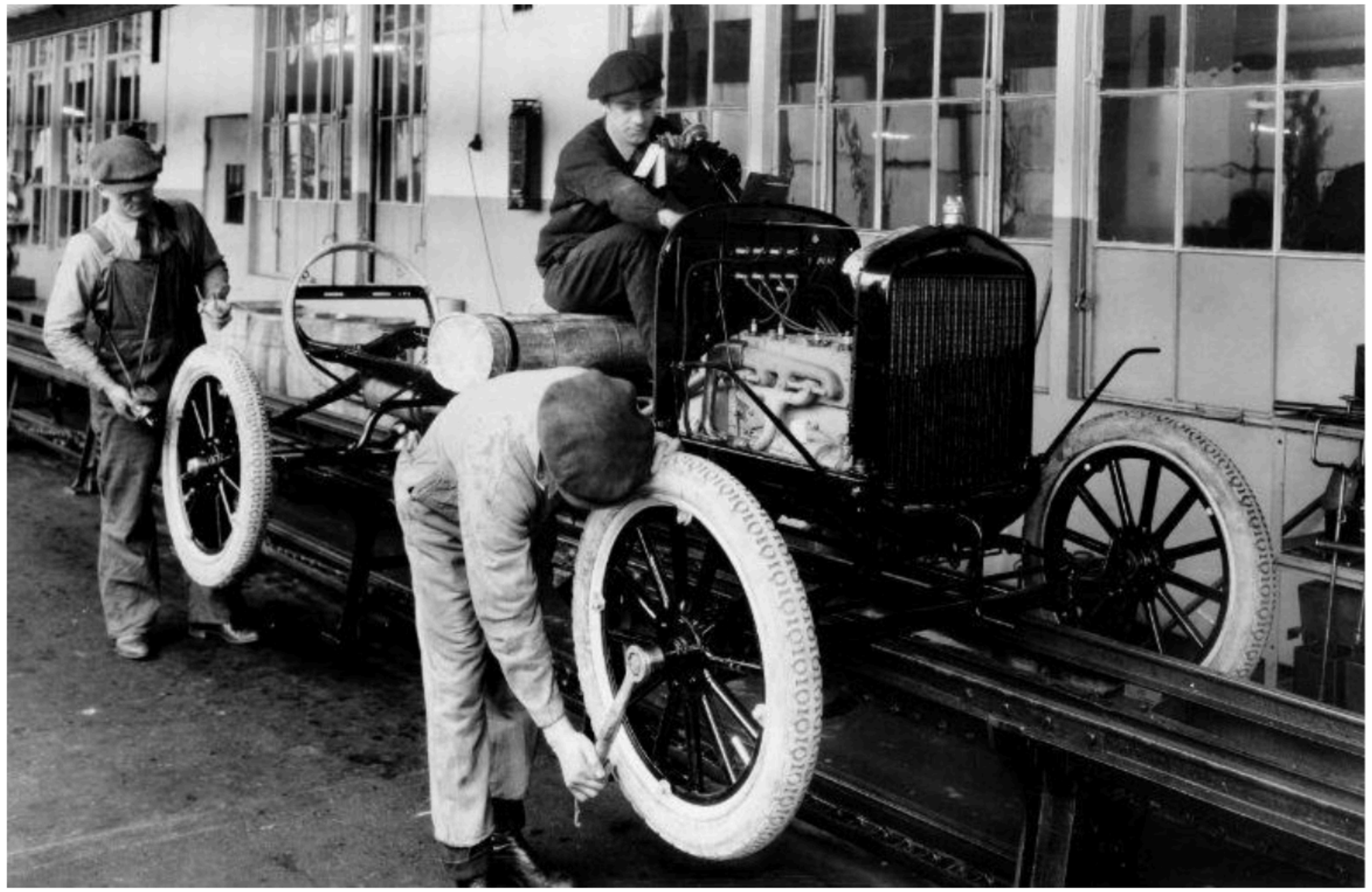
Project

- Team forming
- Proposal
- Presentation
- Report

- Teams of 1-2
- Pick a project:
  - List of suggested projects coming soon on Google Classroom
  - Please talk to me!
- One page: explain what you plan to do and give some evidence that you'vestarted to work on it
- Presentations in last few classes
  - ~10-15 min per project
  - 3-8 pages, structured like a research paper

Lets begin the good stuff...

The goal: Automated Programming



A classic goal

The FORTRAN Automatic Coding System

J. W. BACKUS†, R. J. BEEBER†, S. BEST‡, R. GOLDBERG†, L. M. HAIBT†,
H. L. HERRICK†, R. A. NELSON†, D. SAYRE†, P. B. SHERIDAN†,
H. STERN†, I. ZILLER†, R. A. HUGHES§, AND R. NUTT||

INTRODUCTION

THE FORTRAN project was begun in the summer of 1954. Its purpose was to reduce by a large factor the task of preparing scientific problems for IBM's next large computer, the 704. If it were possible for the 704 to code problems for itself and produce as

system is now complete. It has two components: the FORTRAN language, in which programs are written, and the translator or executive routine for the 704 which effects the translation of FORTRAN language programs into 704 programs. Descriptions of the FORTRAN language and the translator form the principal

On the Synthesis of a Reactive Module

Amir Pnueli and Roni Rosner*
Department of Computer Science
The Weizmann Institute of Science
Rehovot 76100, Israel
amir@wisdom.bitnet, roni@wisdom.bitnet

Abstract

We consider the synthesis of a reactive module with input x and output y, which is specified by the linear temporal formula $\varphi(x, y)$. We show that there exists a program satisfying φ iff the branching time formula $(\forall x)(\exists y)A\varphi(x, y)$ is valid over all tree models. For the restricted case that all variables range over finite domains, the validity problem is decidable, and we present an algorithm for constructing the program with input x and output y, specified by the formula $\varphi(x, y)$, is constructed as a by-product of proving the theorem $(\forall x)(\exists y)\varphi(x, y)$. The specification $\varphi(x, y)$ characterizes the expected relation between the input x presented to the program and the output y computed by the program. For example, the specification for a root extracting program may be presented by the formula $|x - y^2| < \epsilon$.

This approach, which may be called the *AE*-

IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. SE-5, NO. 4, JULY 1979

Synthesis: Dreams ⟹ Programs

ZOHAR MANNA AND RICHARD WALDINGER

techniques are presented for deriving programs iven specifications. The specifications express the d program without giving any hint of the algo-. The basic approach is to transform the specifi-cording to certain rules, until a satisfactory pro-

INTRODUCTION

IN RECENT years there has been increasing activity in the field of program verification. The goal of these efforts is to construct computer systems for determining whether a

What is Program Synthesis

- Automatically finding programs
  - from the underlying programming language/set of components.
  - satisfying the user intent, expressed using some constraints.

This sounds familiar: Compiler/Logic Prog/ML

- Compilers:
  - Fully specified High-Level code → Low-level machine rep.
  - Syntax-directed translation.

Synthesis: discover *how* to perform the desired task. Some notion of *Search*

- Logic Programming
  - dream: express the requirements in a logical form.
  - generic algorithm for all problems.
- ML
  - Find a function/learned model, whose behavior closely matches the dataset.
  - the space of functions that the algorithm considers is very tightly prescribed
    - linear classifiers, decision trees and neural networks

Synthesis: general algorithms for general classes of programs, that support recursion or other forms of iteration

```
append:
    push ebp
    mov ebp, esp
    push eax
    push ebx
    push len
    call malloc
    mov ebx, [ebp + 12]
    mov [eax + info], ebx
    mov dword [eax + next], 0
    mov ebx, [ebp + 8]
    cmp dword [ebx], 0
    je null_pointer
    mov ebx, [ebx]

next_element:
    cmp dword [ebx + next], 0
    je found_last
    mov ebx, [ebx + next]
    jmp next_element

found_last:
    push eax
    push addMes
    call puts
    add esp, 4
    pop eax
    mov [ebx + next], eax

go_out:
    pop ebx
    pop eax
    mov esp, ebp
    pop ebp
    ret 8

null_pointer:
    push eax
    push nullMes
    call puts
    add esp, 4
    pop eax
    mov [ebx], eax
    jmp go_out
```

Assembly

```
void insert(node *xs, int x) {
  node *new;
  node *temp;
  node *prev;

  new = (node *)malloc(sizeof(node));
  if(new == NULL) {
    printf("Insufficient memory.");
    return;
  }
  new->val = x;
  new->next = NULL;
  if (xs == NULL) {
    xs = new;
  } else if(x < xs->val) {
    new->next = xs;
    xs = new;
  } else {
    prev = xs;
    temp = xs->next;
    while(temp != NULL && x > temp->val) {
      prev = temp;
      temp = temp->next;
    }
    if(temp == NULL) {
      prev->next = new;
    } else {
      new->next = temp;
      prev->next = new;
    }
  }
}
```

C

```
insert x []      = [x]
insert x (y:ys)
  | x ≤ y        = x:y:ys
  | otherwise    = y:(insert x ys)
```

Haskell

“Any sufficiently advanced compiler is indistinguishable from a synthesizer”

?

modern program synthesis

Synthesis: Example

Library/Language

library

sort: list -> list

reverse: list -> list

take: list -> int -> list

sum: list -> int

User Intent

query

best_ksum: (l : list) -> (k : int) -> int

i/o examples

49	62	82	54	76

→ **k=2** → 158

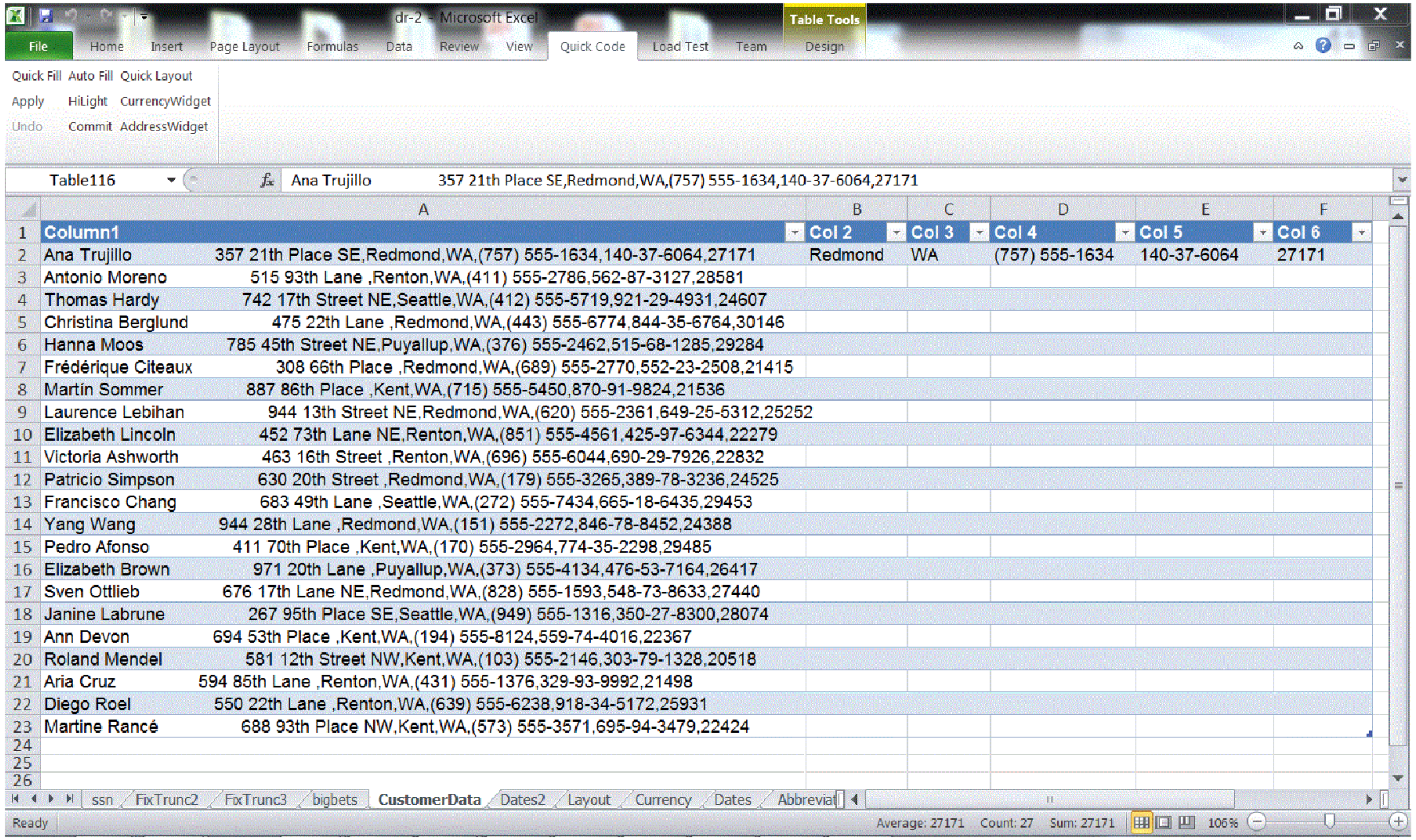
best_ksum l k = sum (take (reverse (sort l)) k)

Modern Program Synthesis: FlashFill

[Gulwani 2011]

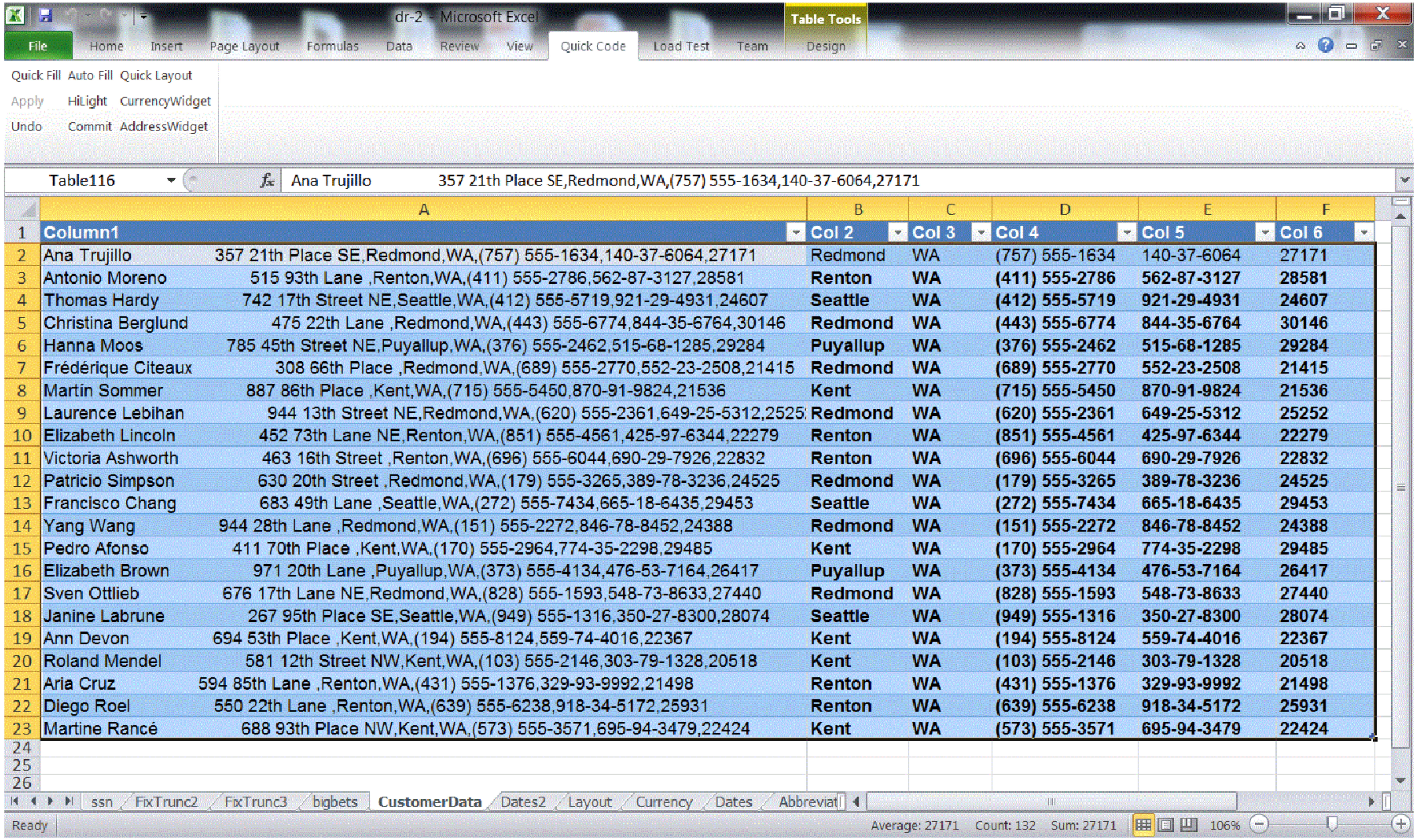
FlashFill : A Feature of Excel 2013

[Gulwani 2011]



	A	B	C	D	E	F
1	Column1	Col 2	Col 3	Col 4	Col 5	Col 6
2	Ana Trujillo 357 21th Place SE,Redmond,WA,(757) 555-1634,140-37-6064,27171	Redmond	WA	(757) 555-1634	140-37-6064	27171
3	Antonio Moreno 515 93th Lane ,Renton,WA,(411) 555-2786,562-87-3127,28581					
4	Thomas Hardy 742 17th Street NE,Seattle,WA,(412) 555-5719,921-29-4931,24607					
5	Christina Berglund 475 22th Lane ,Redmond,WA,(443) 555-6774,844-35-6764,30146					
6	Hanna Moos 785 45th Street NE,Puyallup,WA,(376) 555-2462,515-68-1285,29284					
7	Frédérique Citeaux 308 66th Place ,Redmond,WA,(689) 555-2770,552-23-2508,21415					
8	Martín Sommer 887 86th Place ,Kent,WA,(715) 555-5450,870-91-9824,21536					
9	Laurence Lebihan 944 13th Street NE,Redmond,WA,(620) 555-2361,649-25-5312,25252					
10	Elizabeth Lincoln 452 73th Lane NE,Renton,WA,(851) 555-4561,425-97-6344,22279					
11	Victoria Ashworth 463 16th Street ,Renton,WA,(696) 555-6044,690-29-7926,22832					
12	Patricio Simpson 630 20th Street ,Redmond,WA,(179) 555-3265,389-78-3236,24525					
13	Francisco Chang 683 49th Lane ,Seattle,WA,(272) 555-7434,665-18-6435,29453					
14	Yang Wang 944 28th Lane ,Redmond,WA,(151) 555-2272,846-78-8452,24388					
15	Pedro Afonso 411 70th Place ,Kent,WA,(170) 555-2964,774-35-2298,29485					
16	Elizabeth Brown 971 20th Lane ,Puyallup,WA,(373) 555-4134,476-53-7164,26417					
17	Sven Ottlieb 676 17th Lane NE,Redmond,WA,(828) 555-1593,548-73-8633,27440					
18	Janine Labrune 267 95th Place SE,Seattle,WA,(949) 555-1316,350-27-8300,28074					
19	Ann Devon 694 53th Place ,Kent,WA,(194) 555-8124,559-74-4016,22367					
20	Roland Mendel 581 12th Street NW,Kent,WA,(103) 555-2146,303-79-1328,20518					
21	Aria Cruz 594 85th Lane ,Renton,WA,(431) 555-1376,329-93-9992,21498					
22	Diego Roel 550 22th Lane ,Renton,WA,(639) 555-6238,918-34-5172,25931					
23	Martine Rancé 688 93th Place NW,Kent,WA,(573) 555-3571,695-94-3479,22424					
24						
25						
26						

FlashFill : A Feature of Excel 2013



	Column1	Col 2	Col 3	Col 4	Col 5	Col 6
2	Ana Trujillo 357 21th Place SE,Redmond,WA,(757) 555-1634,140-37-6064,27171	Redmond	WA	(757) 555-1634	140-37-6064	27171
3	Antonio Moreno 515 93th Lane ,Renton,WA,(411) 555-2786,562-87-3127,28581	Renton	WA	(411) 555-2786	562-87-3127	28581
4	Thomas Hardy 742 17th Street NE,Seattle,WA,(412) 555-5719,921-29-4931,24607	Seattle	WA	(412) 555-5719	921-29-4931	24607
5	Christina Berglund 475 22th Lane ,Redmond,WA,(443) 555-6774,844-35-6764,30146	Redmond	WA	(443) 555-6774	844-35-6764	30146
6	Hanna Moos 785 45th Street NE,Puyallup,WA,(376) 555-2462,515-68-1285,29284	Puyallup	WA	(376) 555-2462	515-68-1285	29284
7	Frédérique Citeaux 308 66th Place ,Redmond,WA,(689) 555-2770,552-23-2508,21415	Redmond	WA	(689) 555-2770	552-23-2508	21415
8	Martín Sommer 887 86th Place ,Kent,WA,(715) 555-5450,870-91-9824,21536	Kent	WA	(715) 555-5450	870-91-9824	21536
9	Laurence Lebihan 944 13th Street NE,Redmond,WA,(620) 555-2361,649-25-5312,2525	Redmond	WA	(620) 555-2361	649-25-5312	25252
10	Elizabeth Lincoln 452 73th Lane NE,Renton,WA,(851) 555-4561,425-97-6344,22279	Renton	WA	(851) 555-4561	425-97-6344	22279
11	Victoria Ashworth 463 16th Street ,Renton,WA,(696) 555-6044,690-29-7926,22832	Renton	WA	(696) 555-6044	690-29-7926	22832
12	Patricio Simpson 630 20th Street ,Redmond,WA,(179) 555-3265,389-78-3236,24525	Redmond	WA	(179) 555-3265	389-78-3236	24525
13	Francisco Chang 683 49th Lane ,Seattle,WA,(272) 555-7434,665-18-6435,29453	Seattle	WA	(272) 555-7434	665-18-6435	29453
14	Yang Wang 944 28th Lane ,Redmond,WA,(151) 555-2272,846-78-8452,24388	Redmond	WA	(151) 555-2272	846-78-8452	24388
15	Pedro Afonso 411 70th Place ,Kent,WA,(170) 555-2964,774-35-2298,29485	Kent	WA	(170) 555-2964	774-35-2298	29485
16	Elizabeth Brown 971 20th Lane ,Puyallup,WA,(373) 555-4134,476-53-7164,26417	Puyallup	WA	(373) 555-4134	476-53-7164	26417
17	Sven Ottlieb 676 17th Lane NE,Redmond,WA,(828) 555-1593,548-73-8633,27440	Redmond	WA	(828) 555-1593	548-73-8633	27440
18	Janine Labrune 267 95th Place SE,Seattle,WA,(949) 555-1316,350-27-8300,28074	Seattle	WA	(949) 555-1316	350-27-8300	28074
19	Ann Devon 694 53th Place ,Kent,WA,(194) 555-8124,559-74-4016,22367	Kent	WA	(194) 555-8124	559-74-4016	22367
20	Roland Mendel 581 12th Street NW,Kent,WA,(103) 555-2146,303-79-1328,20518	Kent	WA	(103) 555-2146	303-79-1328	20518
21	Aria Cruz 594 85th Lane ,Renton,WA,(431) 555-1376,329-93-9992,21498	Renton	WA	(431) 555-1376	329-93-9992	21498
22	Diego Roel 550 22th Lane ,Renton,WA,(639) 555-6238,918-34-5172,25931	Renton	WA	(639) 555-6238	918-34-5172	25931
23	Martine Rancé 688 93th Place NW,Kent,WA,(573) 555-3571,695-94-3479,22424	Kent	WA	(573) 555-3571	695-94-3479	22424

Under the hood

The Story of the Flash Fill Feature in Excel

by Sumit Gulwani on Sep 14, 2021 | Tags: Flash Fill, MIP award, program synthesis, Programming by Examples

Automating String Processing in Spreadsheets Using Input-Output Examples

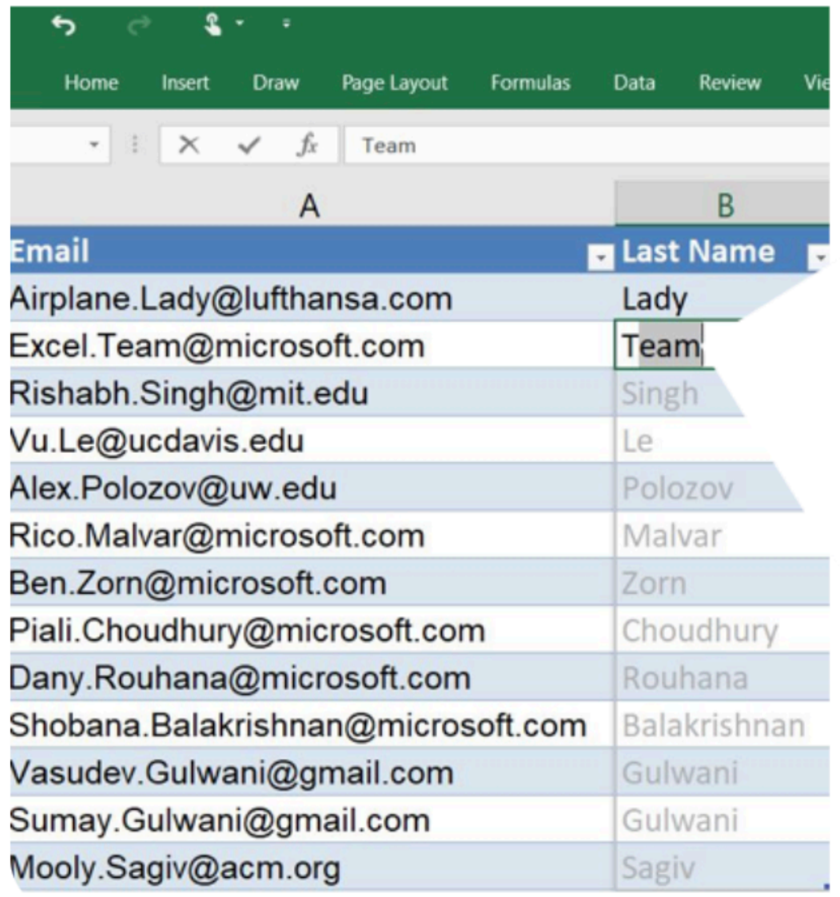
Sumit Gulwani
Microsoft Research, Redmond, WA, USA
sumitg@microsoft.com

Abstract

We describe the design of a string programming/expression language that supports restricted forms of regular expressions, conditionals and loops. The language is expressive enough to represent a wide variety of string manipulation tasks that end-users struggle with. We describe an algorithm based on several novel concepts for

their task [9]. More significantly, programmi
perform tedious and repetitive tasks such as t
like names/phone-numbers/dates from one fo
cleansing, extracting data from several text fil
a single document, etc. Spreadsheet systems
allow users to write macros using a rich inbuil
numerical functions, or to write arbitrary scri

Excel 2013's coolest new feature that
hould have been available years ago"



FlashFill++: Scaling Programming by Example by Cutting to the Chase

JOSÉ CAMBRONERO*, Microsoft, USA
SUMIT GULWANI*, Microsoft, USA
VU LE*, Microsoft, USA

Major Idea is VSAs

Modern Program Synthesis: Sketch [Solar-Lezama 2013]

- Problem: isolate the least significant zero bit in a word
  - example: 0010 0101 → 0000 0010
- Easy to implement with a loop

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

- Can this be done more efficiently with bit manipulation?
  - Trick: adding 1 to a string of ones turns the next zero to a 1
  - i.e. 000111 + 1 = 001000

Sketch: space of possible implementations

```
/**
 * Generate the set of all bit-vector expressions
 * involving +, &, xor and bitwise negation (~).
*/
```

Missing constants!

```
generator bit[W] gen(bit[W] x){
    if(??) return x;
    if(??) return ??;
    if(??) return ~gen(x);
    if(??){
        return {| gen(x) (+ | & | ^) gen(x) |};
    }
}
```

Sketch Idea : program space as a parametric program P[c]

Different values of c gives different program in the space

translate requirements on the behavior of the program P[c] into constraints on the parameters c.

Any value of c that satisfies the constraints is guaranteed to lead to a program satisfying all the requirements.

Sketch: synthesis goal

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

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

Sketch: output

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

~0010 0101
= 1101 1010

0010 0101 + 1
= 0010 0110

&

0000 0010

Modern Program Synthesis: Synquid

[Polikarpova et al. 2016]

Problem: intersection of sets represented as strictly sorted lists

- example: intersect [4, 8, 15, 16, 23, 42] [8, 16, 32, 64] → [8, 16]

Also: we want a guarantee that it's correct on all inputs!

Synquid: synthesis goal and components

Step 1: define synthesis goal as a *type*

```
intersect :: xs:List a  →  ys:List a  →
                 List a
```

sorted list

the set of elements

Step 2: define a set of components

- Which primitive operations is our function likely to use?
- Here: {`Nil`, `Cons`, `<`}

Synquid Output

```
intersection = \xs . \ys .
  match xs with
    Nil -> xs
    Cons x xt ->
      match ys with
        Nil -> ys
        Cons y yt ->
          if x < y
          then intersection xt ys
          else
            if y < x
            then intersection xs yt
            else Cons x (intersection xt yt)
```

xs	ys	result
[4, 8, 15, 16, 23, 42]	[8, 16, 32, 64]	
[8, 15, 16, 23, 42]	[8, 16, 32, 64]	[8]
[15, 16, 23, 42]	[16, 32, 64]	
[16, 23, 42]	[16, 32, 64]	[8, 16]
[23, 42]	[32, 64]	
[42]	[32, 64]	
[42]	[64]	
[]	[64]	

Modern Program Synthesis: GitHub Copilot

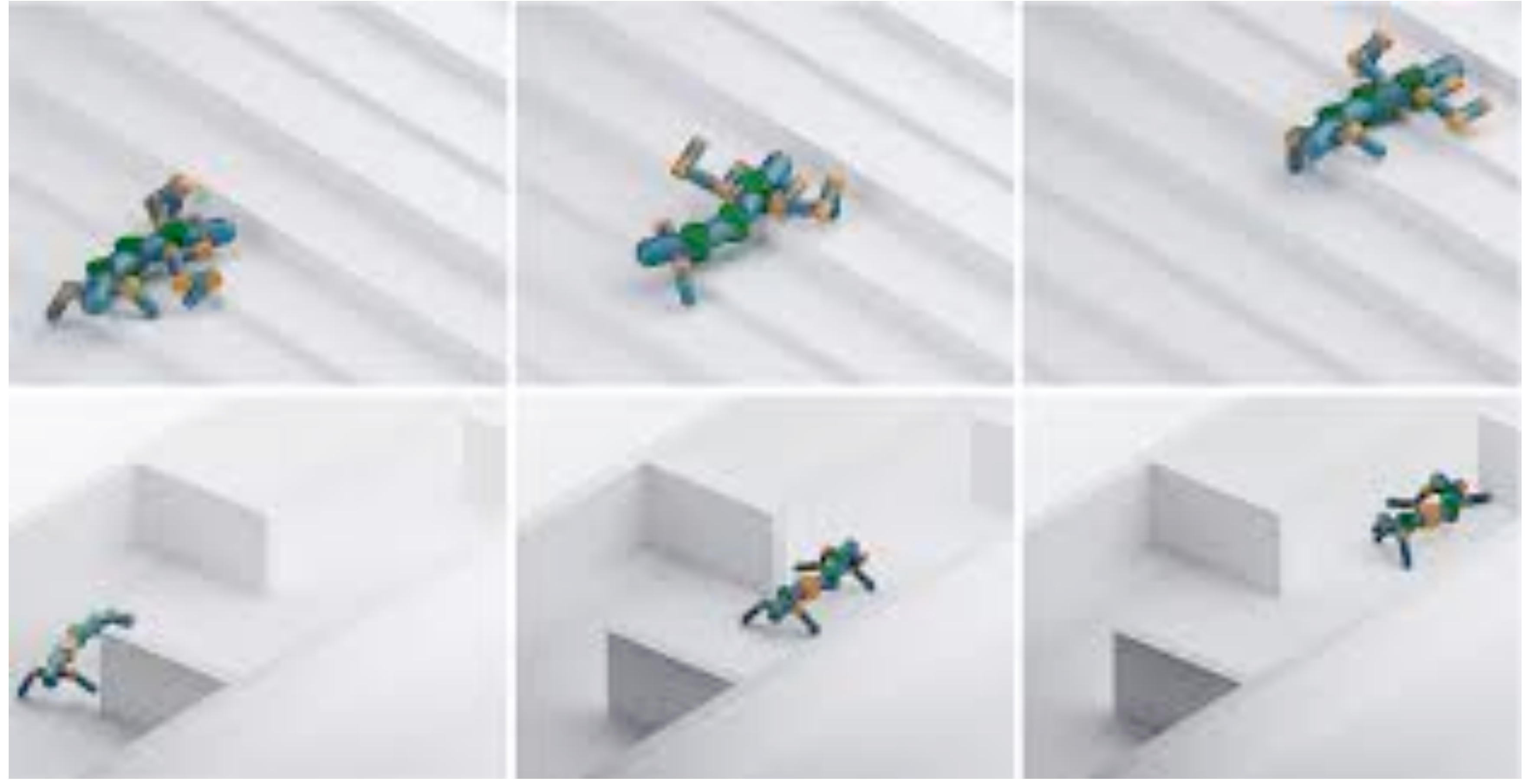
```
// find all images
// and add a green border around them
// and add class “githubCopilot” to them
  function go() {

    var images = document.getElementByTagName(‘img’);
    for (var i = 0; i < images.length; i++) {
      if (images[i].className.indexOf(‘githubCopilot’) == -1) {
        images[i].className += ‘ githubCopilot’;
        images[i].style.border = ‘1px solid green’;
      }
    }
```

input

output

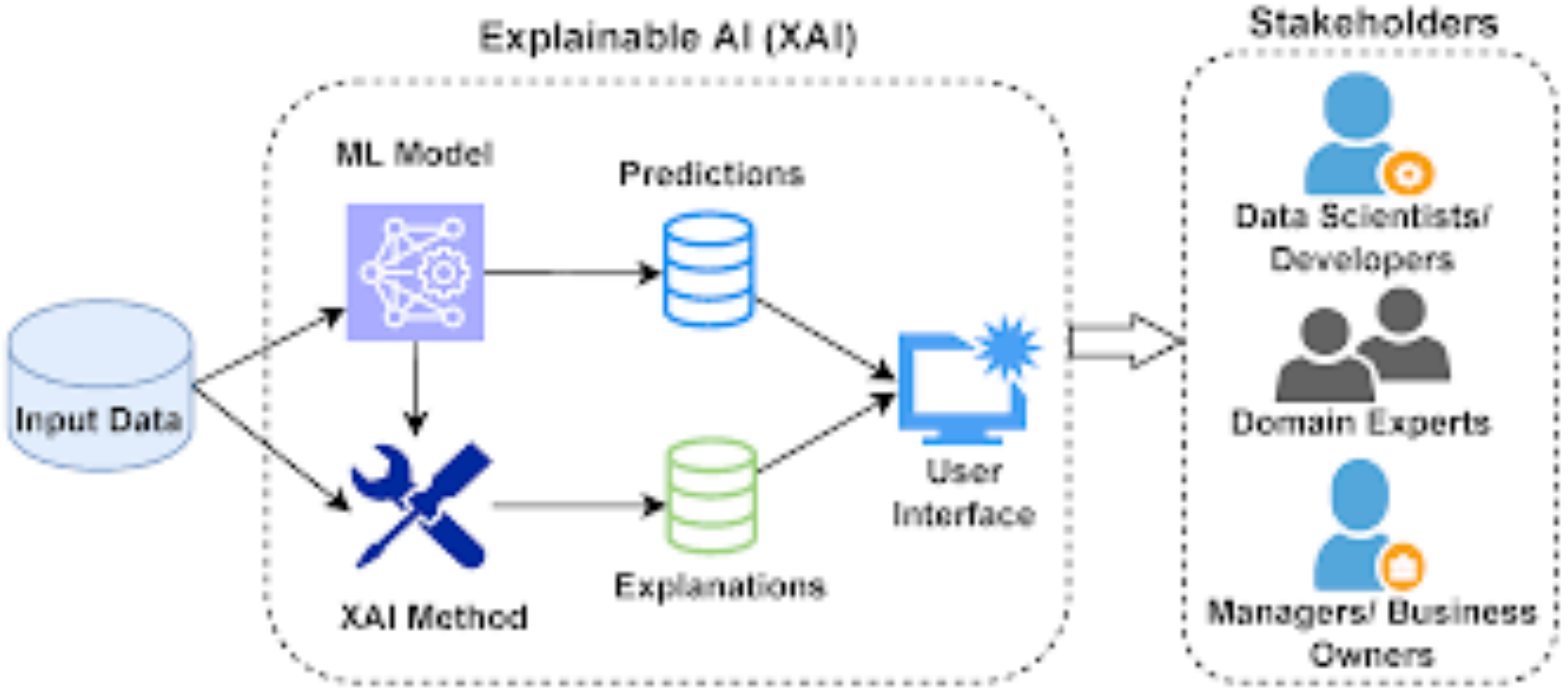
Other Program Synthesis Successes

Robotics design and path planning



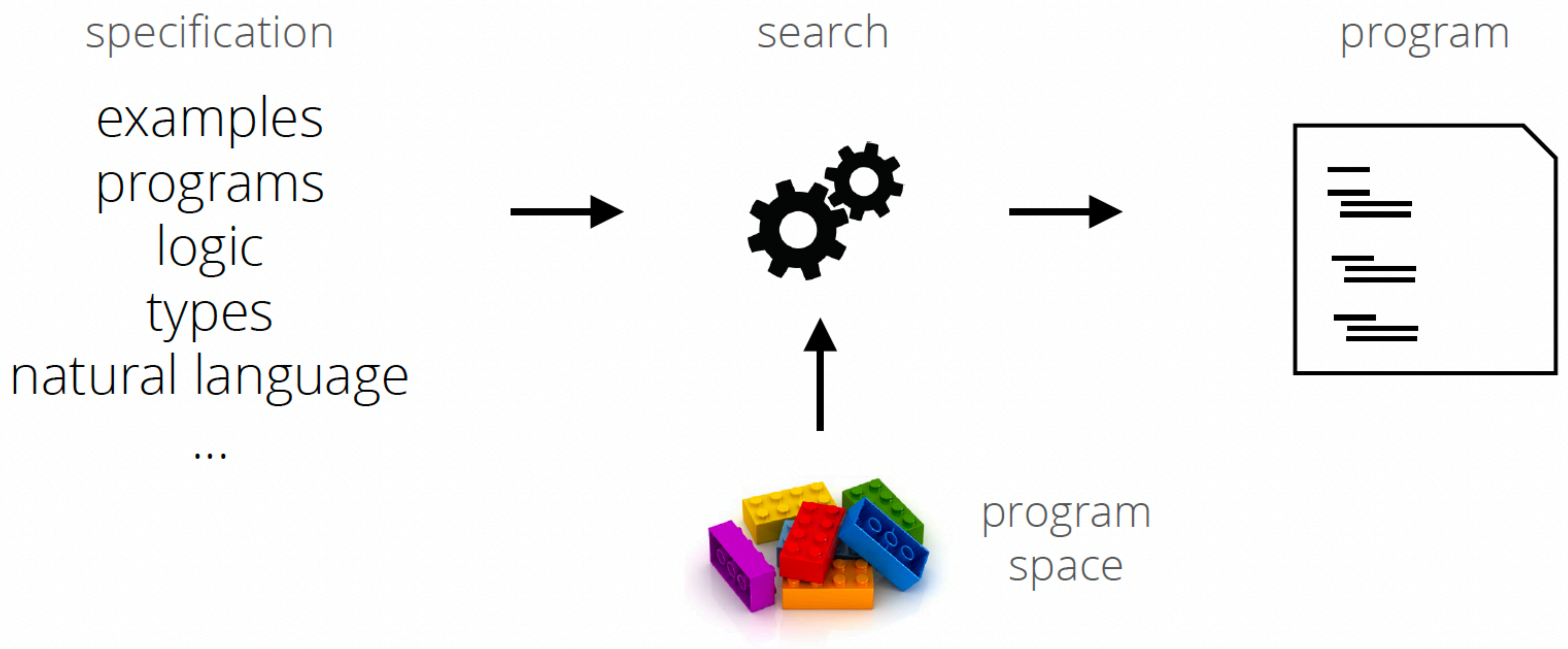
Data Migration

Explainable AI (XAI)



Clement, T. et. al. 2023

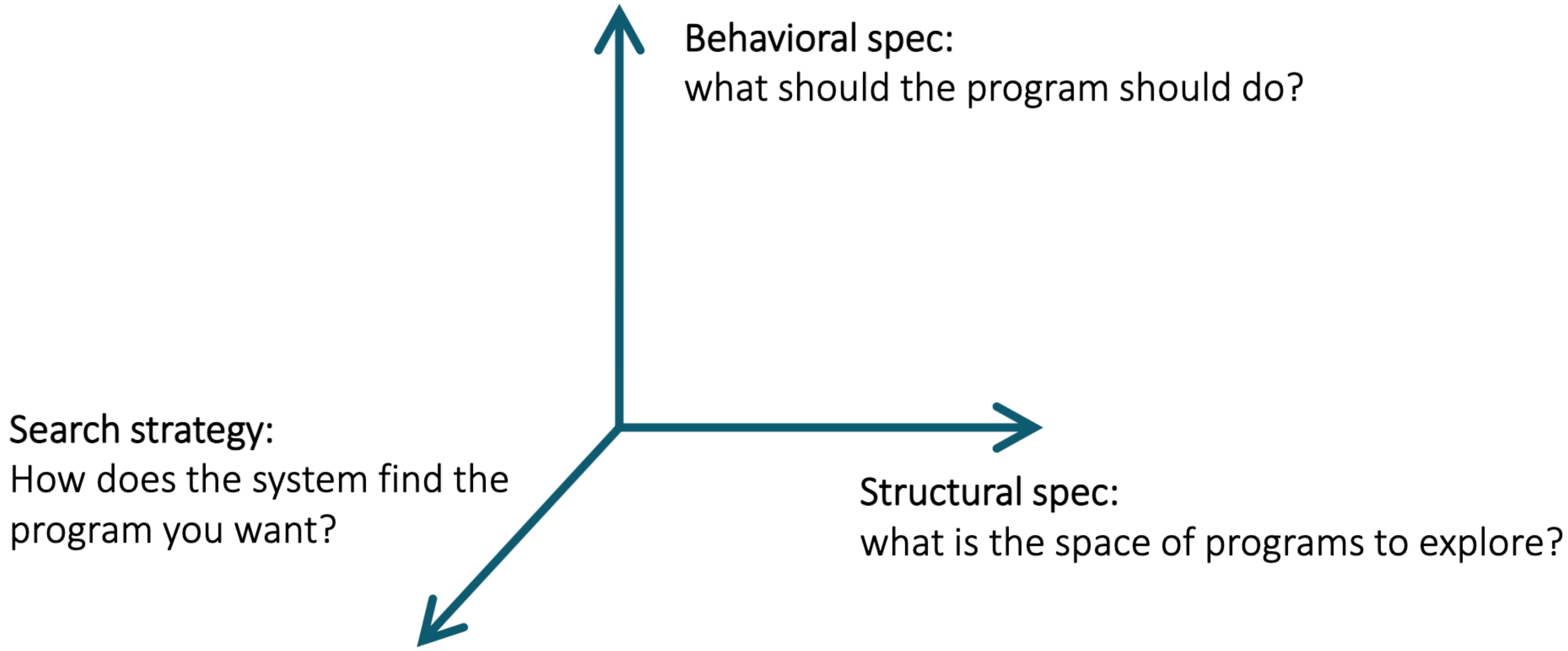
Program Synthesis



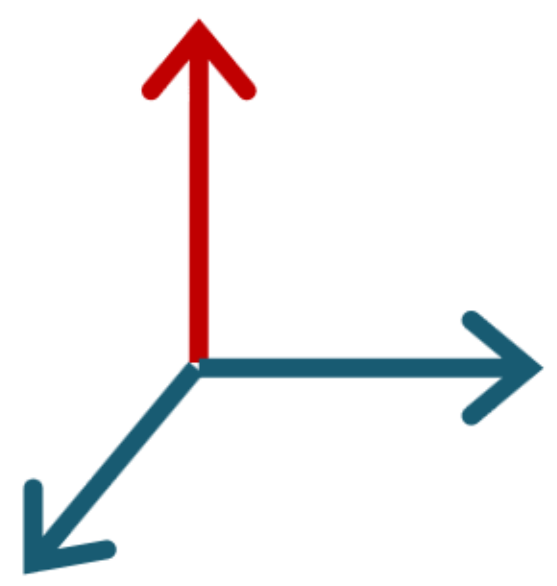
Notice the Duality with Program Verification

Dimensions in Program Synthesis

[Gulwani 2010]



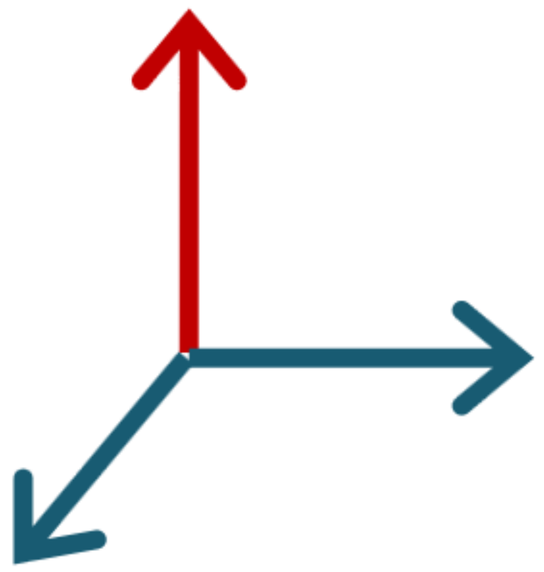
Behavioral Spec



- How do you tell the system what the program should do?
  - What is the input language / format?
  - What is the interaction model?
  - What happens when the intent is ambiguous?

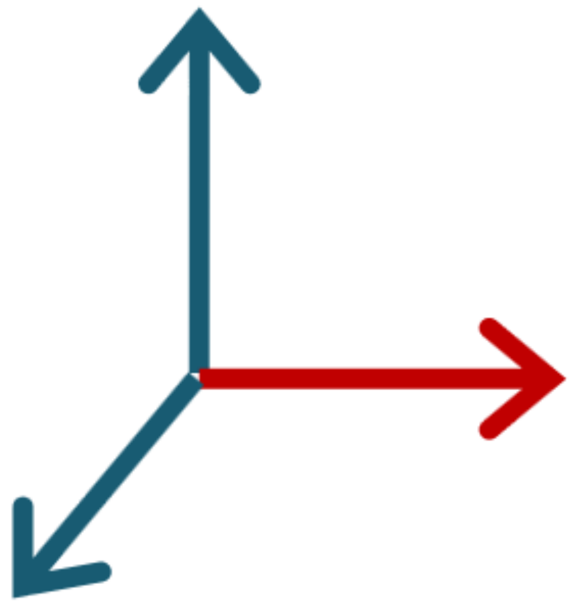
Q: What did the behavioral spec look like in FlashFill / Sketch / Synquid / Copilot?

Behavioral Spec: Examples



- Input/output examples
- Reference implementation
- Formal specifications (pre/post conditions, types, ...)
- Natural language
- Context

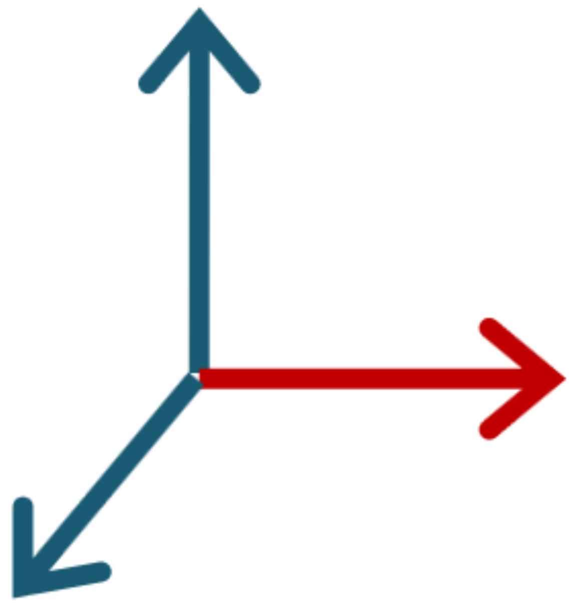
Structural Spec



- What is the space of programs to explore?
  - Large enough to contain interesting programs, yet small enough to exclude garbage and enable efficient search
  - Built-in or user-defined or learned from existing code?

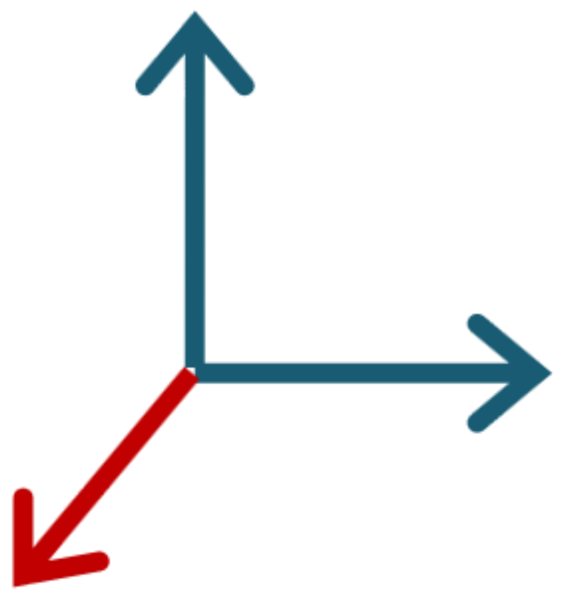
Q: What did the structural spec look like in FlashFill / Sketch / Synquid / Copilot?

Structural Spec: Examples



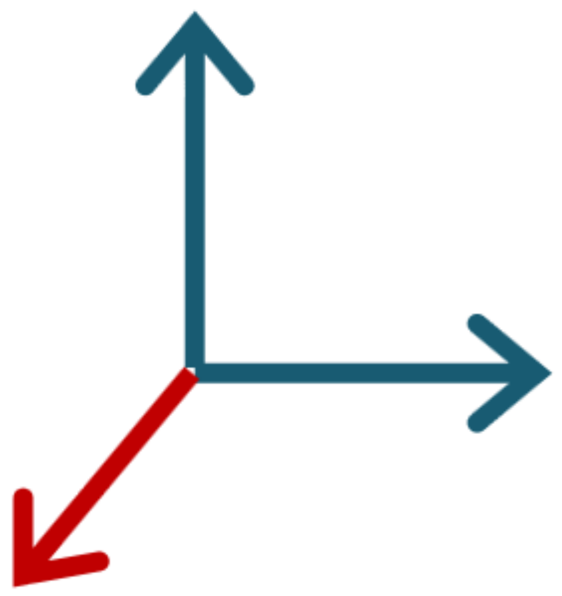
- Built-in DSLs — e.g. DSL for mathematical expressions/ Excel
- User-defined DSL (grammar)
- User-provided components.   — Component-based synthesis —-
- Languages with synthesis constructs
  - e.g. generators in Sketch
- Learned language model

Search Strategies



- Synthesis is search:
  - Find a program in the space defined by structural constraints that satisfies behavioral constraints
- Challenge: the space is astronomically large
  - The search algorithm is the heart of a synthesis technique
- How does the system find the program you want?
  - How does it know it's the program you want?
  - How can it leverage structural constraints to guide the search?
  - How can it leverage behavioral constraints to guide the search?

Search Strategies: Examples



- Enumerative (explicit) search
  - exhaustively enumerate all programs in the language in the order of increasing size
- Deductive
  - Top-down search with recursive reduction of problem to smaller ones.
- Stochastic and Statistical search
  - random exploration of the search space guided by a fitness function
- Representation-based search
  - use a data structure to represent a large set of programs
- Constraint-based search
  - translate to constraints and use a solve

Applications

[Gulwani 2010]

- Data Wrangling
  - Transformations
    - Syntactic String Transformations

      FirstName.Lastname@domain → Firstname Lastname

    - Semantic Transformations

      selling price =

      f (Name, Selling date, MarkupRec, CostRec)

    - Splitting Table Transformations
    - Extractions
    - Layouts

	A	B
1	Email	Column 2
2	Nancy.FreeHafer@fourthcoffee.com	nancy freehafer
3	Andrew.Cencici@northwindtraders.com	andrew cencici
4	Jan.Kotas@litwareinc.com	jan kotas
5	Mariya.Sergienko@gradicdesigninstitute.com	mariya sergienko
6	Steven.Thorpe@northwindtraders.com	steven thorpe
7	Michael.Neipper@northwindtraders.com	michael neipper
8	Robert.Zare@northwindtraders.com	robert zare
9	Laura.Giussani@adventure-works.com	laura giussani
10	Anne.HL@northwindtraders.com	anne hl
11	Alexander.David@contoso.com	alexander david
12	Kim.Shane@northwindtraders.com	kim shane
13	Manish.Chopra@northwindtraders.com	manish chopra
14	Gerwald.Oberleitner@northwindtraders.com	gerwald oberleitner
15	Amr.Zaki@northwindtraders.com	amr zaki
16	Yvonne.McKay@northwindtraders.com	yvonne mckay
17	Amanda.Pinto@northwindtraders.com	amanda pinto

Input v_1	**Input v_2**	**Output**
Stroller	10/12/2010	\$145.67+0.30*145.67
Bib	23/12/2010	\$3.56+0.45*3.56
Diapers	21/1/2011	\$21.45+0.35*21.45
Wipes	2/4/2009	\$5.12+0.40*5.12
Aspirator	23/2/2010	\$2.56+0.30*2.56

MarkupRec

Id	**Name**	**Markup**
S30	Stroller	30%
B56	Bib	45%
D32	Diapers	35%
W98	Wipes	40%
A46	Aspirator	30%
...	...	...

CostRec

Id	**Date**	**Price**
S30	12/2010	\$145.67
S30	11/2010	\$142.38
B56	12/2010	\$3.56
D32	1/2011	\$21.45
W98	4/2009	\$5.12
A46	2/2010	\$2.56
...	...	...

More Applications (Many of these will be part of the Projects)

- Graphics Programming
- Code Repair
- Code Suggestions
- Synthesizing Error-prone, hard to write programs:
  - Distributed Programming : CRDTs
  - Concurrent Programs:
- Modeling of Systems
  - Probabilistic Programs

Structure of the course

- Module 1: Synthesis of Simple Programs
  - Easy to decide when a program is correct
  - Challenge: search in a large space
- Module 2: Synthesis of Complex Programs
  - Deciding when a program is correct can be hard
  - Search in a large space is still a problem
- Module 3: Advance Topics
  - Neural Synthesis,Neural + Symbolic Approaches, Synthesis for xAI
  - Synthesis + X (X $\in$ {Frameworks, Compilers, Network, Databases, etc.})

Reading Weeks 1 and 2

- Topic: Enumerative synthesis from examples
  - Paper: Alur, Radhakrishna, Udupa. Scaling Enumerative Program Synthesis via Divide and Conquer
  - Review due Wednesday
  - Link to PDF on the course web page
- Submit through Google Form (link will be in webpage/Classroom)
- Project:
  - Teams due in two weeks.
  - Submit through a Google Sheet (check email for invite and instructions)

Announcements

- Non-CS students
- Registration on ERP as well as Google Classroom
- Start looking for Projects
- Plagiarism Policy.
  - A bit different.