Introduction to Program Synthesis

CS560: Reasoning About Programs

Roopsha Samanta

Partly based on slides by Armando Solar-Lezama and Xiaokang Qiu

Roadmap

Previously

- Course overview

Today

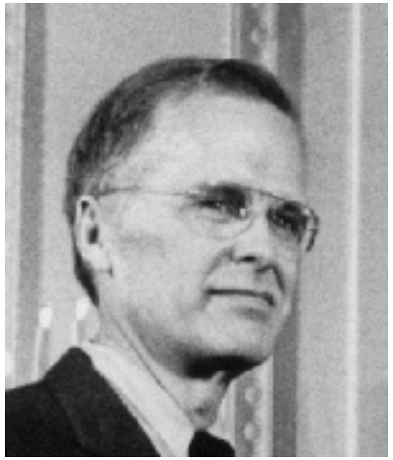
- Introduction to program synthesis
- Project description

What is program synthesis?

1950’s - 1990’s

1950's: Fortran

John Backus



Much of my work has come from being lazy. I didn't like writing programs, and so, when I was working on the IBM 701, writing programs for computing missile trajectories, I started work on a programming system to make it easier to write programs.

Essentially, compilation!

Backus et al., *The FORTRAN Automatic Coding System*, 1957

1950's, 1960's: Church's Synthesis Problem

Ongoing/Reactive Programs

Alonzo Church

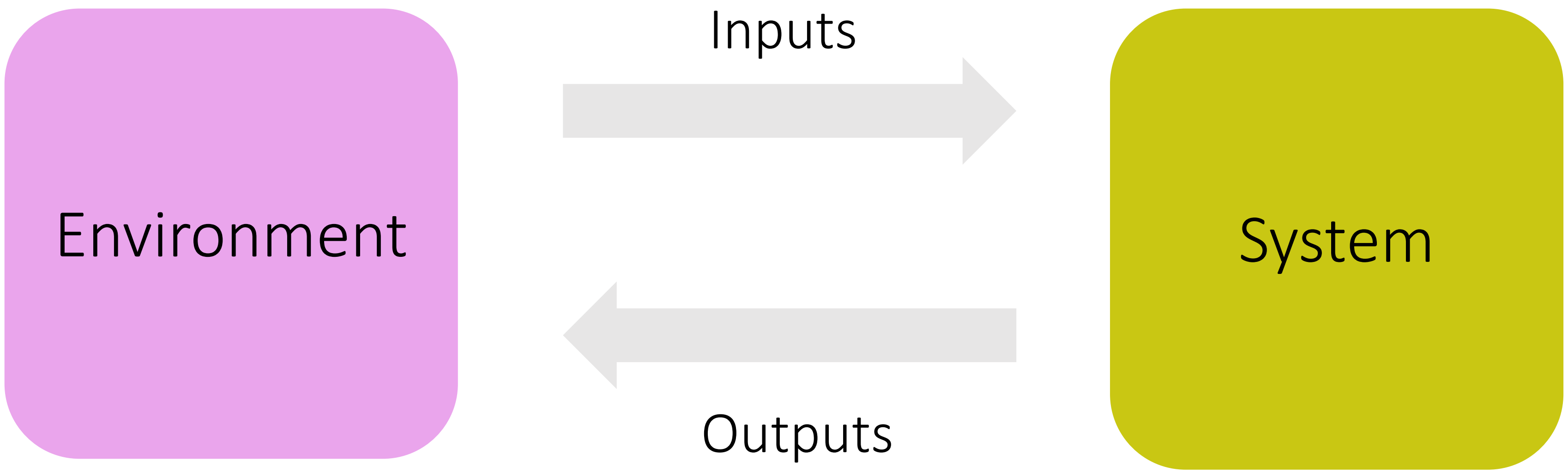
Two problems for recursive arithmetic are studied. The *synthesis problem*: given a requirement $S(t)$ in a logical system which is an extension of recursive arithmetic, to find (if possible) recursion equivalences for a circuit which satisfies the requirement. And the *decision problem*: given both requirement and recursion equivalences, to

Programs represented as circuits/finite automata

Church, *Application of Recursive Arithmetic to the Problem of Circuit Synthesis*, 1957

Church, Logic, Arithmetic and Automata, 1962

The goal of reactive synthesis is to generate a reactive system whose behavior satisfies a temporal specification, in the presence of continuous interaction with an environment

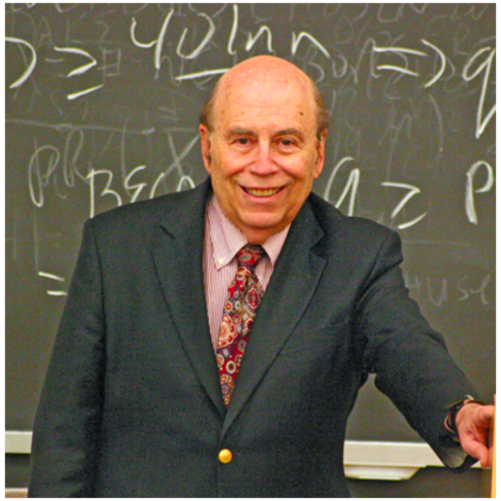


1960's, 1970's: Church's Synthesis Problem Solved!

Julius Richard Buchi

Lawrence Landwebber

Michael O. Rabin



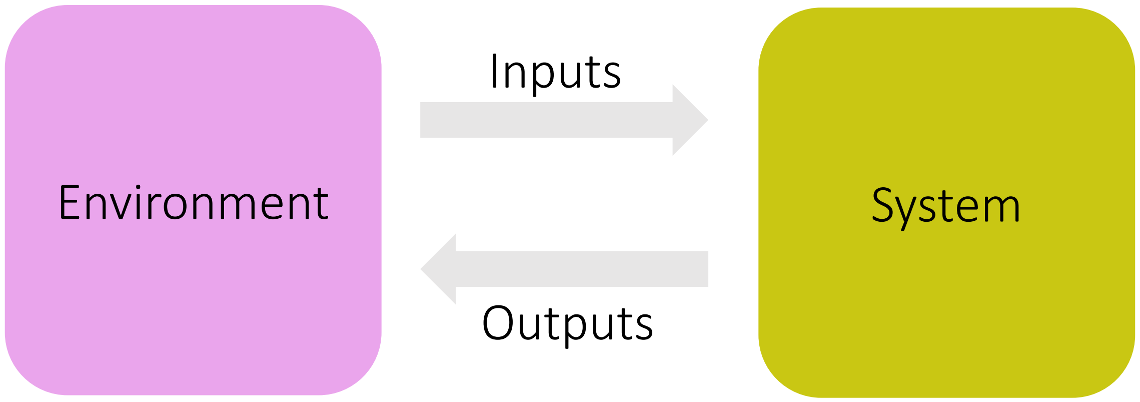
Extract program from finite-state winning strategy of an infinite two-player game

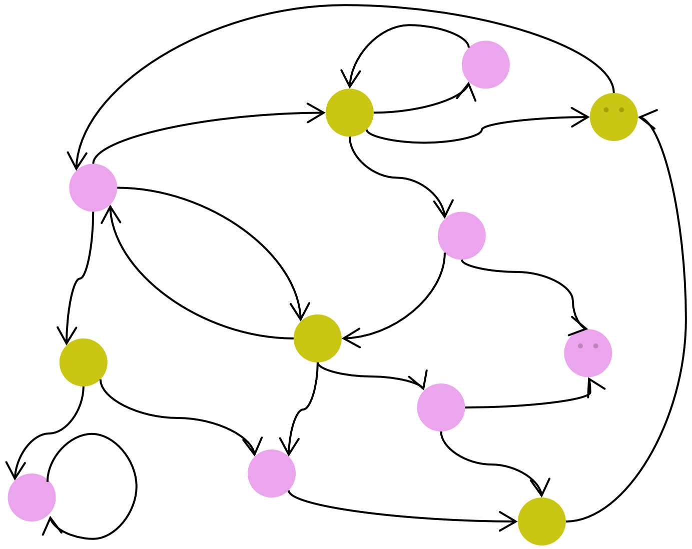
Buchi and Landwebber, *Solving Sequential Conditions by Finite-State Strategies*, 1967

Rabin, *Automata on infinite objects and Church's Problem*, 1972

For every move of the adversary (every action of the environment),
the synthesized program must make a counter-move that maintains correctness.

The game can be modeled as an automaton





1960's, 1970's: Deductive Synthesis

Transformational/Functional Programs

Cordell Green

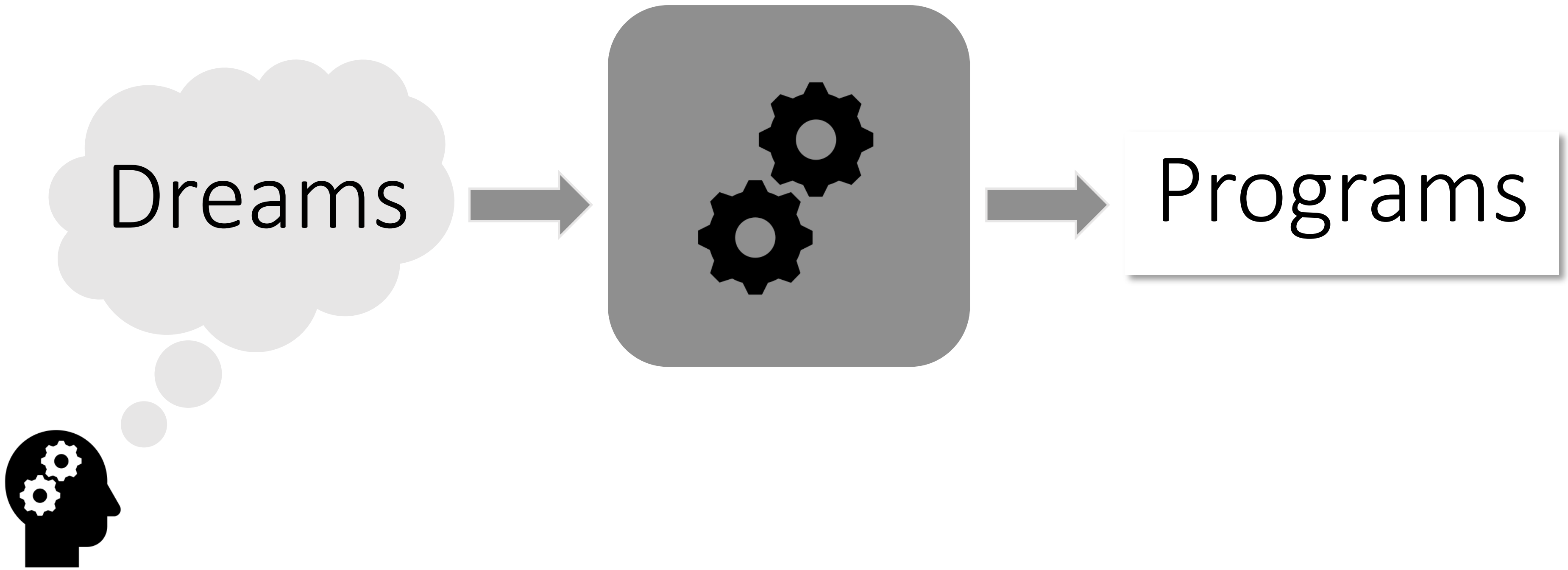
Zohar Manna

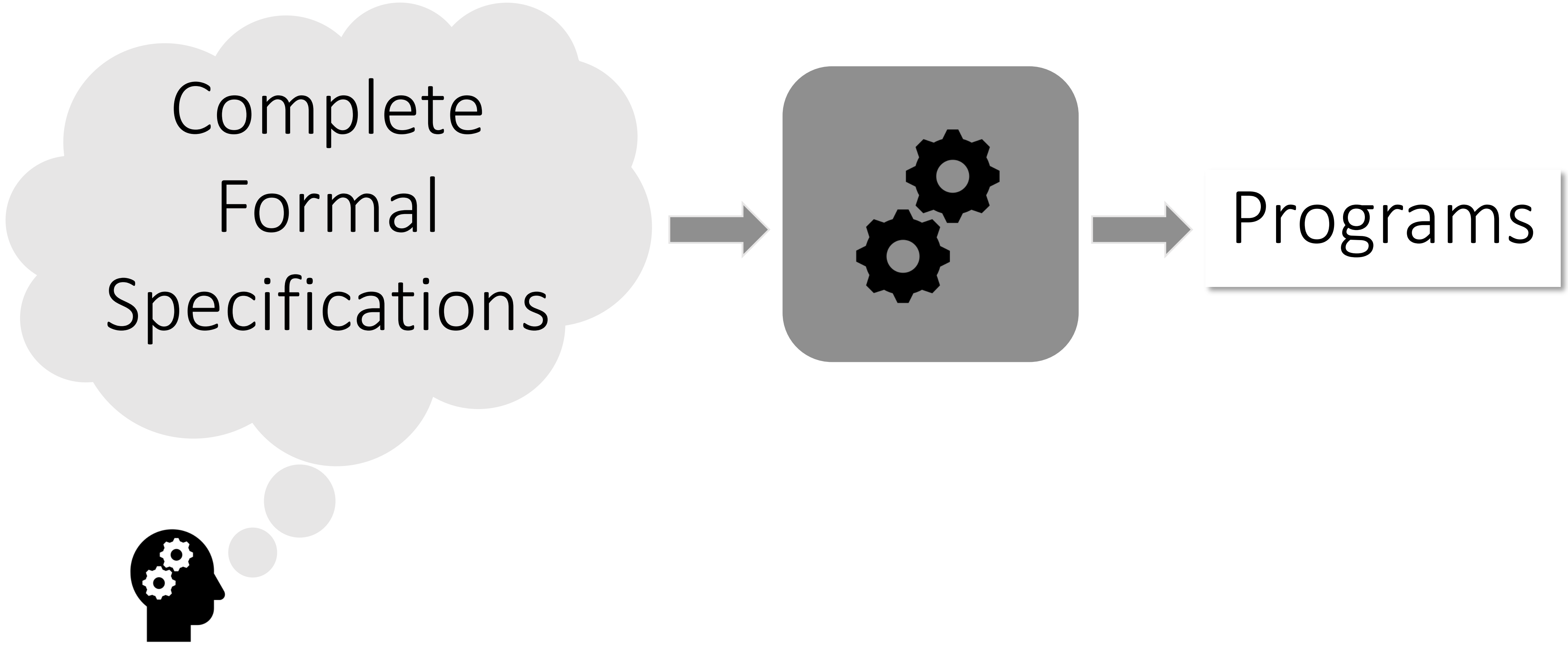
Richard Waldinger

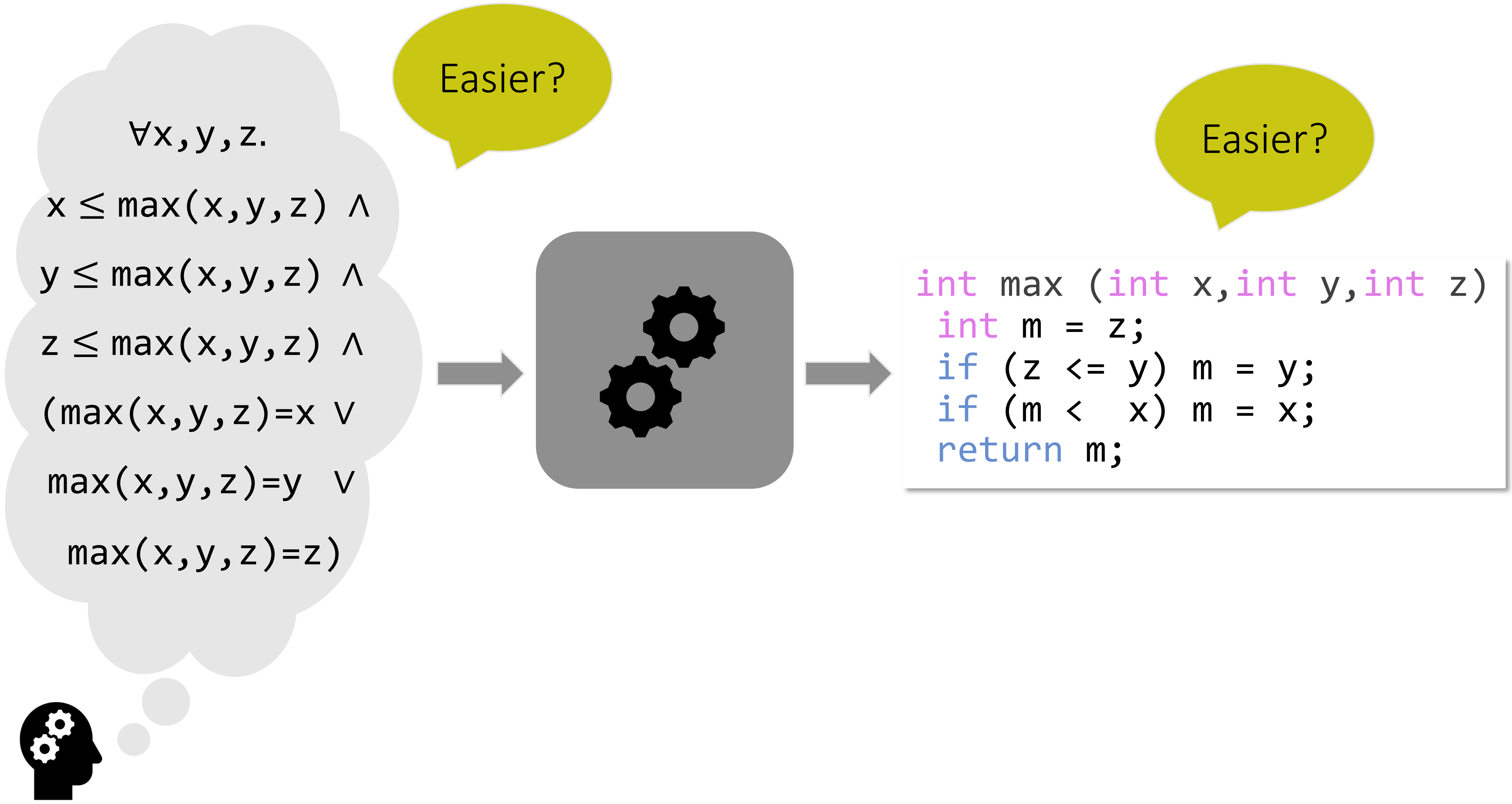
Extract LISP-y program from **proof** of satisfiability of formal specification

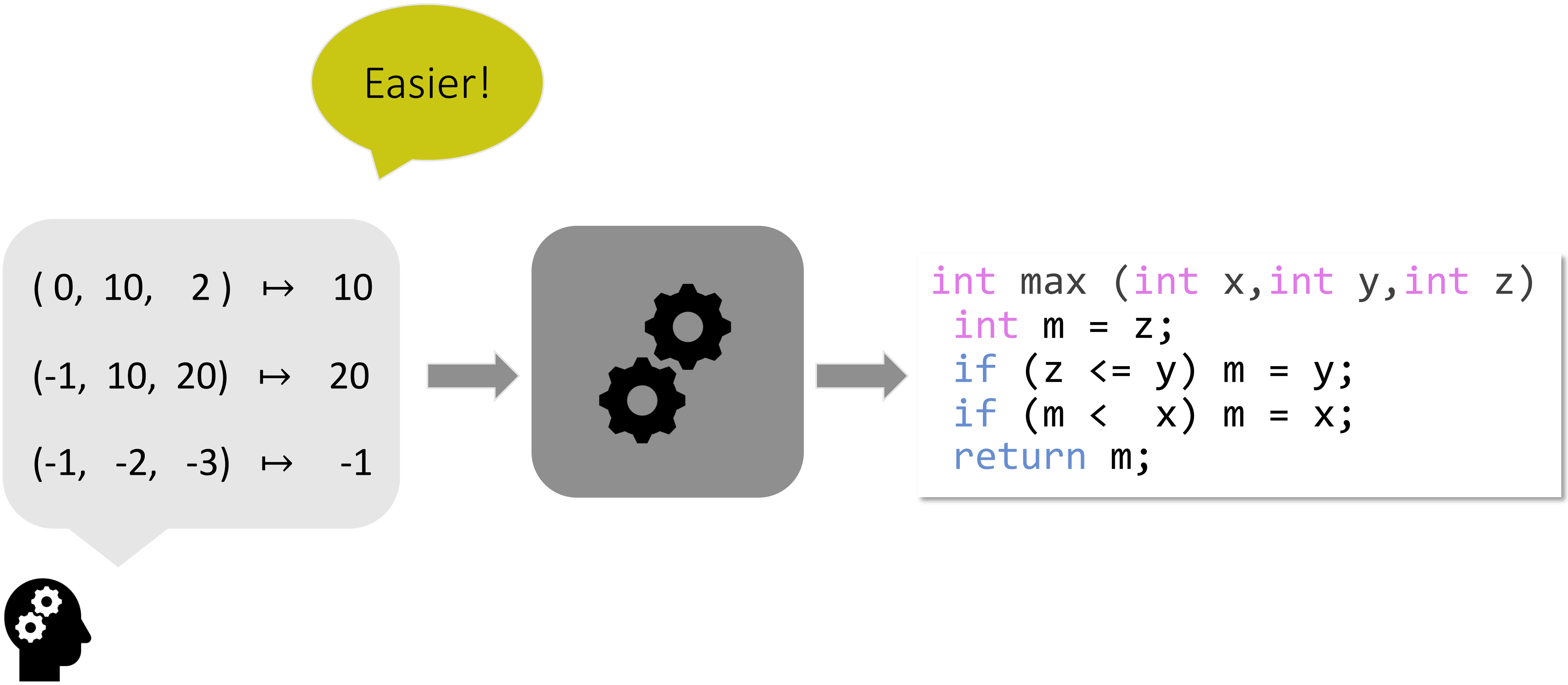
Green, *Application of Theorem Proving to Problem Solving*, 1963

Manna and Waldinger, *Dreams* ⇒ *Programs*, 1979


Dreams
Programs


Complete
Formal
Specifications
Programs


∀x,y,z.
x ≤ max(x,y,z) ∧
y ≤ max(x,y,z) ∧
z ≤ max(x,y,z) ∧
(max(x,y,z)=x ∨
max(x,y,z)=y ∨
max(x,y,z)=z)
Easier?
Easier?
int max (int x,int y,int z)
int m = z;
if (z <= y) m = y;
if (m < x) m = x;
return m;


Easier!
(0, 10, 2) ↦ 10
(-1, 10, 20) ↦ 20
(-1, -2, -3) ↦ -1
int max (int x,int y,int z)
int m = z;
if (z <= y) m = y;
if (m < x) m = x;
return m;

1970's: Inductive Programming

Transformational Programs

Summers, *A Methodology for LISP Program Construction from Examples*, 1977

Biermann, *Inference of Regular LISP Programs from Examples*, 1978

Example 8: In a list of lists, obtain the first element of each list: ((*A*) (*B*)) yields (*A B*). First these lists are converted to *S*-expressions as described in Example 1.

Example Input	Output
$((A \cdot D) \cdot ((B \cdot E) \cdot C))$	$(A \cdot (B \cdot C))$

Program:

$$(F_1\, X) = (\text{COND}((\text{ATOM}\ X)X)$$
$$((\text{ATOM}(\text{CAR}\ X))(F_1(\text{CAR}\ X)))$$
$$(\text{T}(\text{CONS}(F_2\, X)(F_3\, X))))$$
$$(F_2\, X) = (F_1(\text{CAR}\ X))$$
$$(F_3\, X) = (F_1(\text{CDR}\ X)).$$

Time: 18 s.

1980’s: Synthesis of Reactive Programs

Clarke

Emerson

Extract program (model) from **algorithmically-constructed witness** to satisfiability of formal specification.

Clarke & Emerson, *Design and Synthesis of Synchronization Skeletons using Branching-Time Temporal Logic, 1981*

1980's: Synthesis of Reactive Programs

Pnueli

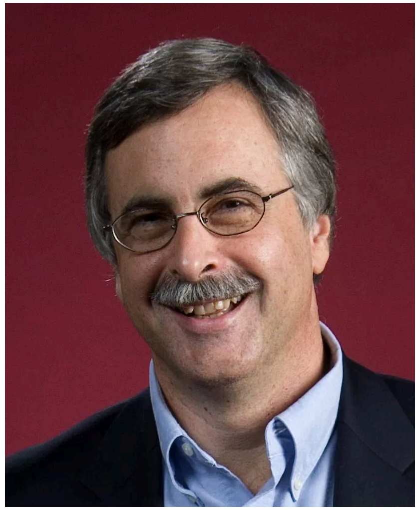
Better algorithms than Buchi, Landwebber, Rabin

Still an active research area!

Pnueli & Rosner, *On the Synthesis of a Reactive Module, 1989*

1980's: Programmer's Apprentice

Charles Rich



Richard C. Waters

- Codify expert knowledge on how to solve programming problems
- User guided synthesis

Rich and Waters, *Programmer's Apprentice*, MIT 1987

1990's: Inductive Learning
Transformational Programs

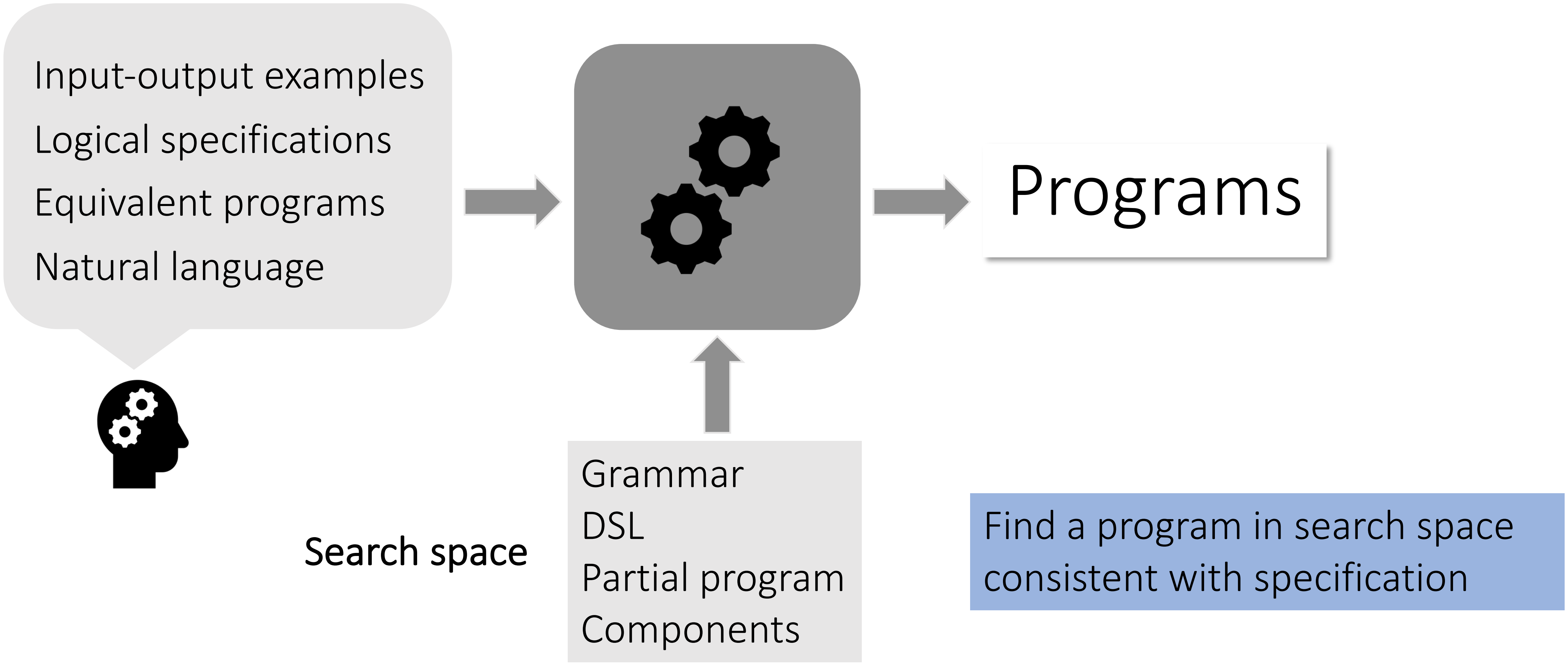
Tessa Lau

Replaced ad-hoc approaches for PBE/PBD with techniques based on version space generalization and inductive logic programming

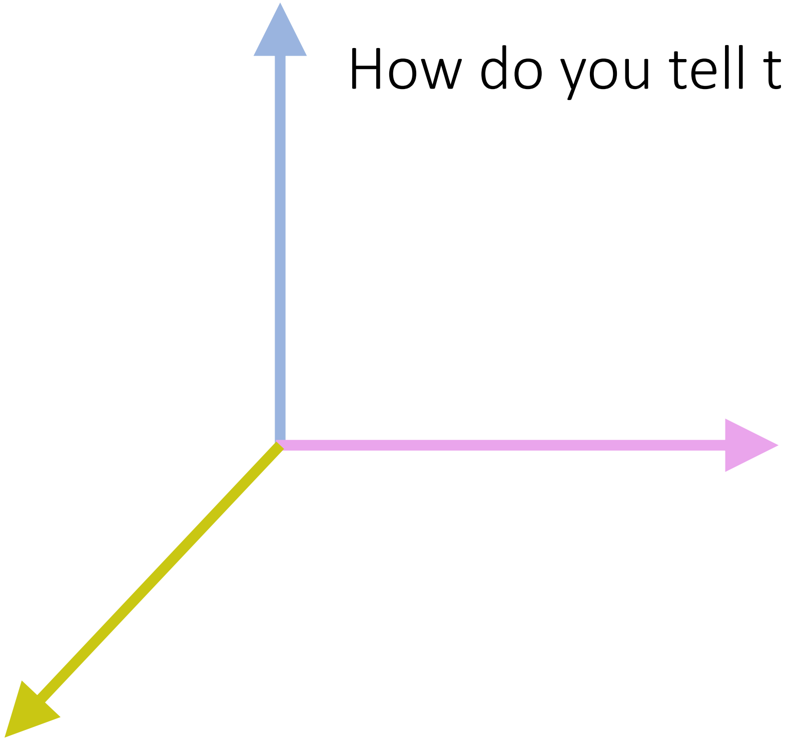
Lau and Weld, *Programming by Demonstration: An Inductive Learning Framework*, 1998

Post 2000: Modern Program Synthesis

Transformational program synthesis: A *search* problem



Dimensions in modern program synthesis



User intent

How do you tell the system what you want?

Search strategy

How does the system find the program you actually want?

Search space

What is the space of programs to explore?

[Gulwani 2010]

Dimensions in modern program synthesis

User intent

How do you tell the system what you want?

Specification formalism?

Interaction model?

Ambiguity?

Search strategy

How does the system find the program you actually want?

How do you guide the system towards relevant programs?

How does the system exploit the structure of the search space?

Search space

What is the space of programs to explore?

How do you represent domain knowledge?

Built-in or user-defined?

Dimensions in modern program synthesis

User intent

Input-output examples

Logical specifications

Equivalent programs

Natural language

Search strategy

Enumerative search + pruning

Constraint-based search

Representation-based search

Stochastic search

ML-based

...

Search space

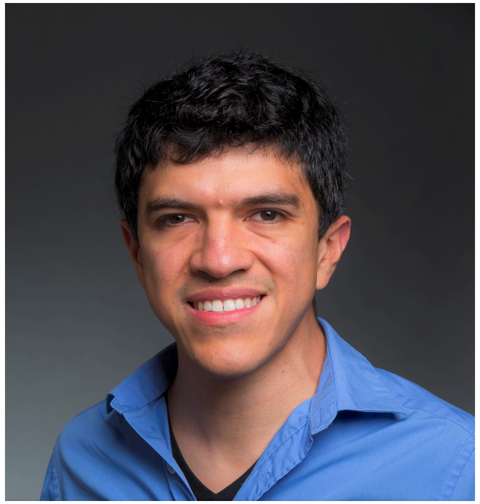
Grammars/DSLs

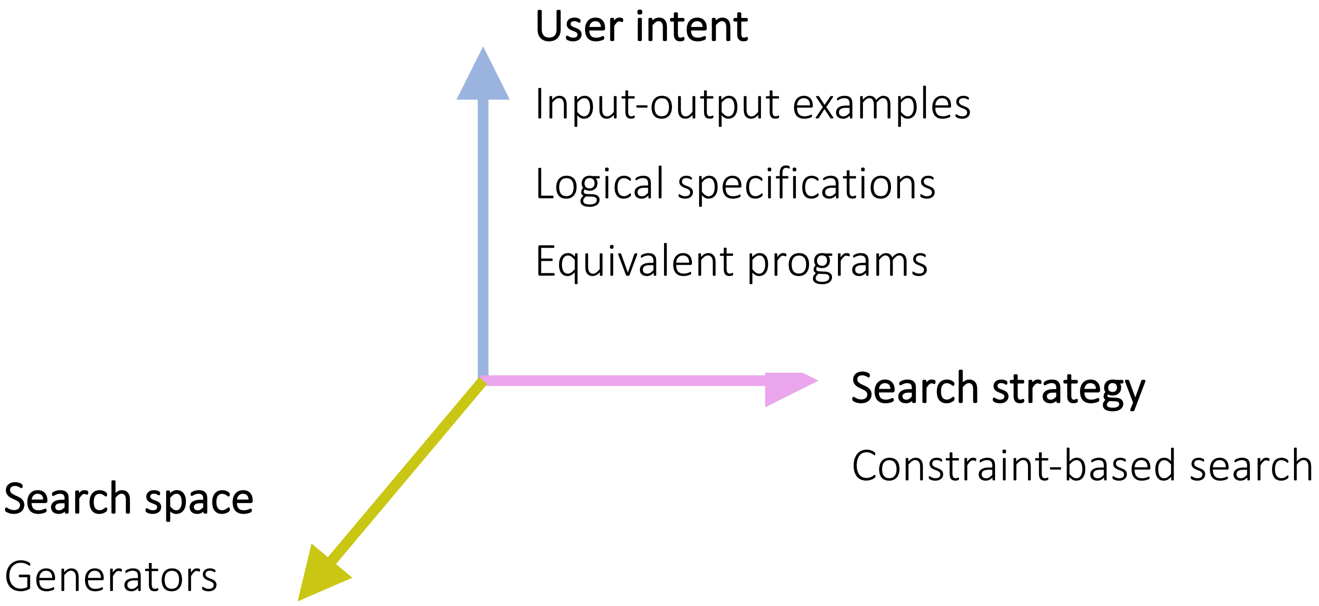
Generators

Components

2006: Sketch

Armando-Solar Lezamma





Solar-Lezamma et al., *Combinatorial Sketching for Finite Programs*, 2006

2006: Sketch

Goal: Output the least significant zero bit in a word/bitvector
Ex: 00100101 → 00000010

```
int W = 32;

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

A possible program
Not the most efficient

Adding 1 to a string of 1's turns the next 0 to a 1!

000111 + 1 = 001000

Trick for an efficient program

Sketch: space of possible implementations

```
/* Generate the set of all bit-vector expressions
 *  involving +, &, xor and bitwise negation (~).
 *  the bound param limits the size of the generated expression.
 */
generator bit[W] gen(bit[W] x, int bound){
    assert bound > 0;
    if(??) return x;
    if(??) return ??;
    if(??) return ~gen(x, bound-1);
    if(??){
      return {| gen(x, bound-1) (+ | & | ^) gen(x, bound-1) |}; }
}
```

Sketch synthesis setup

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

Naïve program

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

Program generator

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

Specification
“Program equivalence”

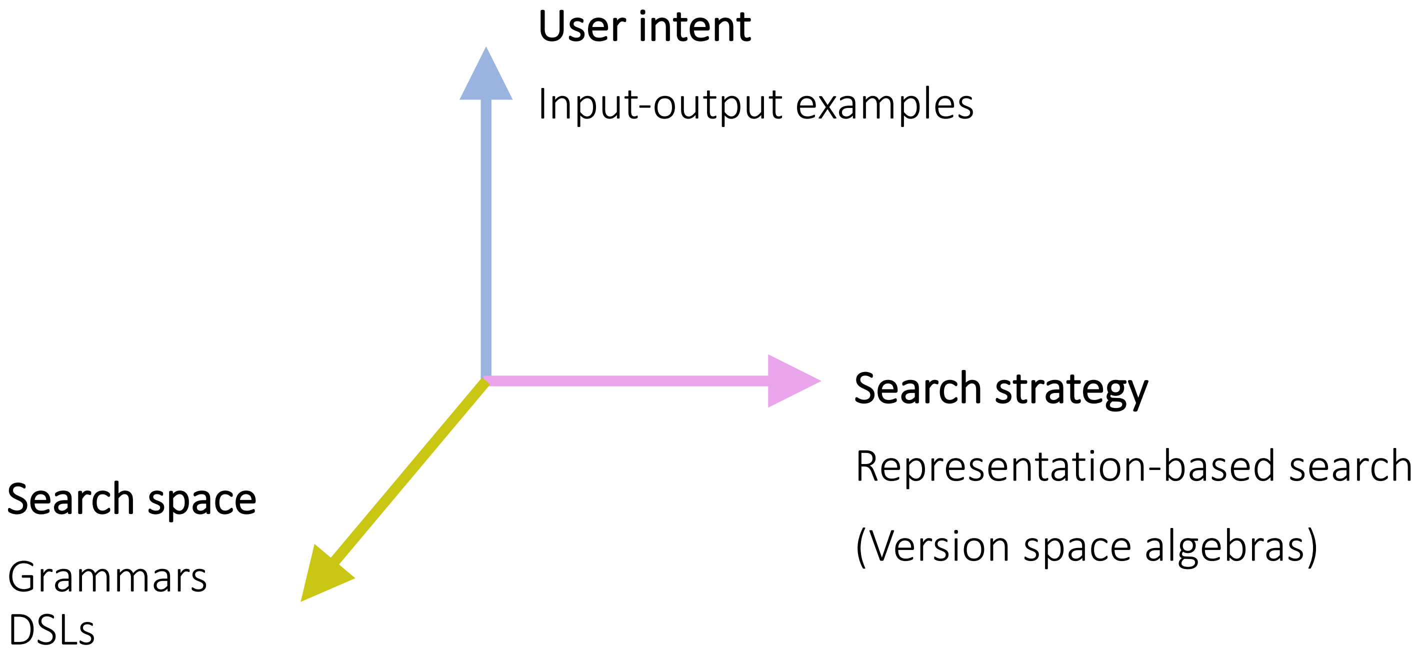
Sketch output

```
bit[W] isolate0fast (bit[W] x)  implements isolate0 {
        return ~x  &  (x+1);
}
```

Desired efficient program

2011: FlashFill

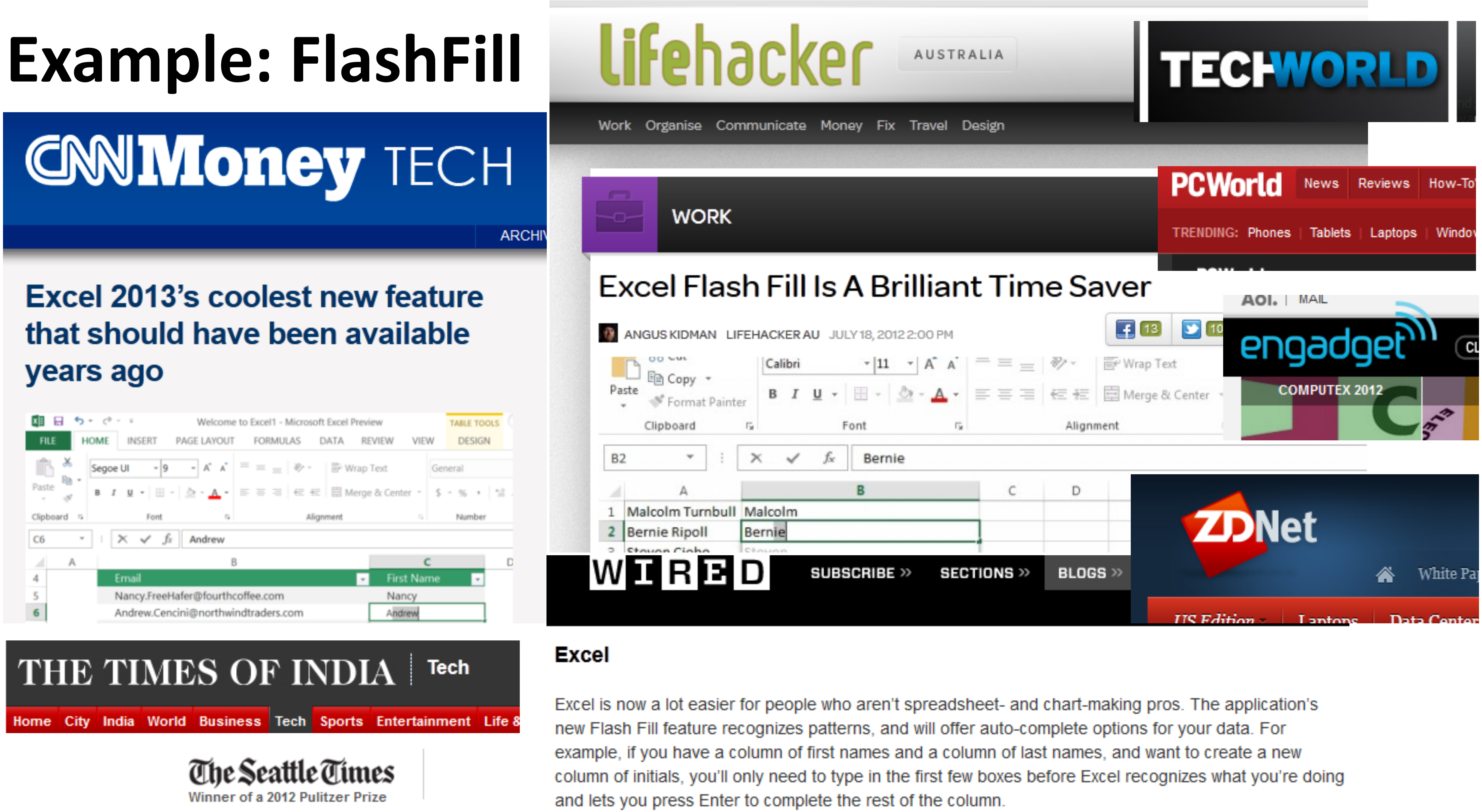
Sumit Gulwani



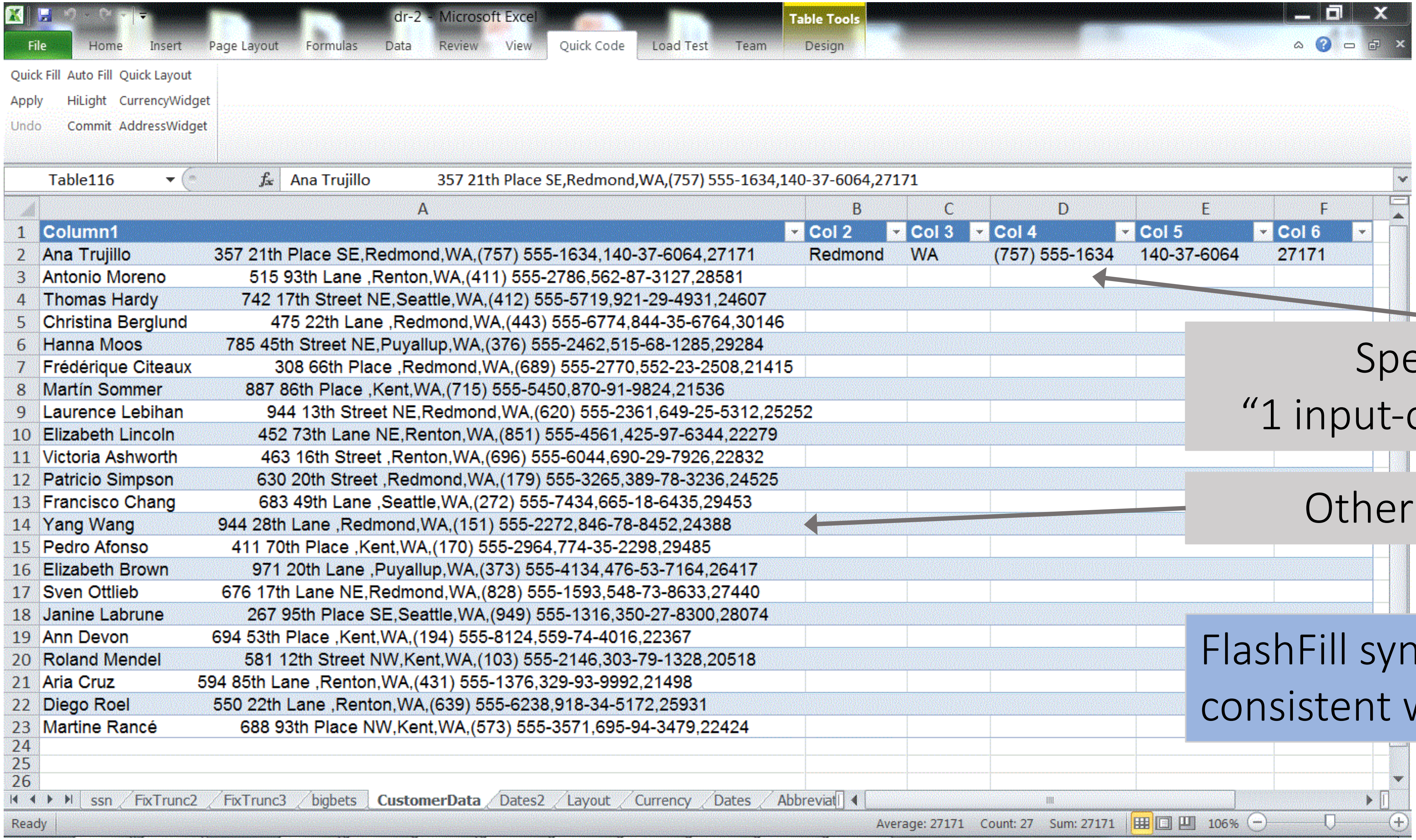
Gulwani, Automatic String Processing in Spreadsheets using Input-Output Examples, 2011

A feature of Excel 2013!

Example: FlashFill



FlashFill synthesis setup



	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						

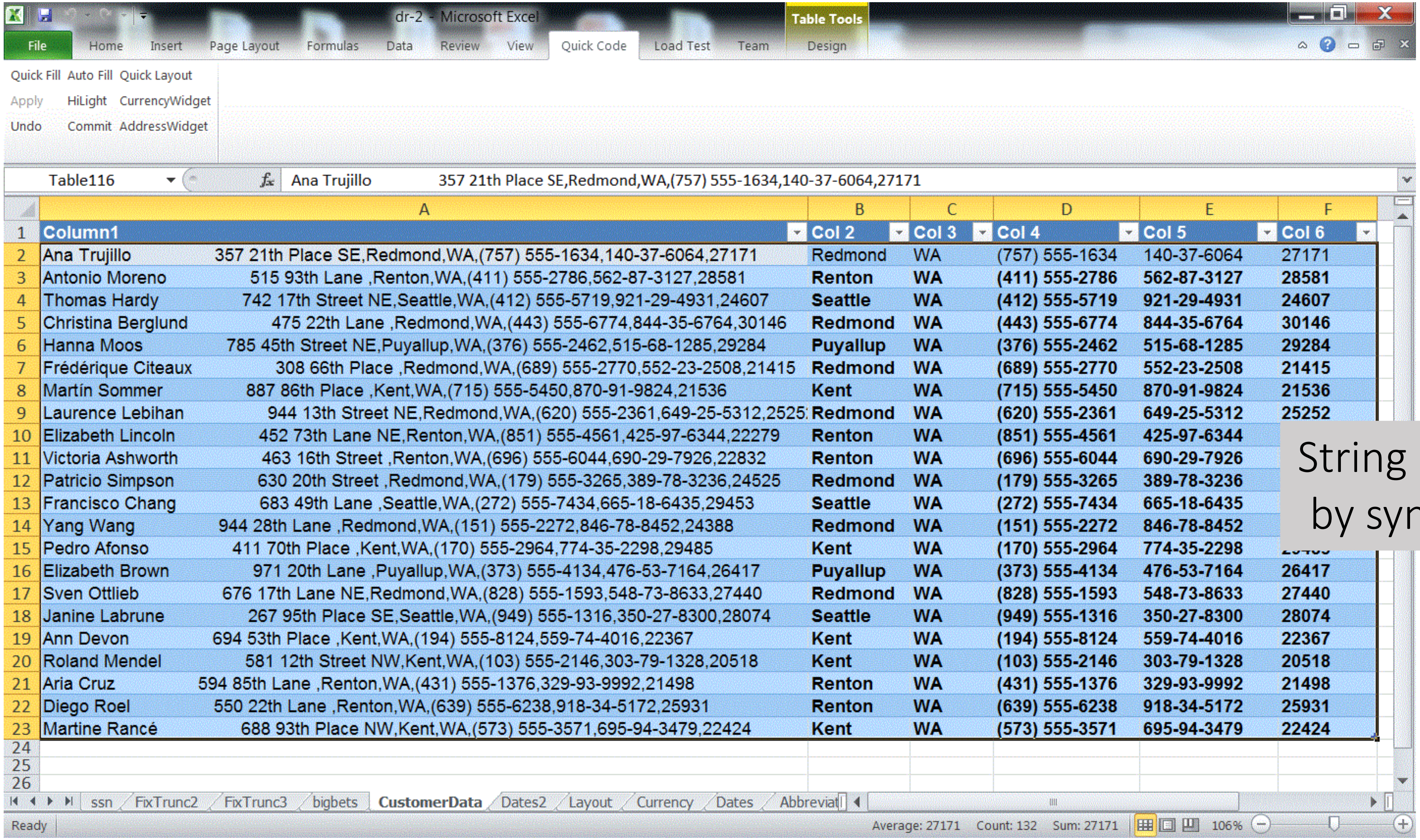
Specification:
“1 input-output example”

Other string inputs

FlashFill synthesizes program consistent with user example

[Gulwani 2010]

FlashFill output



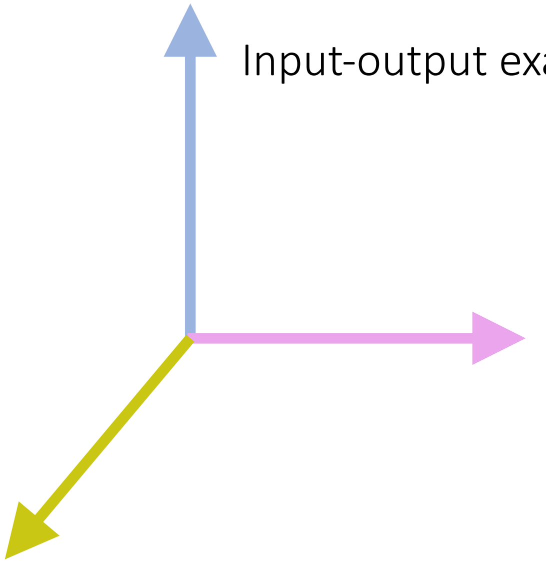
	A	B	C	D	E	F
1	Column1	Col 2	Col 3	Col 4	Col 5	Col 6
2	Ana Trujillo 357 21th Place SE,Redmond,WA,(757) 555-1634,140-37-6064,27171	Redmond	WA	(757) 555-1634	140-37-6064	27171
3	Antonio Moreno 515 93th Lane ,Renton,WA,(411) 555-2786,562-87-3127,28581	Renton	WA	(411) 555-2786	562-87-3127	28581
4	Thomas Hardy 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	[illegible]
11	Victoria Ashworth 463 16th Street ,Renton,WA,(696) 555-6044,690-29-7926,22832	Renton	WA	(696) 555-6044	690-29-7926	[illegible]
12	Patricio Simpson 630 20th Street ,Redmond,WA,(179) 555-3265,389-78-3236,24525	Redmond	WA	(179) 555-3265	389-78-3236	[illegible]
13	Francisco Chang 683 49th Lane ,Seattle,WA,(272) 555-7434,665-18-6435,29453	Seattle	WA	(272) 555-7434	665-18-6435	[illegible]
14	Yang Wang 944 28th Lane ,Redmond,WA,(151) 555-2272,846-78-8452,24388	Redmond	WA	(151) 555-2272	846-78-8452	[illegible]
15	Pedro Afonso 411 70th Place ,Kent,WA,(170) 555-2964,774-35-2298,29485	Kent	WA	(170) 555-2964	774-35-2298	[illegible]
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						

String outputs generated by synthesized program

Many more interesting papers….later!

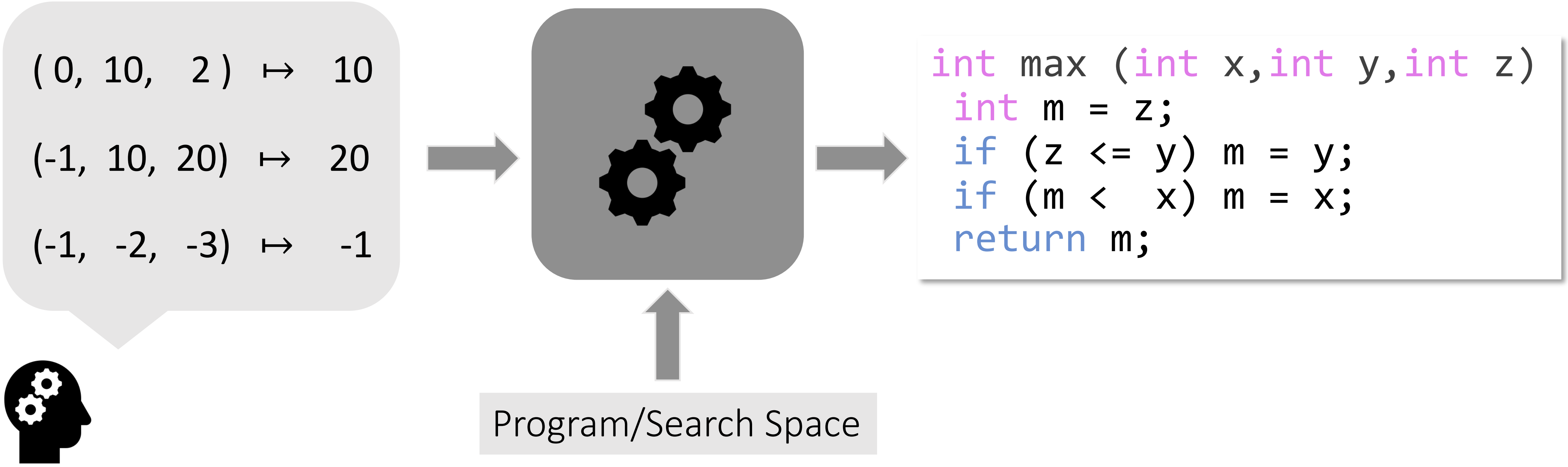
2020: MANTIS

Semantics-guided Inductive Program Synthesis



An et al., *Augmented Example-based Synthesis using Relational Perturbation Properties*, 2020

Problems in inductive program synthesis



Problems in inductive program synthesis

(0, 10, 2) ↦ 10

(-1, 10, 20) ↦ 20

(-1, -2, -3) ↦ -1

Program/Search Space

```
int max (int x,int y,int z)
 int m = z;
 if (z <= y) m = y;
 if (m <  x) m = x;
 return m;
```

```
int max (int x,int y,int z)
 int m = x;
 if (y < z) m = z;
 if (m < y) m = y;
 return m;
```

Ambiguity!

Problems in inductive program synthesis

$(0, 10, 2) \mapsto 10$

$(-1, 10, 20) \mapsto 20$

$(-1, -2, -3) \mapsto -1$

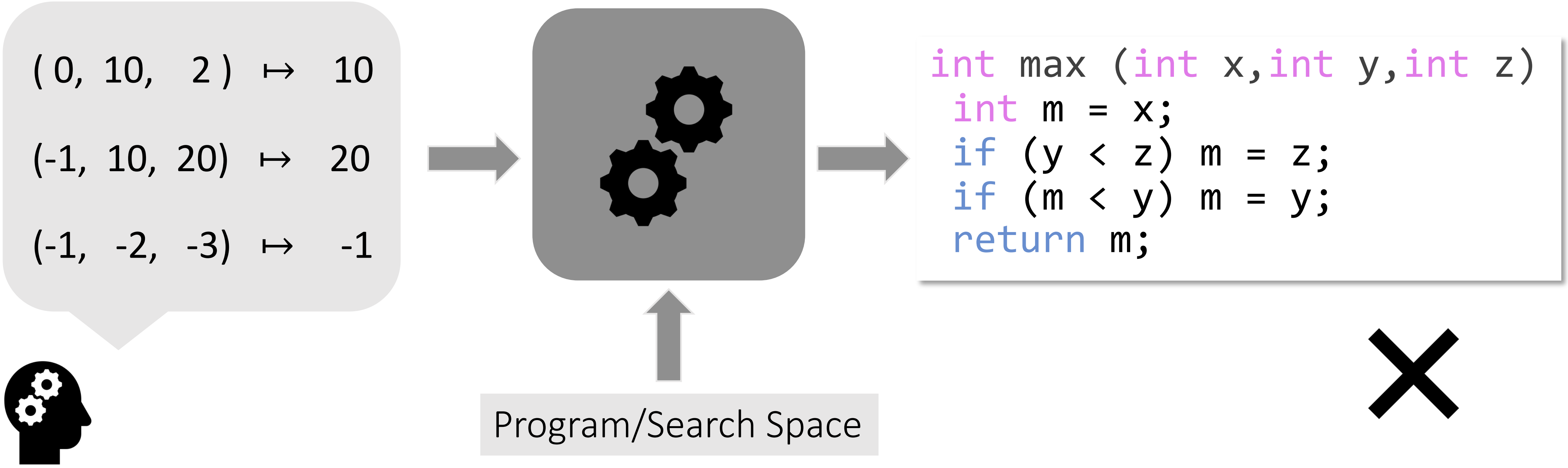
Program/Search Space

```
int max (int x,int y,int z)
 int m = z;
 if (z <= y) m = y;
 if (m <  x) m = x;
 return m;
```

```
int max (int x,int y,int z)
 int m = x;
 if (y < z) m = z;
 if (m < y) m = y;
 return m;
```

Overfitting!

Problems in inductive program synthesis



Overfitting!

Problems in inductive program synthesis

(0, 10, 2) $\mapsto$ 10
(-1, 10, 20) $\mapsto$ 20
(-1, -2, -3) $\mapsto$ -1

```
int max (int x,int y,int z)
 int m = x;
 if (y < z) m = z;
 if (m < y) m = y;
 return m;
```

✗

(0, 10, 2) $\mapsto$ 10
(-1, 10, 20) $\mapsto$ 20
(-1, -3, -2) $\mapsto$ -1

```
int max (int x,int y,int z)
 int m = x;
 if (y < z) m = z;
 if (m < y) m = y;
 return m;
```

✓

Brittleness!

Syntactic Bias

Occam's Razor
Structured DSL
Ranking Function

(0, 10, 2) ↦ 10

(-1, 10, 20) ↦ 20

(-1, -2, -3) ↦ -1

Program/Search Space

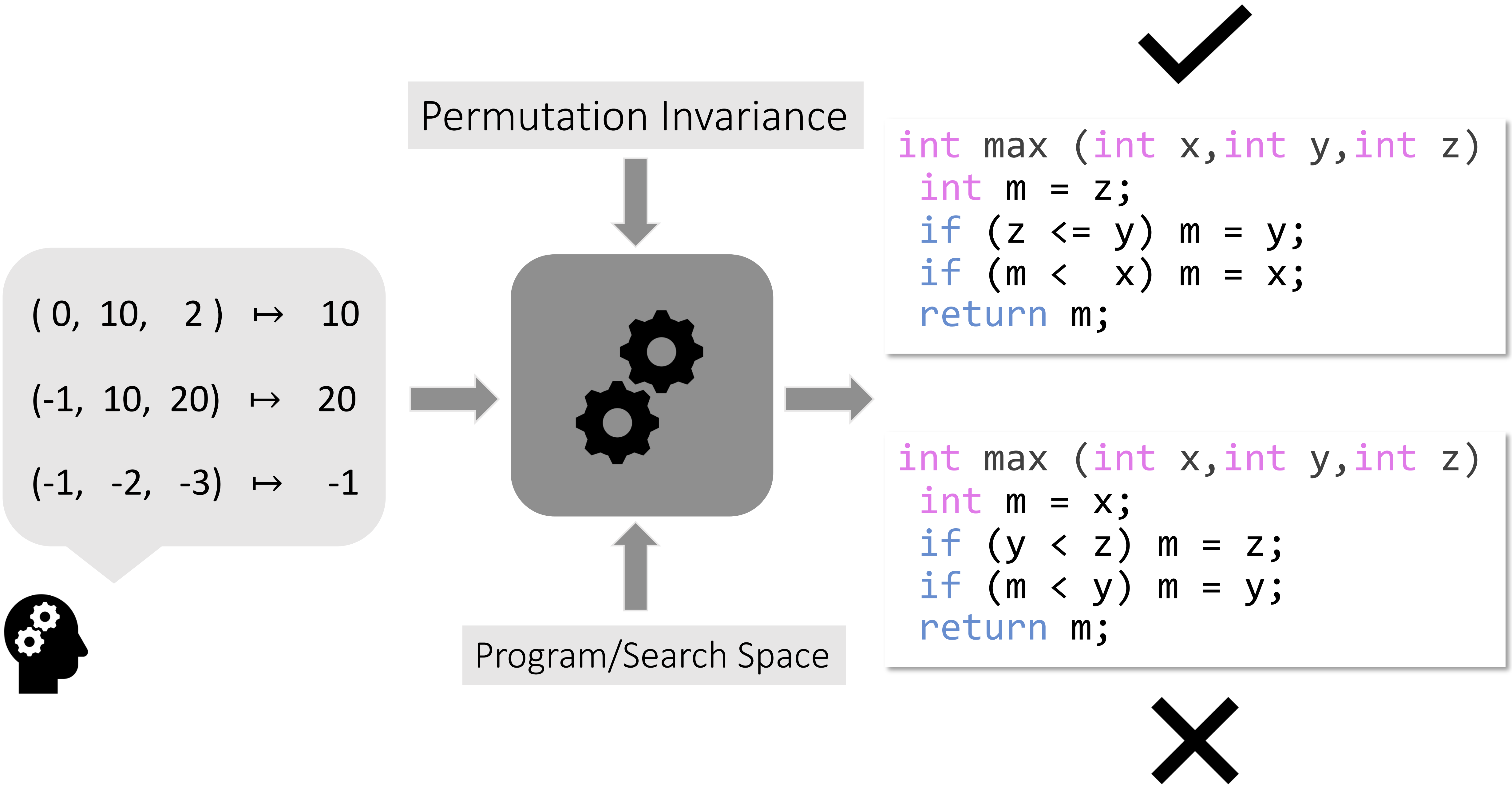
```
int max (int x,int y,int z)
 int m = z;
 if (z <= y) m = y;
 if (m <  x) m = x;
 return m;
```

✓

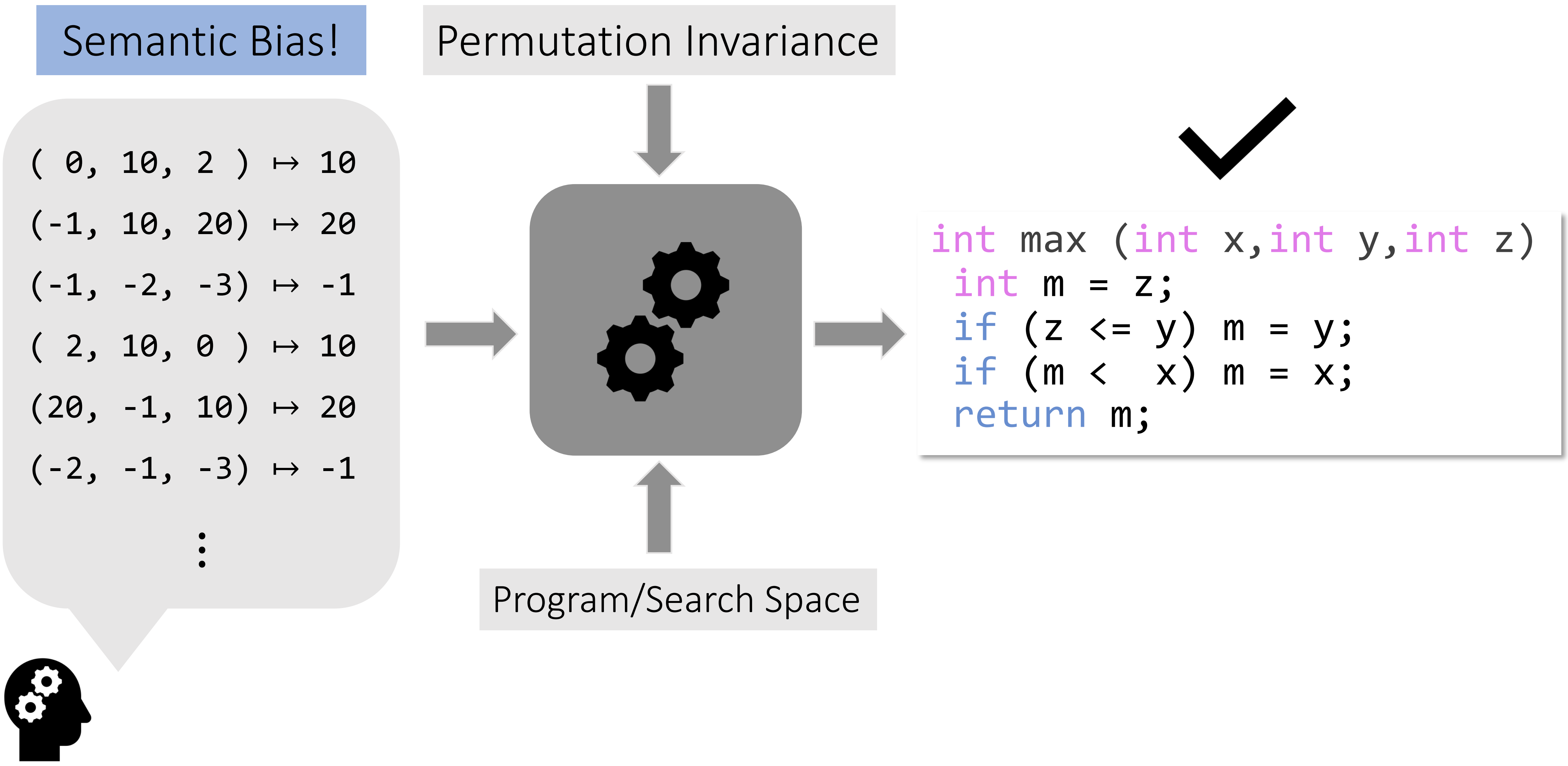
```
int max (int x,int y,int z)
 int m = x;
 if (y < z) m = z;
 if (m < y) m = y;
 return m;
```

✗

Syntactic bias can also be inadequate!



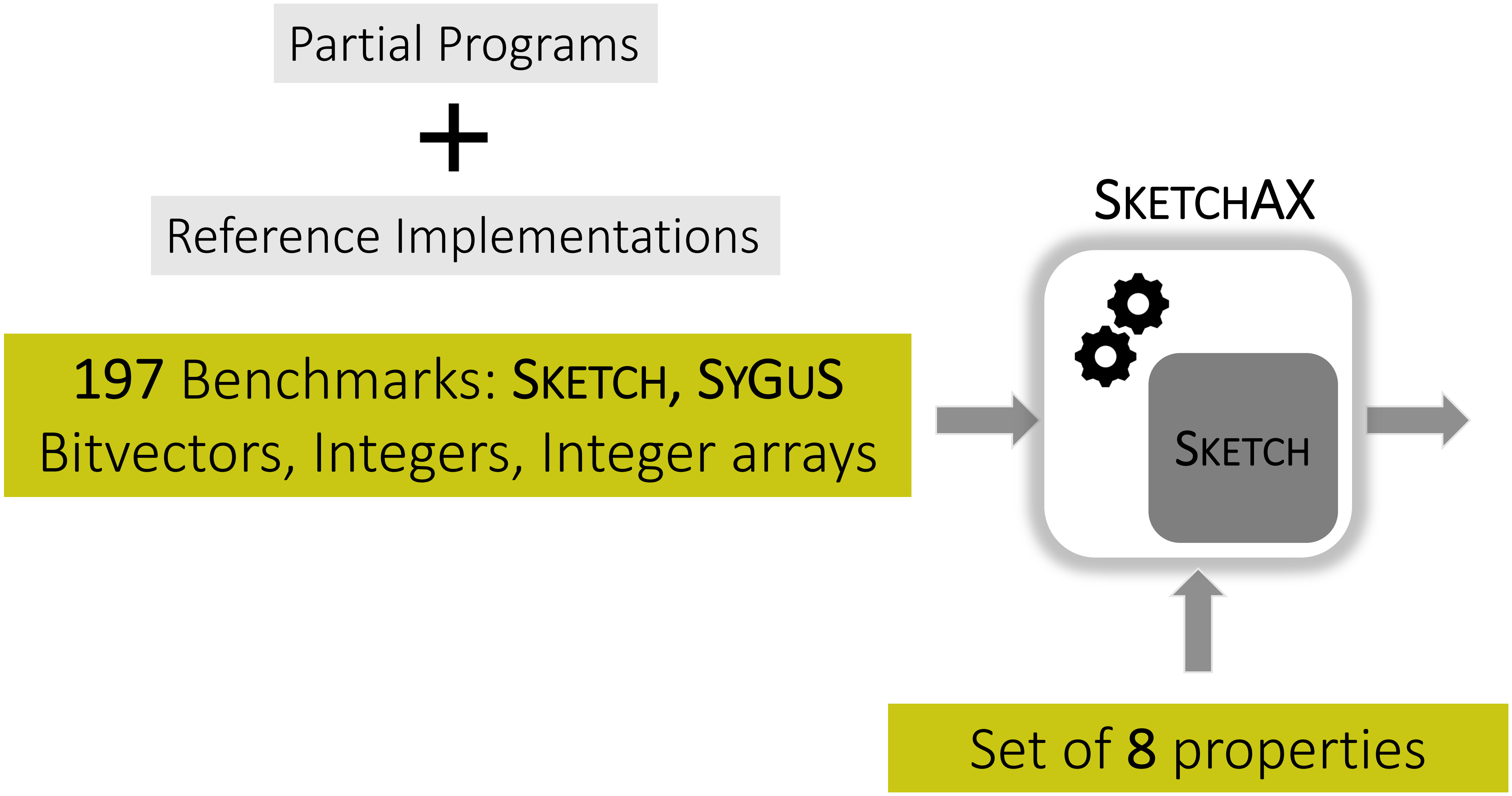
(0, 10, 2) ↦ 10
(-1, 10, 20) ↦ 20
(-1, -2, -3) ↦ -1
Permutation Invariance
Program/Search Space
int max (int x,int y,int z)
int m = z;
if (z <= y) m = y;
if (m < x) m = x;
return m;
int max (int x,int y,int z)
int m = x;
if (y < z) m = z;
if (m < y) m = y;
return m;

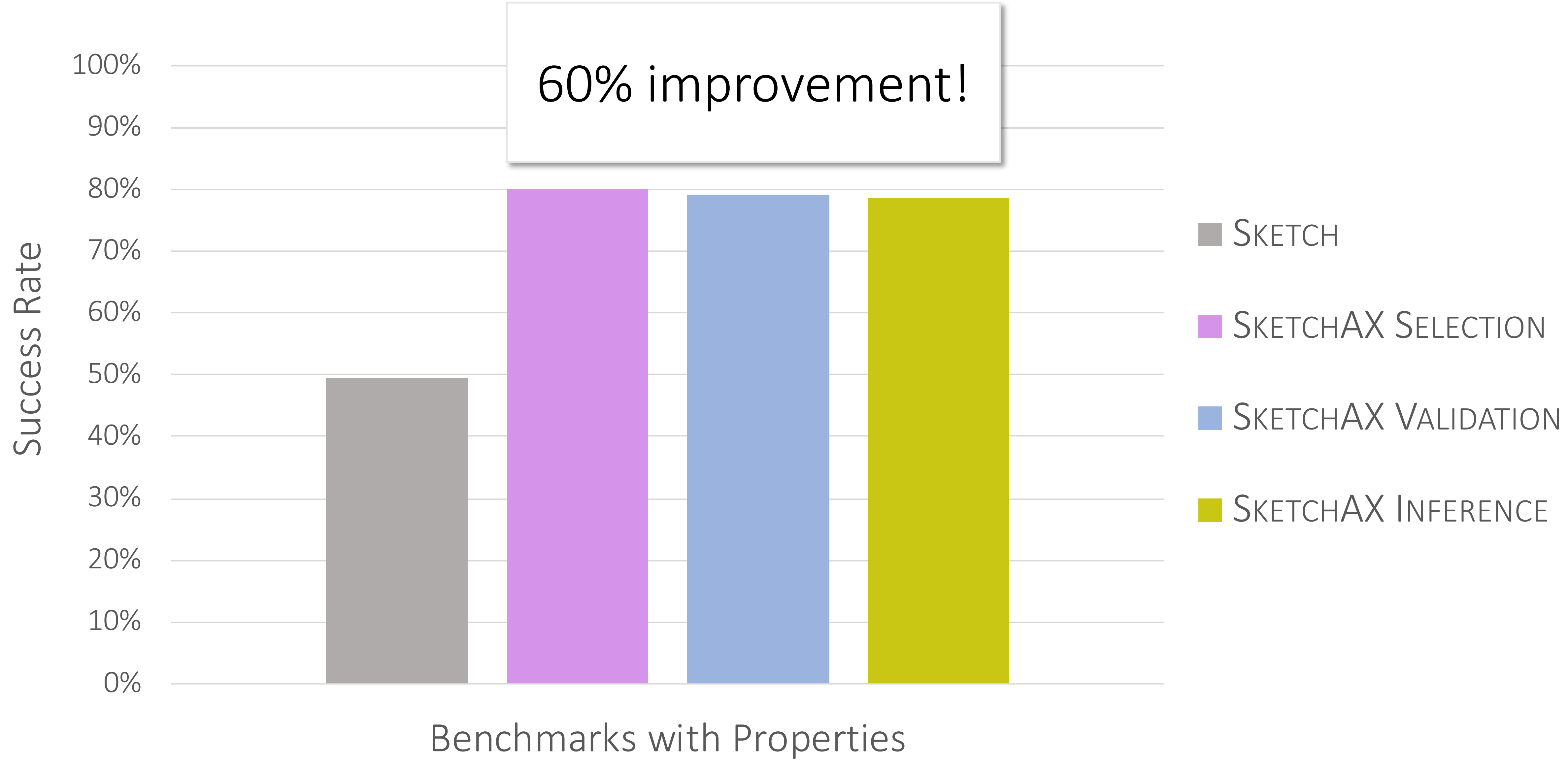

Semantic Bias!
(0, 10, 2) ↦ 10
(-1, 10, 20) ↦ 20
(-1, -2, -3) ↦ -1
(2, 10, 0) ↦ 10
(20, -1, 10) ↦ 20
(-2, -1, -3) ↦ -1
⋮
Permutation Invariance
Program/Search Space
int max (int x,int y,int z)
int m = z;
if (z <= y) m = y;
if (m < x) m = x;
return m;

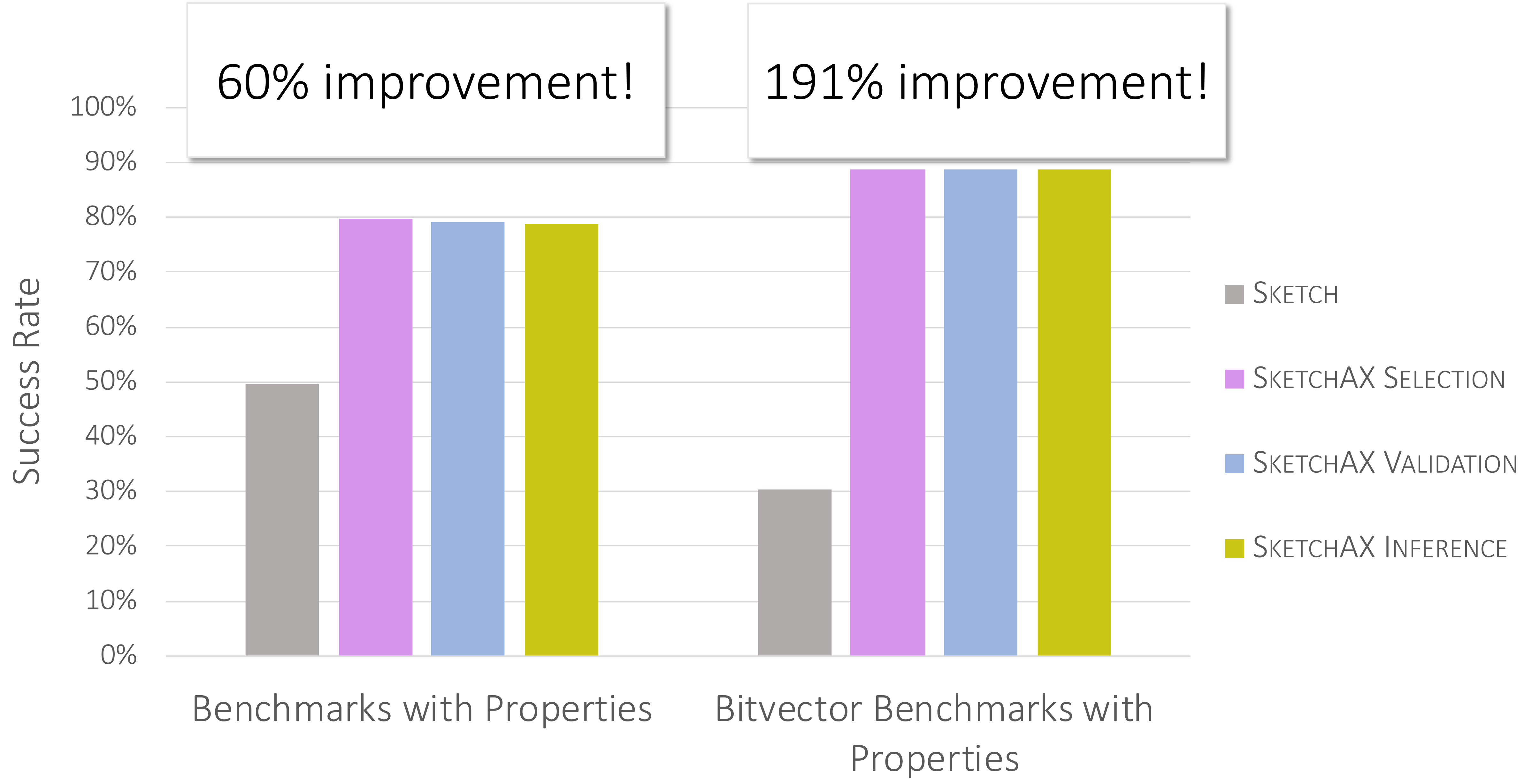
Key Strategy

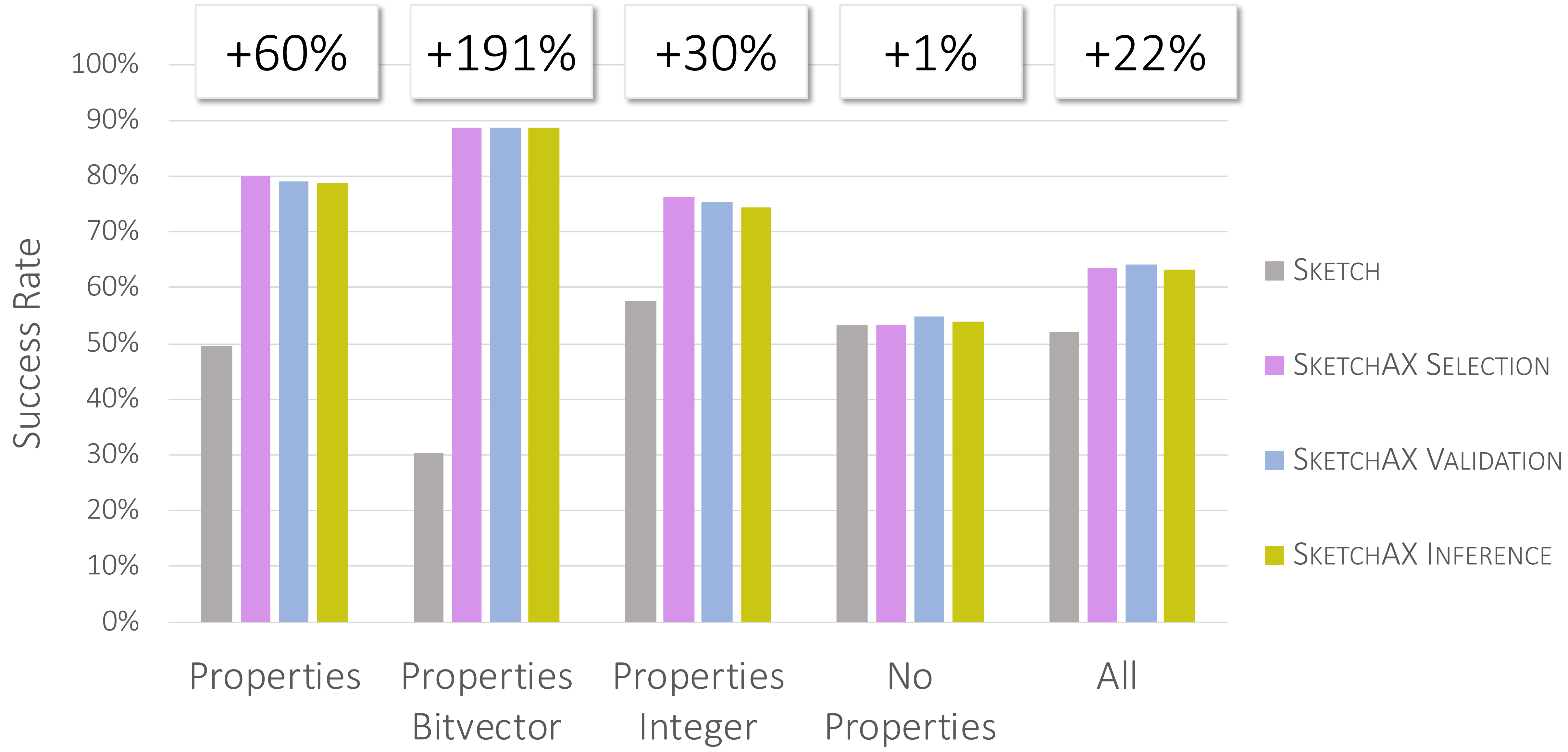
1. Augment examples by applying properties

2. Use PBE engine to synthesize program from augmented examples


Partial Programs
+
Reference Implementations
197 Benchmarks: SKETCH, SYGUS
Bitvectors, Integers, Integer arrays
SKETCHAX
SKETCH
Set of 8 properties


60% improvement!
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
Success Rate
Benchmarks with Properties
SKETCH
SKETCHAX SELECTION
SKETCHAX VALIDATION
SKETCHAX INFERENCE


60% improvement!
191% improvement!
Success Rate
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
Benchmarks with Properties
Bitvector Benchmarks with Properties
SKETCH
SKETCHAX SELECTION
SKETCHAX VALIDATION
SKETCHAX INFERENCE


+60%
+191%
+30%
+1%
+22%
Success Rate
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
Properties
Properties Bitvector
Properties Integer
No Properties
All
SKETCH
SKETCHAX SELECTION
SKETCHAX VALIDATION
SKETCHAX INFERENCE

Your project – Apply MANTIS to your domain!

Example Domains

- Imperative programs over simple datatypes (integers, Booleans, arrays, matrices)
- Heap-manipulating (recursive) imperative programs
- Functional programs with algebraic datatypes
- Relational queries (SQL, Datalog)
- String-manipulating programs
- Table-manipulating programs
- Parser synthesis
- Map-reduce distributed programs
- Web programming
- Regular expressions
- …

1. Identify domain

2. Identify applicable semantic properties

3. Use existing inductive synthesizer as black-box

or

Build your own search algorithm/synthesizer!

Summary

Today

- Introduction to program synthesis
- Project description

Next

- Propositional logic
- Normal Forms