Advanced Software Security

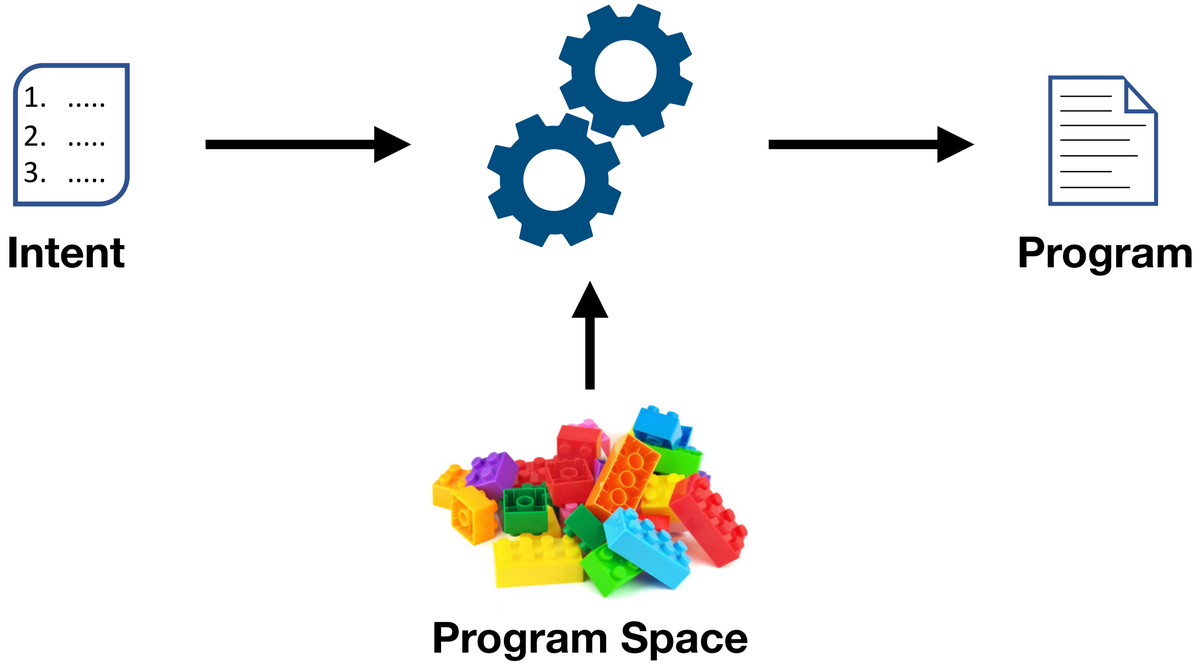
2. Introduction to Program Synthesis

Kihong Heo

KAIST

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.

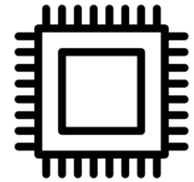
...

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

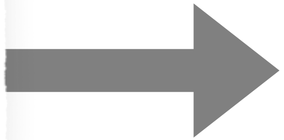
History of Programming

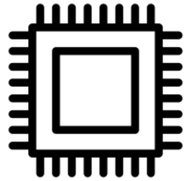
```
quickSort:
.LFB2:
        .cfi_startproc
        endbr64
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        subq    $32, %rsp
        movq    %rdi, -24(%rbp)
        movl    %esi, -28(%rbp)
        movl    %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
```



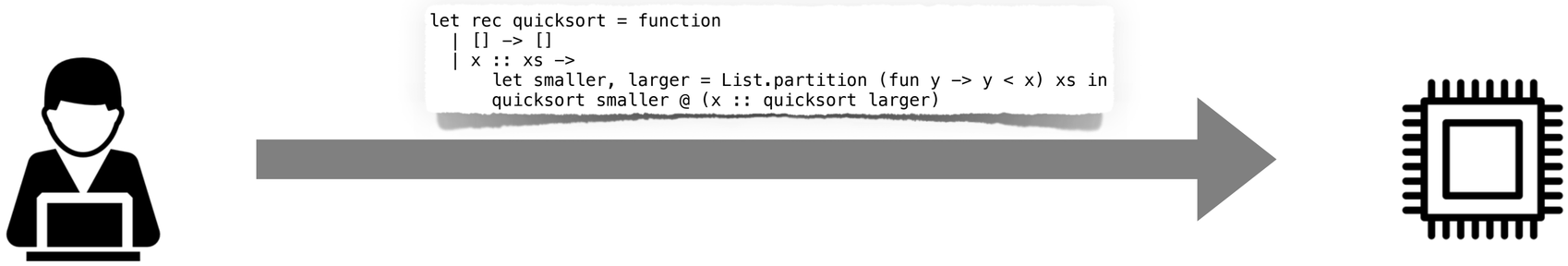
History of Programming

```
void swap(int *a, int *b) {
  int t = *a;
  *a = *b;
  *b = t;
}

int partition(int array[], int low, int high) {
  int pivot = array[high];
  int i = (low - 1);
  for (int j = low; j < high; j++) {
    if (array[j] <= pivot) {
      i++;
      swap(&array[i], &array[j]);
    }
  }
  swap(&array[i + 1], &array[high]);
  return (i + 1);
}

void quickSort(int array[], int low, int high) {
  if (low < high) {
    int pi = partition(array, low, high);
    quickSort(array, low, pi - 1);
    quickSort(array, pi + 1, high);
  }
}
```

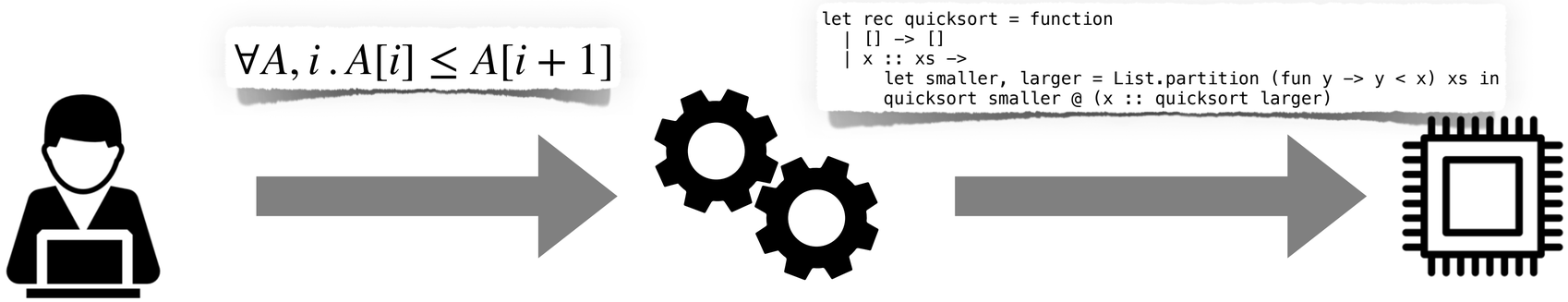




History of Programming



Future of Programming



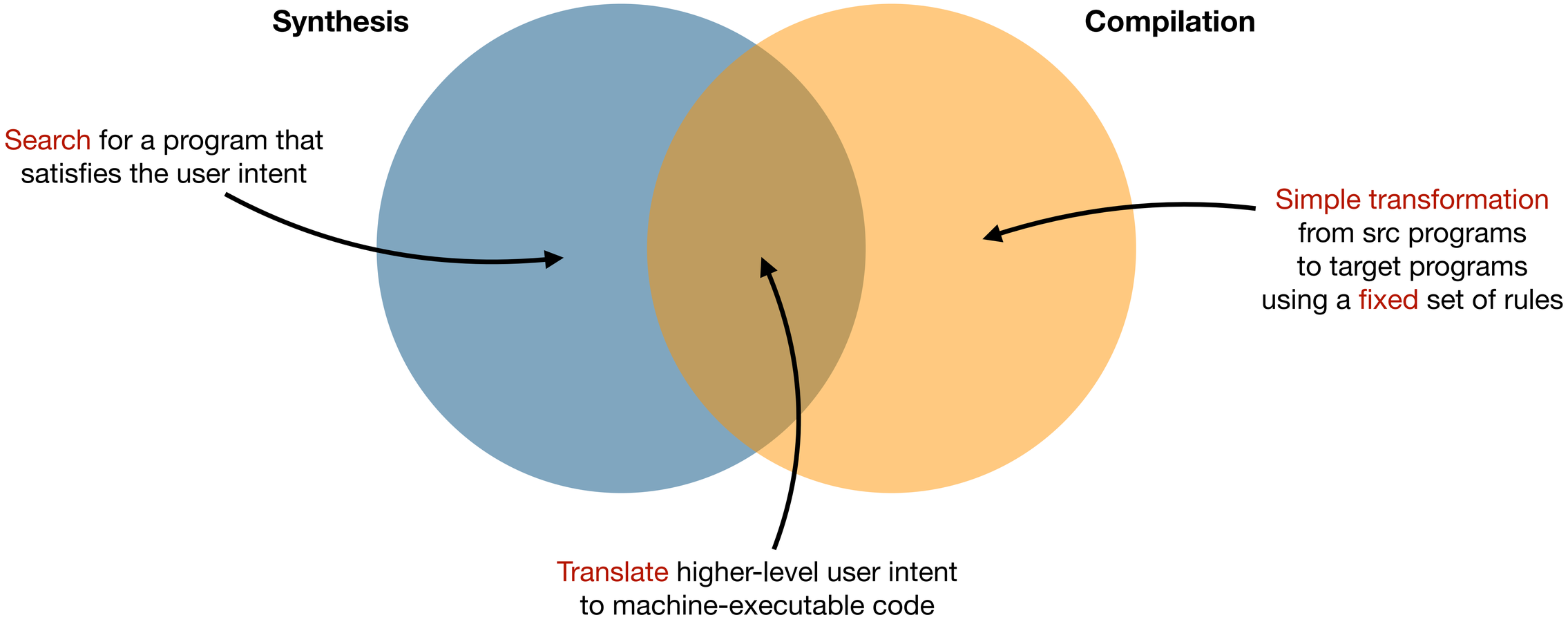
Why Synthesis?

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

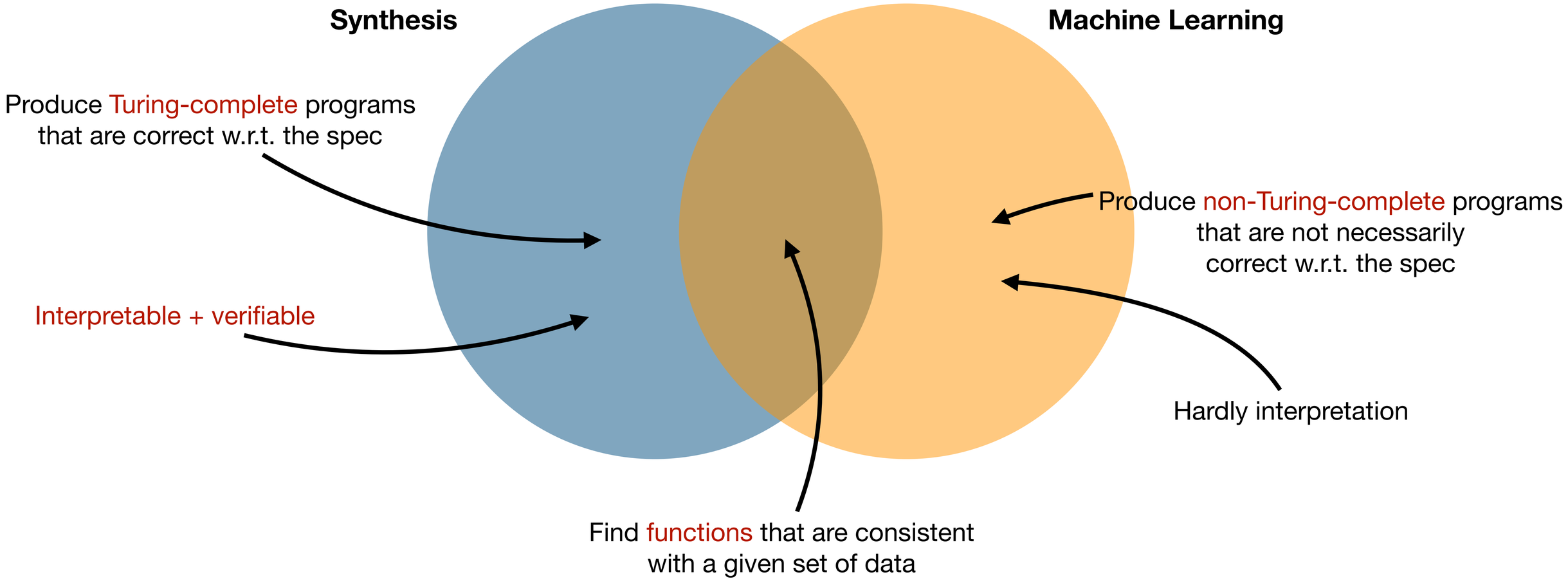

```
  ...
  hashOut.data = hashes + SSL_MD5_DIGEST_LEN;
  hashOut.length = SSL_SHA1_DIGEST_LEN;
  if ((err = SSLFreeBuffer(&hashCtx)) != 0)
      goto fail;
  if ((err = ReadyHash(&SSLHashSHA1, &hashCtx)) != 0)
      goto fail;
  if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
      goto fail;
  if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
      goto fail;
  if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
      goto fail;
      goto fail;  /* MISTAKE! THIS LINE SHOULD NOT BE HERE */
  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

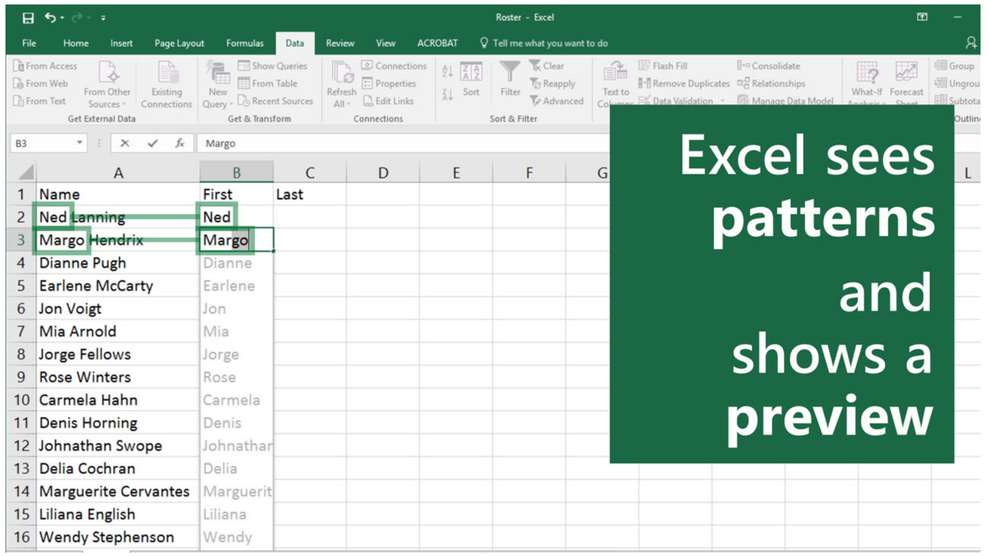


Synthesis vs Machine Learning



Application: End-user Programming

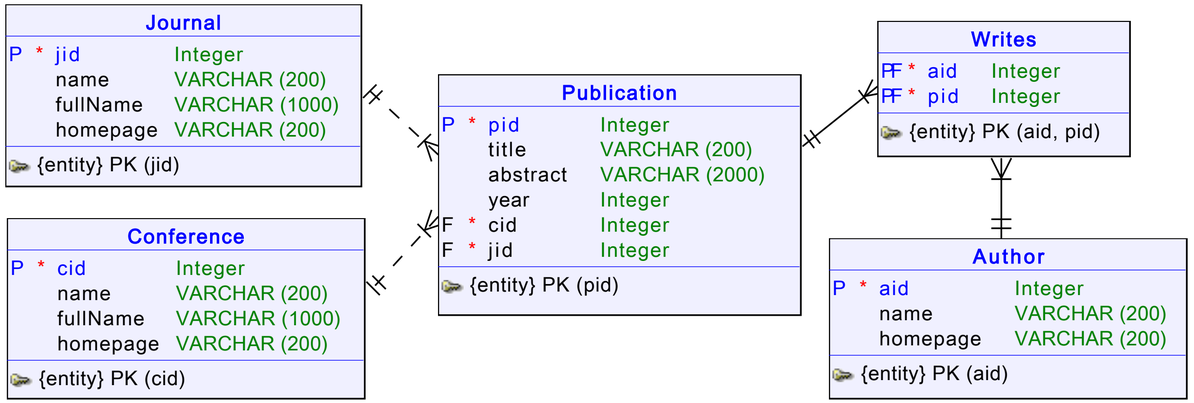
- String transformation (e.g., MS Excel's FlashFill)



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

Application: End-user Programming

- 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

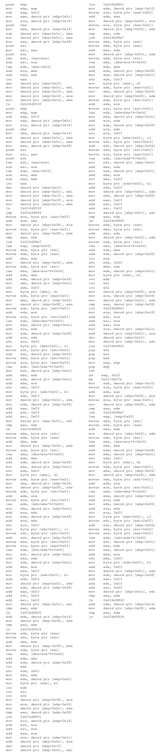
```
1  # gcc -O3
2
3  .L0:
4    movq rsi, r9
5    movl ecx, ecx
6    shrq 32, rsi
7    andl 0xffffffff, r9d
8    movq rcx, rax
9    movl edx, edx
10   imulq r9, rax
11   imulq rdx, r9
12   imulq rsi, rdx
13   imulq rsi, rcx
14   addq rdx, rax
15   jae .L2
16   movabsq 0x100000000, rdx
17   addq rdx, 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
```

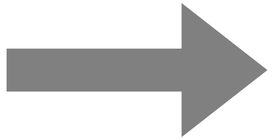
```
1  # STOKE
2
3  .L0:
4    shlq 32, rcx
5    movl edx, edx
6    xorq rdx, rcx
7    movq rcx, rax
8    mulq rsi
9    addq r8, rdi
10   adcq 0, rdx
11   addq rdi, rax
12   adcq 0, rdx
13   movq rdx, r8
14   movq rax, rdi
```

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

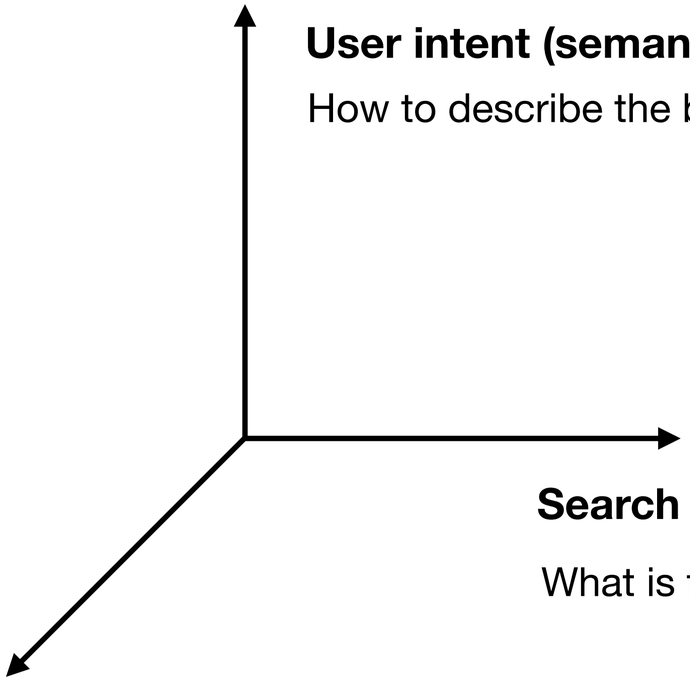
Challenges

시작하기 × int check_buffer_overflow() { Untitled-2 ●

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

```
1   // read an image file
2   int read_file() {
3       FILE *fp;
        int i, j, k;
        int n;
        int m;
        int nc;
        int nr;
        int nb;
        int nc_max;
        int nr_max;
        int nb_max;
        int nc_min;
        int nr_min;
        int nb_min;
        int nc_avg;
        int nr_avg;
        int nb_avg;
        int nc_sum;
        int nr_sum;
        int nb_sum;
        int nc_var;
        int nr_var;
        int nb_var;
        int nc_std;
        int nr_std;
        int nb_std;
        int nc_med;
        int nr_med;
        int nb_med;
        int nc_mode;
        int nr_mode;
        int nb_mode;
        int nc_min_index;
        int nr_min_index;
        int nb_min_index;
        int nc_max_index;
        int nr_max_index;
        int nb_max_index;
        int nc_med_index;
        int nr_med_index;
        int nb_med_index;
        int nc_mode_index;
        int nr_mode_index;
        int nb_mode_index;
        int nc_sum_index;
        int nr_sum_index;
        int nb_sum_index;
        int nc_var_index;
        int nr_var_index;
        int nb_var_index;
        int nc_std_index;
        int nr_std_index;
        int nb_std_index;
        int nc_avg_index;
        int nr_avg_index;
        int nb_avg_index;
        int nc_min_index_index;
        int nr_min
4   }
```

Dimensions in Program Synthesis



User intent (semantic specification)

How to describe the behavior of the program?

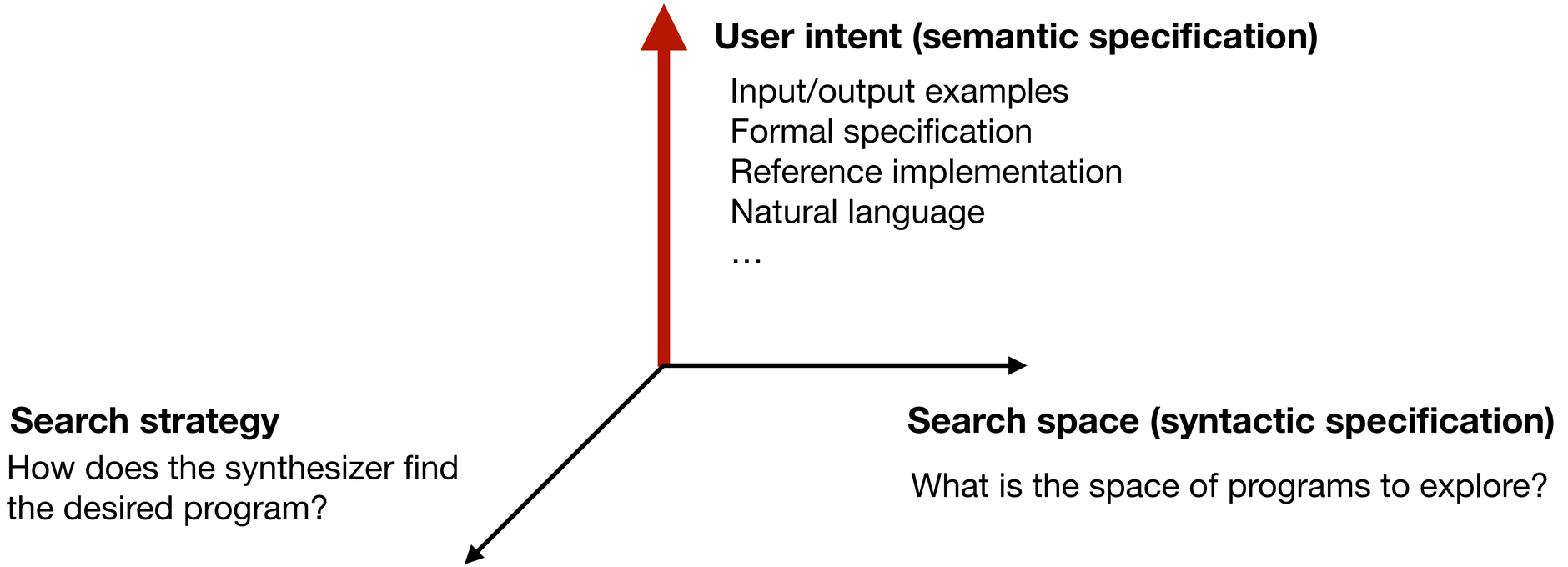
Search strategy

How does the synthesizer find the desired program?

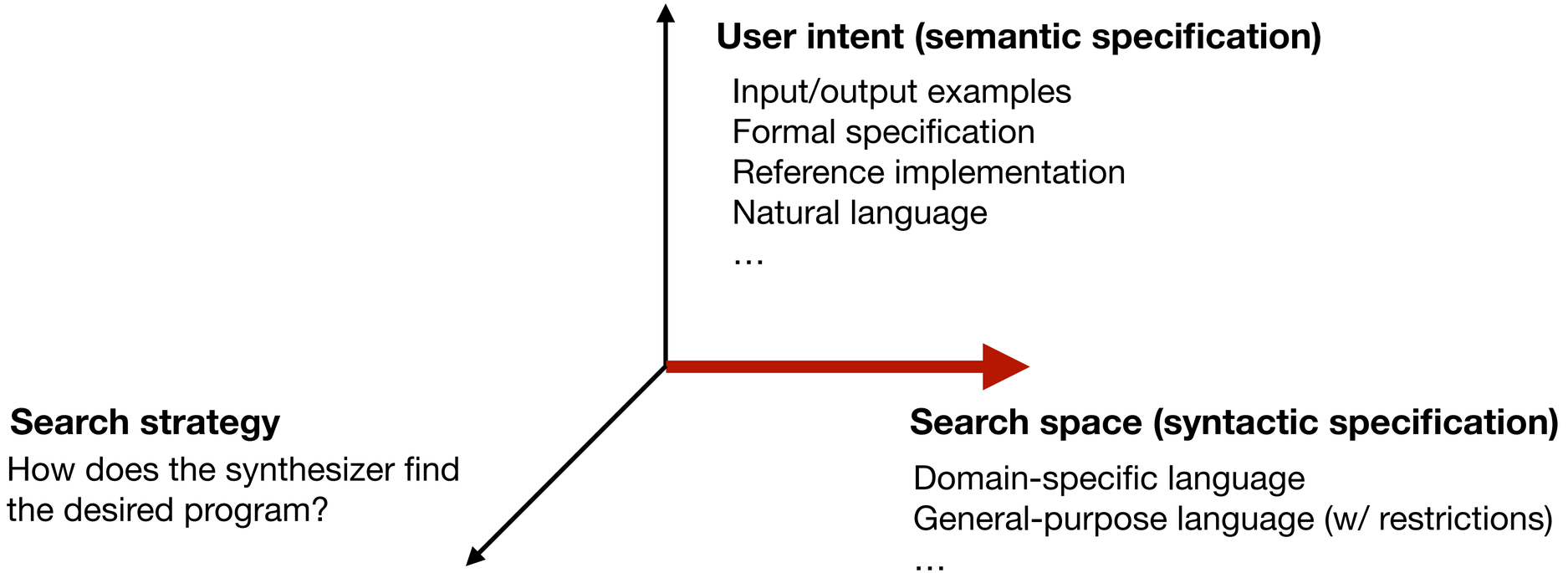
Search space (syntactic specification)

What is the space of programs to explore?

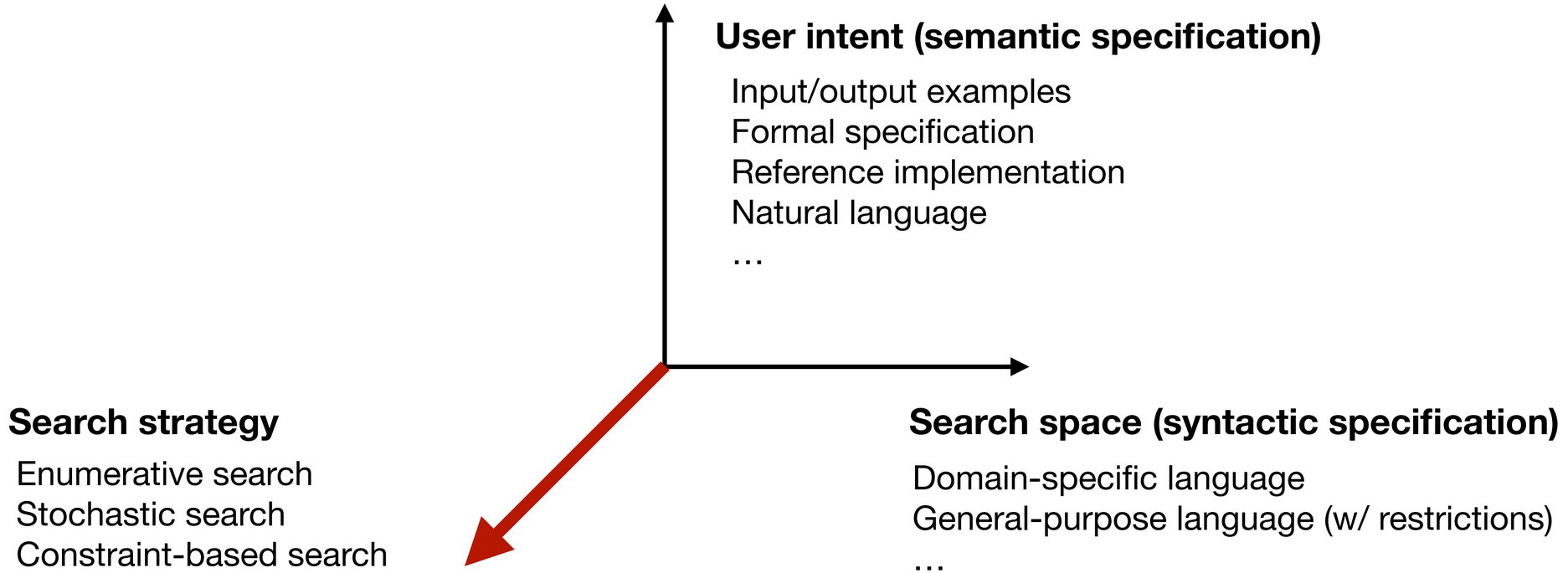
Dimensions in Program Synthesis



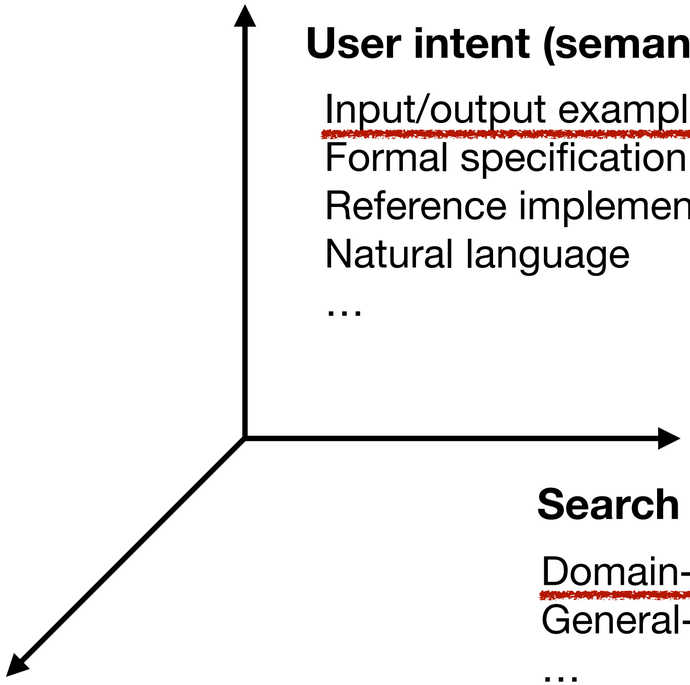
Dimensions in Program Synthesis



Dimensions in Program Synthesis



Dimensions in Program Synthesis



User intent (semantic specification)

Input/output examples
Formal specification
Reference implementation
Natural language
…

Search strategy

Enumerative search
Stochastic search
Constraint-based search
…

Search space (syntactic specification)

Domain-specific language
General-purpose language (w/ restrictions)
…

***This course**

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, expresiveness, etc