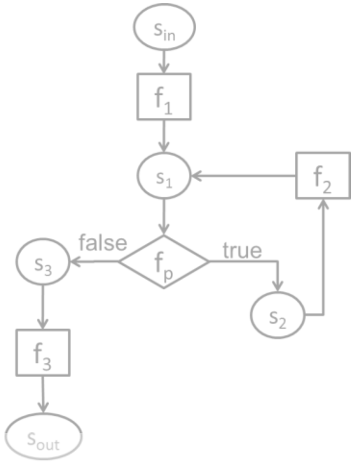
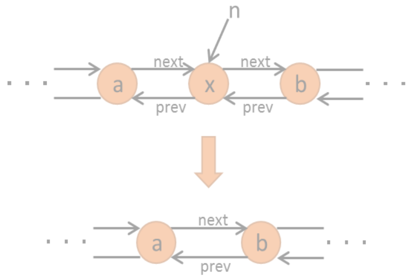
$$\exists c \forall in\ Q(c, in)$$

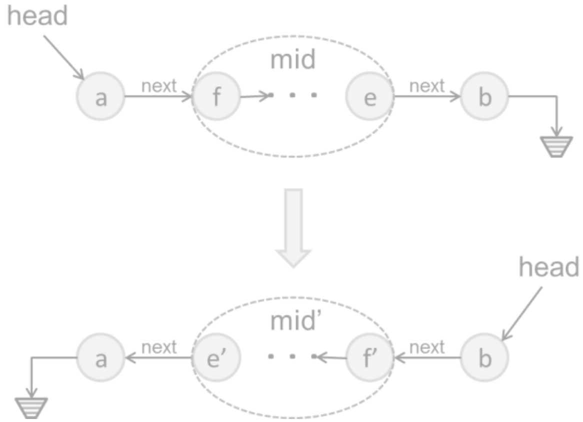
```c
/* Average of x and y without using x+y (avoid overflow)*/
int avg(int x, int y){
  int t = expr({x/2, y/2, x%2, y%2, 2 }, {PLUS, DIV});
  assert t == (x+y)/2;
  return t;
}
```

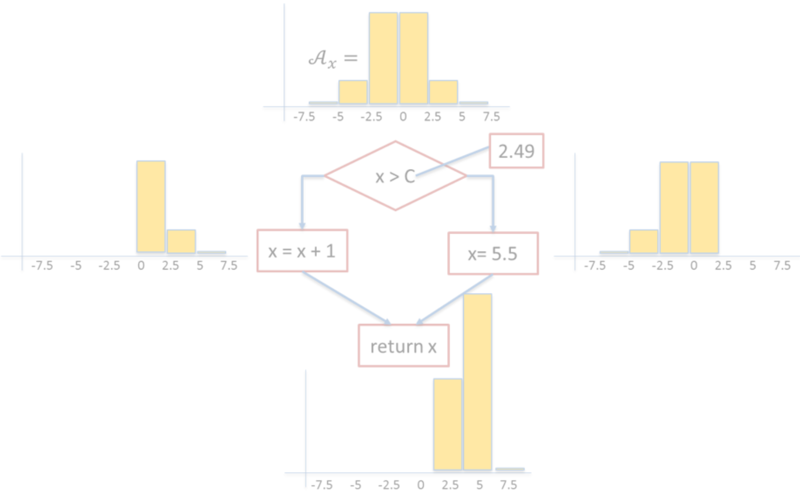





```
{
  s = n.succ;
  p = n.pred;
  p.succ = s;
  s.pred = p;
}
```

Program Synthesis





$$\varphi(p)$$

$$Sk[c](in)$$

Lecture 1
Course Overview and Introduction to Synthesis

Instructor

Loris D'Antoni

- Associate Professor
- Before that:
  - Faculty at UW-Madison for 9 years
  - PhD at Upenn
- Research areas:
  - program synthesis and program verification
  - Large language models for code
- he / him
- This course (and the book I'm sort of working on) is codesigned with Nadia Polikarpova (UCSD).

Logistics

Lecture

- When: Tue/Thu 11230-150
- Where: EBU3B 2154

Office Hours

- When: Thu after class (1230-130)
- Where: CSE 3214

Course Website

- https://github.com/lorisdanto/cse291-program-synthesis-loris/
- Discussions: on Slack

Goals and activities

1. Understand what program verification/synthesis can do and how

2. Use existing tools

3. Contribute to existing techniques and tools towards a publication in an academic conference

lectures
read and discuss research papers

project

Evaluation

Class Participation: 5%

- answer questions in class
- participate in paper discussion on Slack

Paper reviews: 45%

- 9 papers, 5% each

Final Project: 50%

- Team formed by deadline: 5%
- 1-page project proposal: 15%
- Project presentation: 15%
- Final report: 15%

Papers reviews

Due on dates set on Canvas

Posted on the Reading List at least a week before due date

Reviews submitted via Canvas

Review content: see wiki

Project

Kinds of projects:

- re-implement a technique 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

Teams of 1/2/3

Pick a project:

- List of suggested projects on the wiki (but feel free to propose your own)
- Talk to me!

Proposal

One page: explain what you plan to do and give some evidence that you've started to work on it

Presentation

Last days of class

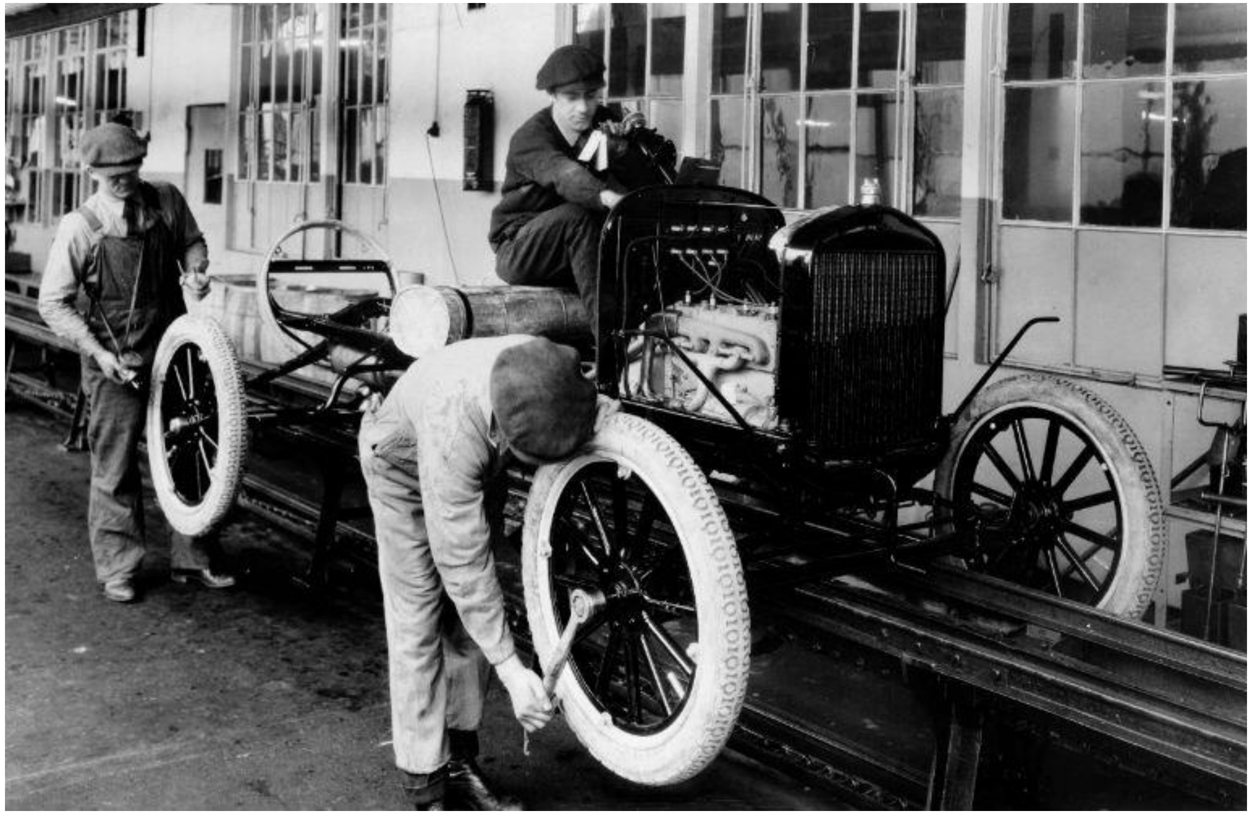
- ~10 min per project

Report

3-8 pages, structured like a research paper

And now the good stuff

The goal: automate programming



What is program synthesis?

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

```
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
```

```
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;
    }
  }
}
```

```
insert x xs =
  match xs with
    Nil → Cons x Nil
    Cons h t →
      if x ≤ h
        then Cons x xs
        else Cons h (insert x t)
```

?

"Any sufficiently advanced compiler is indistinguishable from a synthesizer"

Assembly

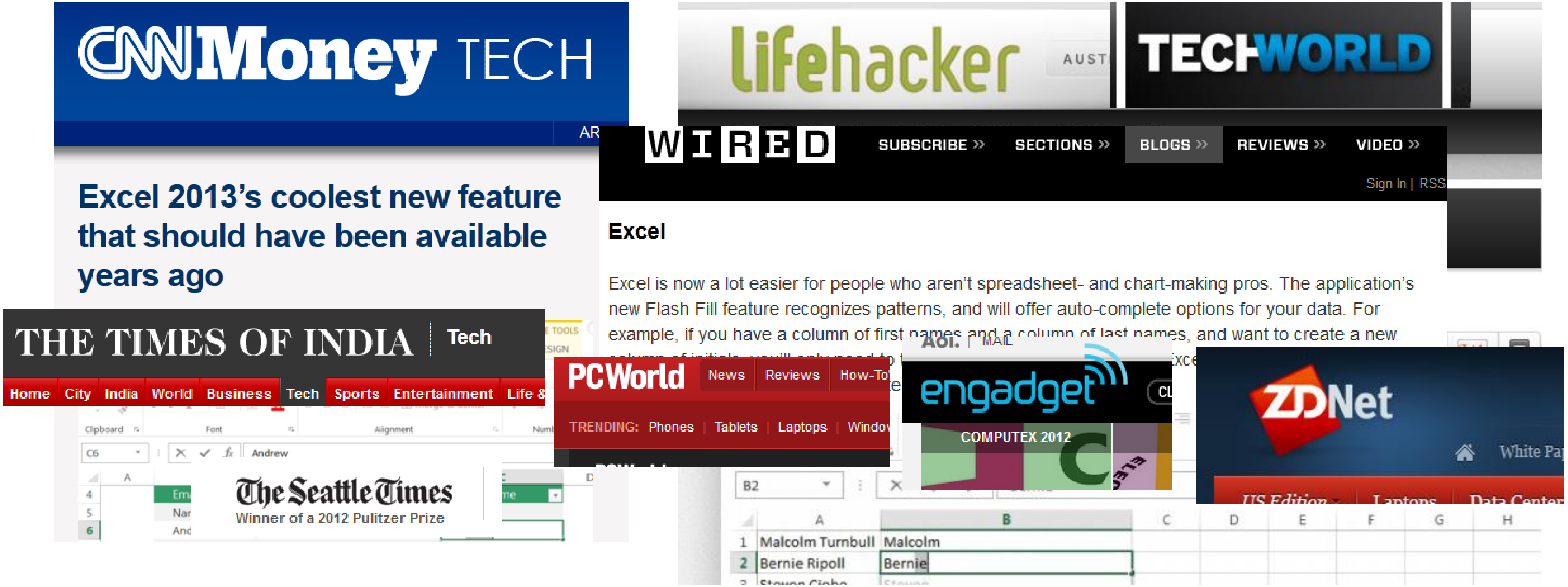
C

Haskell

modern program synthesis

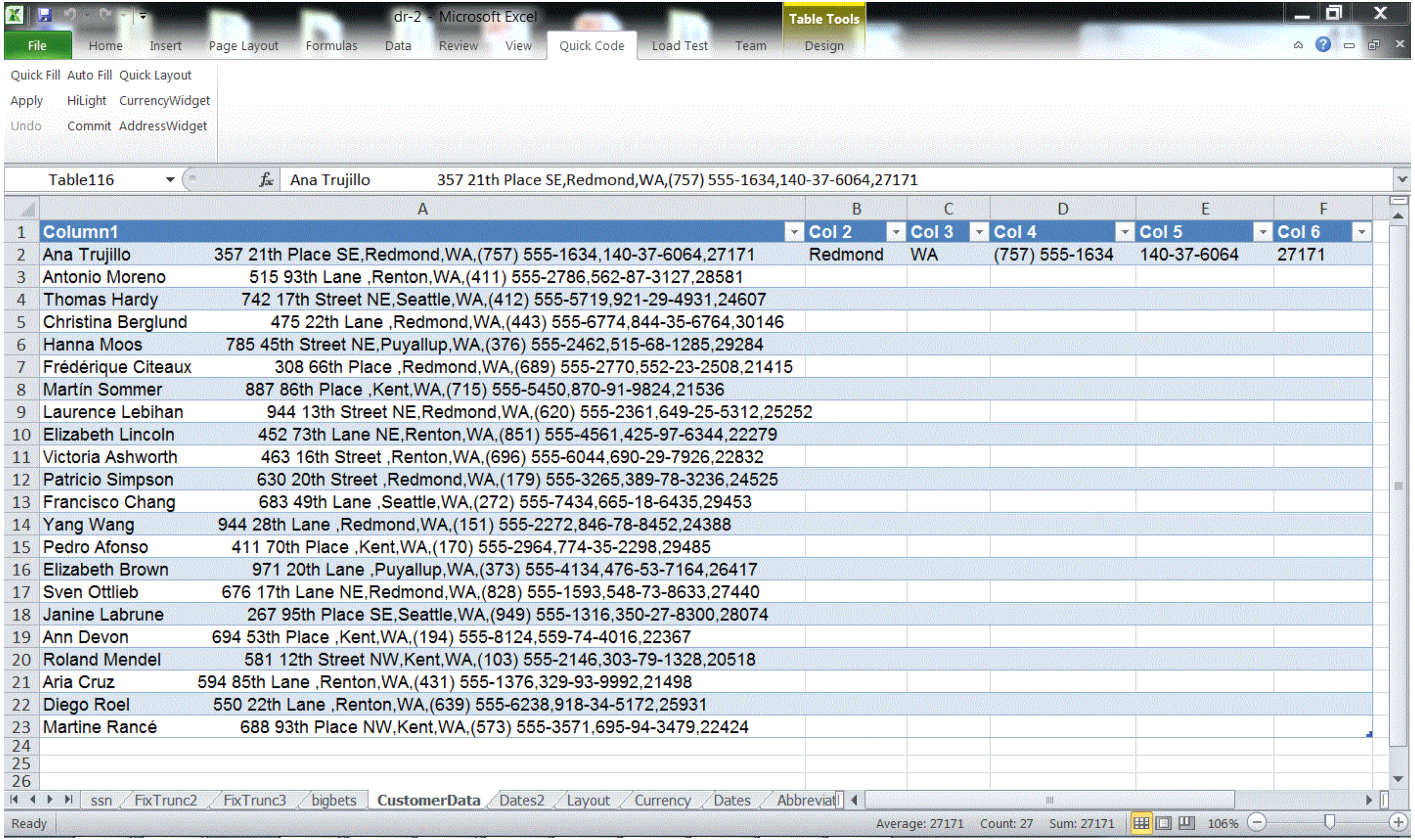
Modern program synthesis: FlashFill

[Gulwani 2011]



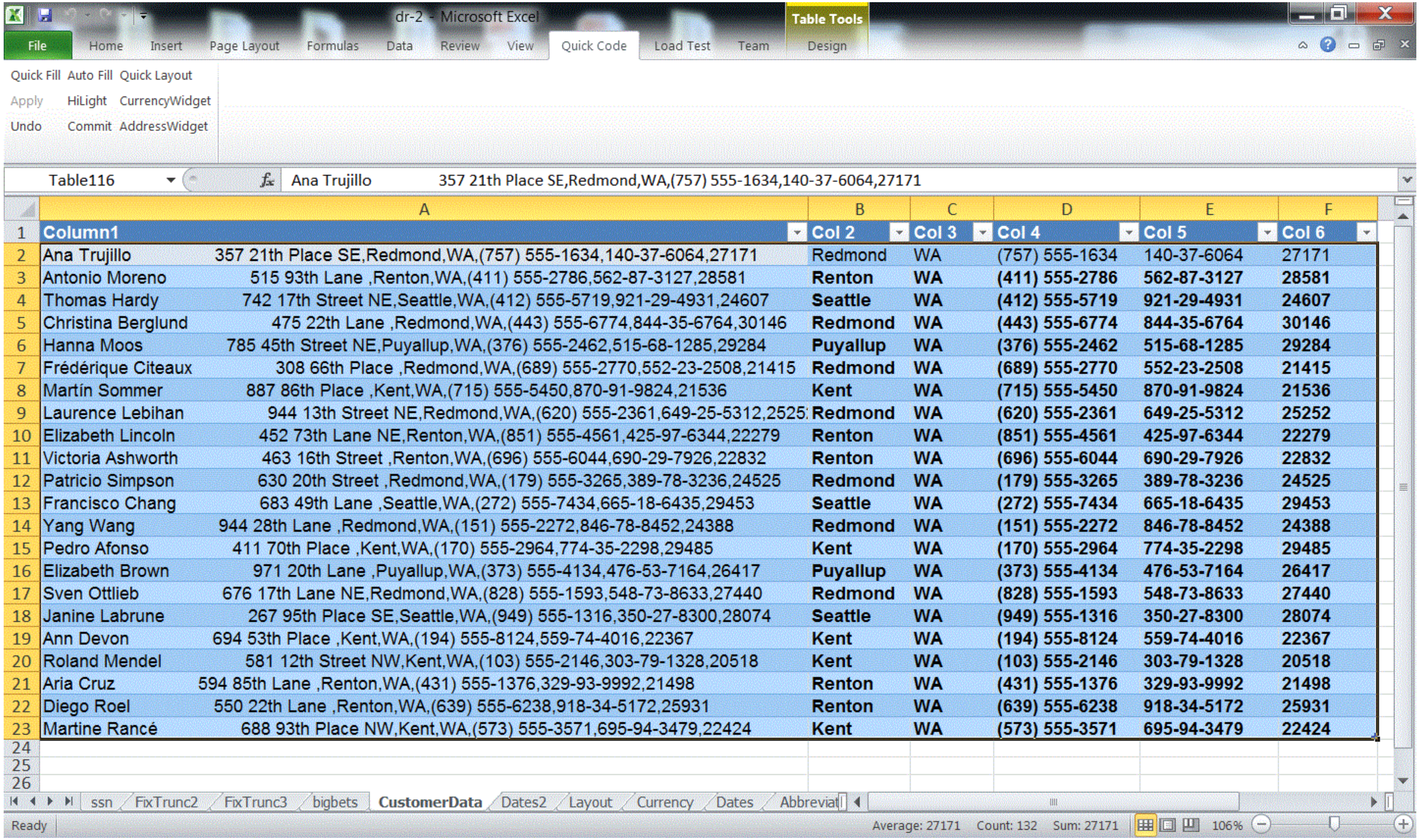
FlashFill: a feature of Excel 2013

[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



	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	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
24						
25						
26						

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 (~).
*/

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

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 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*

sorted list

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

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: LLMs

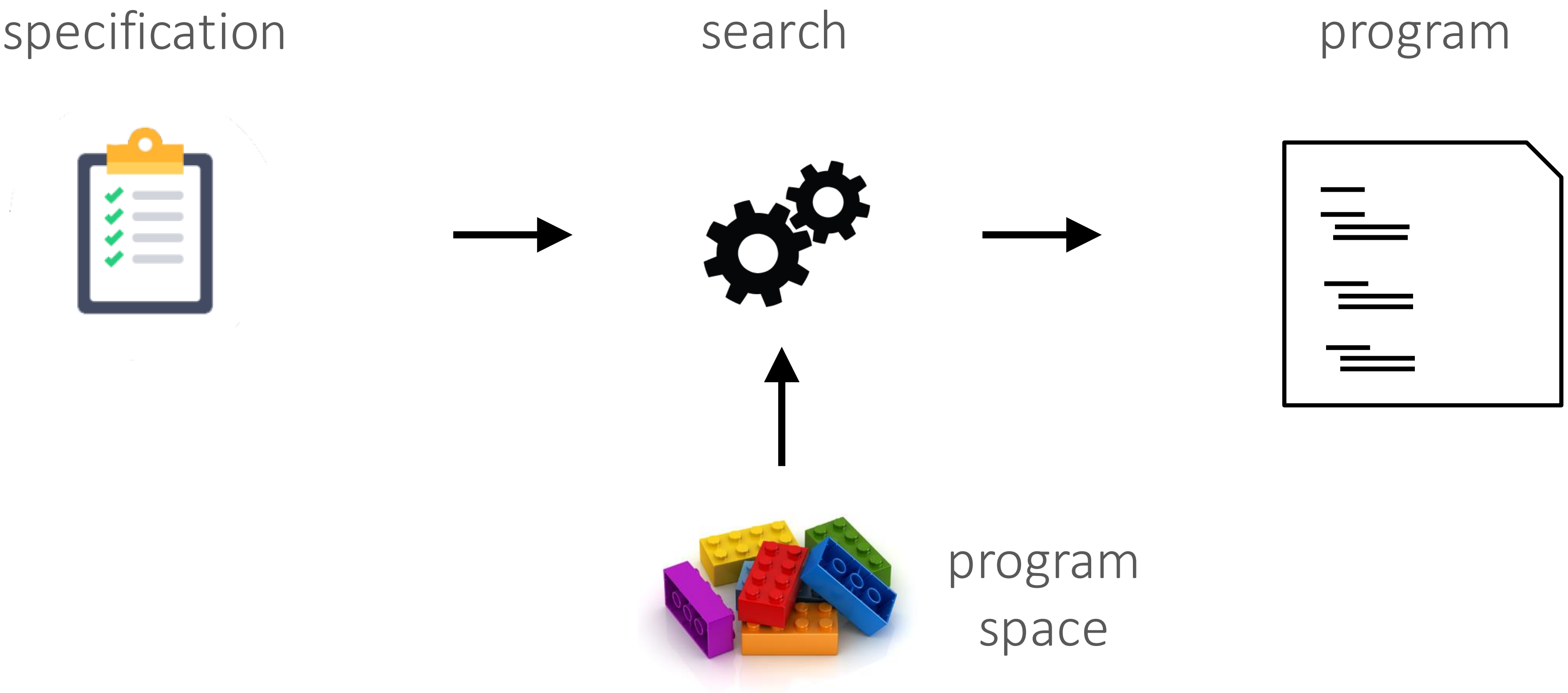
```
// find all images
// and add a green border around them
// and add class “githubCopilot” to them
function go() {

 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

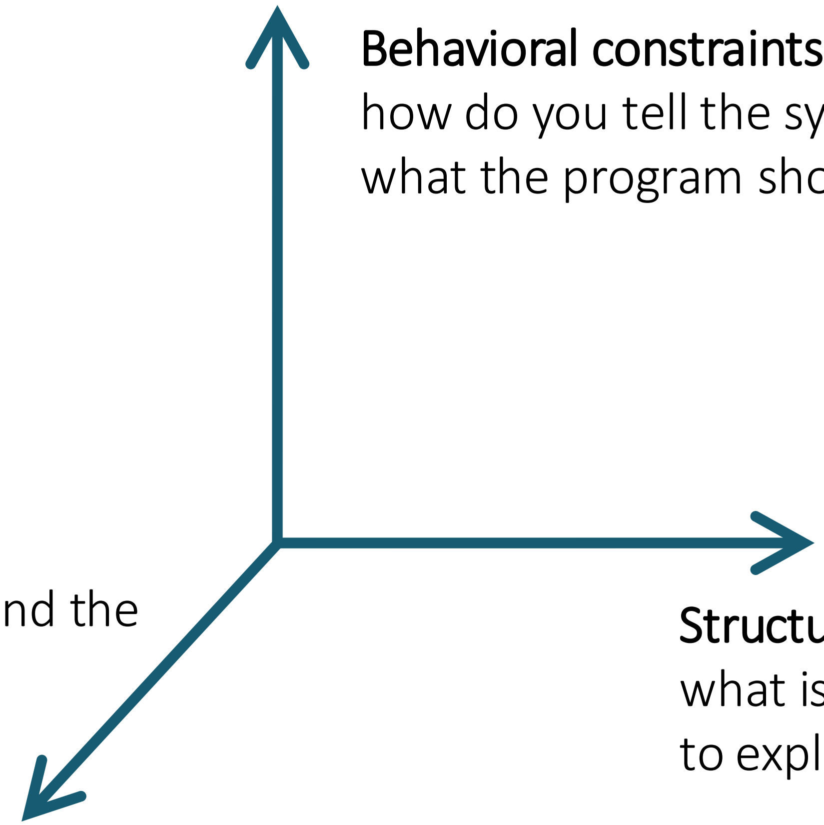
output

What is program synthesis?



Dimensions in program synthesis

[Gulwani 2010]

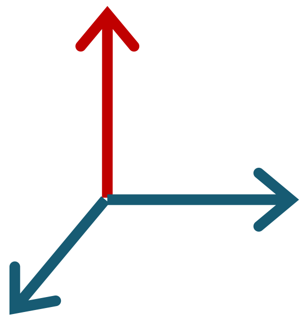


Behavioral constraints (aka specification):
how do you tell the system
what the program should do?

Search strategy:
How does the system find the
program you want?

Structural constraints:
what is the space of programs
to explore?

Behavioral constraints

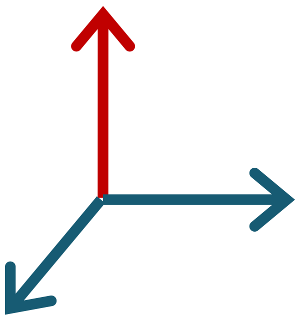


How do you tell the system what the program should do?

- What is the input language / format?
- What is the interaction model?
- What happens when the intent is ambiguous?

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

Behavioral constraints: examples



Input/output examples

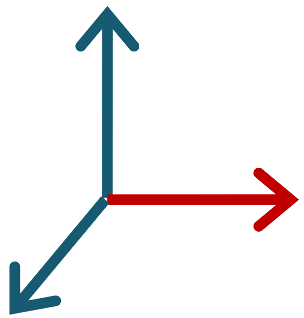
Reference implementation

Formal specifications (pre/post conditions, types, ...)

Natural language

Context

Structural constraints

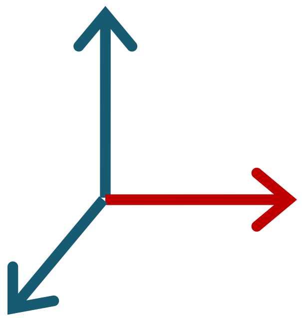


What is the space of programs to explore?

- Large enough to contain interesting programs, yet small enough to exclude garbage and enable efficient search
- Built-in or user defined?
- Can we extract domain knowledge from existing code?

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

Structural constraints: examples



Built-in DSL

User-defined DSL (grammar)

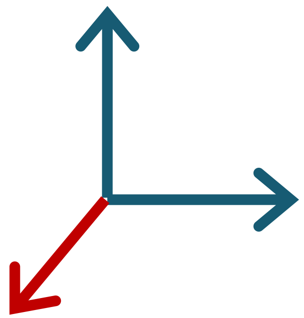
User-provided components

Languages with synthesis constructs

- e.g. generators in Sketch

General-purpose language + learned model

Search strategies



Synthesis is search:

- Find a program in the space defined by *structural constraints* that satisfies *behavioral constraints*

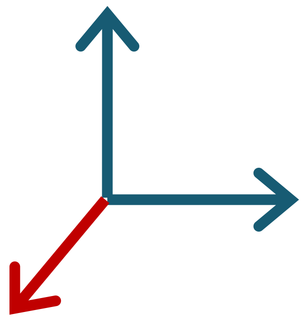
Challenge: the space is astronomically large

- 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

Stochastic 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 solver

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: Advanced Topics

- Human aspects, applications, neural synthesis

Module 1: Searching for Simple Programs

Example: FlashFill

specification

1: “Dantoni, Loris” → “Loris”

2: “Van Damme, Jean Claude” → “Jean”

program space

constant string:

“…”

or substring of input:

`between(“…”, “…”)`

→

“” … “hello” … “Loris” …

`between(“ ”, “ ”)` … `between(“, ”, “ ”)`

too many

Module 2: Searching for Complex Programs

Example: Synquid

specification

```
intersect :: xs:SList a  →
  ys:SList a  →
  {ν:SList a | elems ν = elems xs ∩
                         elems ys}
```

program

```
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)
```

How do we know this program always produces a sorted list that is the intersection?

Module 3: Advanced Topics

Synthesis using LLMs and other ML techniques

Weeks 1-2

Topic: Enumerative synthesis from examples

Paper: Alur, Radhakrishna, Udupa. Scaling Enumerative Program Synthesis via Divide and Conquer