

Program Reasoning

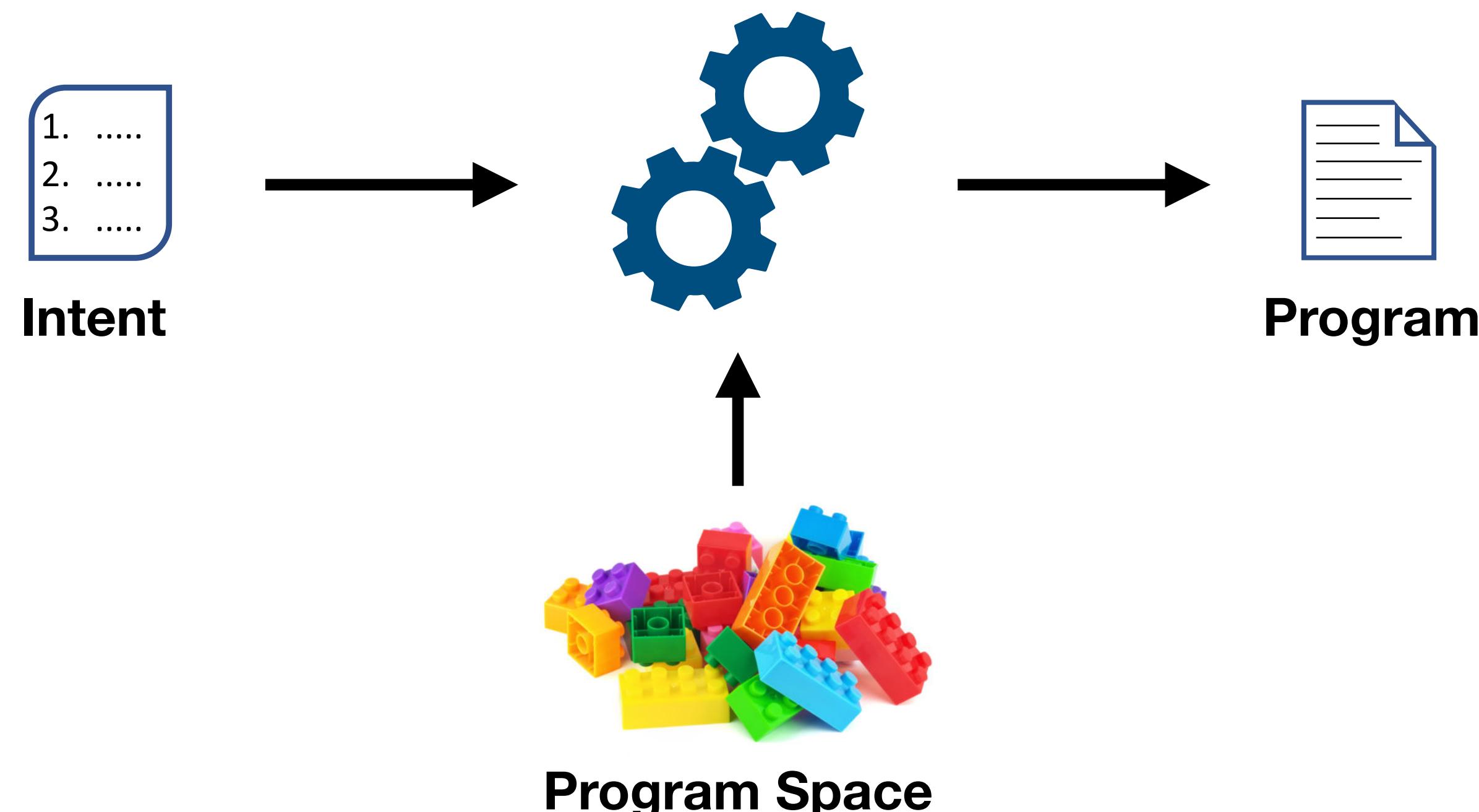
9. Overview of Program Synthesis

Kihong Heo



Program Synthesis

- A task of automatically finding a program
 - that satisfy user intent from the underlying program space



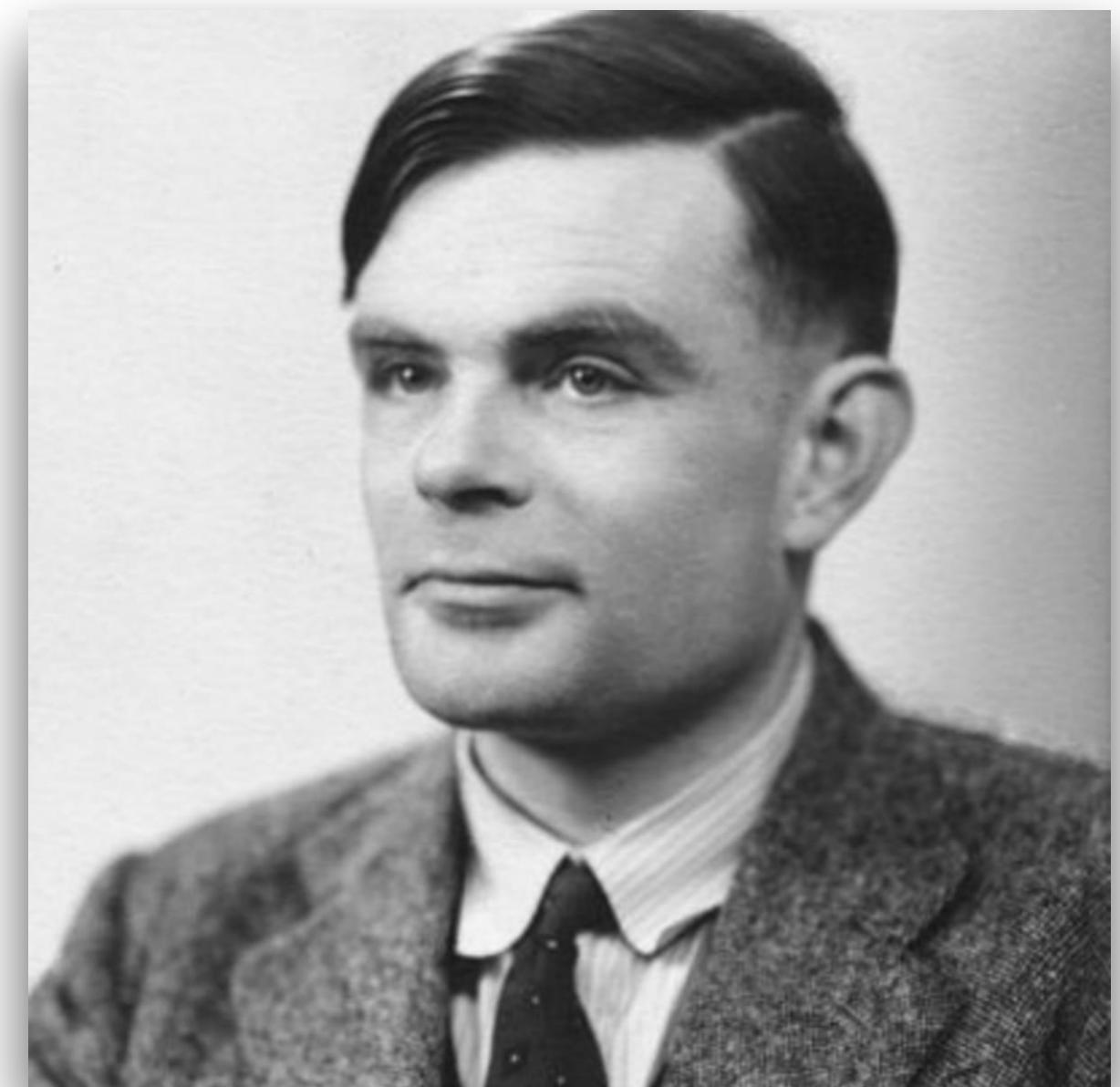
History

“Instruction tables will have to be made up by mathematicians with computing experience and perhaps a certain puzzle-solving ability.

...

There need be no real danger of it ever becoming a drudge, for any processes that are quite mechanical may be turned over to the machine itself.”

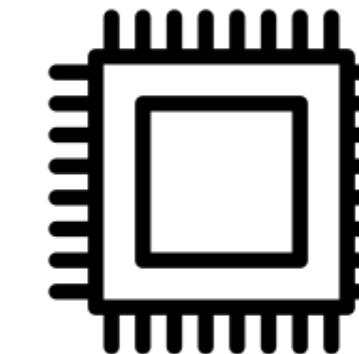
- A. M. Turing, “Proposed Electronic Calculator”, 1946



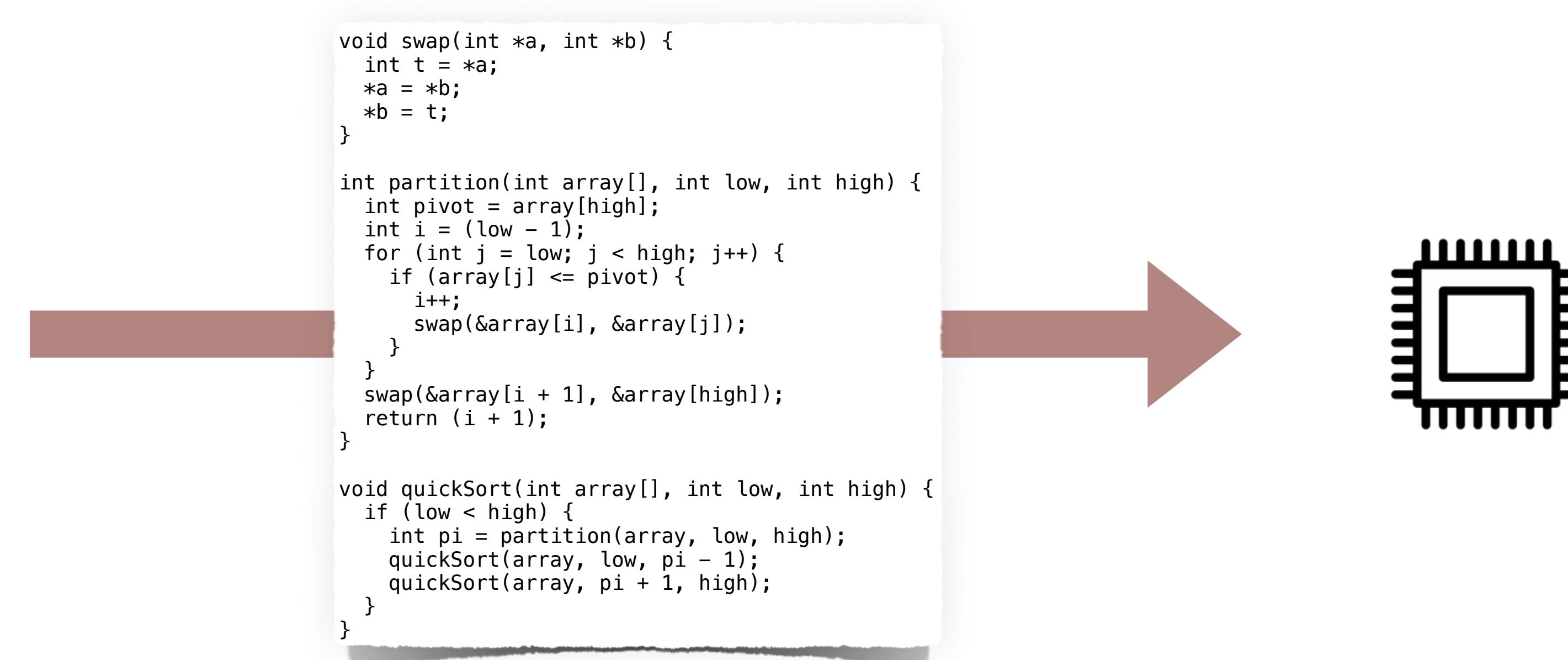
History of Programming



```
quickSort:  
.LFB2:  
.cfi_startproc  
endbr64  
pushq %rbp  
.cfi_offset 16  
.cfi_offset 6, -16  
movq %rsp, %rbp  
.cfi_def_cfa_register 6  
subq $32, %rsp  
movq %rdi, -24(%rbp)  
movl %esi, -28(%rbp)  
movl %edx, -32(%rbp)  
movl -28(%rbp), %eax  
cmpl -32(%rbp), %eax  
jge .L9  
movl -32(%rbp), %edx  
movl -28(%rbp), %ecx  
movq -24(%rbp), %rax  
movl %ecx, %esi  
movq %rax, %rdi  
call partition  
movl %eax, -4(%rbp)  
movl -4(%rbp), %eax  
leal -1(%rax), %edx  
movl -28(%rbp), %ecx  
movq -24(%rbp), %rax  
movl %ecx, %esi  
movq %rax, %rdi  
call quickSort  
movl -4(%rbp), %eax  
leal 1(%rax), %ecx  
movl -32(%rbp), %edx  
movq -24(%rbp), %rax  
movl %ecx, %esi  
movq %rax, %rdi  
call quickSort  
.L9:  
nop  
leave  
.cfi_def_cfa 7, 8  
ret  
.cfi_endproc
```



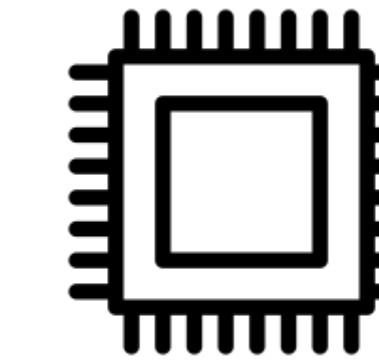
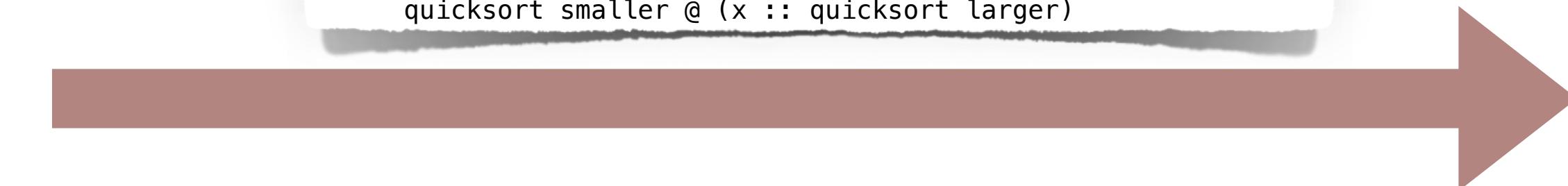
History of Programming



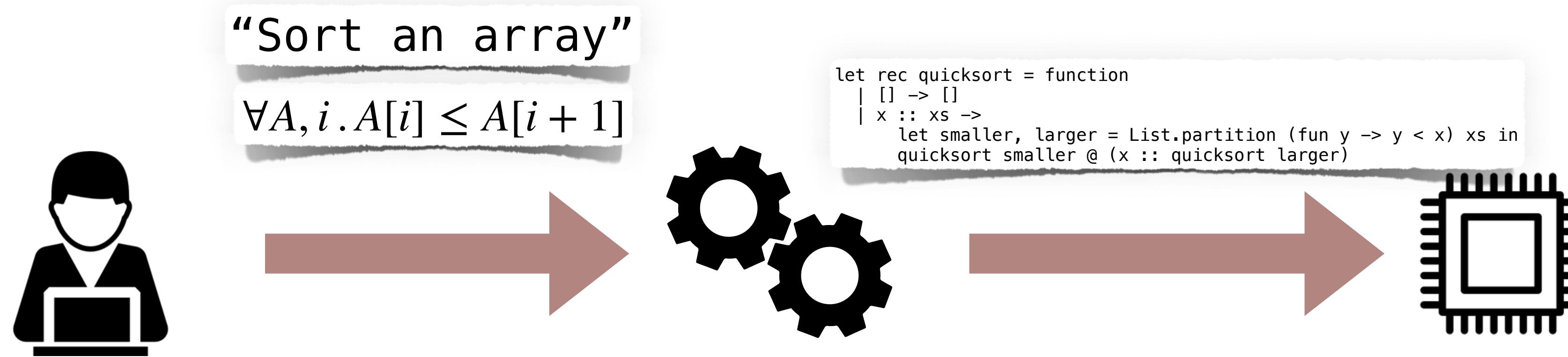
History of Programming



```
let rec quicksort = function
| [] -> []
| x :: xs ->
  let smaller, larger = List.partition (fun y -> y < x) xs in
  quicksort smaller @ (x :: quicksort larger)
```



Future of Programming



Why Synthesis?

- Human makes mistakes
- #developers << #programming tasks (feature reqs, bug fixes)
 - E.g., Linux kernel's open bug reports
- Programming for everybody



```
...
hashOut.data = hashes + SSL_MD5_DIGEST_LEN;
hashOut.length = SSL_SHA1_DIGEST_LEN;
if ((err = SSLFreeBuffer(&hashCtx)) != 0)
    goto fail;
if ((err = ReadyHash(&SSLHashSHA1, &hashCtx)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
    goto fail;
if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
    goto fail;

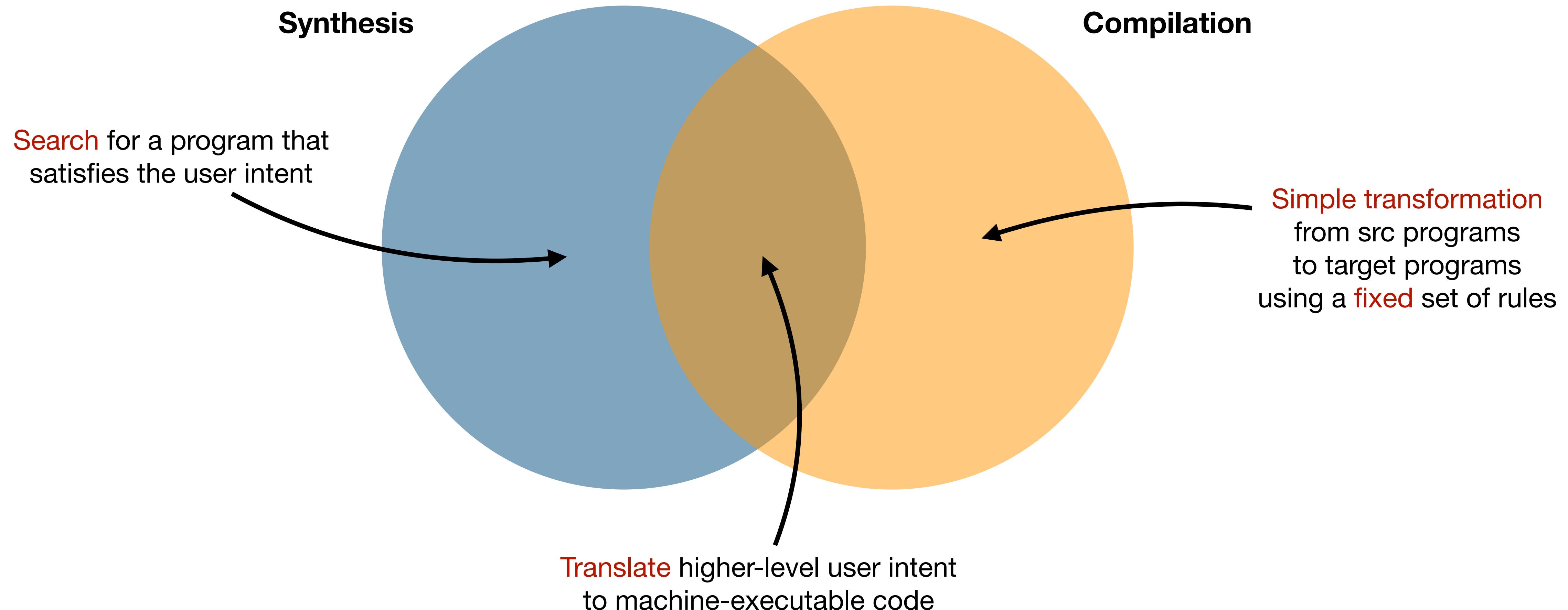
err = sslRawVerify(...);

fail:
...
return err;
```

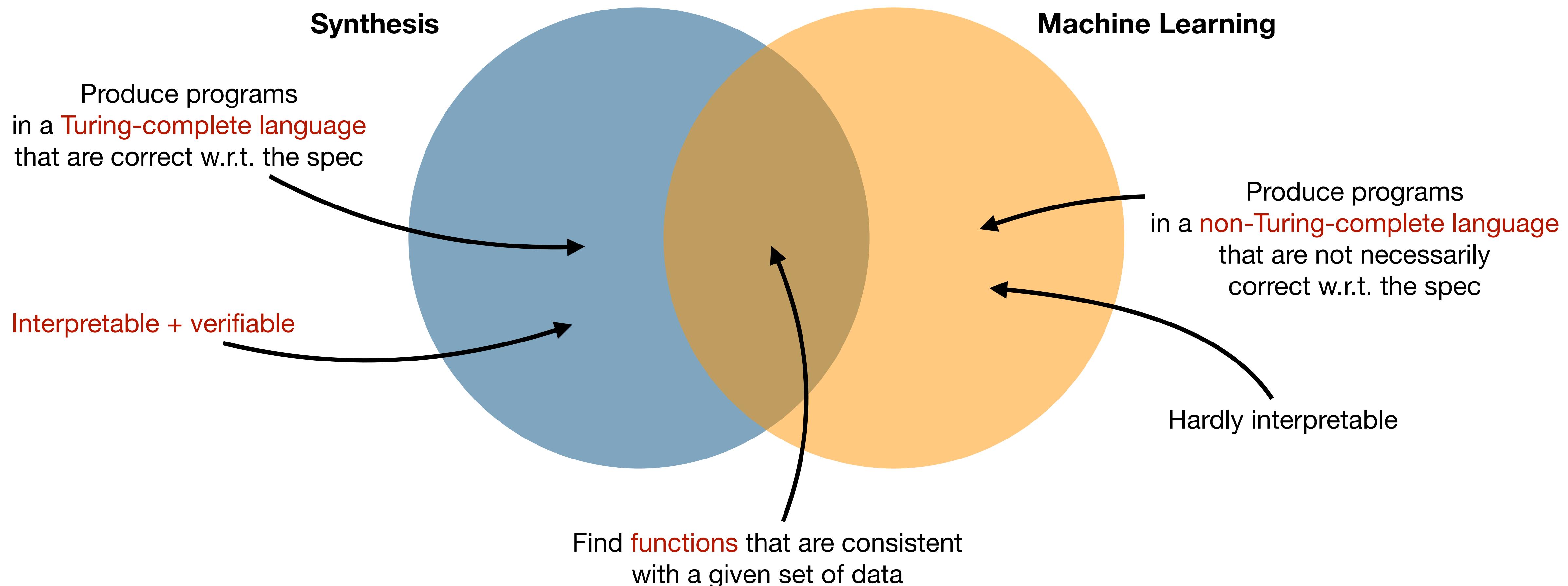


goto fail, 2014
MacOS / iOS
CVE-2014-1266

Synthesis vs Compilation

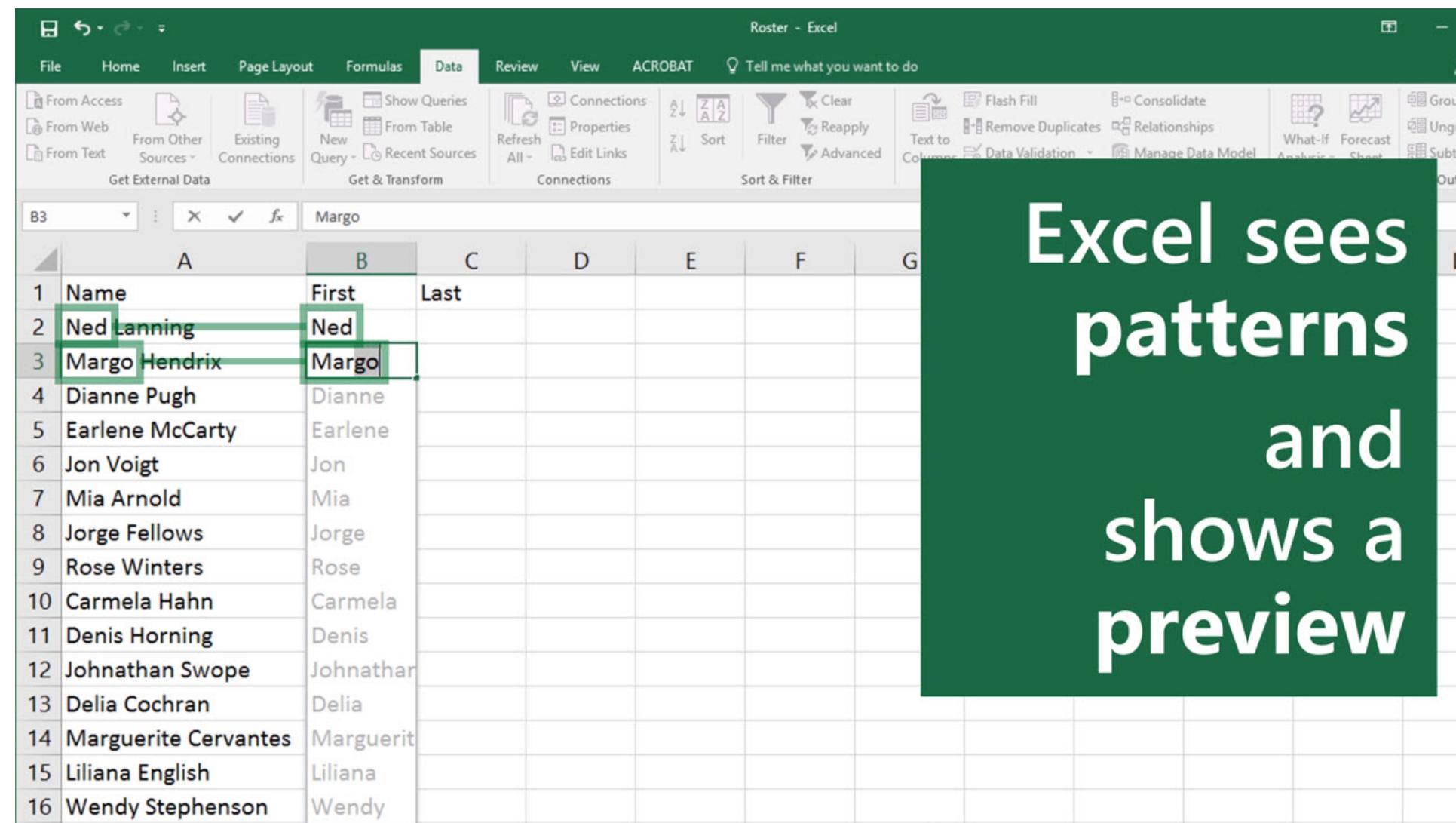


Synthesis vs Machine Learning



Application: End-user Programming (1)

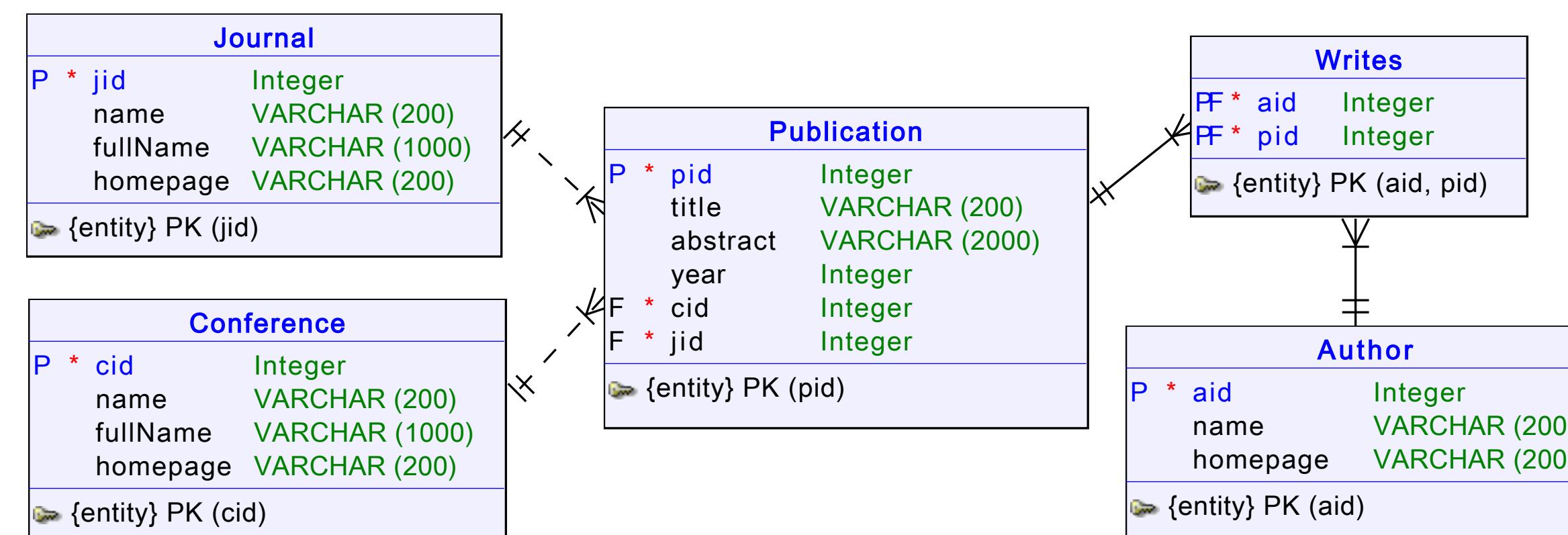
- String transformation (e.g., MS Excel's FlashFill)



Automating String Processing in Spreadsheets using Input-Output Examples, POPL'11

Application: End-user Programming (2)

- SQL query synthesis



“Find the number of papers in OOPSLA 2020”

```
SELECT count(Publication.pid)
FROM Publication JOIN Conference ON Publication.cid = Conference.cid
WHERE Conference.name = "OOPSLA" AND Publication.year = 2010
```

SQLizer: Query Synthesis from Natural Language, OOPSLA'17

Application: Super Optimization

- Montgomery multiplication kernel from the OpenSSL RSA library

```
1 # gcc -O3                      1 # STOKE
2                                         2
3 .L0:                                3 .L0:
4  movq rsi, r9                         4  shlq 32, rcx
5  movl ecx, ecx                         5  movl edx, edx
6  shrq 32, rsi                          6  xorq rdx, rcx
7  andl 0xffffffff, r9d                 7  movq rcx, rax
8  movq rcx, rax                         8  mulq rsi
9  movl edx, edx                         9  addq r8, rdi
10 imulq r9, rax                        10 adcq 0, rdx
11 imulq rdx, r9                         11 addq rdi, rax
12 imulq rsi, rdx                        12 adcq 0, rdx
13 imulq rsi, rcx                        13 movq rdx, r8
14 addq rdx, rax                        14 movq rax, rdi
15 jae .L2
16 movabsq 0x100000000, rdx
17 addq rdx, rcx
18 .L2:
19  movq rax, rsi
20  movq rax, rdx
21  shrq 32, rsi
22  salq 32, rdx
23  addq rsi, rcx
24  addq r9, rdx
25  adcq 0, rcx
26  addq r8, rdx
27  adcq 0, rcx
28  addq rdi, rdx
29  adcq 0, rcx
30  movq rcx, r8
31  movq rdx, rdi
```

Stochastic Super Optimization, ASPLOS'13

Application: Reverse Engineering

- Binary lifting

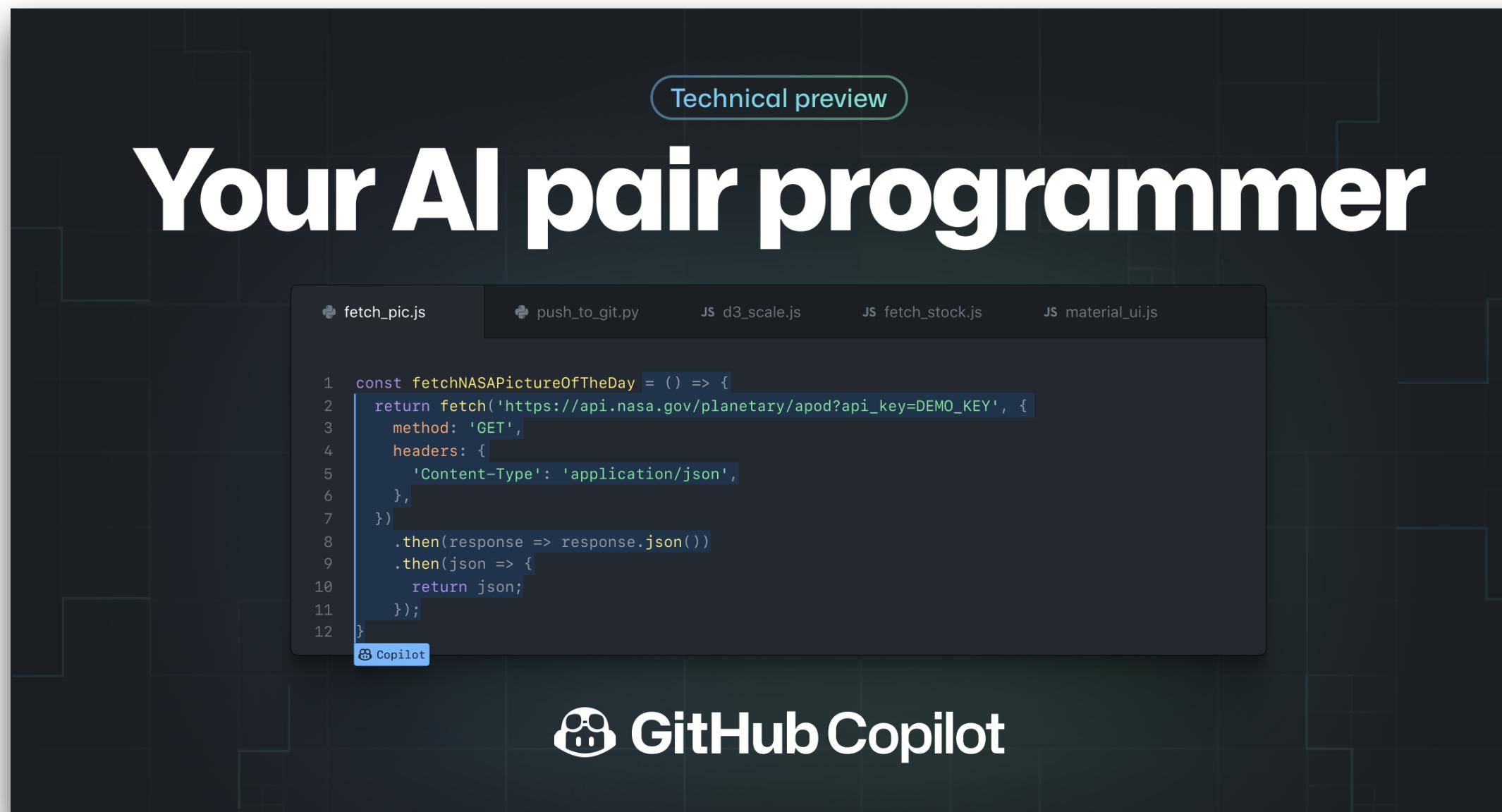


```
#include <Halide.h>
#include <vector>
using namespace std;
using namespace Halide;

int main() {
    Var x_0;
    Var x_1;
    ImageParam input_1(UInt(8), 2);
    Func output_1;
    output_1(x_0, x_1) =
        cast<uint8_t>((((((2 +
            (2 * cast<uint32_t>(input_1(x_0+1, x_1+1)))
            cast<uint32_t>(input_1(x_0, x_1+1)) +
            cast<uint32_t>(input_1(x_0+2, x_1+1)))
        >> cast<uint32_t>(2))) & 255));
    vector<Argument> args;
    args.push_back(input_1);
    output_1.compile_to_file("halide_out_0", args)
    return 0;
}
```

Helium: Lifting High-Performance Stencil Kernels from Stripped x86 Binaries to Halide DSL Code, PLDI'15

Application: Code Suggestion

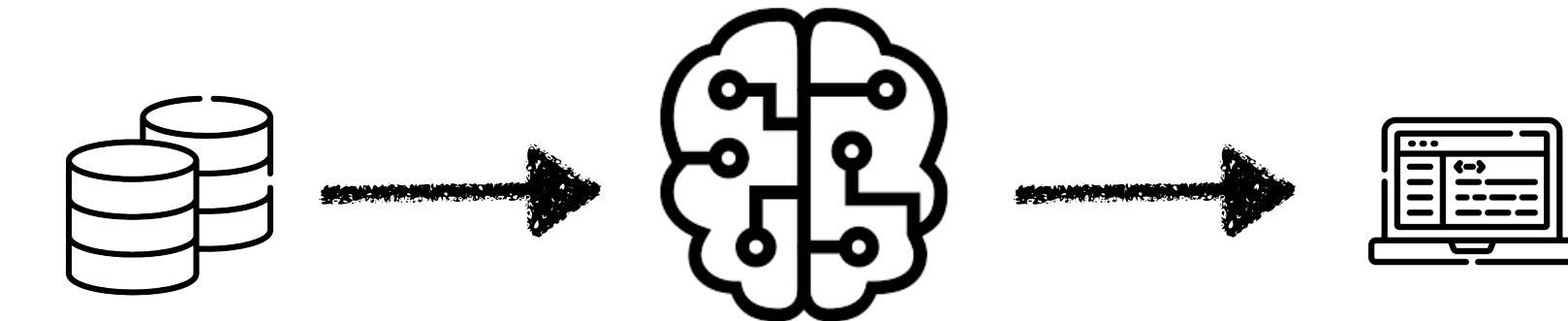


The screenshot shows the GitHub Copilot interface with a dark theme. At the top, there are tabs for "sentiments.ts", "write_sql.go", "parse_expenses.py", and "addresses.rb". The main content area contains a Python script for parsing expense strings:

```
1 import datetime
2
3 def parse_expenses(expenses_string):
4     """Parse the list of expenses and return the list of triples (date, value, currency).
5     Ignore lines starting with #.
6     Parse the date using datetime.
7     Example expenses_string:
8         2016-01-02 -34.01 USD
9         2016-01-03 2.59 DKK
10        2016-01-03 -2.72 EUR
11    """
12     expenses = []
13     for line in expenses_string.splitlines():
14         if line.startswith("#"):
15             continue
16         date, value, currency = line.split(" ")
17         expenses.append((datetime.datetime.strptime(date, "%Y-%m-%d"),
18                           float(value),
19                           currency))
20
21     return expenses
```

A "Copilot" button is visible at the bottom right of the code editor.

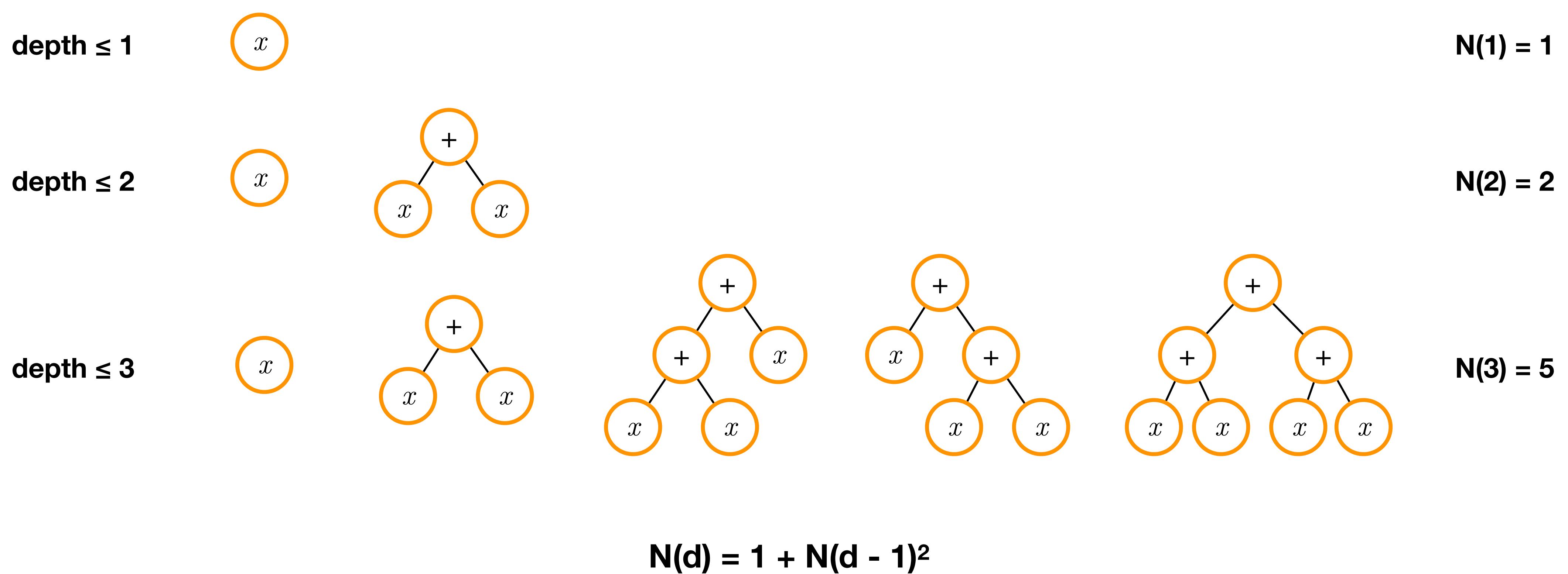
Two Directions: PL vs ML



- Semantics (logical reasoning)
- Correctness guaranteed
- Not very scalable
- Search algorithms, constraint solving
- Example: Excel FlashFill
- Likelihood (probabilistic reasoning)
- Correctness not guaranteed
- Very scalable
- Probabilistic inference
- Example: Copilot

Challenge 1: Search Space (1)

$$S \rightarrow x \mid S + S$$



Challenge 1: Search Space (2)

$$S \rightarrow x \mid S + S$$

$$N(d) = 1 + N(d - 1)^2$$

$$N(1) = 1$$

$$N(2) = 2$$

$$N(3) = 5$$

$$N(4) = 26$$

$$N(5) = 677$$

$$N(6) = 458330$$

$$N(7) = 210066388901$$

$$N(8) = 44127887745906175987802$$

$$N(9) = 1947270476915296449559703445493848930452791205$$

$$N(10) = 3791862310265926082868235028027893277370233152247388584761734150717768254410341175325352026$$

!!

*Examples from Nadia Polikarpova's slides

Challenge 2: Correctness (1)

A screenshot of a code editor window titled "Untitled-2". The code is a simple C++ function:

```
1 int check_buffer_overflow() {
2     int buffer[5];
3     buffer[6] = 0;
4     return 0;
5 }
```

The line `buffer[6] = 0;` is highlighted with a red rectangle, indicating it is the source of the buffer overflow error.

```
1 // read an image file
2 int read_file() {
3     FILE *fp;
4     int i, j, k;
5     int n;
6     int m;
7     int nc;
8     int nr;
9     int nb;
10    int nc_max;
11    int nr_max;
12    int nb_max;
13    int nc_min;
14    int nr_min;
15    int nb_min;
16    int nc_avg;
17    int nr_avg;
18    int nb_avg;
19    int nc_sum;
20    int nr_sum;
21    int nb_sum;
22    int nc_var;
23    int nr_var;
24    int nb_var;
25    int nc_std;
26    int nr_std;
27    int nb_std;
28    int nc_med;
29    int nr_med;
30    int nb_med;
31    int nc_mode;
32    int nr_mode;
33    int nb_mode;
34    int nc_min_index;
35    int nr_min_index;
36    int nb_min_index;
37    int nc_max_index;
38    int nr_max_index;
39    int nb_max_index;
40    int nc_med_index;
41    int nr_med_index;
42    int nb_med_index;
43    int nc_mode_index;
44    int nr_mode_index;
45    int nb_mode_index;
46    int nc_sum_index;
47    int nr_sum_index;
48    int nb_sum_index;
49    int nc_var_index;
50    int nr_var_index;
51    int nb_var_index;
52    int nc_std_index;
53    int nr_std_index;
54    int nb_std_index;
55    int nc_avg_index;
56    int nr_avg_index;
57    int nb_avg_index;
58    int nc_min_index_index;
59    int nr_min_index_index
```

Challenge 2: Correctness (2)

- Recurring SW vulnerabilities by human developers
- Example: gimp-2.6.7 (CVE-2009-1570)

```
long ToL (char *pbuffer) { return (puffer[0] | puffer[1]<<8 | puffer[2]<<16 | puffer[3]<<24); } 3  
short ToS (char *pbuffer) { return ((short)(puffer[0] | puffer[1]<<8)); }  
gint32 ReadBMP (gchar *name, GError **error) { 1  
    if (fread(buffer, Bitmap_File_Head.biSize - 4, fd) != 0)  
        FATALP ("BMP: Error reading BMP file header #3");  
    Bitmap_Head.biWidth = ToL (&buffer[0x00]); 2  
    Bitmap_Head.biBitCnt = ToS (&buffer[0x0A]);  
  
    rowbytes = ((Bitmap_Head.biWidth * Bitmap_Head.biBitCnt - 1) / 32) * 4 + 4; 4  
    image_ID = ReadImage (rowbytes); 5  
    ...  
}  
  
gint32 ReadImage (int rowbytes) { 6  
    buffer = malloc(rowbytes); // malloc with overflowed size  
    ...  
}
```

Challenge 2: Correctness (3)

- Recurring SW vulnerabilities by human developers
- Example: sam2p-0.49.4 (CVE-2017-1663)

```
long ToL (char *pbuffer) { return (puffer[0] | puffer[1]<<8 | puffer[2]<<16 | puffer[3]<<24); } 3  
short ToS (char *pbuffer) { return ((short)(puffer[0] | puffer[1]<<8)); }  
  
bitmap_type bmp_load_image (FILE* filename) { 1  
    if (fread(buffer, Bitmap_File_Head.biSize - 4, fd) != 0)  
        FATALP ("BMP: Error reading BMP file header #3");  
    Bitmap_Head.biWidth = ToL (&buffer[0x00]); 2  
    Bitmap_Head.biBitCnt = ToS (&buffer[0x0A]);  
  
    rowbytes = ((Bitmap_Head.biWidth * Bitmap_Head.biBitCnt - 1) / 32) * 4 + 4; 4  
    image.bitmap = ReadImage (rowbytes); 5  
    ...  
}  
  
unsigned char* ReadImage (int rowbytes) { 6  
    unsigned char *buffer = (unsigned char*) new char[rowbytes]; // malloc with overflowed size  
    ...  
}
```

Challenge 2: Correctness (3)

- Recurring SW vulnerabilities by AI developers
- Example (feat. Copilot):

```
int toLong(char *buffer) {
    return (buffer[0]) | (buffer[1] << 8) | (buffer[2] << 16) | (buffer[3] << 24);
}

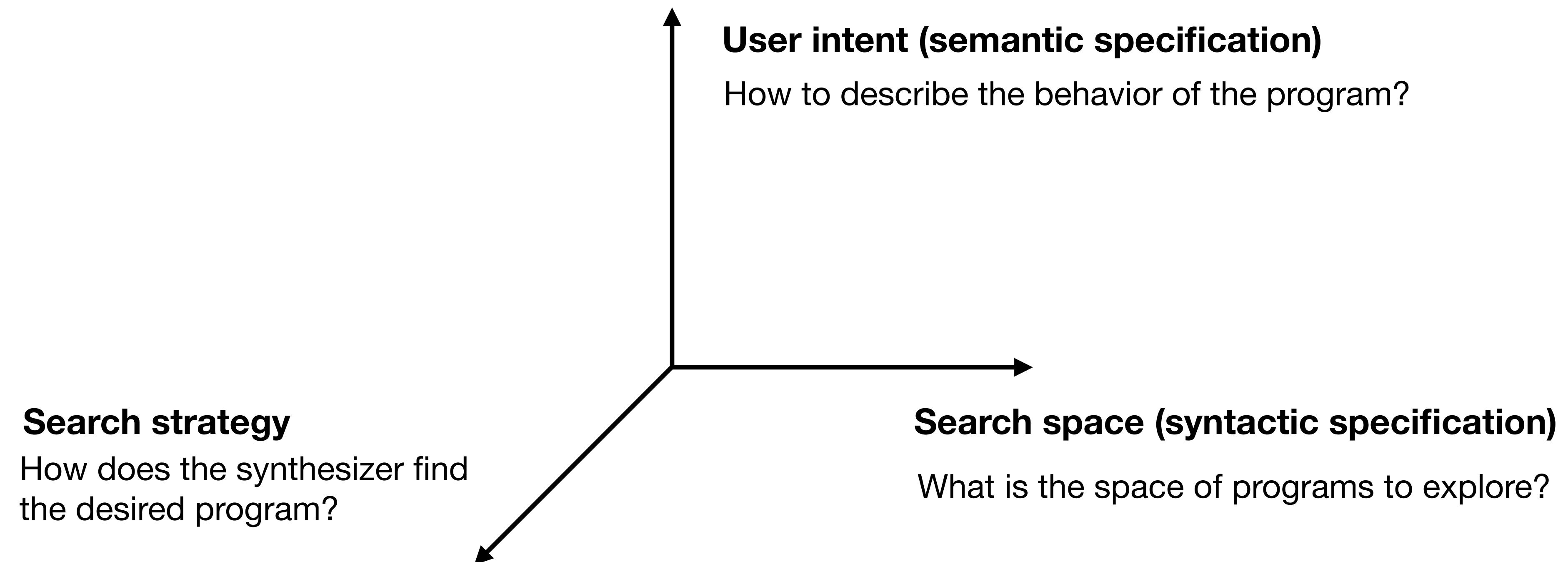
int f(char *name) {
    int width, height, area;
    char buffer[10];
    FILE *fd = fopen(name, "rb");
    fread(buffer, 10, 1, fd);
    fclose(fd);

    // Copilot, fill it!
    width = toLong(buffer + 18);
    height = toLong(buffer + 22);
    area = width * height;
```

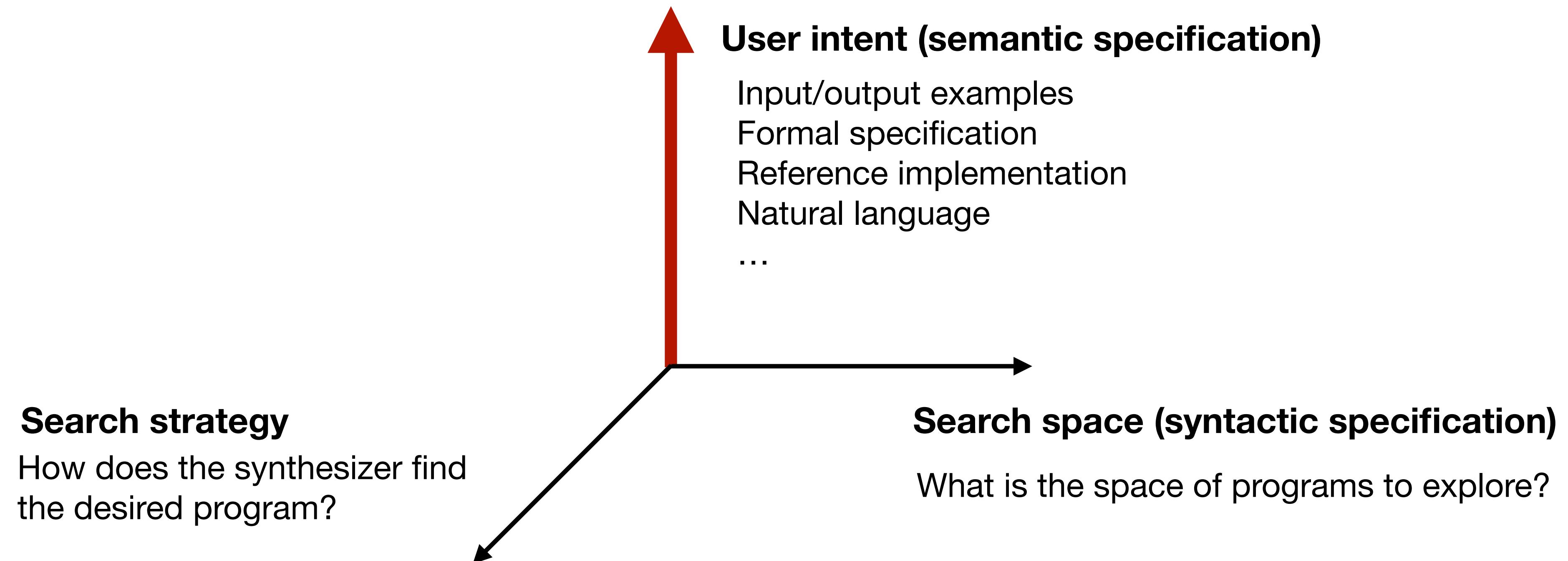
Grand Challenge

- How to achieve scalable and correct program synthesizers?
 - Logical inference and search algorithms are enough?
 - Large language models are enough?
- In this semester
 - “PL-based” techniques: synthesize a program that satisfies a given specification
- After this semester
 - Hope you develop a new to combine PL and ML

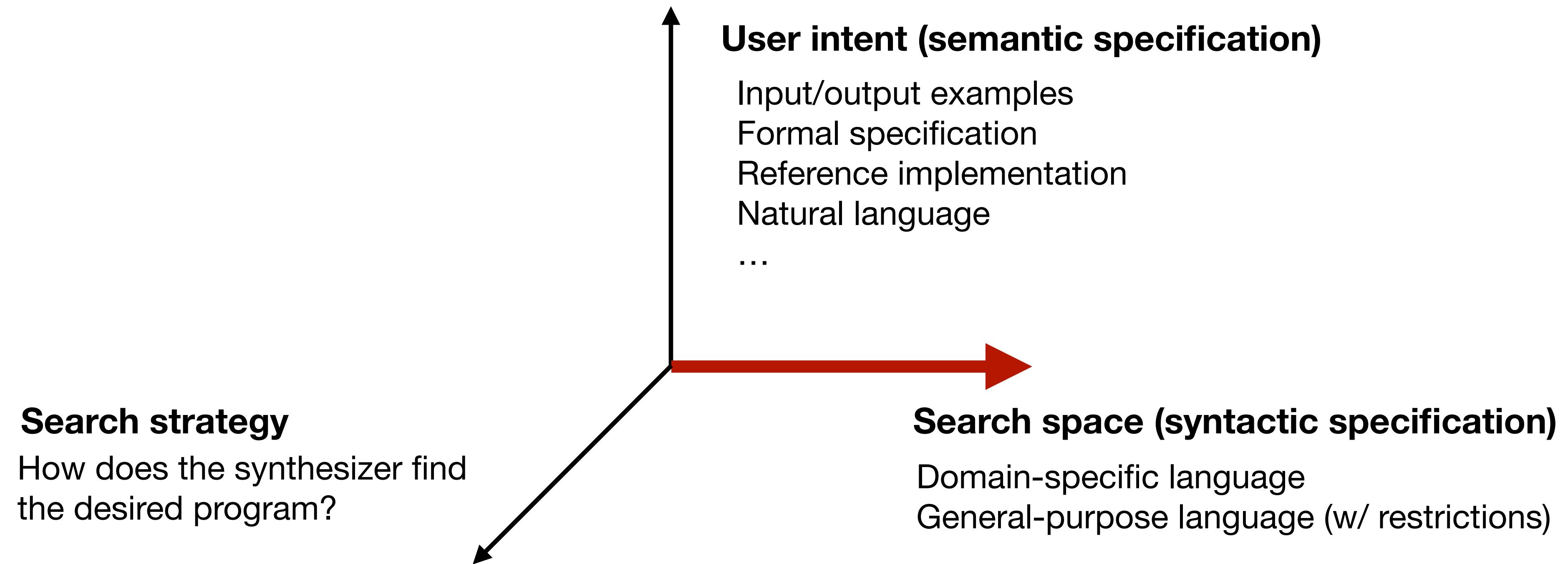
Dimensions in Program Synthesis



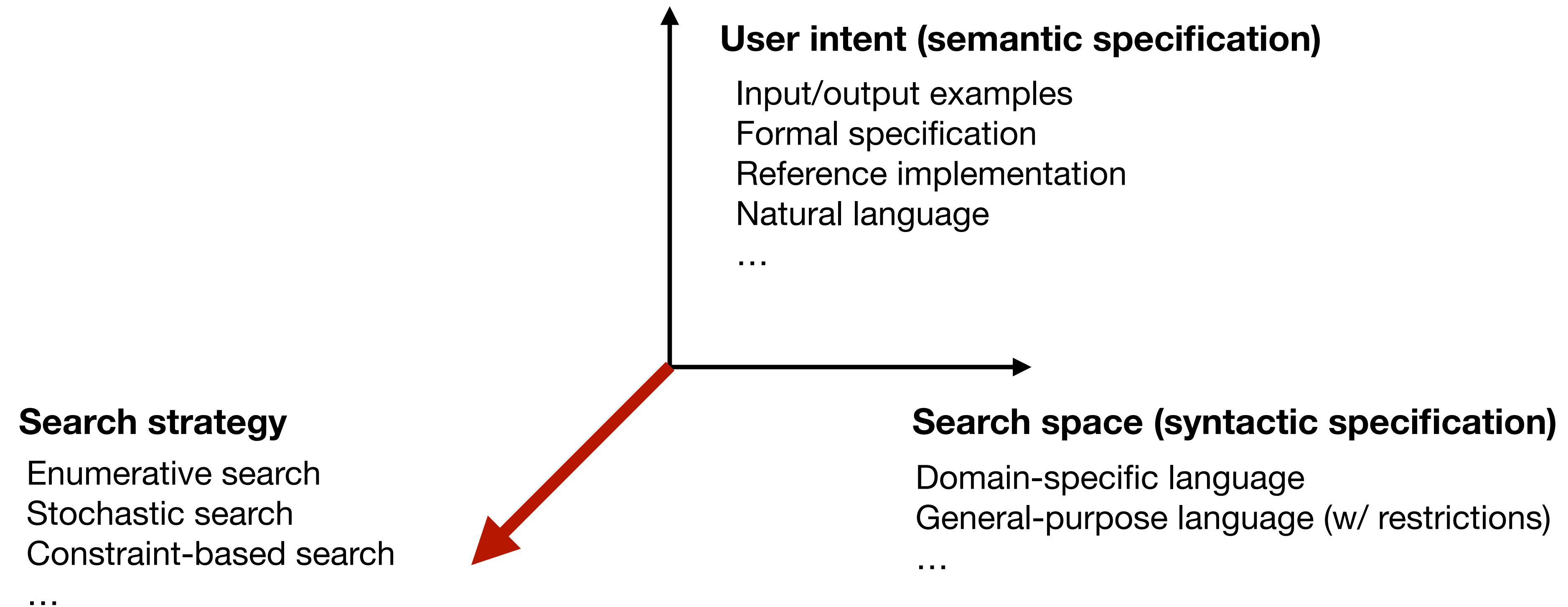
Dimensions in Program Synthesis



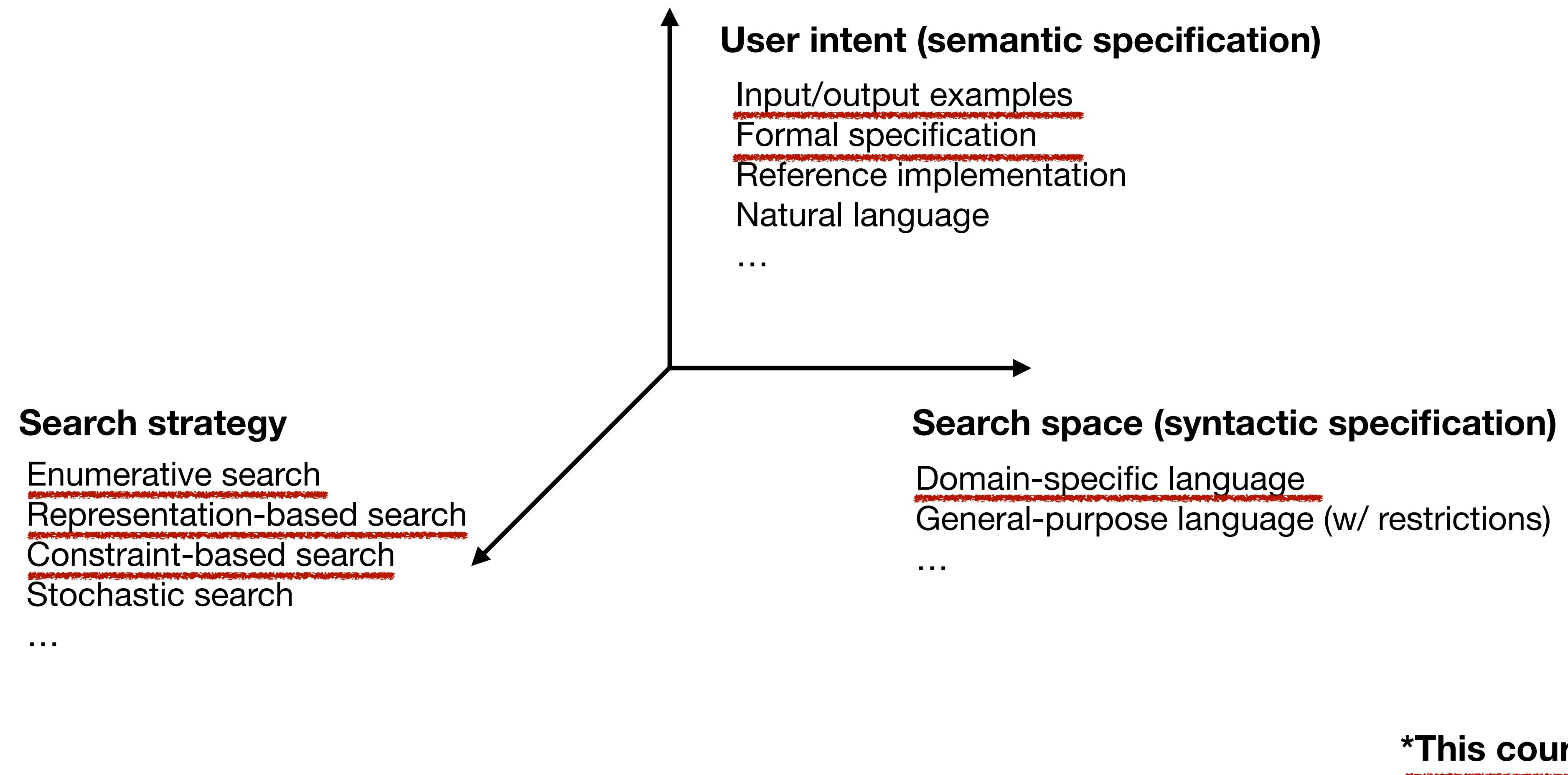
Dimensions in Program Synthesis



Dimensions in Program Synthesis



Dimensions in Program Synthesis



Summary

- Program synthesis: **automated programming** systems (the holy grail of computer science)
- Opportunities: many applications in software engineering, compiler, security, etc
- Principles: syntactic / semantics specification, search strategy
- Challenges: scalability, correctness, expressiveness, etc