프로그래밍 언어와 기계 학습의 만남

허 기 홍

전산학부 / 프로그래밍 시스템 연구실

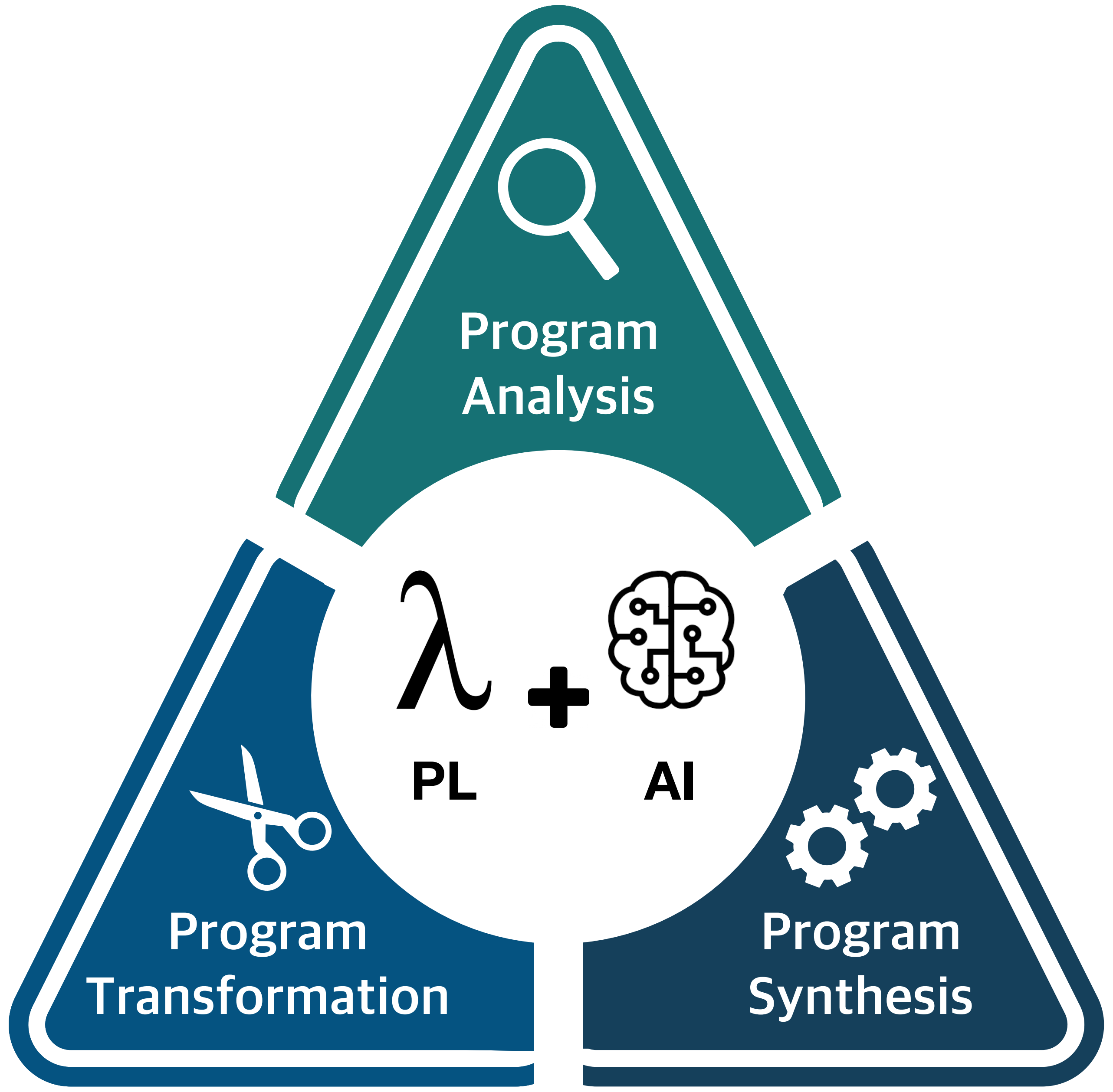
제 2회 KAIST PL 워크샵

KAIST

Programming
Systems Laboratory

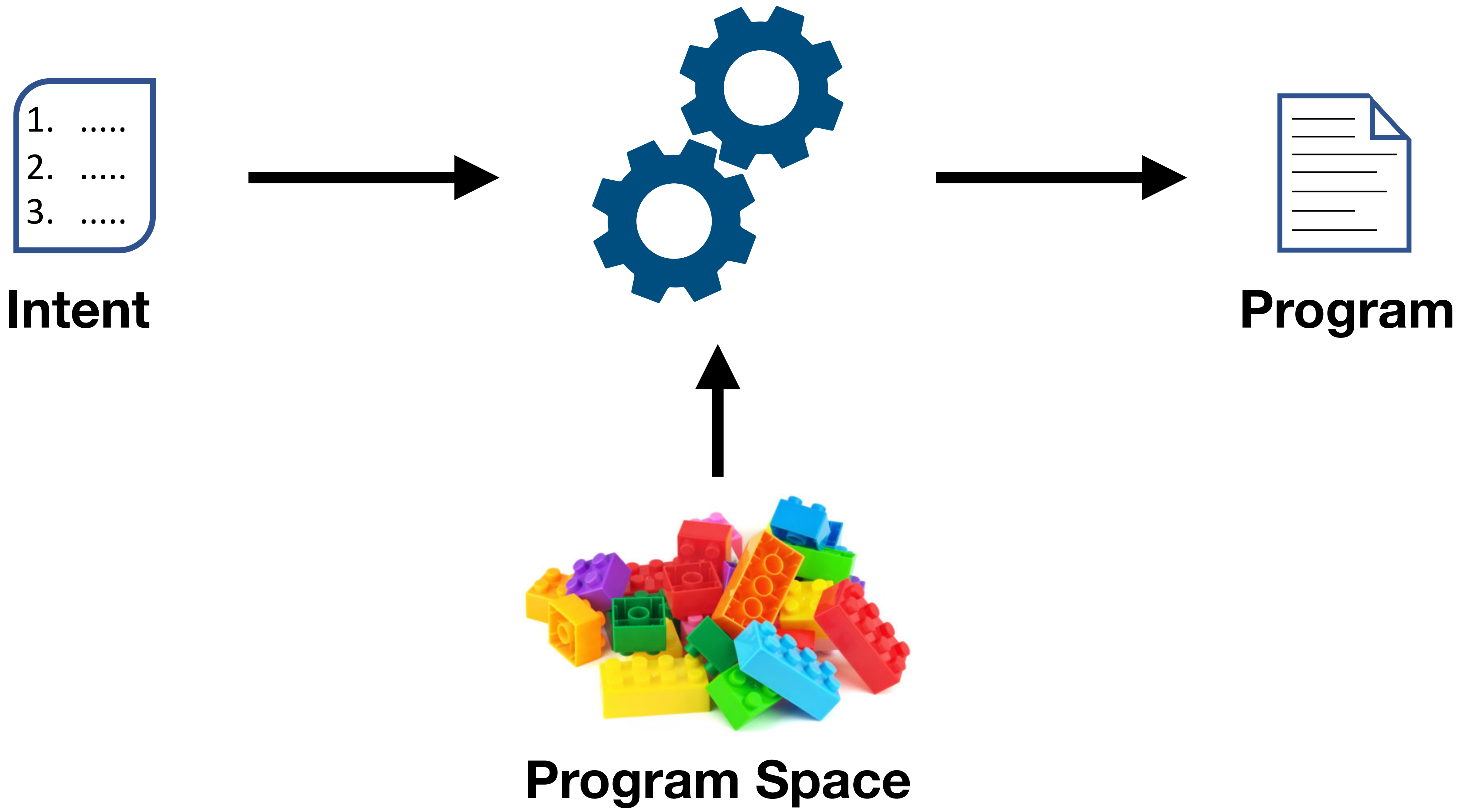
몇 가지 경험과 사례

- 프로그램 분석
  - 유연한 프로그램 분석을 위한 지도 학습, 강화 학습
  - 편리한 프로그램 분석을 위한 베이지안 추론
- 프로그램 변환과 합성
  - 빠른 합성을 위한 언어 모델
  - 빠른 변환을 위한 강화 학습



프로그램 합성 / 변환

- 사용자에 **의도**에 맞는 프로그램을 주어진 **탐색 공간**에서 **자동**으로 찾기



Intent

Program

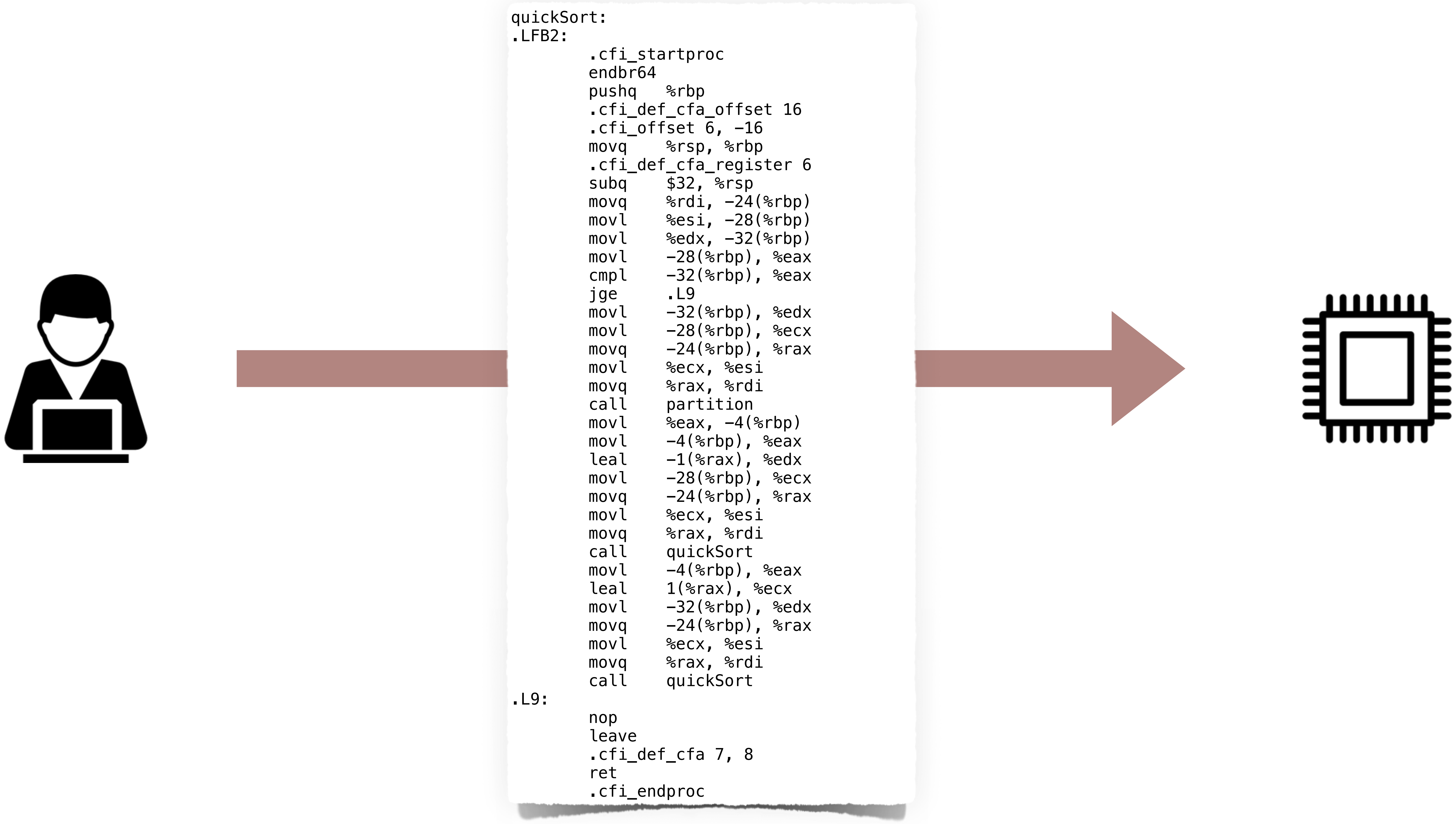
Program Space

자동 프로그래밍

***“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**.”*

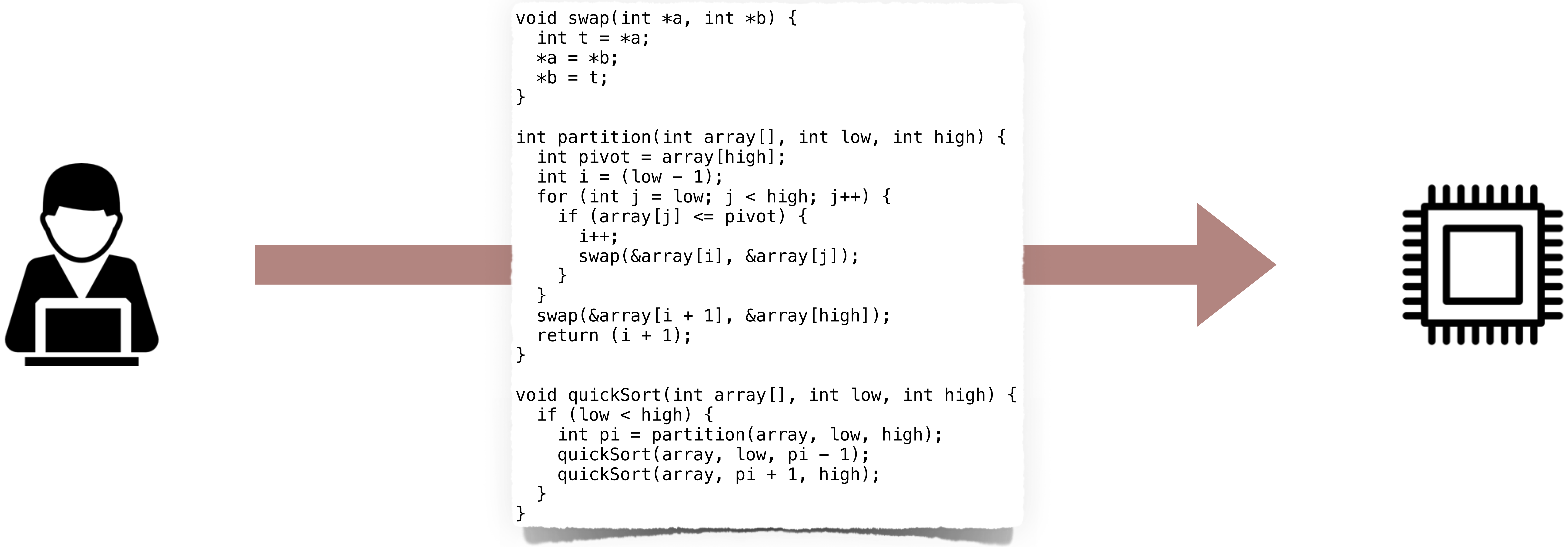
- A. M. Turing, “Proposed Electronic Calculator”, 1946

프로그래밍의 역사



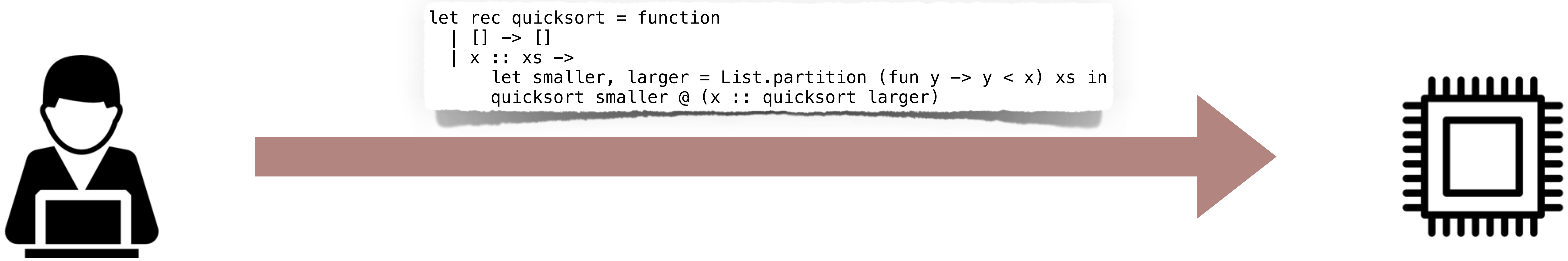
```
quickSort:
.LFB2:
        .cfi_startproc
        endbr64
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        subq    $32, %rsp
        movq    %rdi, -24(%rbp)
        movl    %esi, -28(%rbp)
        movl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

프로그래밍의 역사



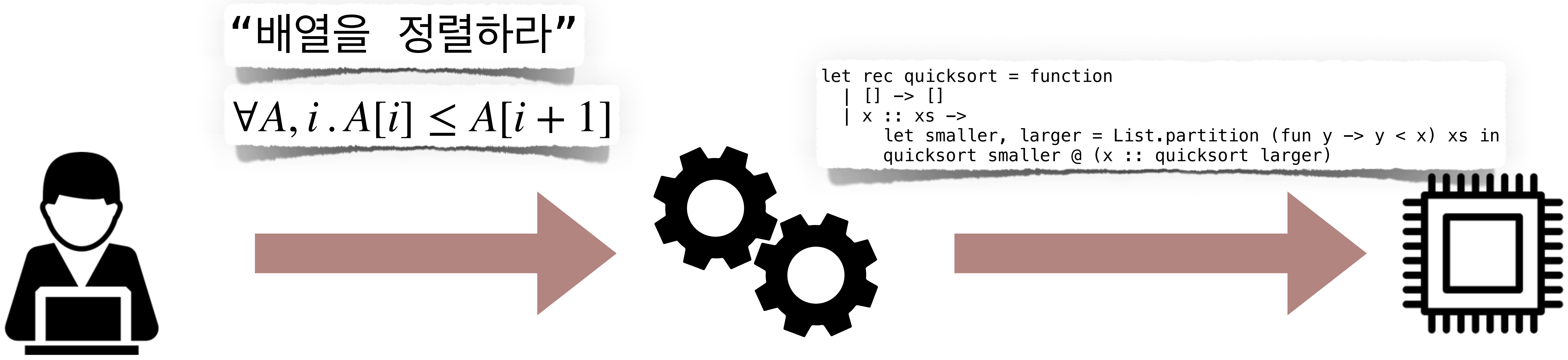
```
void swap(int *a, int *b) {
  int t = *a;
  *a = *b;
  *b = t;
}

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);
  }
}
```

프로그래밍의 역사



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

프로그래밍의 미래



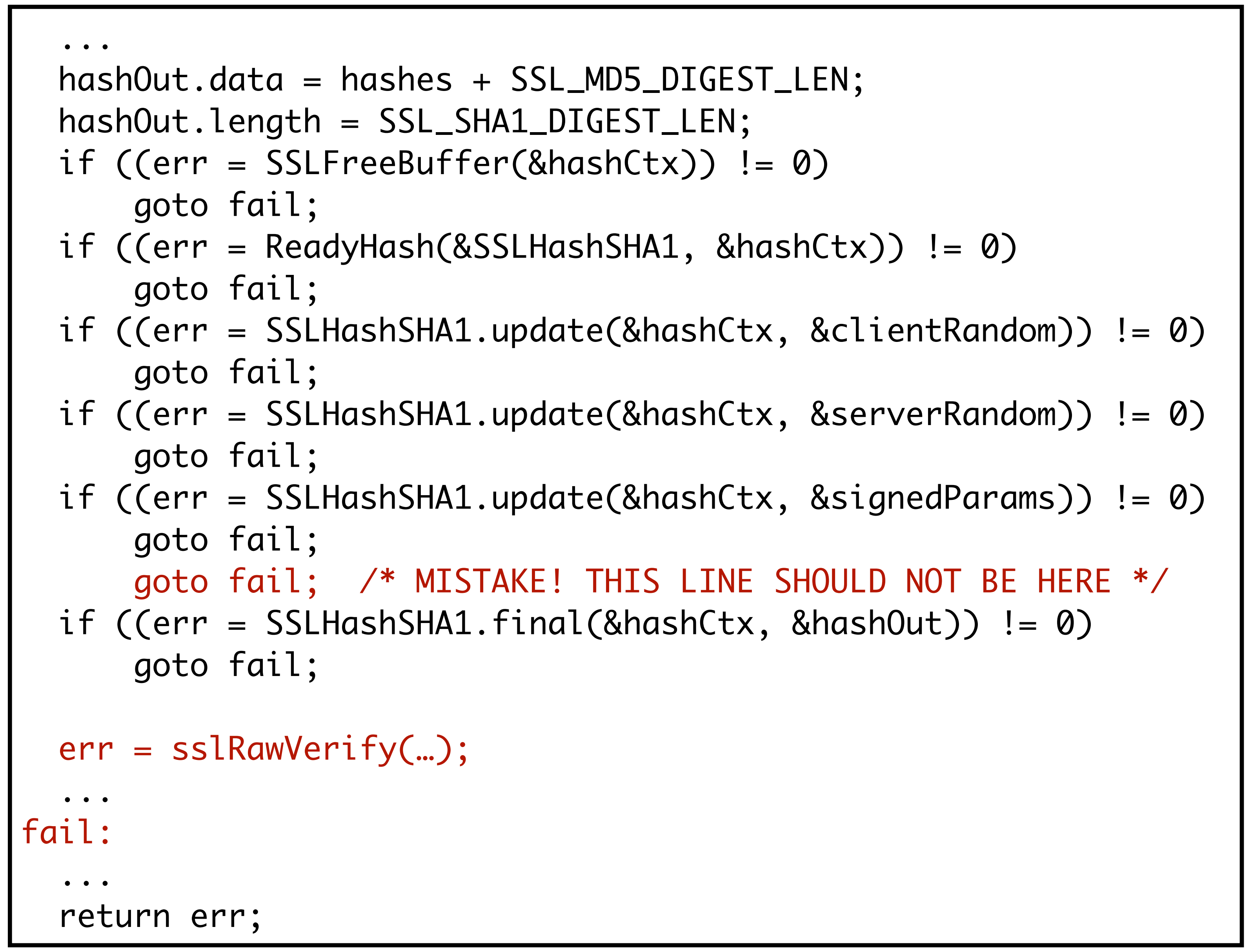
“배열을 정렬하라”

$\forall A, i \,.\, A[i] \leq A[i+1]$

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

왜 자동 프로그래밍이 필요한가?

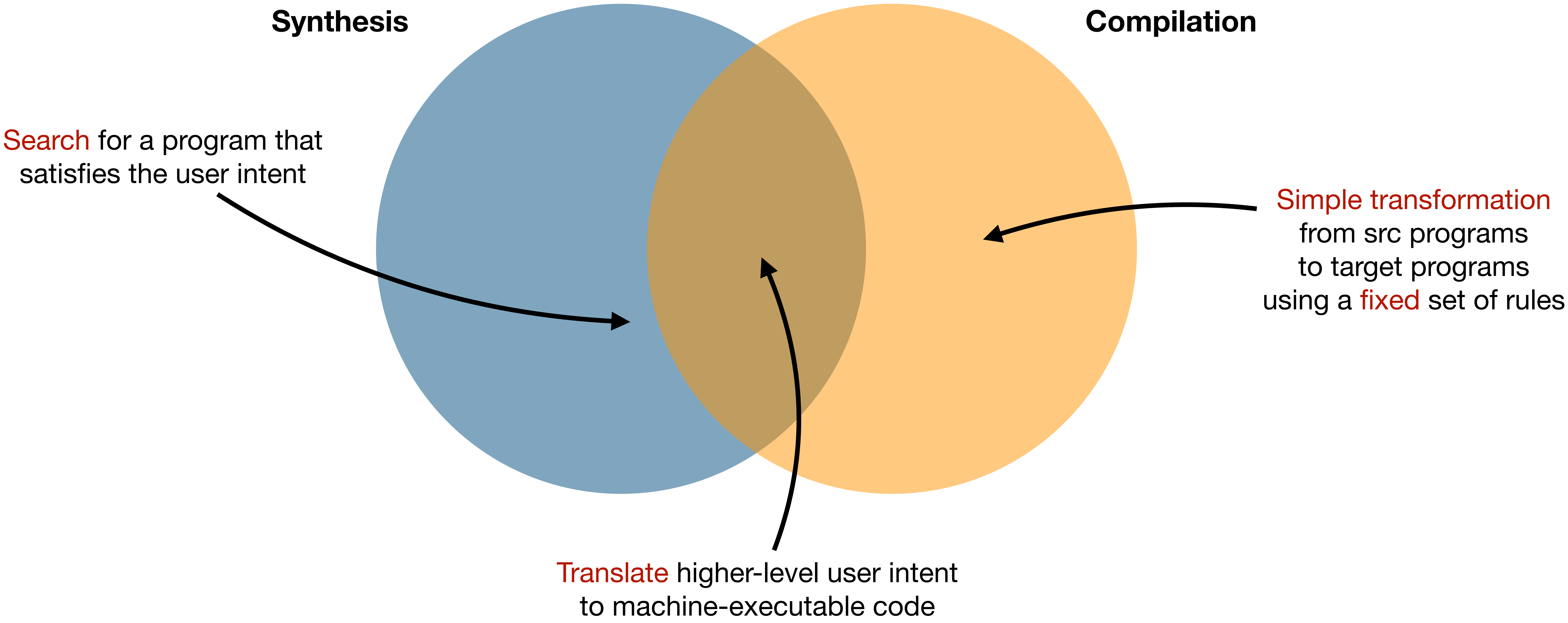
- 사람은 누구나 실수를 한다, 전문 개발자일지라도
- #developers << #programming tasks (feature reqs, bug fixes)
  - E.g., Linux kernel's open bug reports
- 모두를 위한 프로그래밍



```
  ...
  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

합성 vs 컴파일

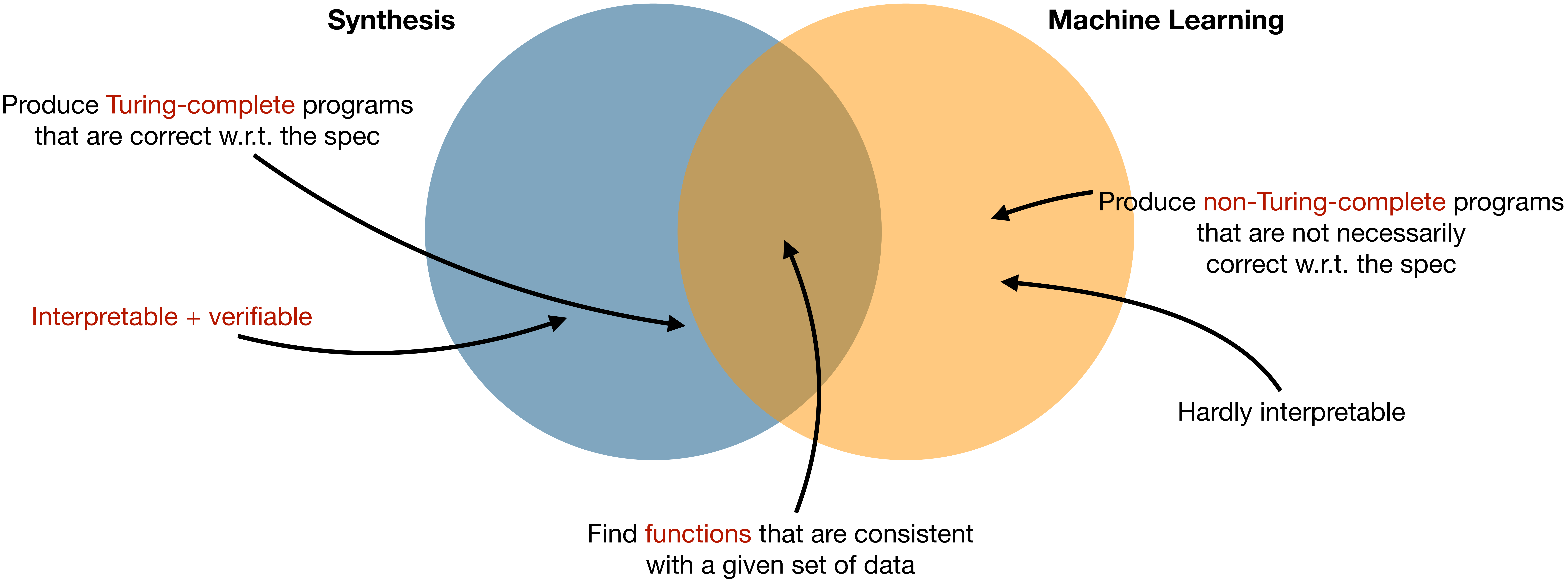


Search for a program that satisfies the user intent

Simple transformation from src programs to target programs using a fixed set of rules

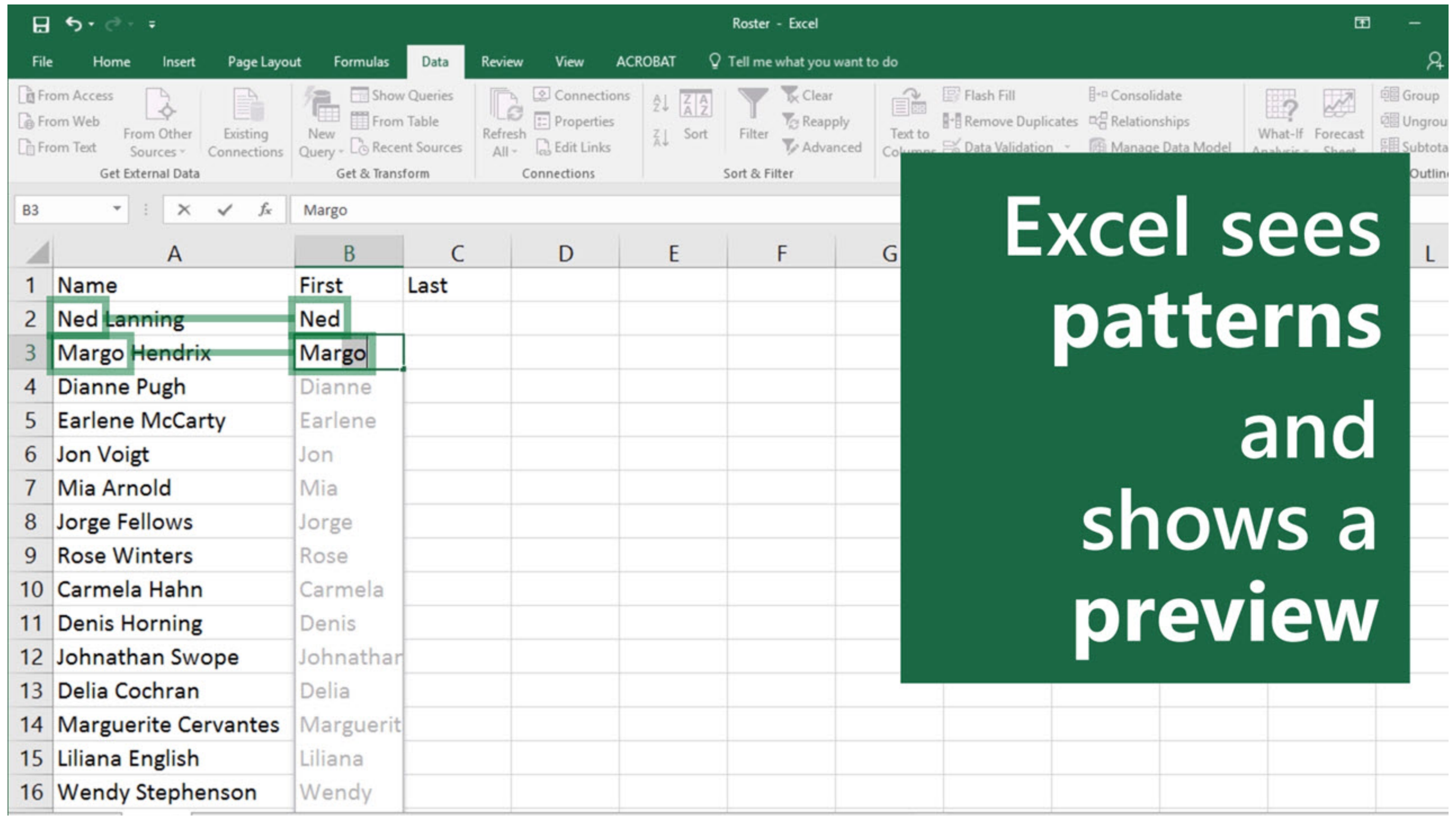
Translate higher-level user intent to machine-executable code

합성 vs 기계 학습


Synthesis
Machine Learning
Produce Turing-complete programs that are correct w.r.t. the spec
Interpretable + verifiable
Produce non-Turing-complete programs that are not necessarily correct w.r.t. the spec
Hardly interpretable
Find functions that are consistent with a given set of data

예제: 모두를 위한 프로그래밍 (1)

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



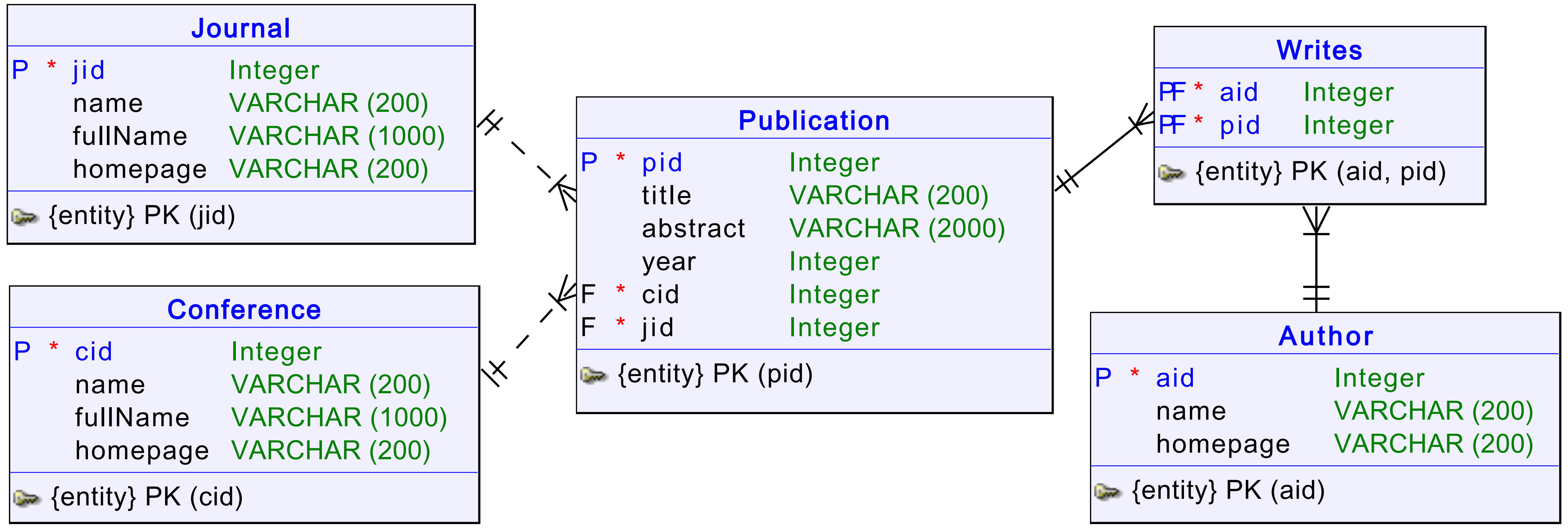
	A	B	C
1	Name	First	Last
2	Ned Lanning	Ned	
3	Margo Hendrix	Margo	
4	Dianne Pugh	Dianne	
5	Earlene McCarty	Earlene	
6	Jon Voigt	Jon	
7	Mia Arnold	Mia	
8	Jorge Fellows	Jorge	
9	Rose Winters	Rose	
10	Carmela Hahn	Carmela	
11	Denis Horning	Denis	
12	Johnathan Swope	Johnathar	
13	Delia Cochran	Delia	
14	Marguerite Cervantes	Marguerit	
15	Liliana English	Liliana	
16	Wendy Stephenson	Wendy	

Excel sees patterns and shows a preview

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

예제: 모두를 위한 프로그래밍 (2)

- SQL query synthesis



Journal

P	*	jid	Integer
		name	VARCHAR (200)
		fullName	VARCHAR (1000)
		homepage	VARCHAR (200)

{entity} PK (jid)

Conference

P	*	cid	Integer
		name	VARCHAR (200)
		fullName	VARCHAR (1000)
		homepage	VARCHAR (200)

{entity} PK (cid)

Publication

P	*	pid	Integer
		title	VARCHAR (200)
		abstract	VARCHAR (2000)
		year	Integer
F	*	cid	Integer
F	*	jid	Integer

{entity} PK (pid)

Writes

PF	*	aid	Integer
PF	*	pid	Integer

{entity} PK (aid, pid)

Author

P	*	aid	Integer
		name	VARCHAR (200)
		homepage	VARCHAR (200)

{entity} PK (aid)

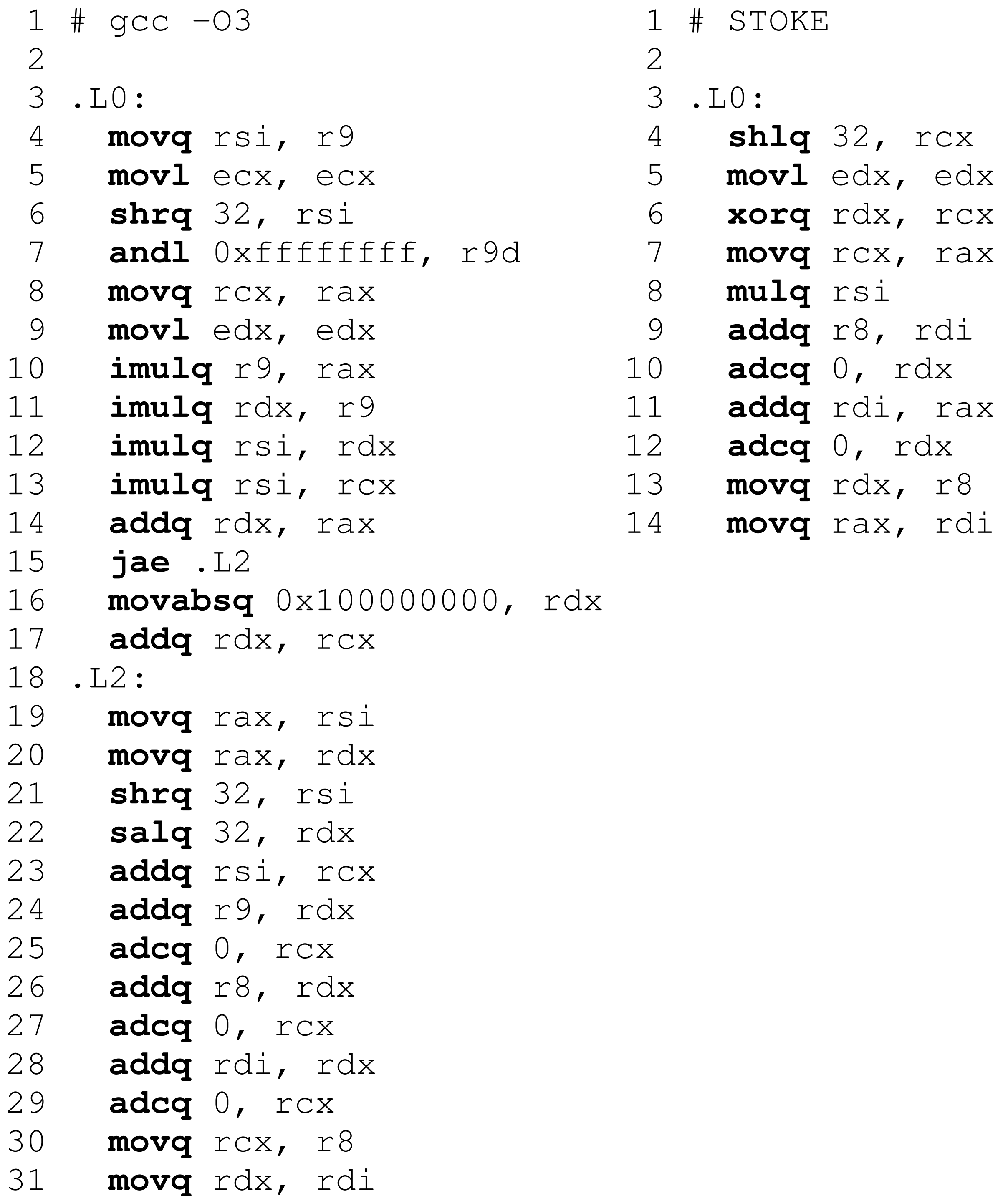
“Find the number of papers in OOPSLA 2020”

```
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

예제: 특급 최적화

- Montgomery multiplication kernel from the OpenSSL RSA library



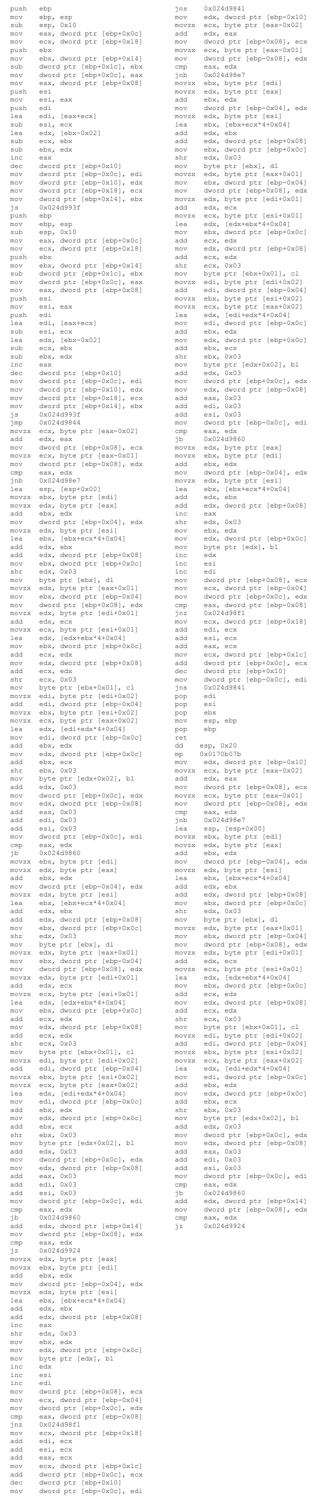
```
# gcc -O3

.L0:
  movq rsi, r9
  movl ecx, ecx
  shrq 32, rsi
  andl 0xffffffff, r9d
  movq rcx, rax
  movl edx, edx
  imulq r9, rax
  imulq rdx, r9
  imulq rsi, rdx
  imulq rsi, rcx
  addq rdx, rax
  jae .L2
  movabsq 0x100000000, rdx
  addq rdx, rcx
.L2:
  movq rax, rsi
  movq rax, rdx
  shrq 32, rsi
  salq 32, rdx
  addq rsi, rcx
  addq r9, rdx
  adcq 0, rcx
  addq r8, rdx
  adcq 0, rcx
  addq rdi, rdx
  adcq 0, rcx
  movq rcx, r8
  movq rdx, rdi
```

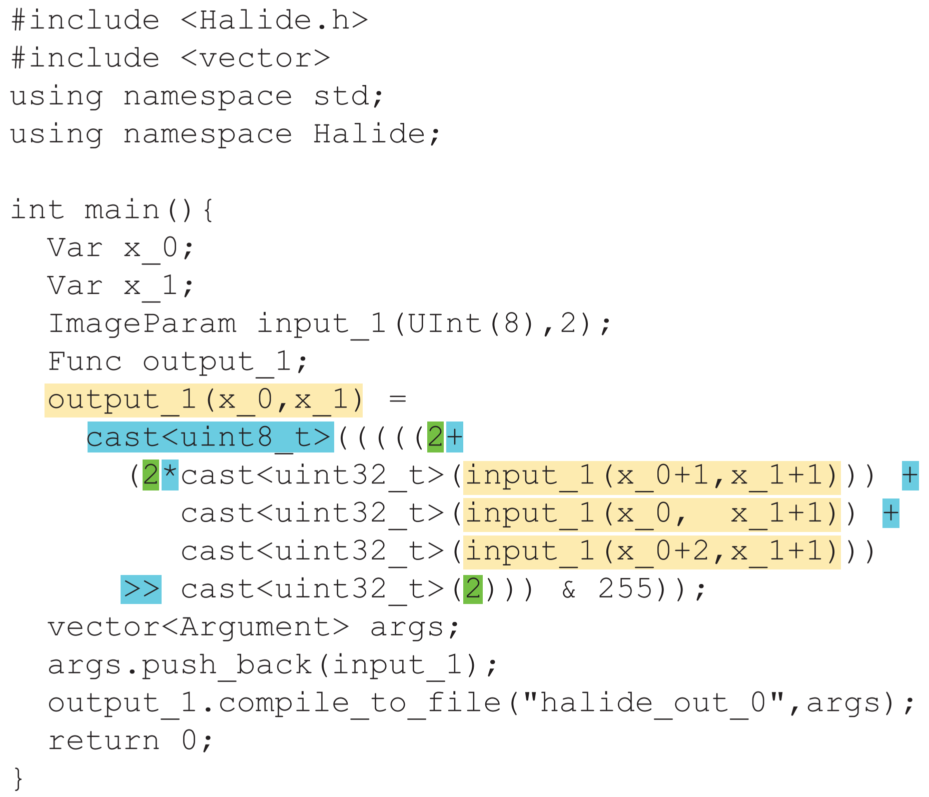
```
# STOKE

.L0:
  shlq 32, rcx
  movl edx, edx
  xorq rdx, rcx
  movq rcx, rax
  mulq rsi
  addq r8, rdi
  adcq 0, rdx
  addq rdi, rax
  adcq 0, rdx
  movq rdx, r8
  movq rax, rdi
```

Stochastic Super Optimization, ASPLOS'13

예제: 역공학

- Binary lifting

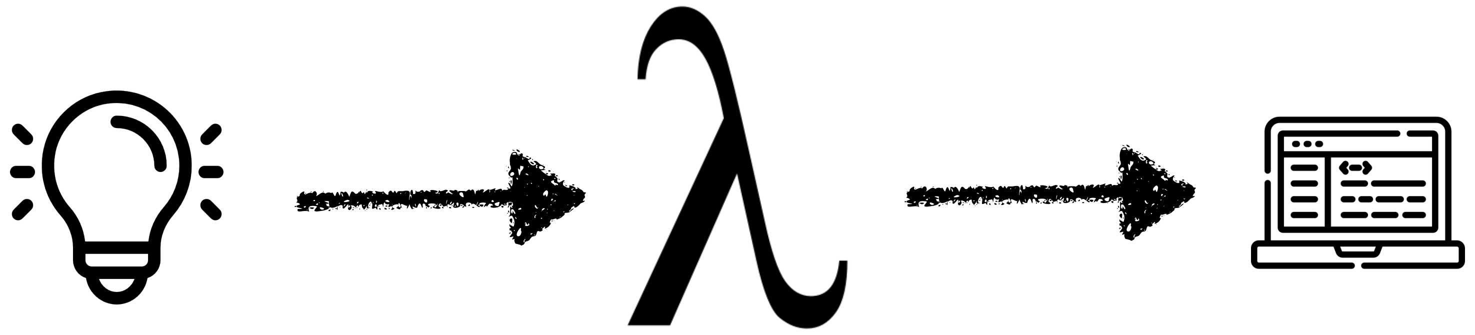


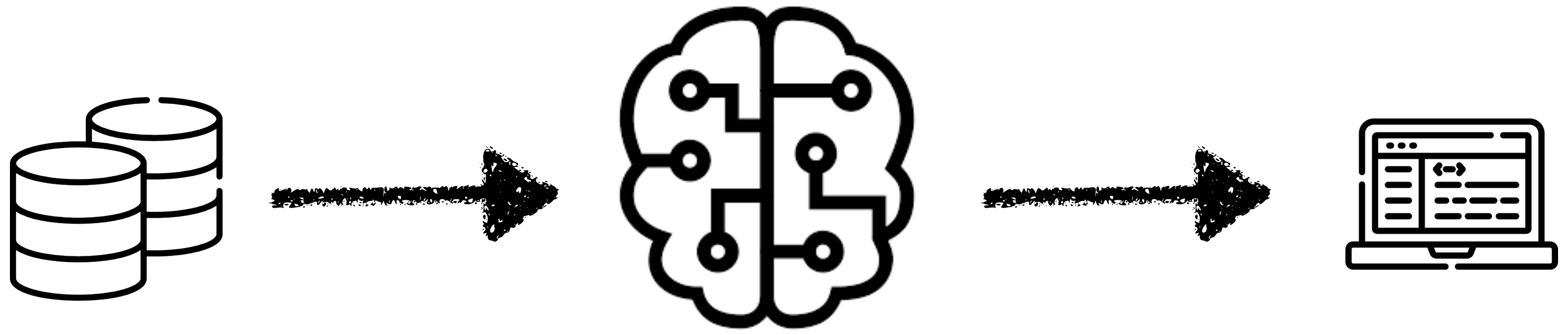


```
#include <Halide.h>
#include <vector>
using namespace std;
using namespace Halide;

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

두 가지 방향: PL vs ML





- 논리 기반
- 올바름 보장 O
- 느림
- Excel FlashFill 등에서 활용
- 공간 탐색 알고리즘 기반

- 확률 기반
- 올바름 보장 X
- 빠름
- Gihub Copilot 등에서 활용
- 확률 추론 알고리즘 기반

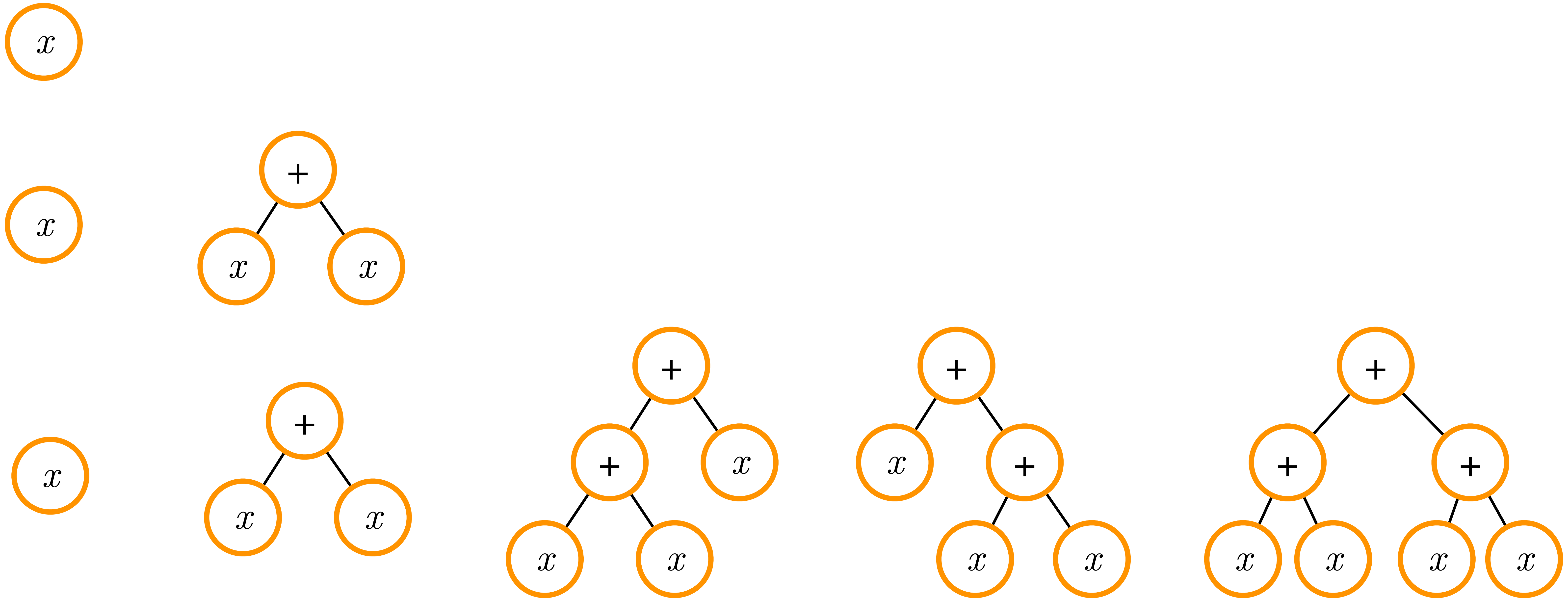
도전 과제 1: 공간 탐색

$$S \to x \mid S + S$$

depth ≤ 1

depth ≤ 2

depth ≤ 3



N(1) = 1

N(2) = 2

N(3) = 5

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

*Examples from Nadia Polikarpova’s slides

도전 과제 1: 공간 탐색

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

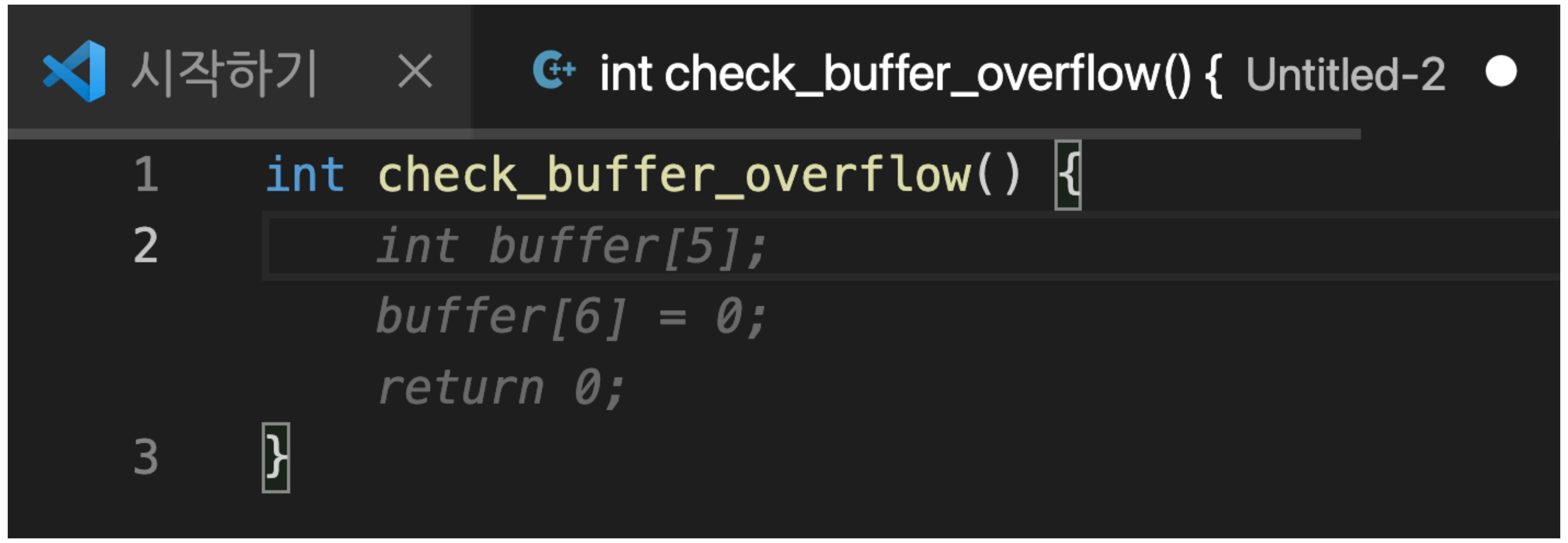
N(d) = 1 + N(d - 1)²

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

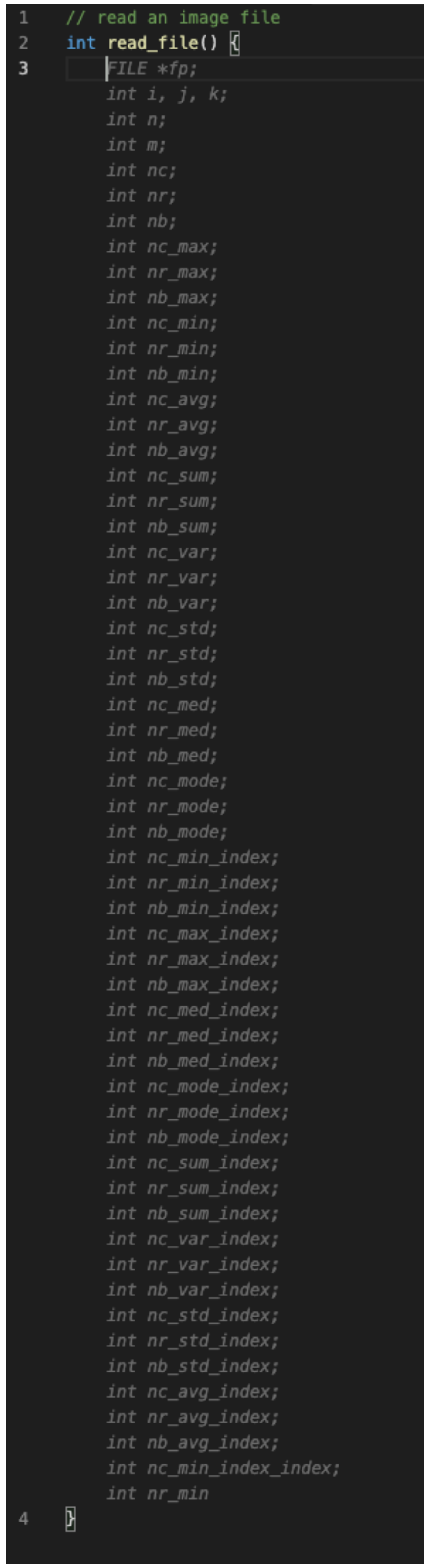
‼

*Examples from Nadia Polikarpova's slides

도전 과제 2: 올바름 보장



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

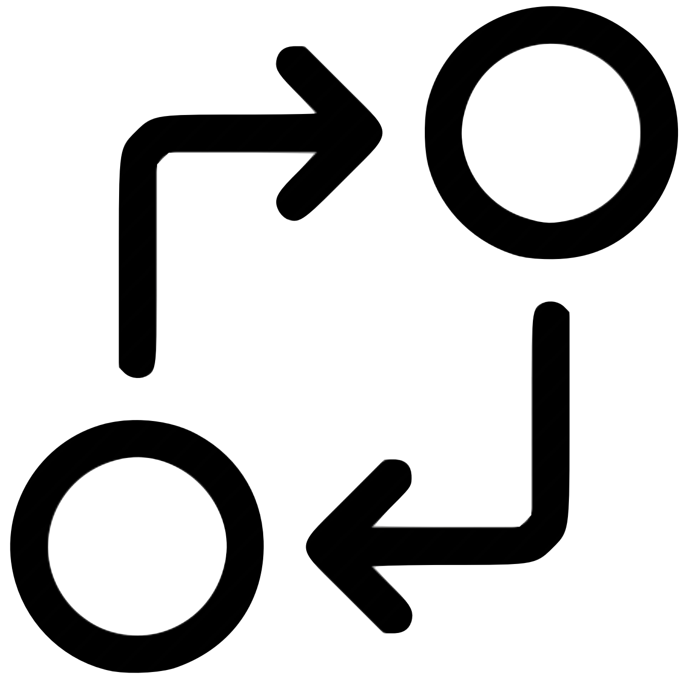


```c
// read an image file
int read_file() {
    FILE *fp;
    int i, j, k;
    int n;
    int m;
    int nc;
    int nr;
    int nb;
    int nc_max;
    int nr_max;
    int nb_max;
    int nc_min;
    int nr_min;
    int nb_min;
    int nc_avg;
    int nr_avg;
    int nb_avg;
    int nc_sum;
    int 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
}
```

오늘 이야기

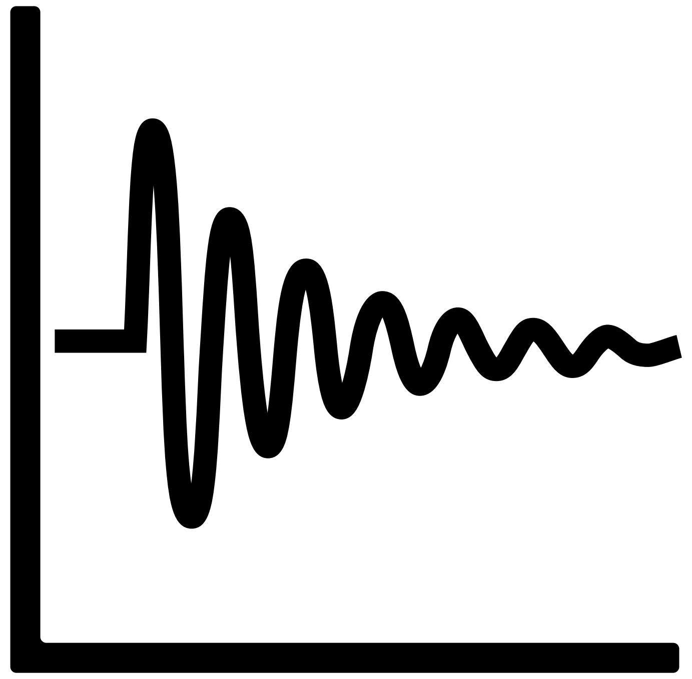
- “PL” 세팅에서 프로그램 변환과 합성을 빠르게 하는 방법
    - “PL” 세팅: 언어의 구문과 의미 구조 + 올바름 판단 방법 (예: 테스트 케이스)이 주어진 상황
- 탐색을 위한 핵심 아이디어
    - 연관성: “프로그램 구문들 사이에는 의미적 연관성이 있다”
    - 규칙성: “프로그램 구문에는 전형적인 패턴이 있다”
    - 연속성: “프로그램 구문이 조금 바뀌면 의미도 조금 바뀐다”

세 가지 아이디어



연관성 [CCS’18]

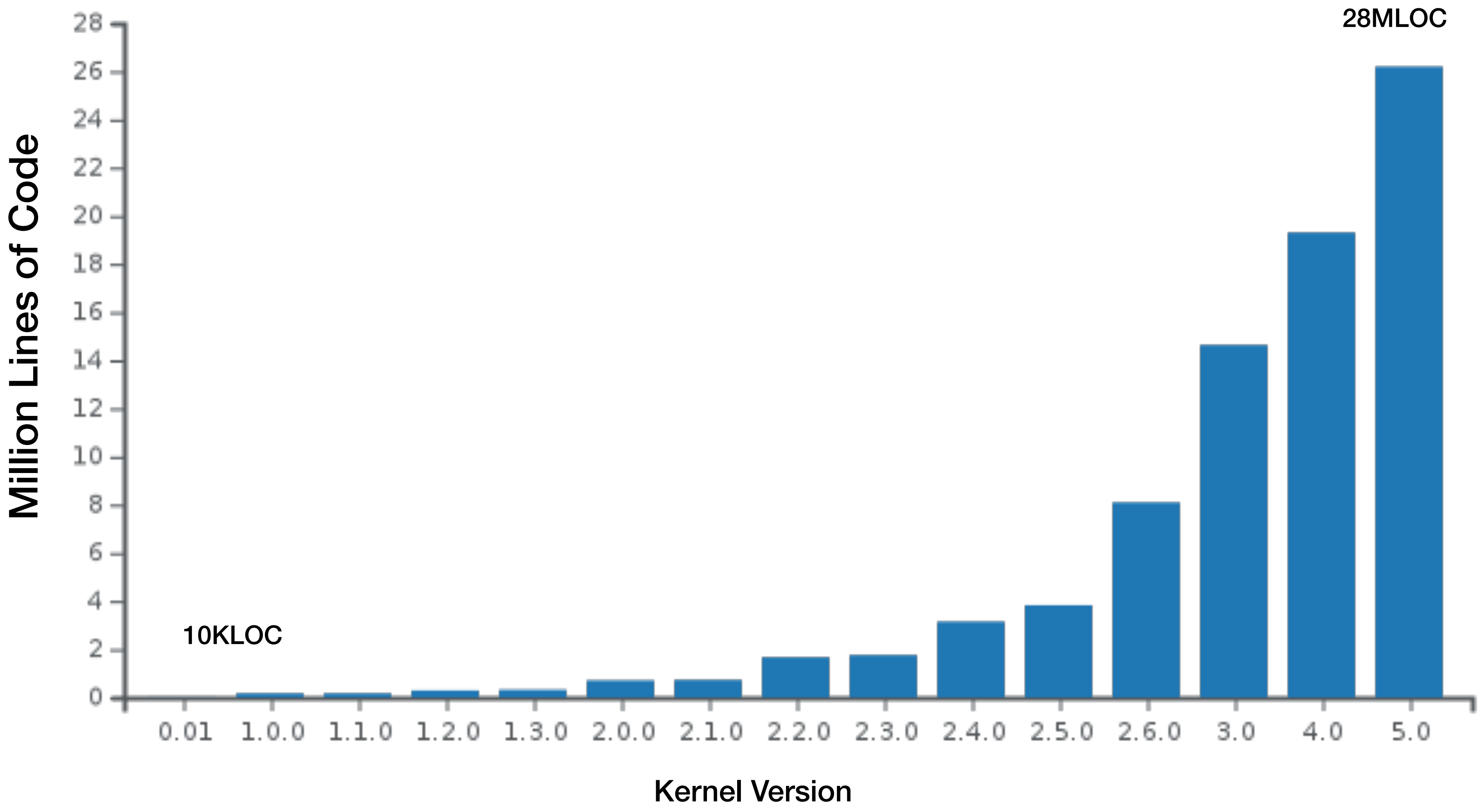
규칙성 [PLDI’18]



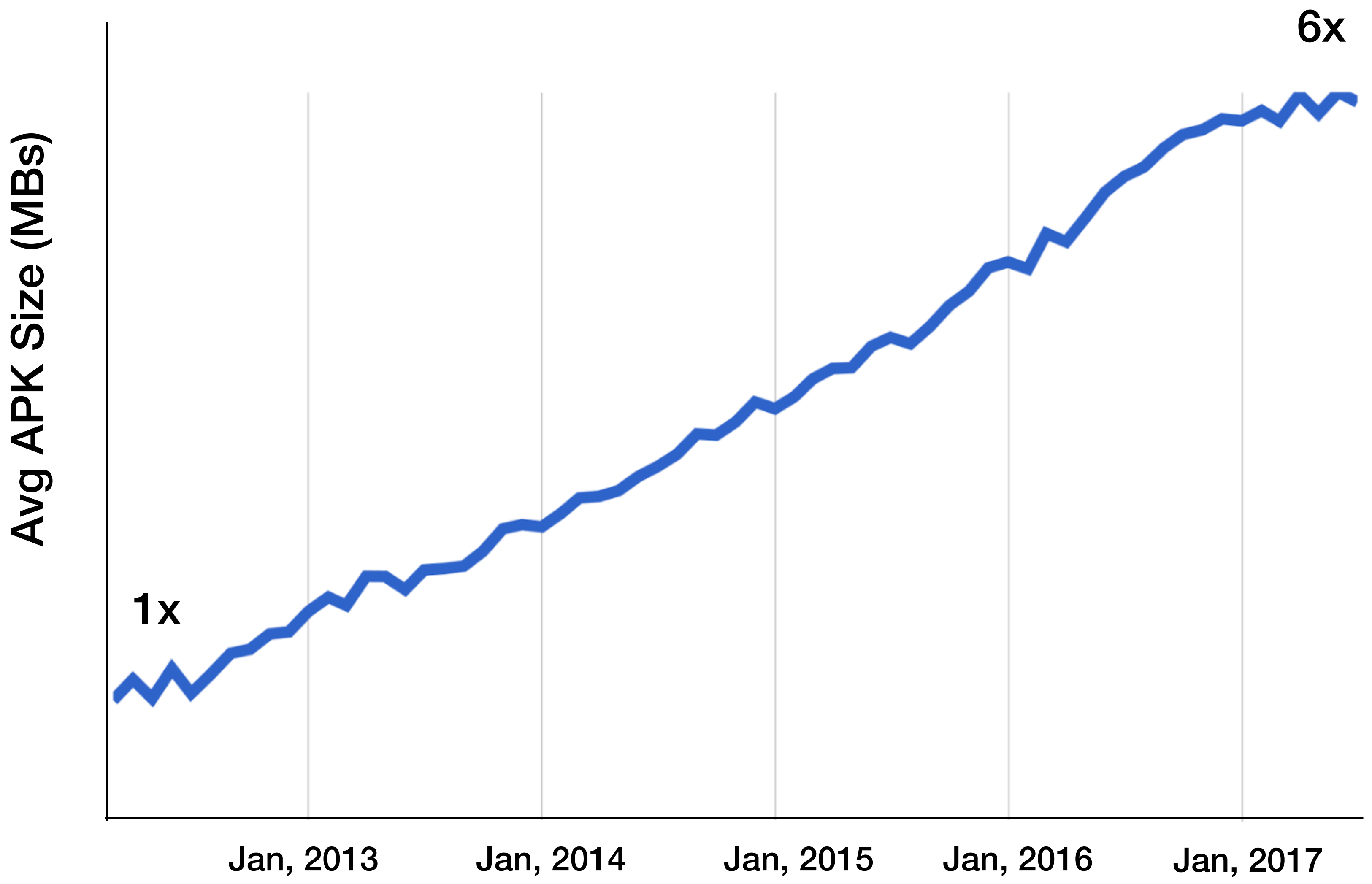
연속성 [IJCAI’19]

소프트웨어의 크기와 복잡도

Size of Linux Kernel



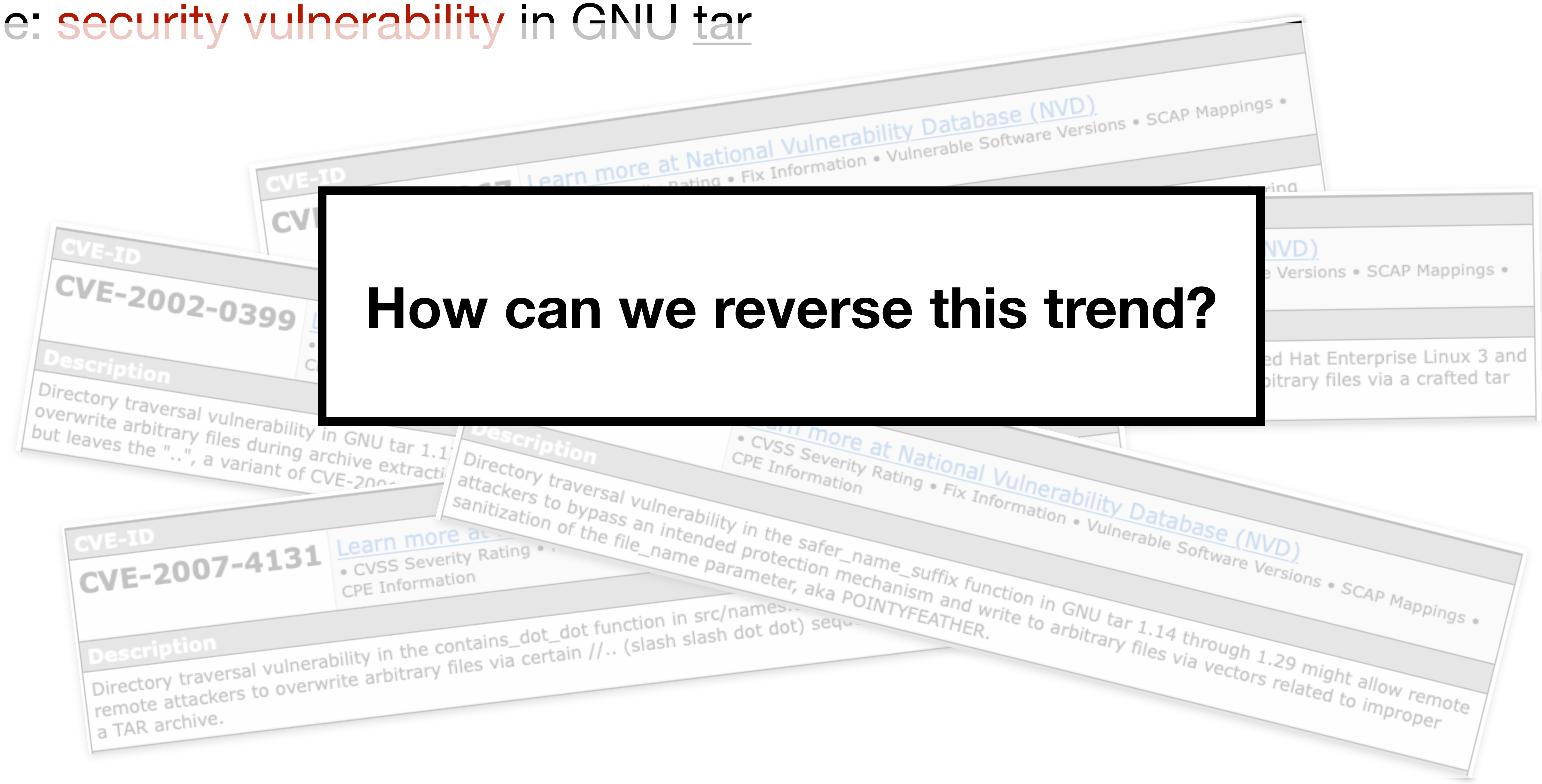
Avg. Size of Android Apps



소프트웨어 거품 (Software Bloat)

Performance | **Maintainability** | **Security**

- Example: security vulnerability in GNU tar



CVE-ID

Learn more at National Vulnerability Database (NVD)

Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings •

CVE-ID

CVE-2002-0399

Description

Directory traversal vulnerability in GNU tar 1.1
overwrite arbitrary files during archive extracti
but leaves the "..", a variant of CVE-200

NVD)

Versions • SCAP Mappings •

ed Hat Enterprise Linux 3 and
bitrary files via a crafted tar

CVE-ID

CVE-2007-4131

Learn more at

• CVSS Severity Rating •

CPE Information

Description

Directory traversal vulnerability in the contains_dot_dot function in src/names.
remote attackers to overwrite arbitrary files via certain //.. (slash slash dot dot) seq
a TAR archive.

Description

Learn more at National Vulnerability Database (NVD)

• CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings •

CPE Information

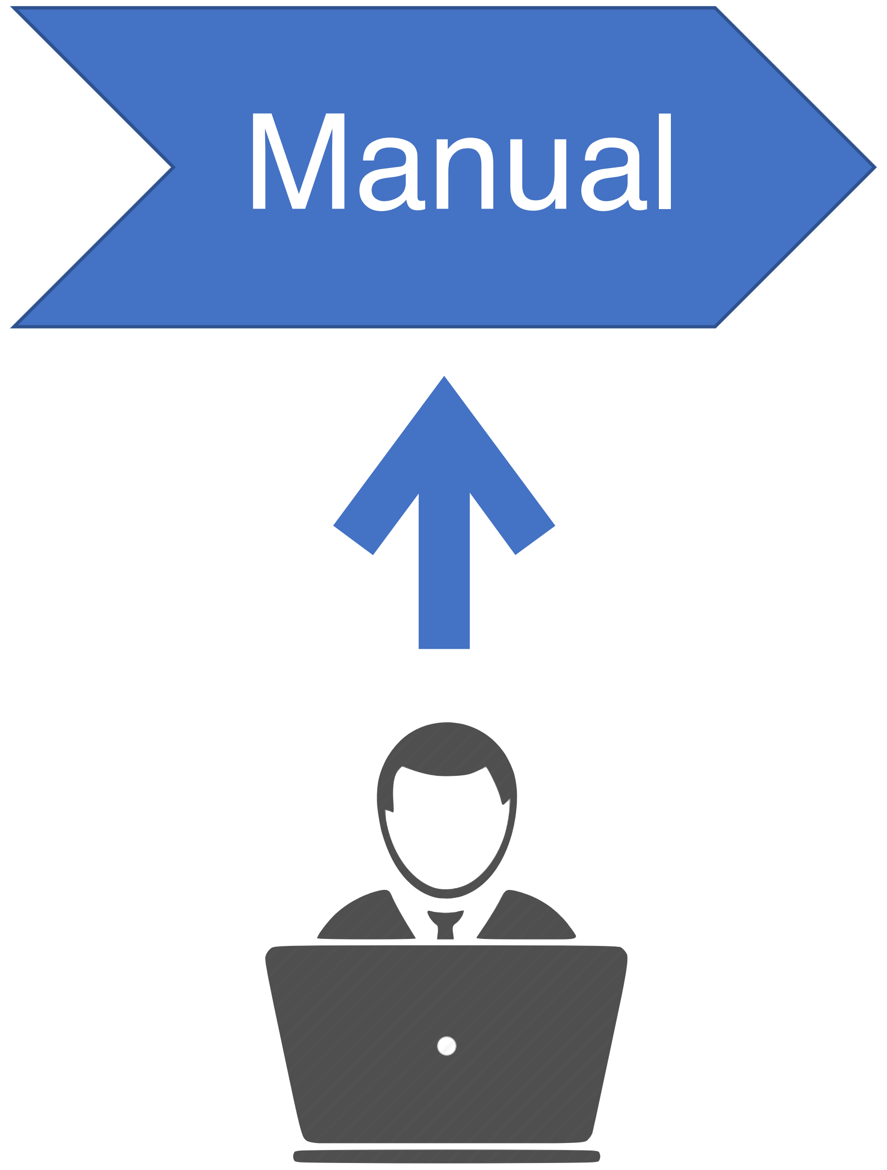
Directory traversal vulnerability in the safer_name_suffix function in GNU tar 1.14 through 1.29 might allow remote
attackers to bypass an intended protection mechanism and write to arbitrary files via vectors related to improper
sanitization of the file_name parameter, aka POINTYFEATHER.

How can we reverse this trend?

현실

General-purpose tar

- Out-of-the-box Linux
- 97 cmd line options
- 45,778 LOC
- 13,227 statements
- CVE-2016-6321



Manual

Customized tar

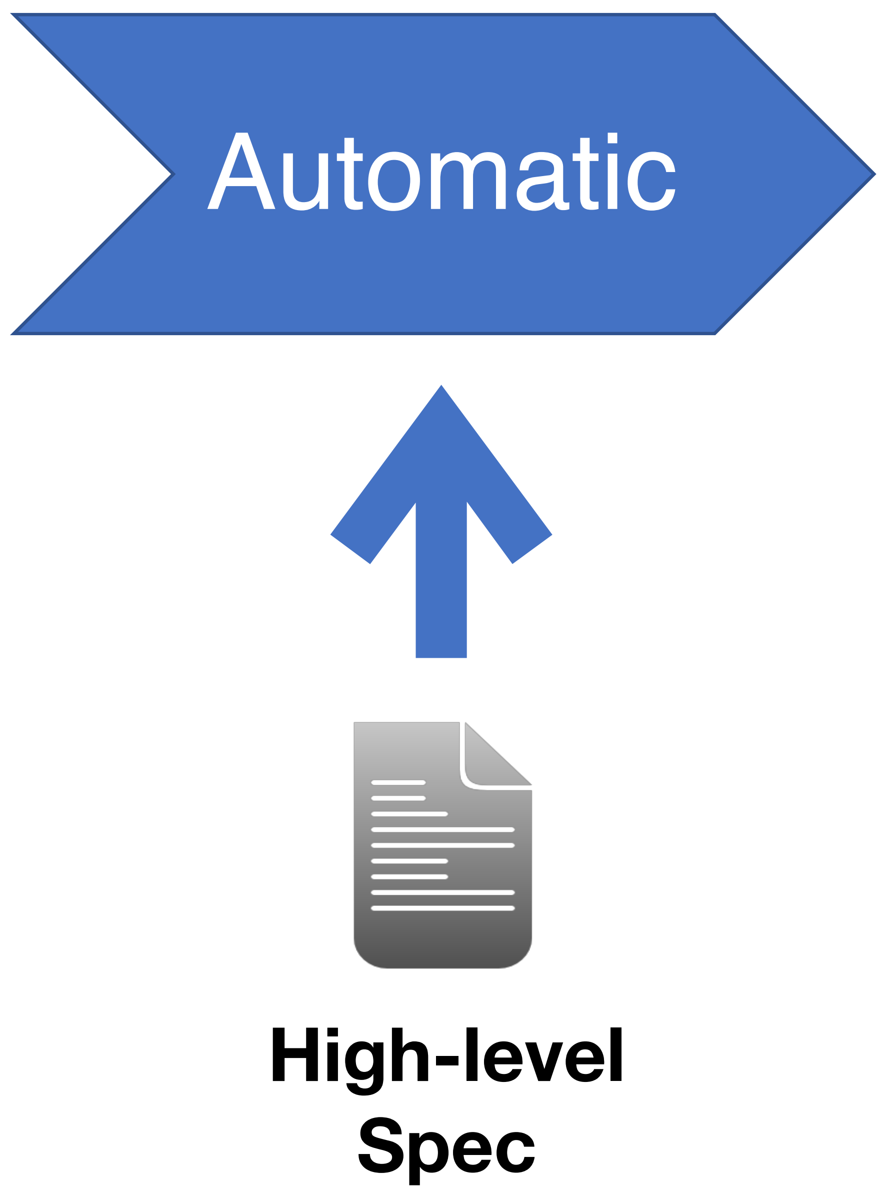
- BusyBox Utility Package*
- 8 cmd line options
- 3,287 LOC
- 403 statements
- No known CVEs

*https://busybox.net

목표

General-purpose tar

- Out-of-the-box Linux
- 97 cmd line options
- 45,778 LOC
- 13,227 statements
- CVE-2016-6321



Customized tar

- BusyBox Utility Package*
- 8 cmd line options
- **1,646** ~~3,287~~ LOC
- **518** ~~403~~ statements
- No known CVEs

Chisel: A Program Debloating System*

- 다섯 가지 목표:
    - **minimality**: trim code as aggressively as possible
    - **efficiency**: scale to large programs
    - **robustness**: avoid introducing new vulnerabilities
    - **naturalness**: produce maintainable code
    - **generality**: handle a variety of programs and specs

*https://chisel.cis.upenn.edu

Example: tar-1.14

```c
int absolute_names;
int ignore_zeros_option;
struct tar_stat_info stat_info;

char *safer_name_suffix (char *file_name, int link_target) {
    int prefix_len;
    char *p;

    if (absolute_names) {
        p = file_name;
    } else {
       /* CVE-2016-6321 */
       /* Incorrect sanitization if “file_name” contains ".." */
       ...
    }
    ...
    return p;
}

void extract_archive() {
    char *file_name = safer_name_suffix(stat_info.file_name, 0);
    /* Overwrite “file_name” if exists */
    ...
}

void list_archive() { ... }
```

```c
void read_and(void *(do_something)(void)) {
    enum read_header status;
    while (...) {
        status = read_header();
        switch (status) {
        case HEADER_SUCCESS: (*do_something)(); continue;
        case HEADER_ZERO_BLOCK:
          if (ignore_zeros_option) continue;
          else break;
        ...
        default: break;
        }
    }
    ...
}

/* Supports all options: -x, -t, -P, -i, ... */
int main(int argc, char **argv) {
    int optchar;
    while (optchar = getopt_long(argc, argv) != -1) {
        switch(optchar) {
        case 'x': read_and(&extract_archive); break;
        case 't': read_and(&list_archive); break;
        case 'P': absolute_names = 1; break;
        case 'i': ignore_zeros_option = 1; break;
        ...
        }
    }
    ...
}
```

Example: tar-1.14

Global variable declarations removed

```
int absolute_names;
int ignore_zeros_option;
struct tar_stat_info stat_info;

char *safer_name_suffix (char *file_name, int link_target) {
    int prefix_len;
    char *p;

    if (absolute_names) {
        p = file_name;
    } else {
       /* CVE-2016-6321 */
       /* Incorrect sanitization if “file_name” contains ".." */
       ...
    }
    ...
    return p;
}

void extract_archive() {
    char *file_name = safer_name_suffix(stat_info.file_name, 0);
    /* Overwrite “file_name” if exists */
    ...
}

void list_archive() { ... }
```

Code containing CVE removed

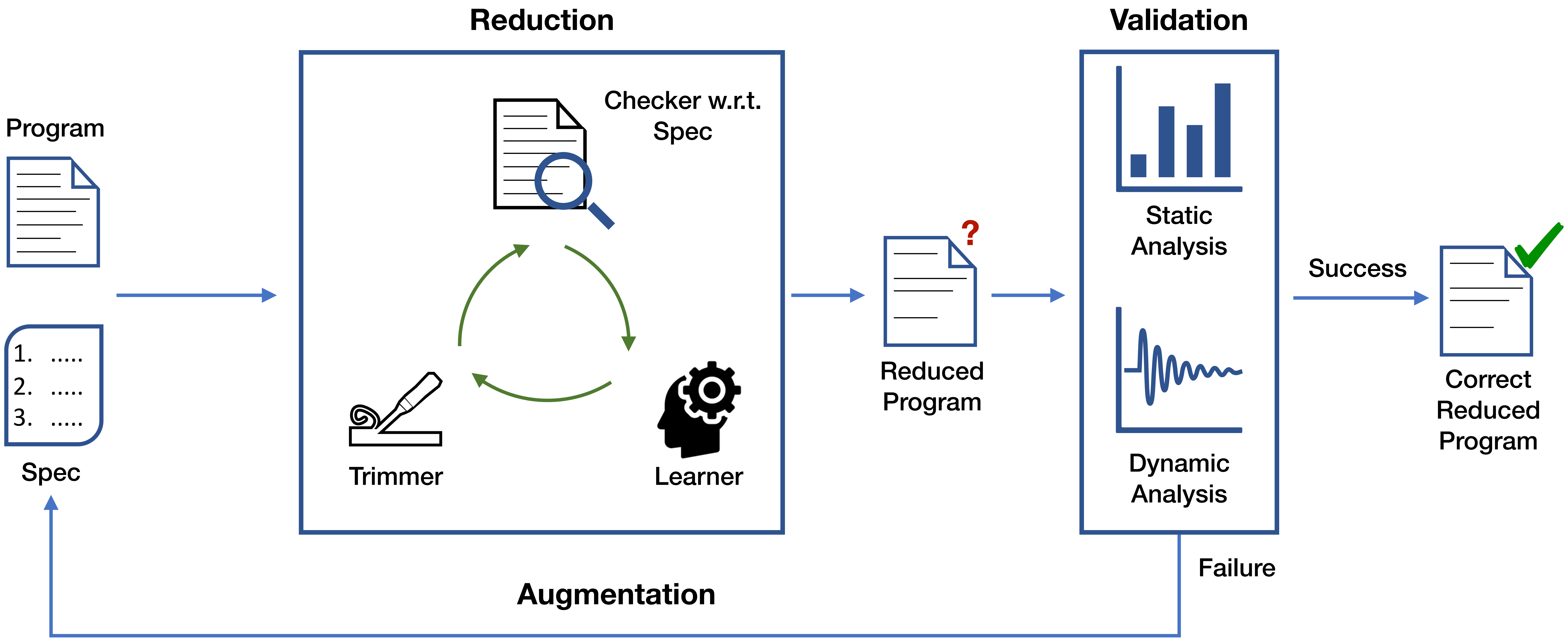
Overwriting functionalities removed

```
void read_and(void *(do_something)(void)) {
    enum read_header status;
    while (...) {
        status = read_header();
        switch (status) {
        case HEADER_SUCCESS: (*do_something)(); continue;
        case HEADER_ZERO_BLOCK:
          if (ignore_zeros_option) continue;
          else break;
        ...
        default: break;
        }
    }
    ...
}

/* Supports all options: -x, -t, -P, -i, ... */
int main(int argc, char **argv) {
    int optchar;
    while (optchar = getopt_long(argc, argv) != -1) {
        switch(optchar) {
        case 'x': read_and(&extract_archive); break;
        case 't': read_and(&list_archive); break;
        case 'P': absolute_names = 1; break;
        case 'i': ignore_zeros_option = 1; break;
        ...
        }
    }
    ...
}
```

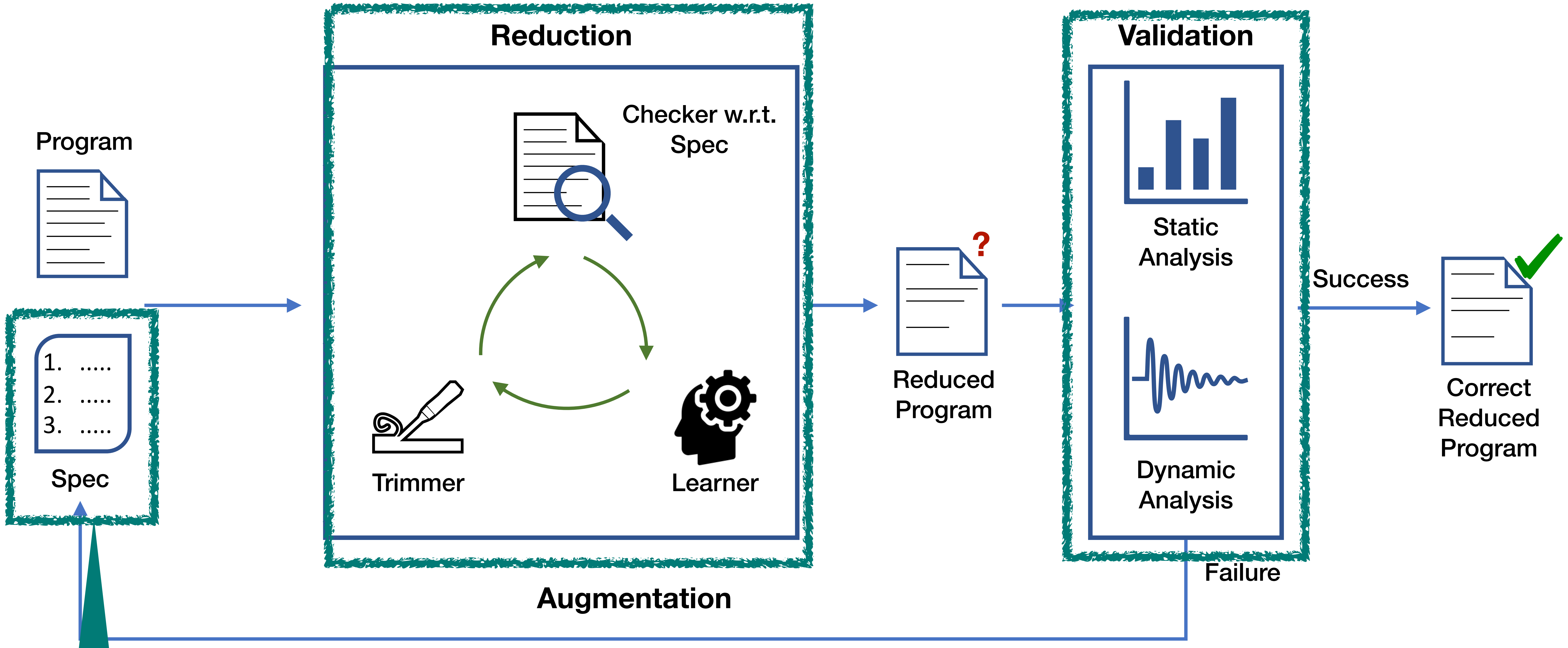
Unnecessary functionalities removed

Unsupported options removed

System Architecture



Key Questions



2. How to effectively reduce programs?

3. How to validate robustness?

1. How to provide high-level specification?

High-level Specification

```
#!/bin/bash

function compile {
  clang -o tar.debloat tar-1.14.c
  return $?
}

# tests for the desired functionalities
function desired {
  # 1. archiving multiple files
  touch foo bar
  ./tar.debloat cf foo.tar foo bar
  rm foo bar
  ./tar.debloat xf foo.tar
  test -f foo -a -f bar || exit 1

  # 2. extracting from stdin
  touch foo
  ./tar.debloat cf foo.tar foo
  rm foo
  cat foo.tar | ./tar.debloat x
  test -f foo || exit 1

  # other tests
  …
  return 0
}
```

```
# tests for the undesired functionalities
function undesired {
  for test_script in `ls other_tests/*.sh`
  do
    { sh -x -e $test_script; } >& log
    grep 'Segmentation fault' log && exit 1
  done
  return 0
}

compile || exit 1
desired || exit 1
undesired || exit 1
```

High-level Specification

```
#!/bin/bash

function compile {
  clang -o tar.debloat tar-1.14.c
  return $?
}

# tests for the desired functionalities
function desired {
  # 1. archiving multiple files
  touch foo bar
  ./tar.debloat cf foo.tar foo bar
  rm foo bar
  ./tar.debloat xf foo.tar
  test -f foo -a -f bar || exit 1

  # 2. extracting from stdin
  touch foo
  ./tar.debloat cf foo.tar foo
  rm foo
  cat foo.tar | ./tar.debloat x
  test -f foo || exit 1

  # other tests
  …
  return 0
}
```

1. The program is compilable.

```
# tests for the undesired functionalities
function undesired {
  for test_script in `ls other_tests/*.sh`
  do
    { sh -x -e $test_script; } >& log
    grep 'Segmentation fault' log && exit 1
  done
  return 0
}

compile || exit 1
desired || exit 1
undesired || exit 1
```

High-level Specification

```
#!/bin/bash

function compile {
  clang -o tar.debloat tar-1.14.c
  return $?
}

# tests for the desired functionalities
function desired {
  # 1. archiving multiple files
  touch foo bar
  ./tar.debloat cf foo.tar foo bar
  rm foo bar
  ./tar.debloat xf foo.tar
  test -f foo -a -f bar || exit 1

  # 2. extracting from stdin
  touch foo
  ./tar.debloat cf foo.tar foo
  rm foo
  cat foo.tar | ./tar.debloat x
  test -f foo || exit 1

  # other tests
  …
  return 0
}
```

2. The program produces the same results with the desired functionalities.
(e.g., using regression test suites)

```
# tests for the undesired functionalities
function undesired {
  for test_script in `ls other_tests/*.sh`
  do
    { sh -x -e $test_script; } >& log
    grep 'Segmentation fault' log && exit 1
  done
  return 0
}

compile || exit 1
desired || exit 1
undesired || exit 1
```

High-level Specification

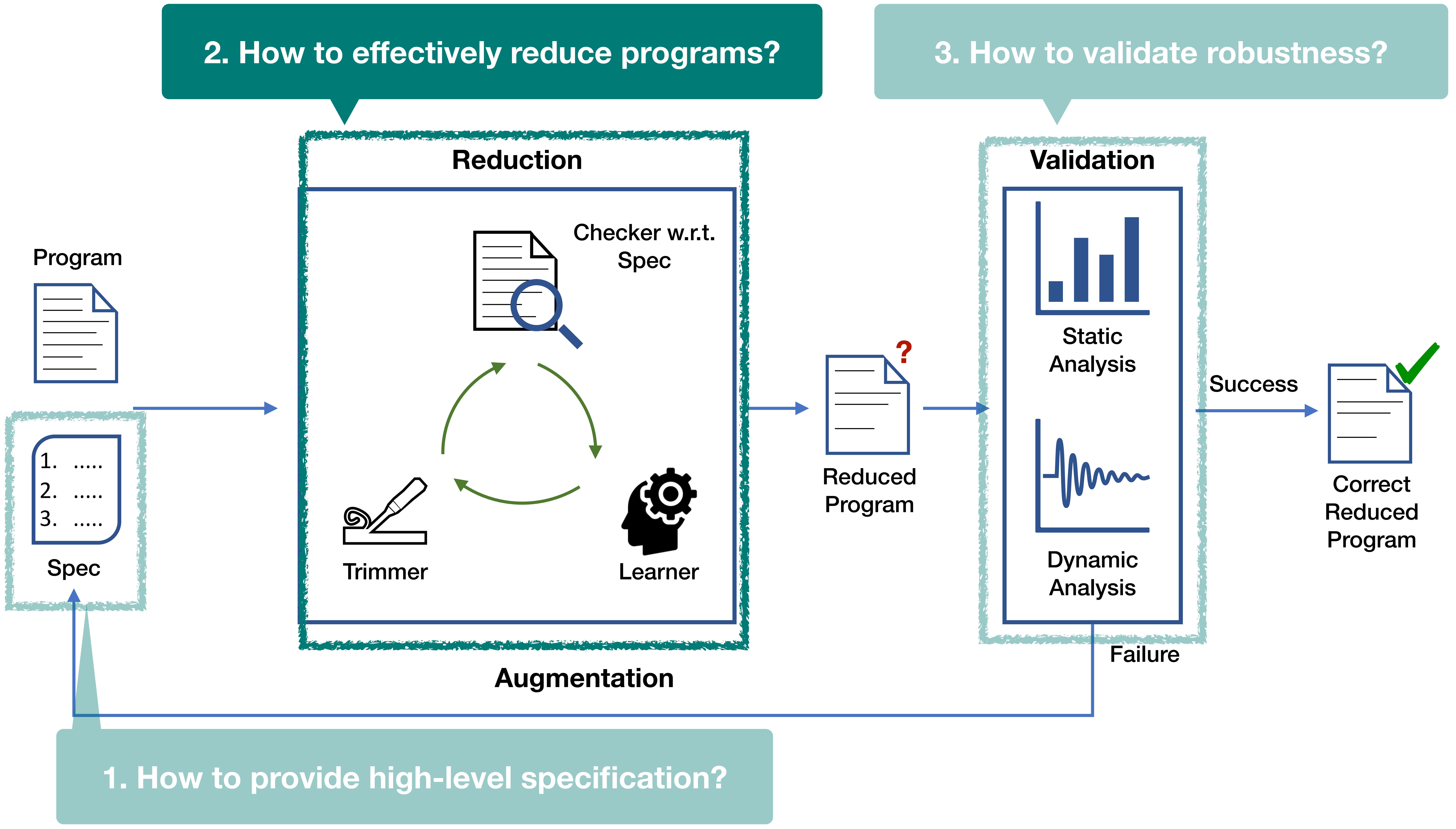
```
#!/bin/bash
```

3. The program does not crash with the undesired functionalities. (e.g., using Clang sanitizers)

```
function desired {
  # 1. archiving multiple files
  touch foo bar
  ./tar.debloat cf foo.tar foo bar
  rm foo bar
  ./tar.debloat xf foo.tar
  test -f foo -a -f bar || exit 1

  # 2. extracting from stdin
  touch foo
  ./tar.debloat cf foo.tar foo
  rm foo
  cat foo.tar | ./tar.debloat x
  test -f foo || exit 1

  # other tests
  …
  return 0
}
```

```
# tests for the undesired functionalities
function undesired {
  for test_script in `ls other_tests/*.sh`
  do
    { sh -x -e $test_script; } >& log
    grep 'Segmentation fault' log && exit 1
  done
  return 0
}
compile || exit 1
desired || exit 1
undesired || exit 1
```

Key Questions



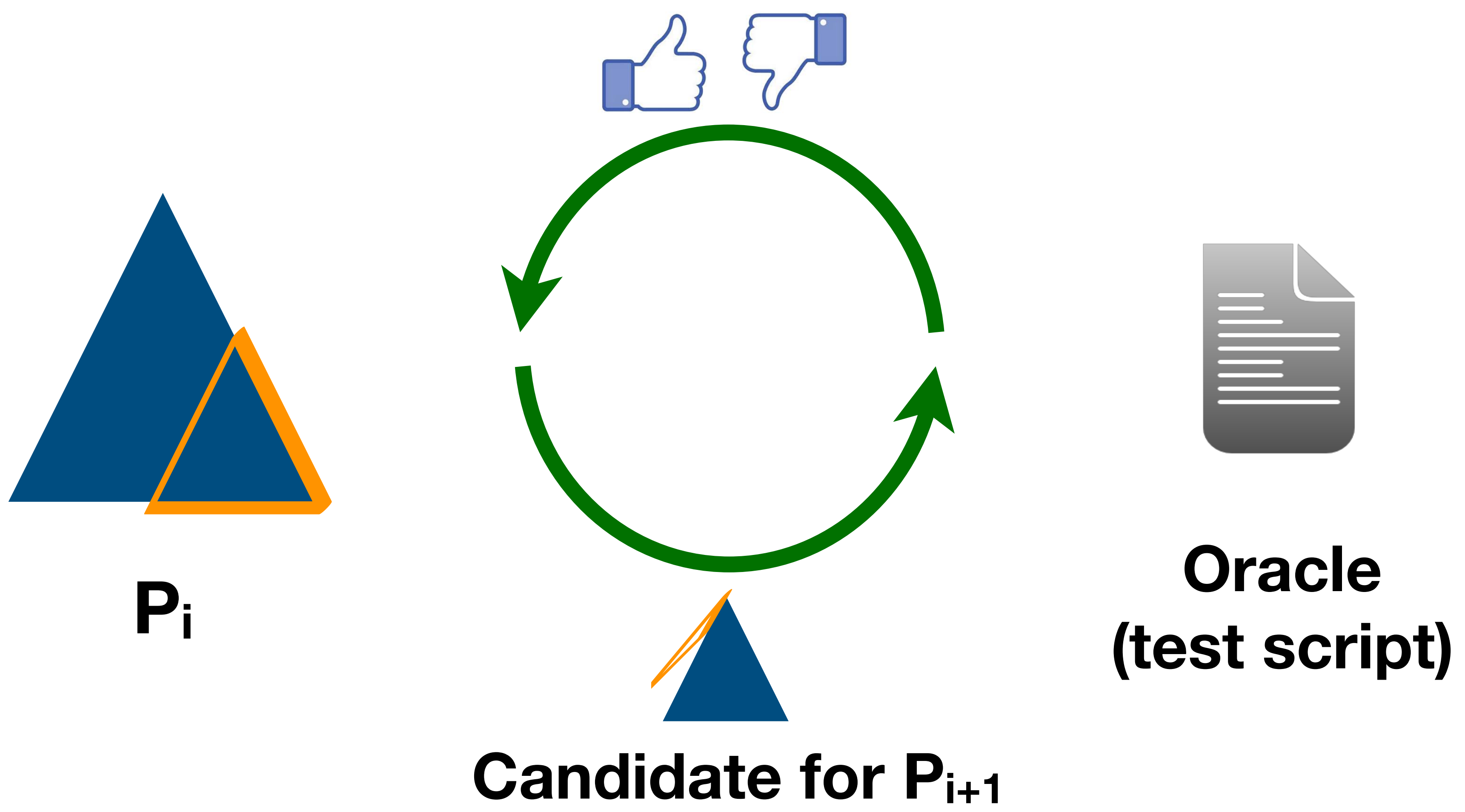
2. How to effectively reduce programs?

3. How to validate robustness?

1. How to provide high-level specification?

Program Debloating by Delta Debugging*

- Oracle O takes a program and returns **PASS** or **FAIL**
- Given a program P, find a **1-minimal** P^* such that $O(P^*)$ = **PASS**
- **1-minimal:** removing any single element of P^* does not pass O
- Time complexity: $O(|P|^2)$



P_i

Candidate for P_{i+1}

Oracle (test script)

*Zeller and Hildebrandt, simplifying and isolating failure-inducing input, TSE, 2002

Example

- Property of interest: termination with return code zero

Original

```
int f1() { return 0; }
int f2() { return 1; }
int f3() { return 1; }
int f4() { return 1; }
int f5() { return 1; }
int f6() { return 1; }
int f7() { return 1; }
int main() { return f1(); }
```

Minimal version

```
int f1() { return 0; }






int main() { return f1(); }
```

Example (Cont’d)

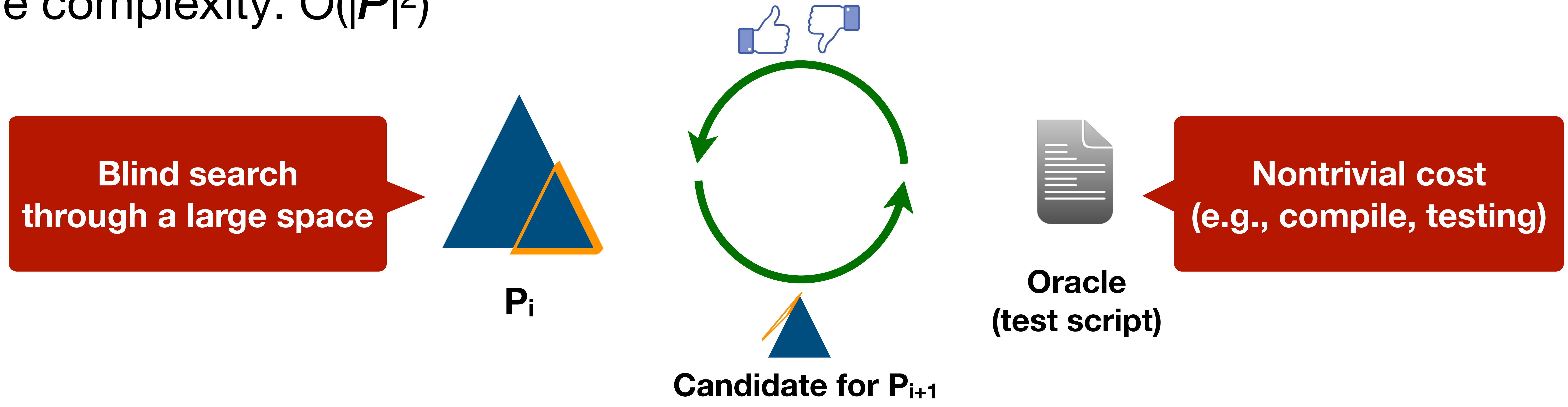
(**included**)

	f1	**f2**	**f3**	**f4**	**f5**	**f6**	**f7**	**main**	✔
1	**f1**	**f2**	**f3**	**f4**	f5	f6	f7	main	✘
2	f1	f2	f3	f4	**f5**	**f6**	**f7**	**main**	✘
3	**f1**	**f2**	f3	f4	f5	f6	f7	main	✘
4	f1	f2	**f3**	**f4**	f5	f6	f7	main	✘
5	f1	f2	f3	f4	**f5**	**f6**	f7	main	✘
6	f1	f2	f3	f4	f5	f6	**f7**	**main**	✘
7	f1	f2	**f3**	**f4**	**f5**	**f6**	**f7**	**main**	✘
8	**f1**	**f2**	f3	f4	**f5**	**f6**	**f7**	**main**	✔
9	**f1**	**f2**	f3	f4	f5	f6	**f7**	**main**	✔
10	**f1**	f2	f3	f4	f5	f6	f7	main	✘
11	f1	**f2**	f3	f4	f5	f6	f7	main	✘
12	f1	f2	f3	f4	f5	f6	**f7**	main	✘
13	f1	f2	f3	f4	f5	f6	f7	**main**	✘
14	f1	**f2**	f3	f4	f5	f6	**f7**	**main**	✘
15	**f1**	f2	f3	f4	f5	f6	**f7**	**main**	✔
16	**f1**	f2	f3	f4	f5	f6	f7	**main**	✔

*All duplications are omitted

델타 디버깅의 문제

- Oracle O takes a program and returns **PASS** or **FAIL**
- Given a program *P*, find a **1-minimal** *P** such that O(*P**) = **PASS**
- **1-minimal:** removing any single element of *P** does not pass O
- Time complexity: O(|*P*|^2)



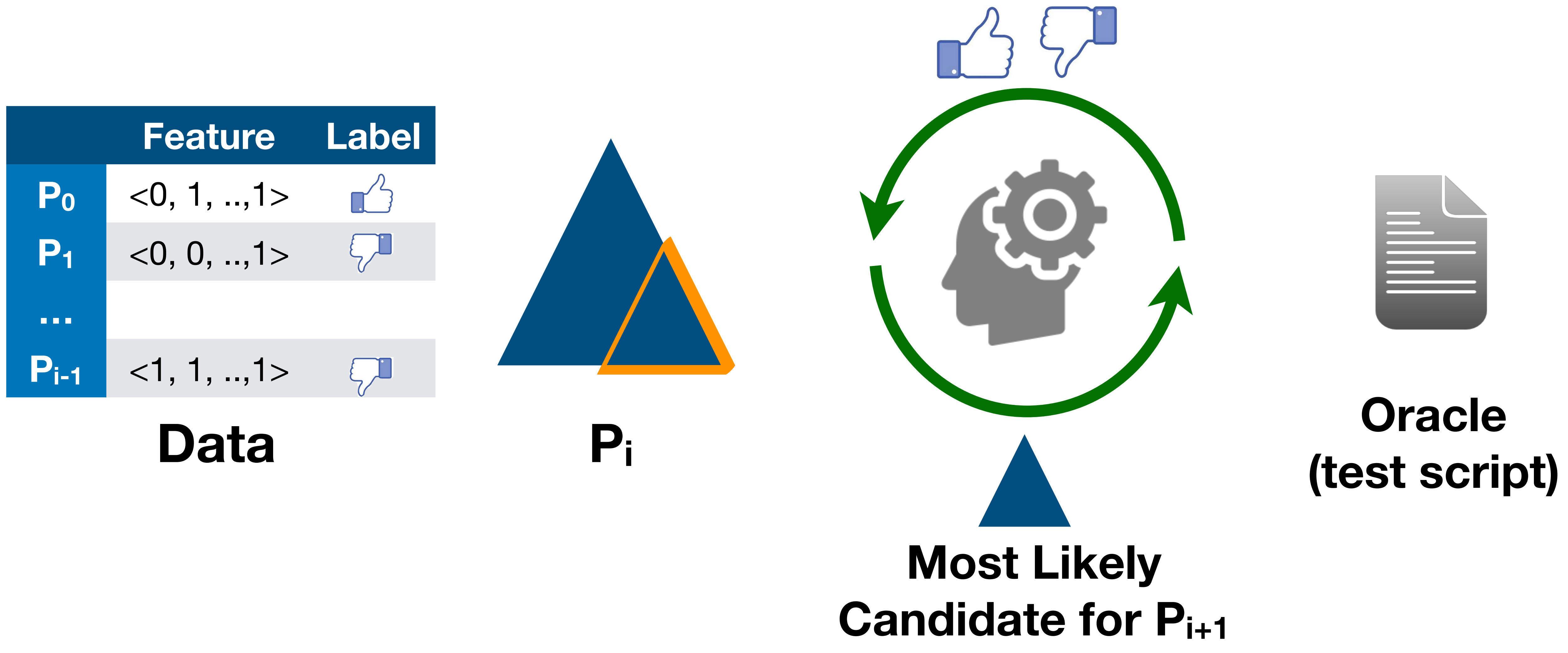
Blind search through a large space

Nontrivial cost (e.g., compile, testing)

*Zeller and Hildebrandt, simplifying and isolating failure-inducing input, TSE, 2002

Learning-guided Delta Debugging

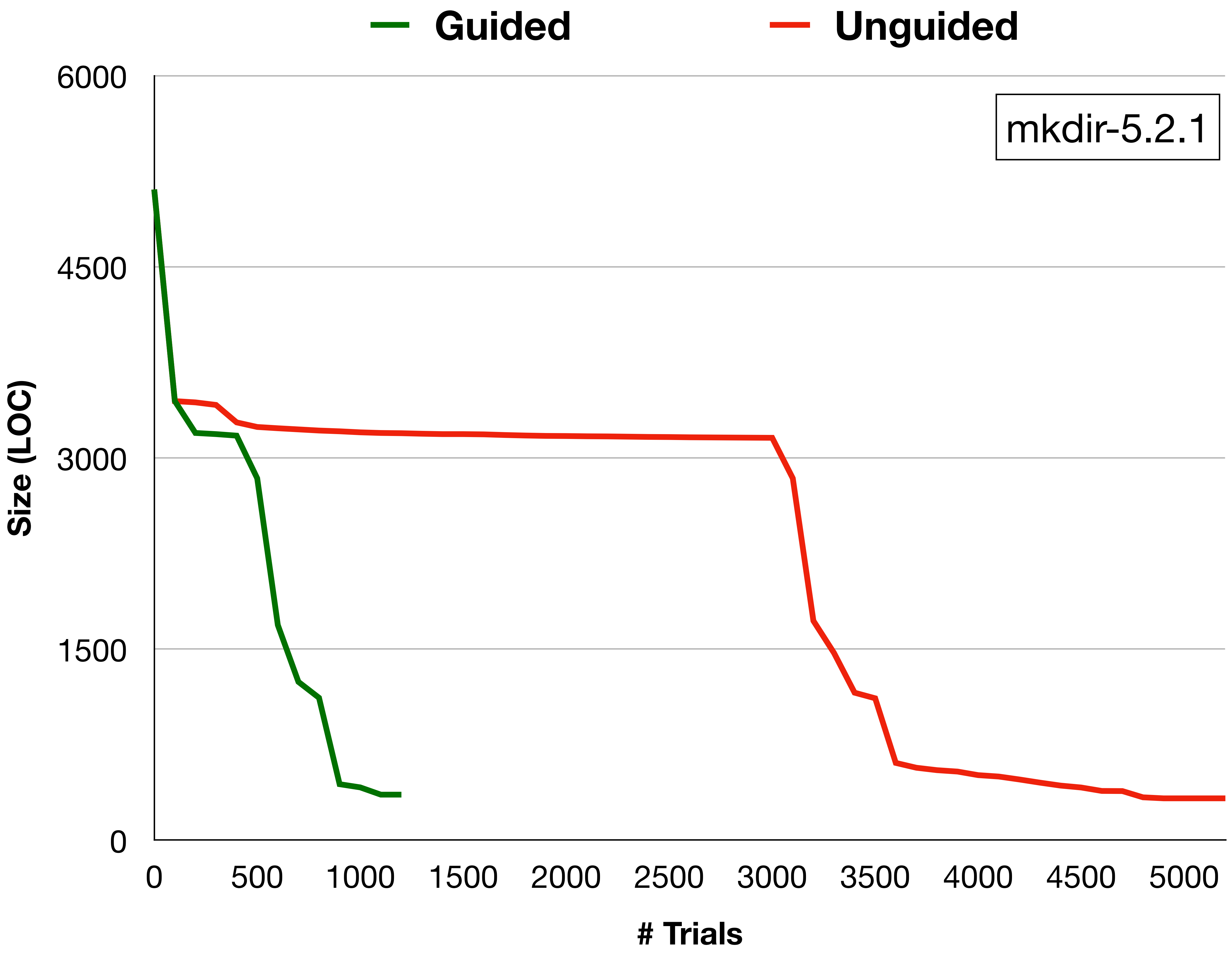
- **Learn a policy** for DD using reinforcement learning (RL)
- **Guide the search** based on the prediction of the learned policy
- Still guarantee **1-minimality** and **O(|*P*|^2)** time complexity



	Feature	Label
P_0	<0, 1, ..,1>	👍
P_1	<0, 0, ..,1>	👎
…		
P_{i-1}	<1, 1, ..,1>	👎

Data

Effectiveness



Example

```
/* mkdir-5.2.1 */
int xstrtol(char *s, char **ptr, int strtol_base, strtol_t *val,
            char *valid_suffixes) {
    err = 0;
    assert(0 <= strtol_base && strtol_base <= 36);
    p = ptr ? ptr : &t_ptr;
    q = s;
    while(ISSPACE (*q)) ++q;
    if (*q == ‘-‘) return LONGINT_INVALID;
    errno = 0;
    tmp = strtol(s, p, strtol_base);
    if (*p == s) { … }
    if (!valid_suffixes) { … }
    if (**p != ‘\0’) { … }
    *val = tmp;
    return err;
}
```

▨ : removed code

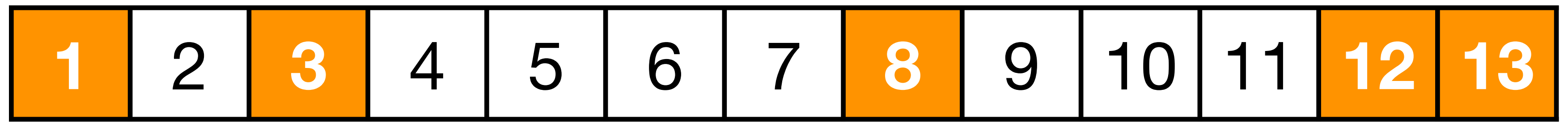
Example

```
/* mkdir-5.2.1 */
int xstrtol(char *s, char **ptr, int strtol_base, strtol_t *val,
            char *valid_suffixes) {
 1: err = 0;
 2: assert(0 <= strtol_base && strtol_base <= 36);
 3: p = ptr ? ptr : &t_ptr;
 4: q = s;
 5: while(ISSPACE (*q)) ++q;
 6: if (*q == ‘-‘) return LONGINT_INVALID;
 7: errno = 0;
 8: tmp = strtol(s, p, strtol_base);
 9: if (*p == s) { … }
10: if (!valid_suffixes) { … }
11: if (**p != ‘\0’) { … }
12: *val = tmp;
13: return err;
}
```

▨ : removed code

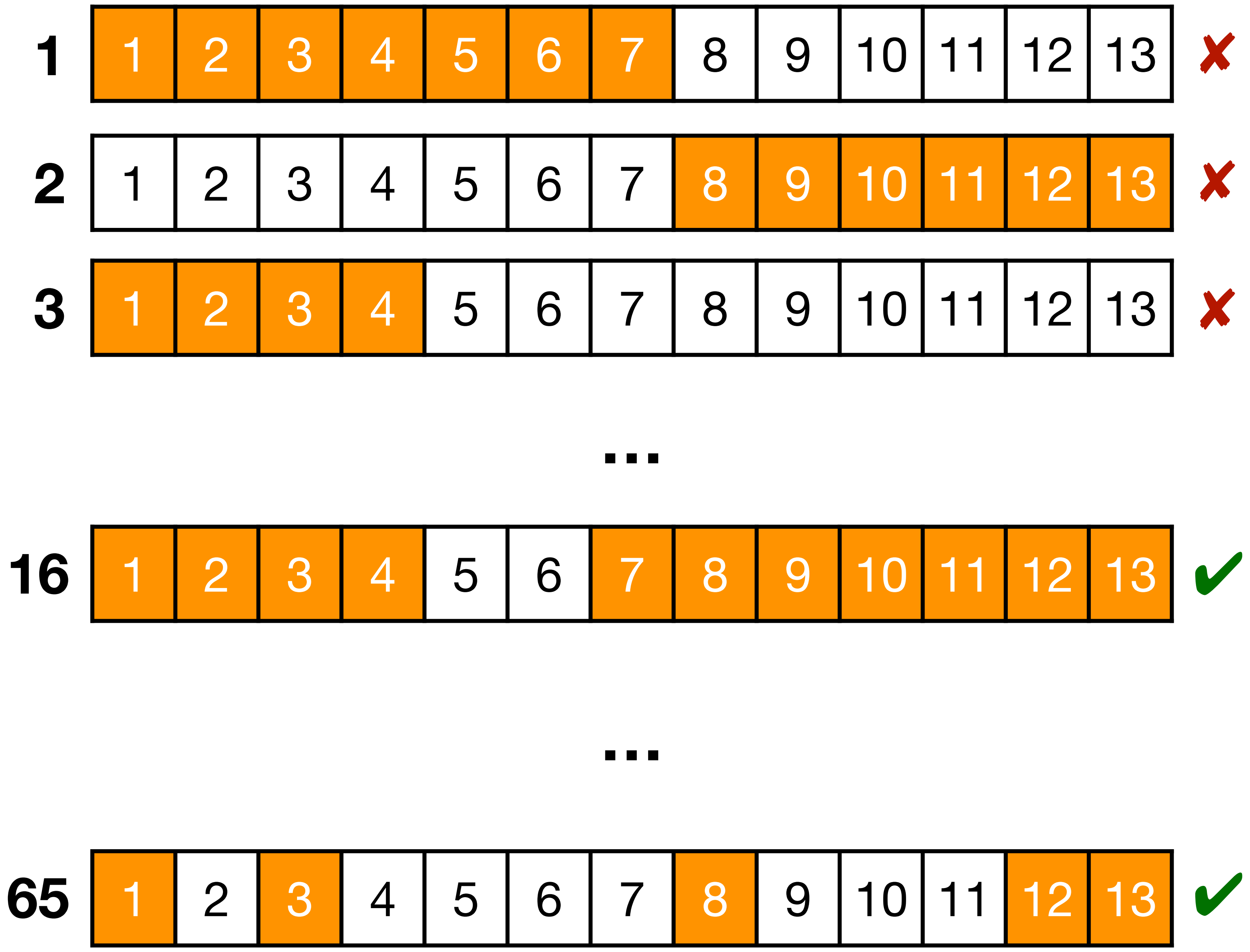
Minimal Desired Program:

1	2	3	4	5	6	7	8	9	10	11	12	13



1	2	**3**	4	5	6	7	**8**	9	10	11	**12**	**13**

Unguided Delta-Debugging



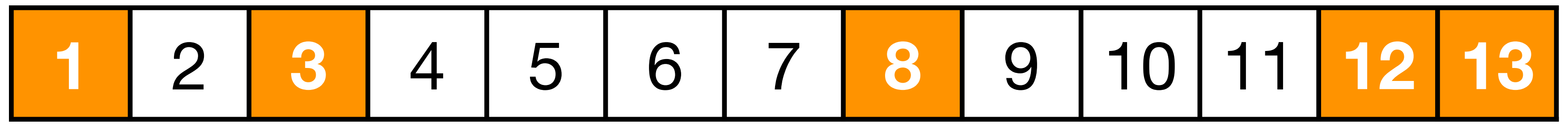
1	**1**	**2**	**3**	**4**	**5**	**6**	**7**	8	9	10	11	12	13	✗
2	1	2	3	4	5	6	7	**8**	**9**	**10**	**11**	**12**	**13**	✗
3	**1**	**2**	**3**	**4**	5	6	7	8	9	10	11	12	13	✗

...

16	**1**	**2**	**3**	**4**	5	6	**7**	**8**	**9**	**10**	**11**	**12**	**13**	✓

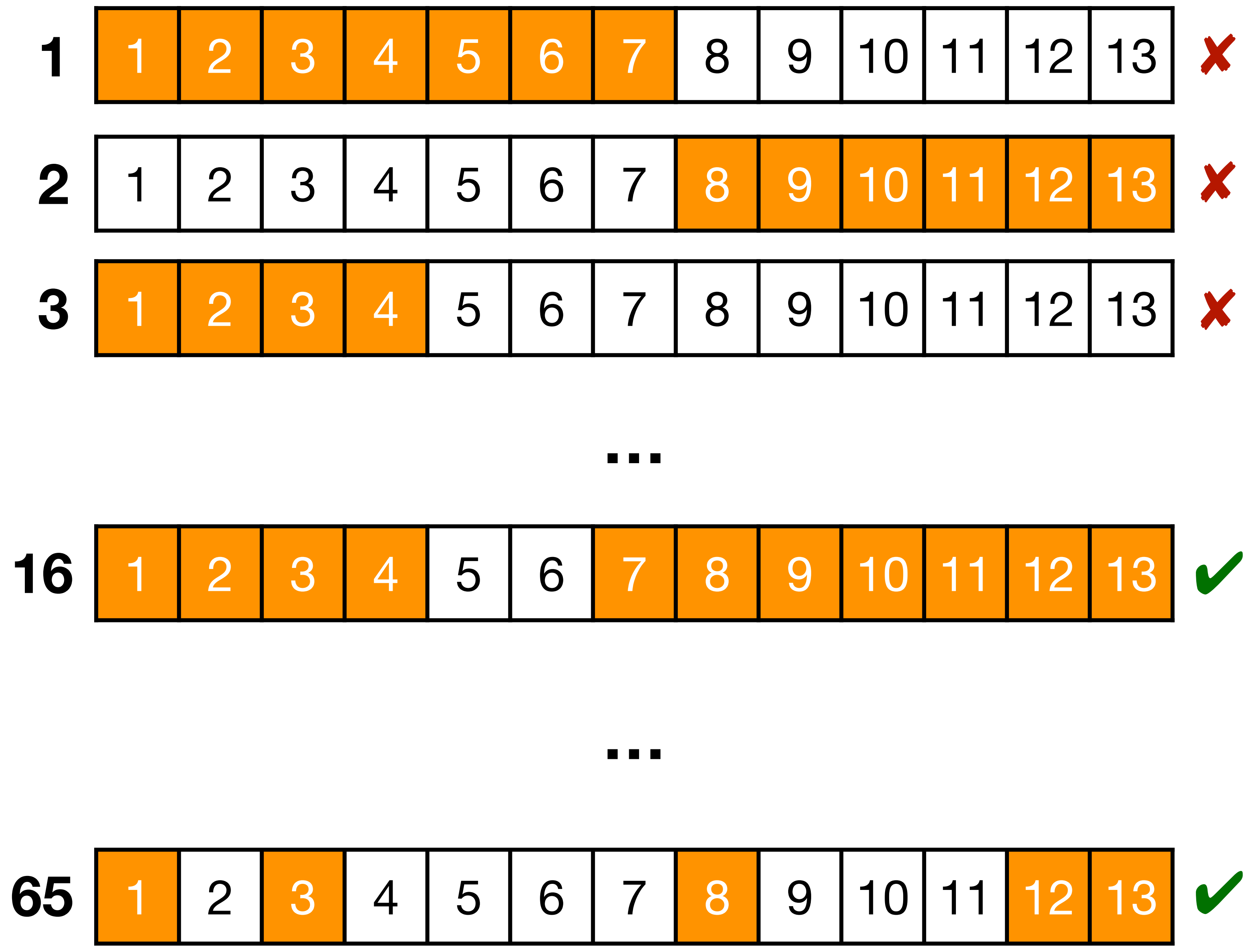
...

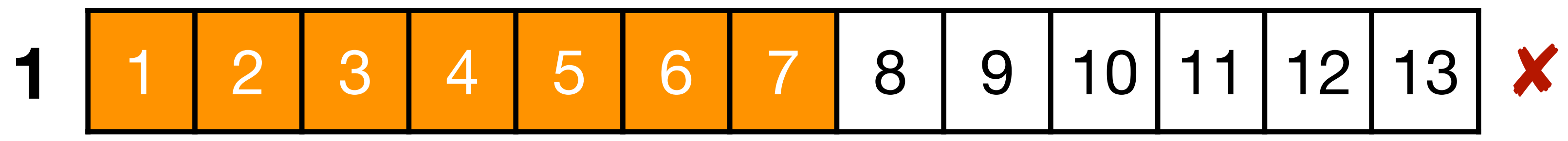
65	**1**	2	**3**	4	5	6	7	**8**	9	10	11	**12**	**13**	✓

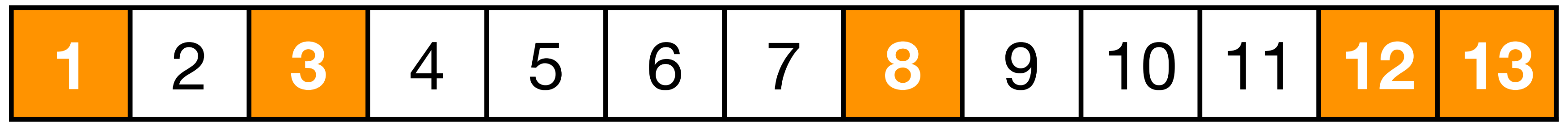


Unguided Delta-Debugging

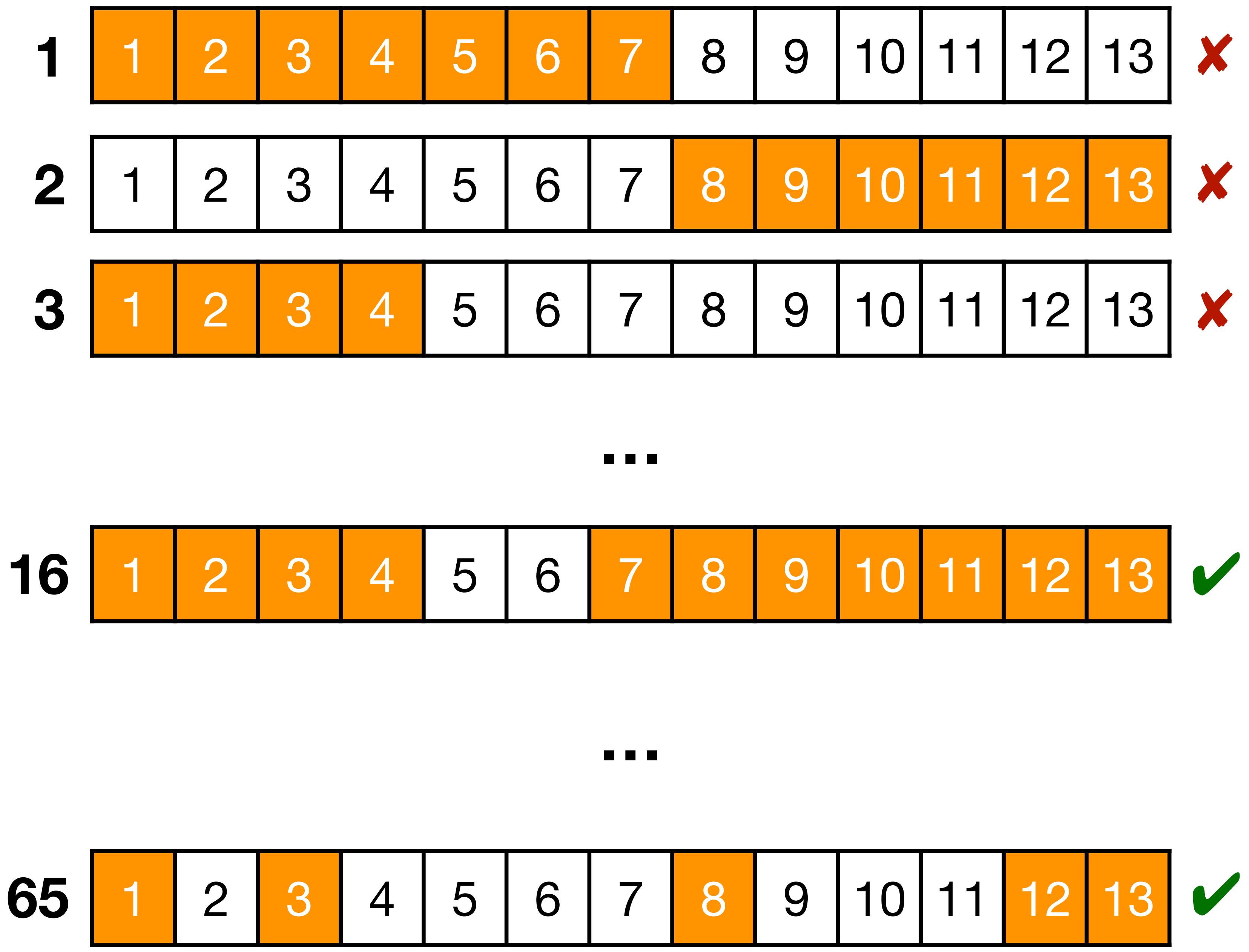
Guided Delta-Debugging



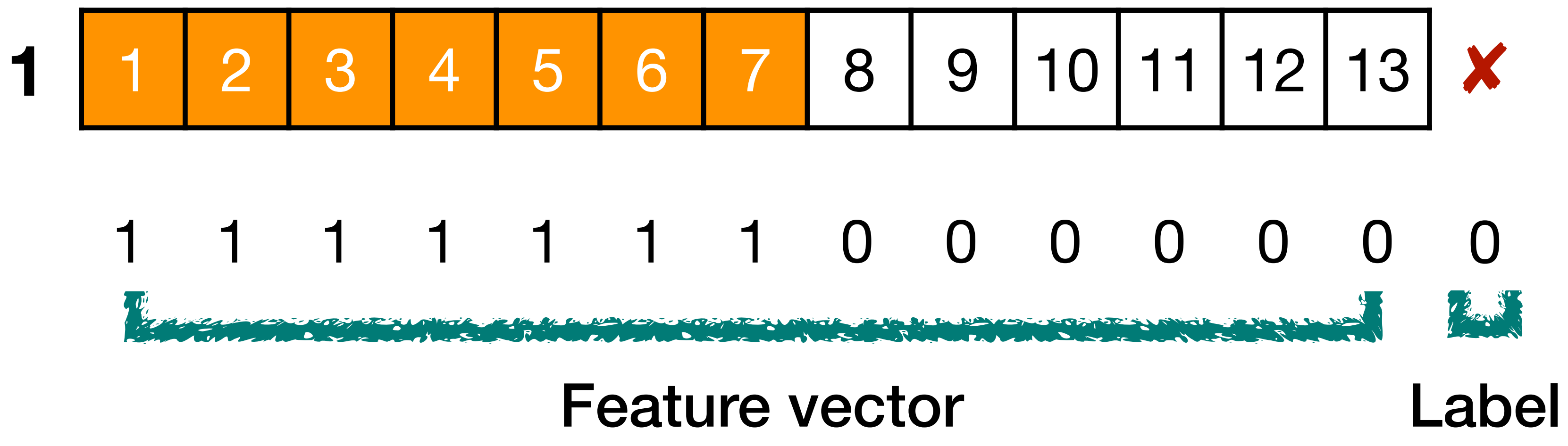


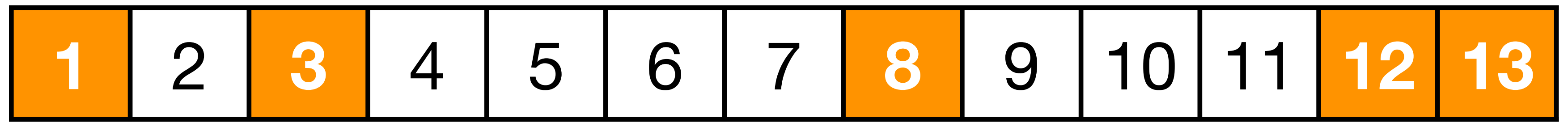


Unguided Delta-Debugging

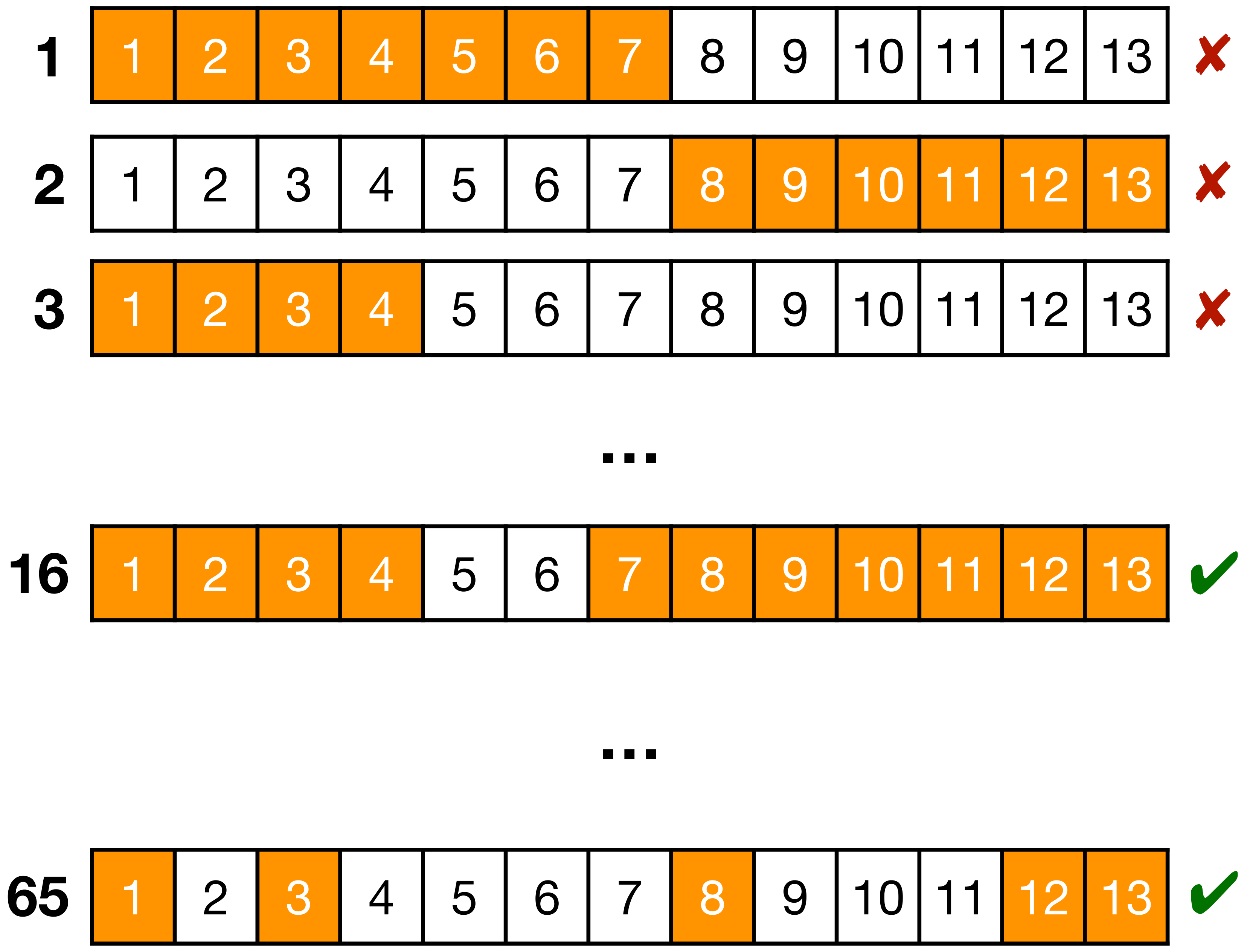


Guided Delta-Debugging

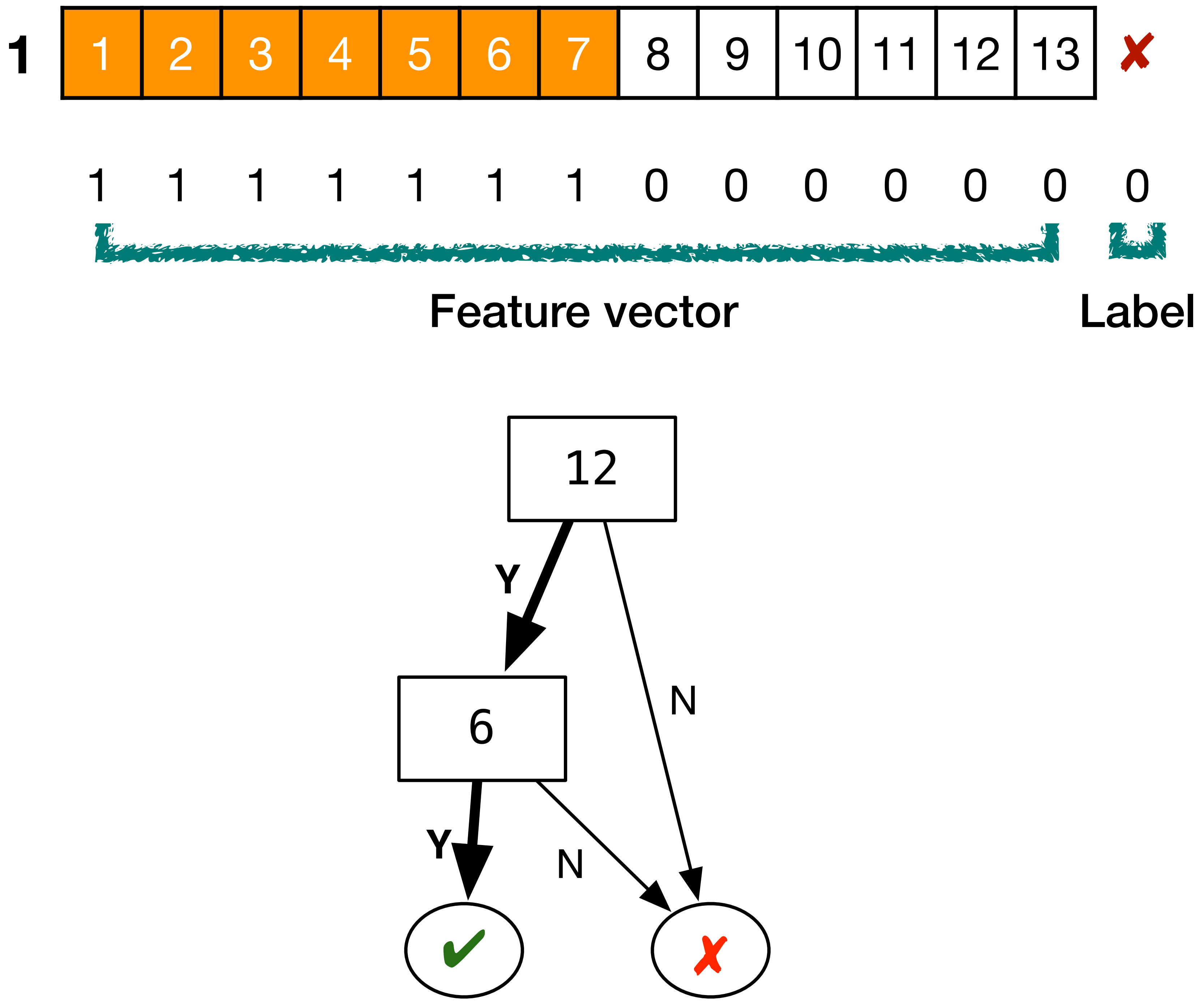




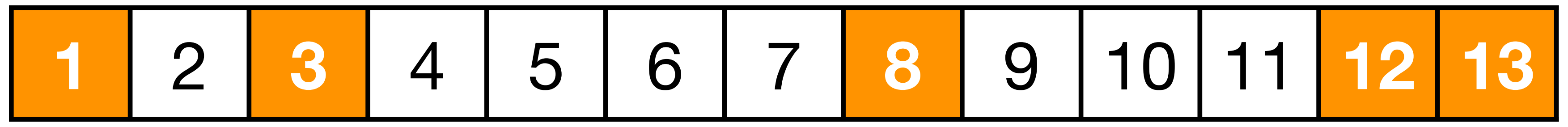
Unguided Delta-Debugging



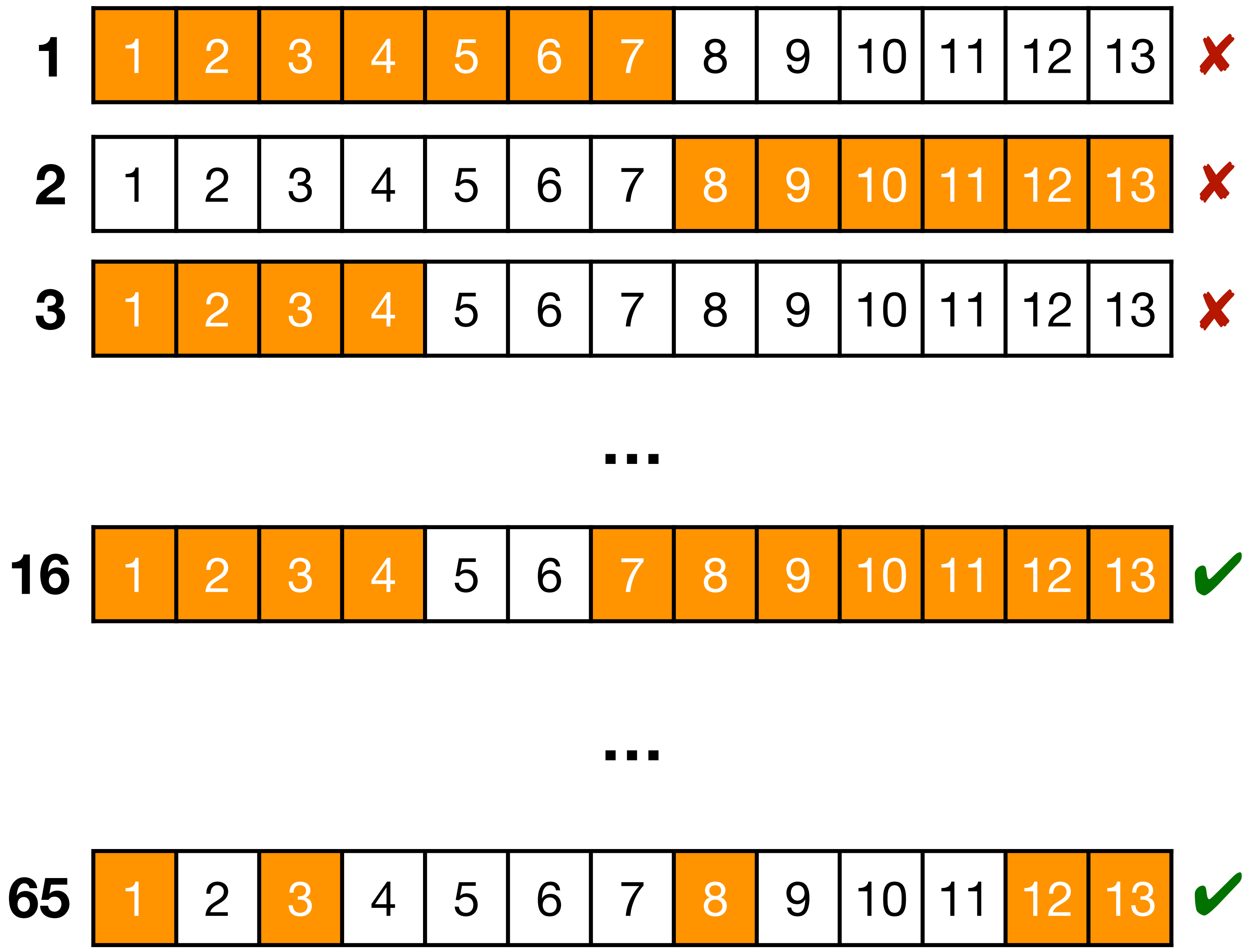
Guided Delta-Debugging



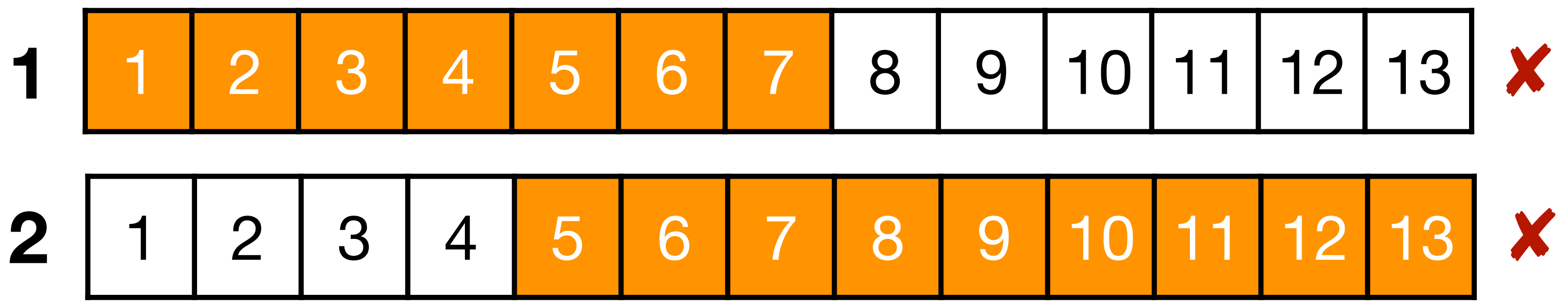
*P** should include 6 and 12

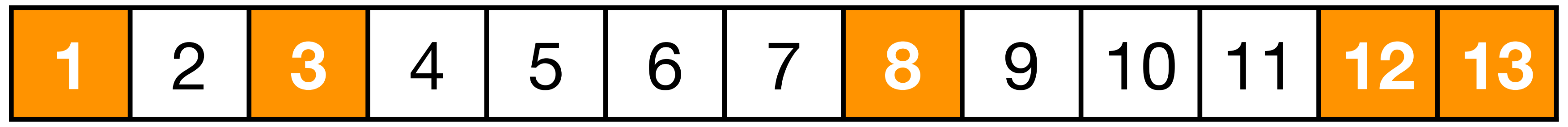


Unguided Delta-Debugging

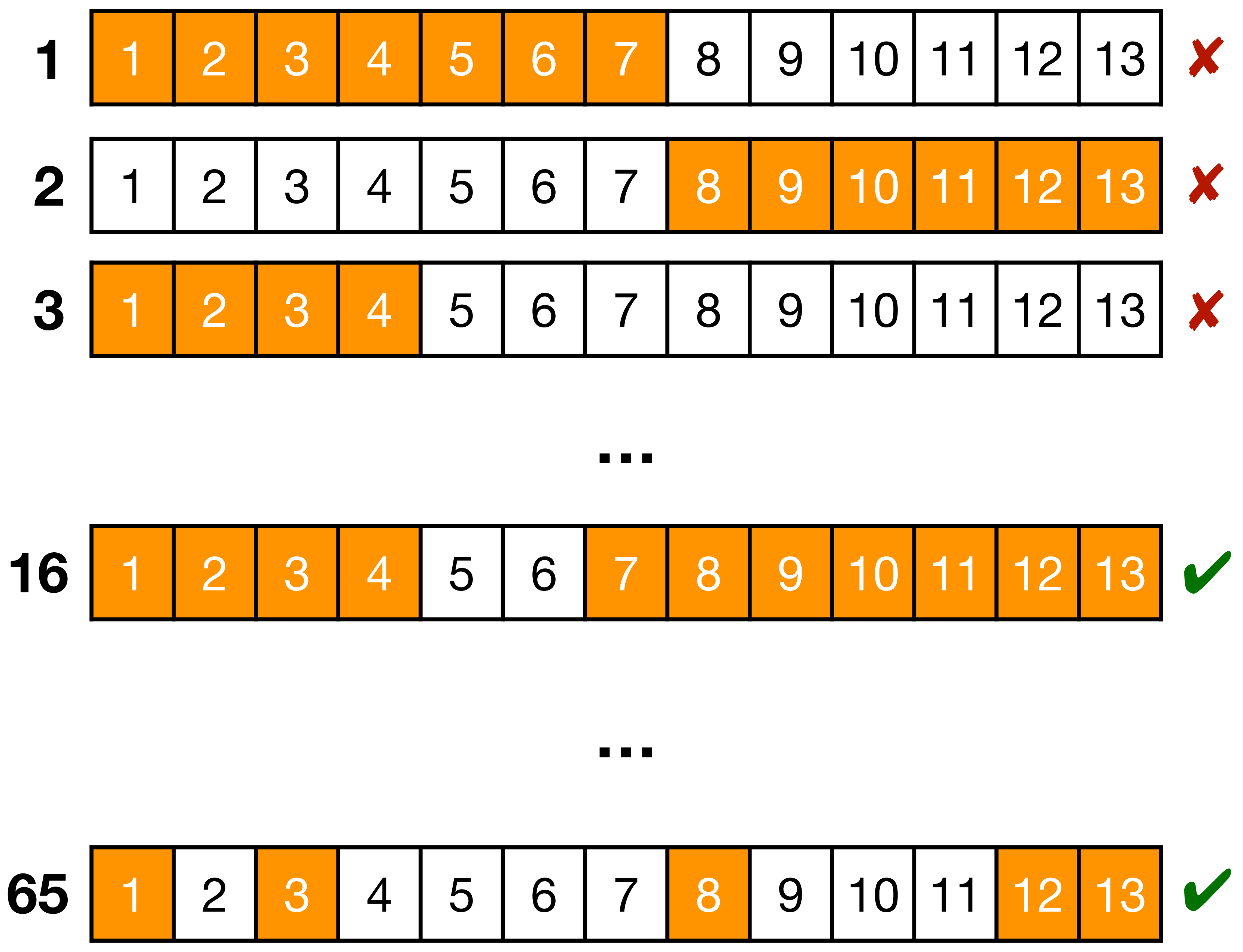


Guided Delta-Debugging

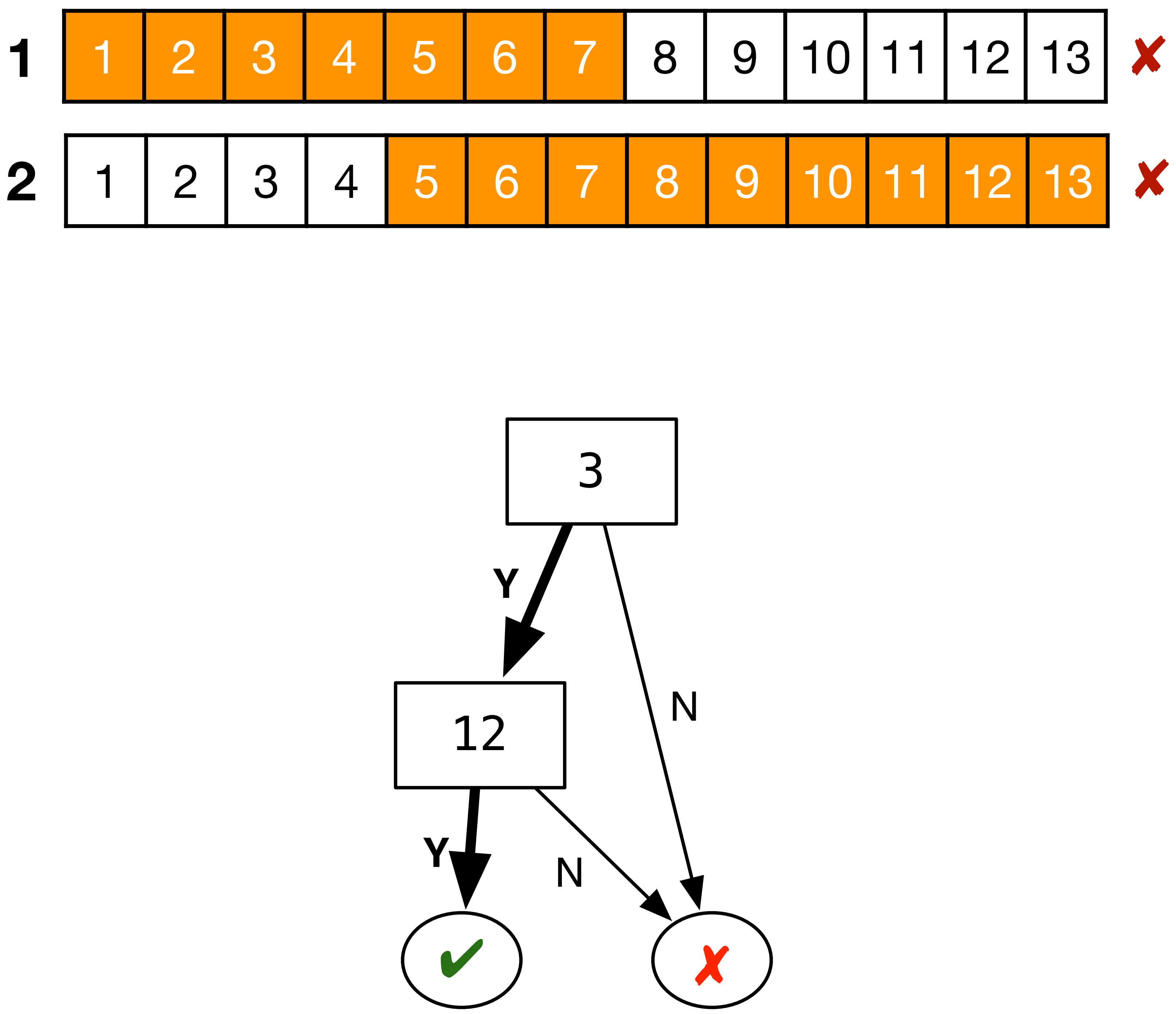


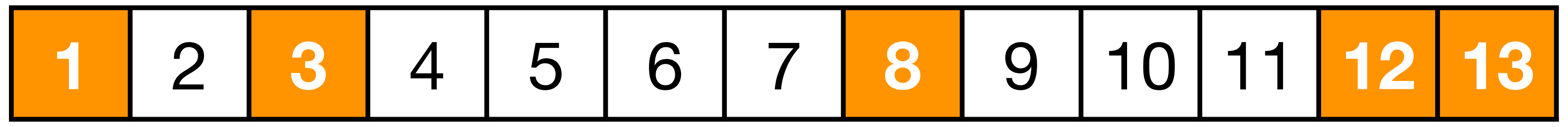


Unguided Delta-Debugging

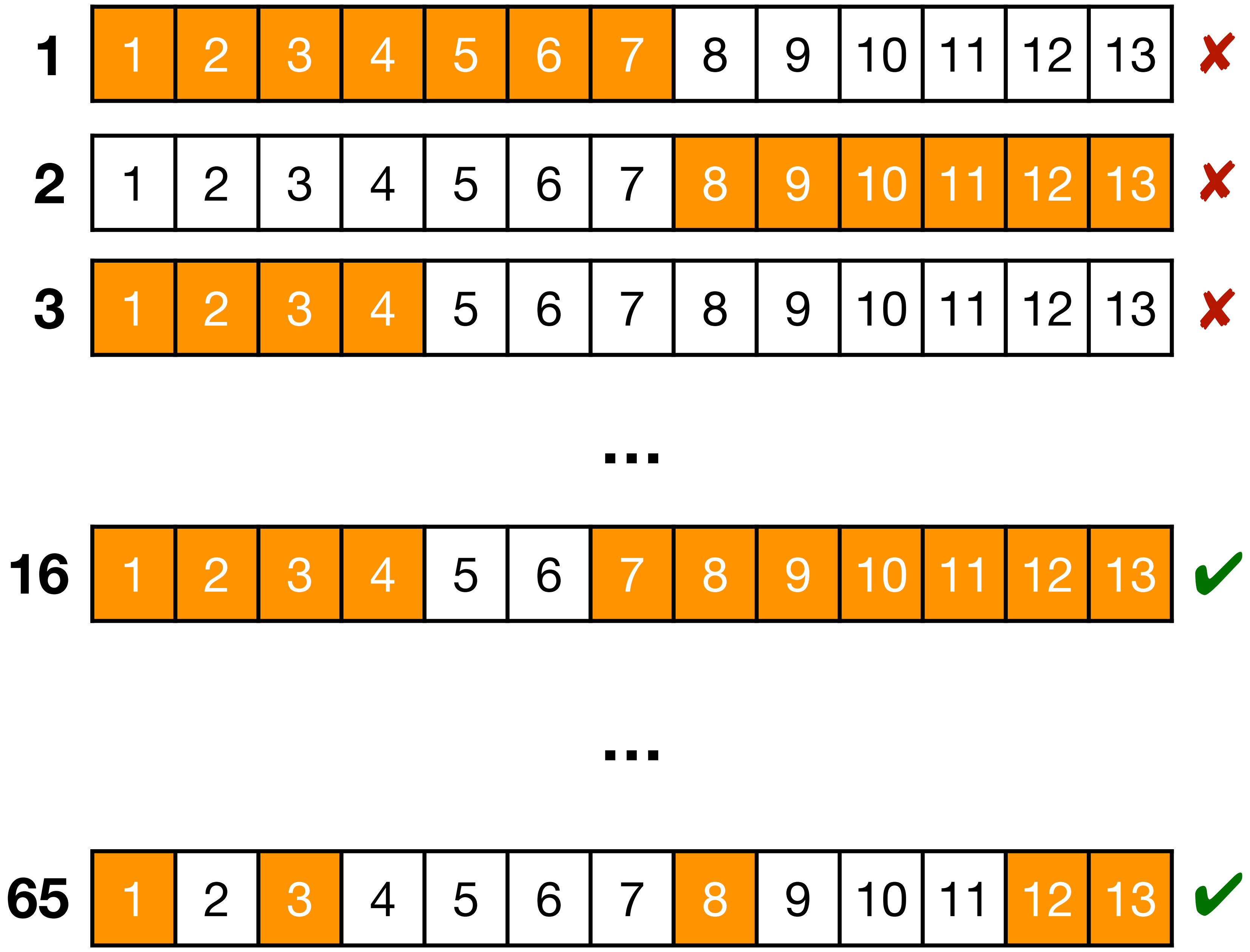


Guided Delta-Debugging

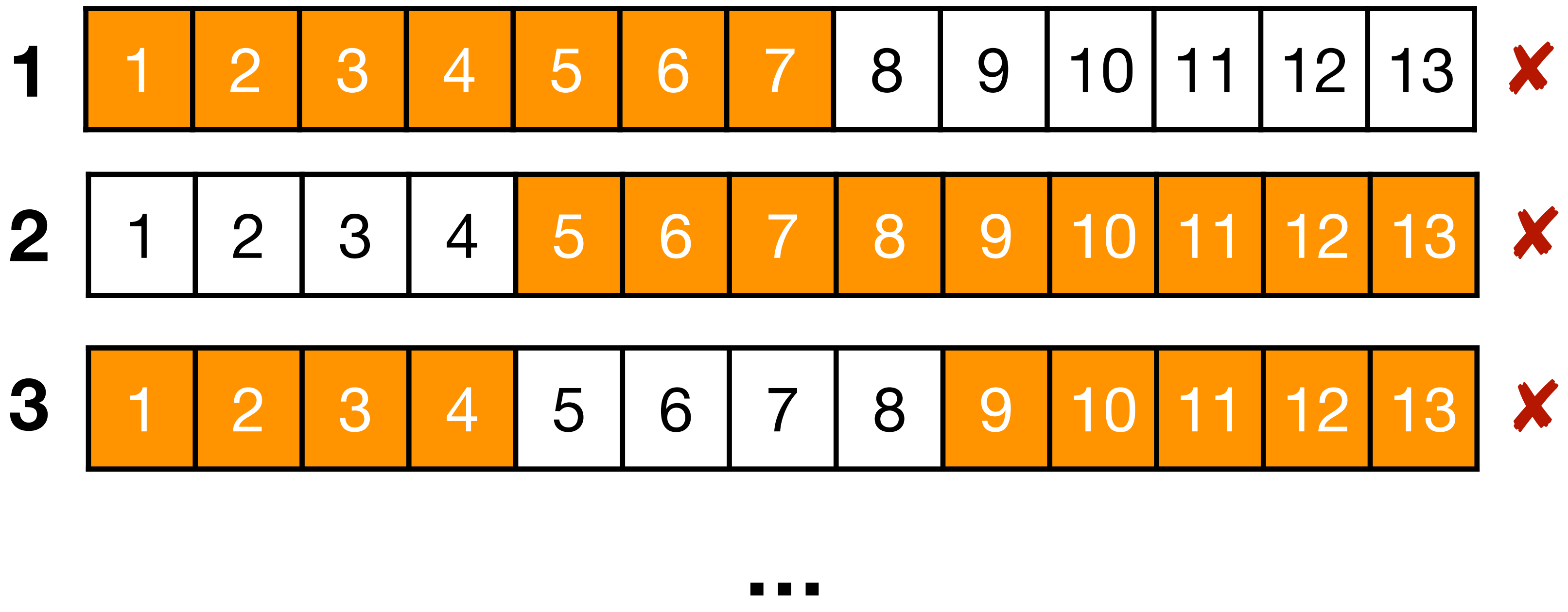


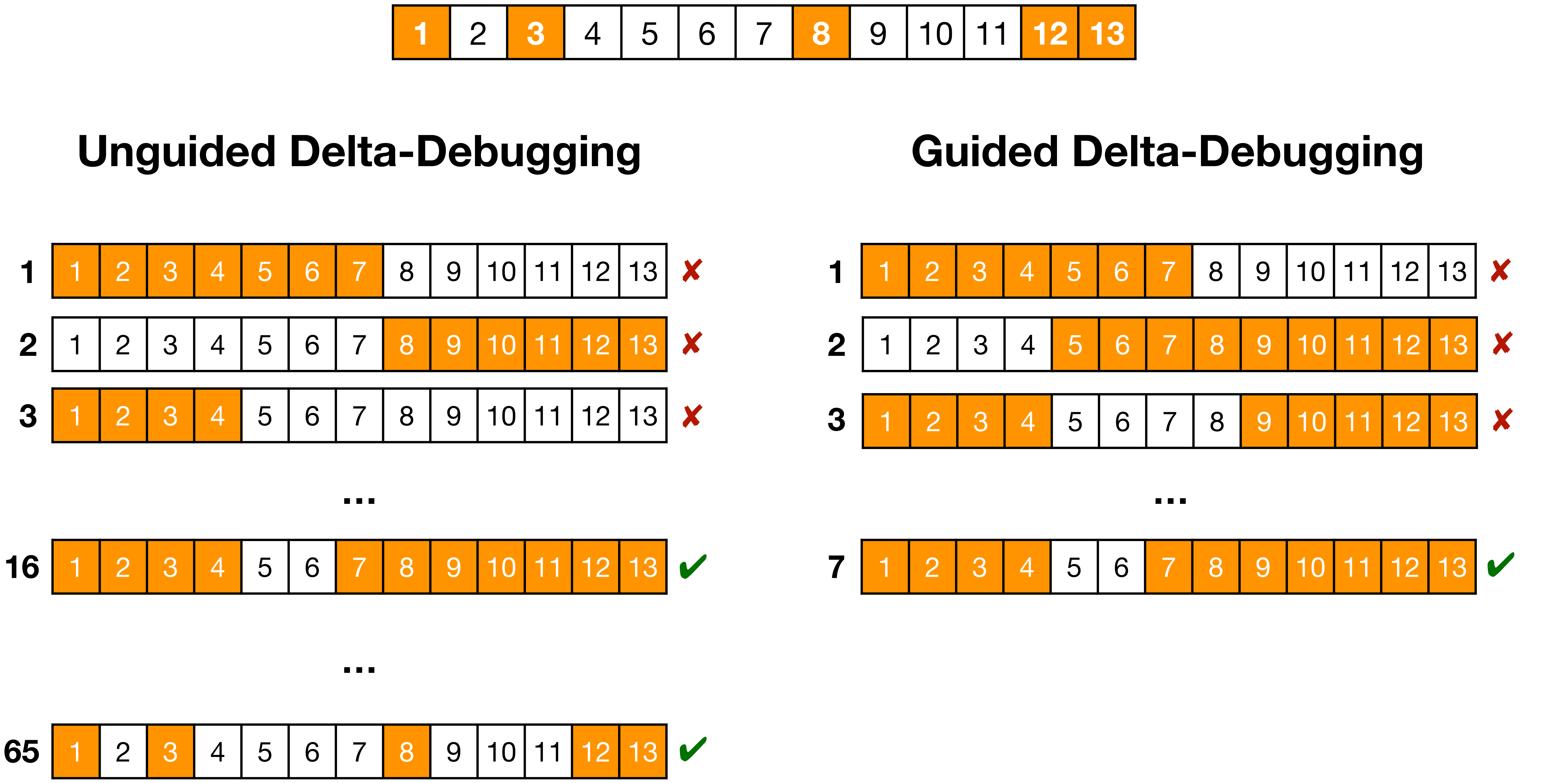


Unguided Delta-Debugging



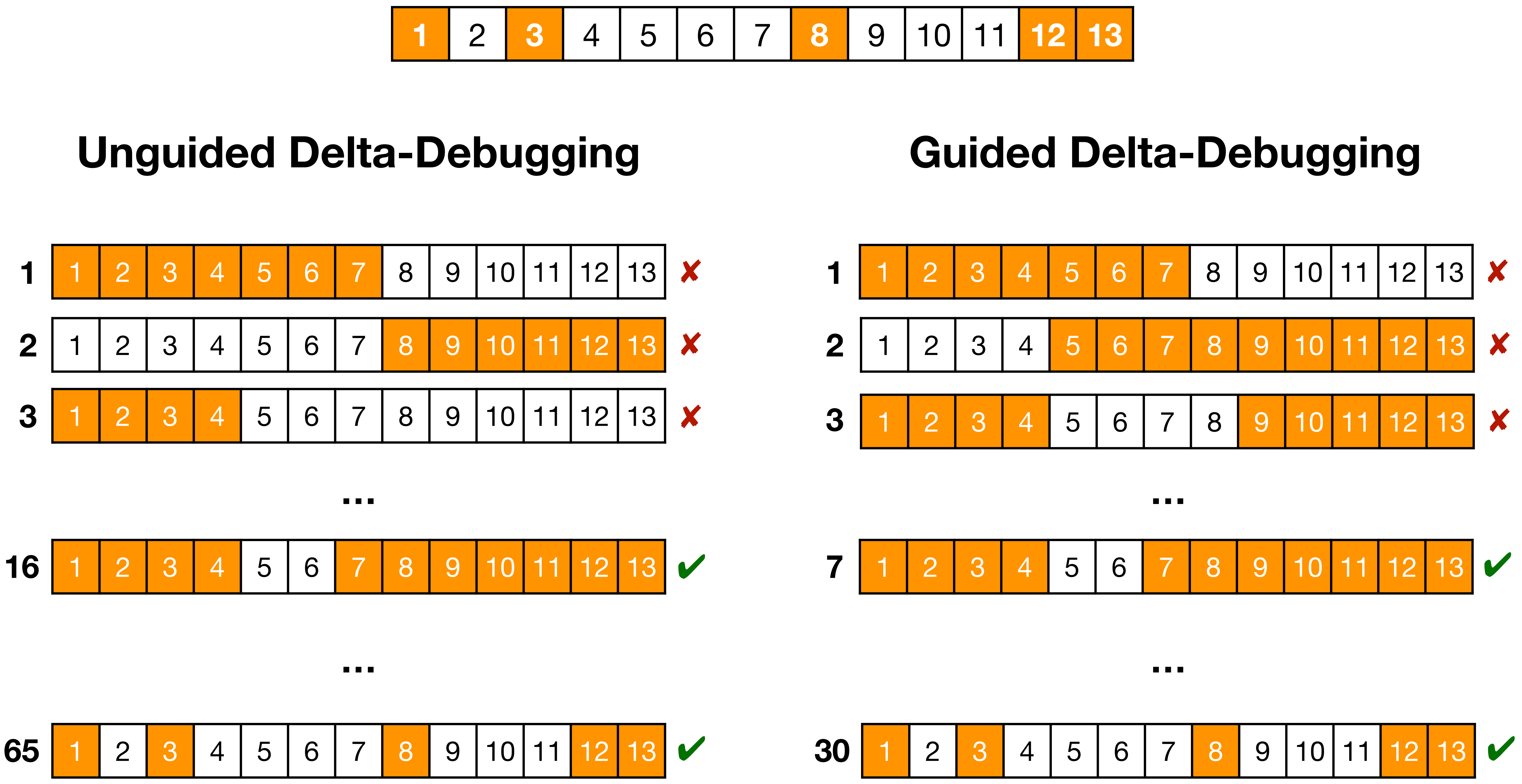
Guided Delta-Debugging





Unguided Delta-Debugging

Guided Delta-Debugging



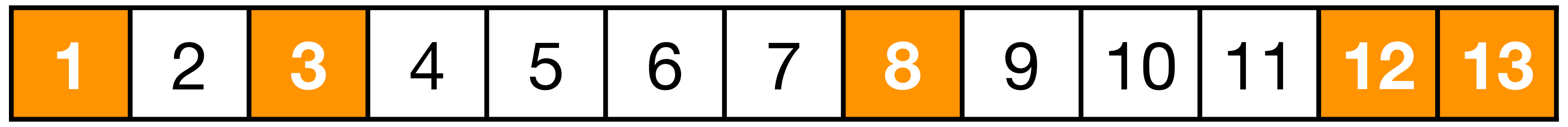
1	2	**3**	4	5	6	7	**8**	9	10	11	**12**	**13**

Unguided Delta-Debugging

1	**1**	**2**	**3**	**4**	**5**	**6**	**7**	8	9	10	11	12	13	✗
2	1	2	3	4	5	6	7	**8**	**9**	**10**	**11**	**12**	**13**	✗
3	**1**	**2**	**3**	**4**	5	6	7	8	9	10	11	12	13	✗
							...							
16	**1**	**2**	**3**	**4**	5	6	**7**	**8**	**9**	**10**	**11**	**12**	**13**	✓
							...							
65	**1**	2	**3**	4	5	6	7	**8**	9	10	11	**12**	**13**	✓

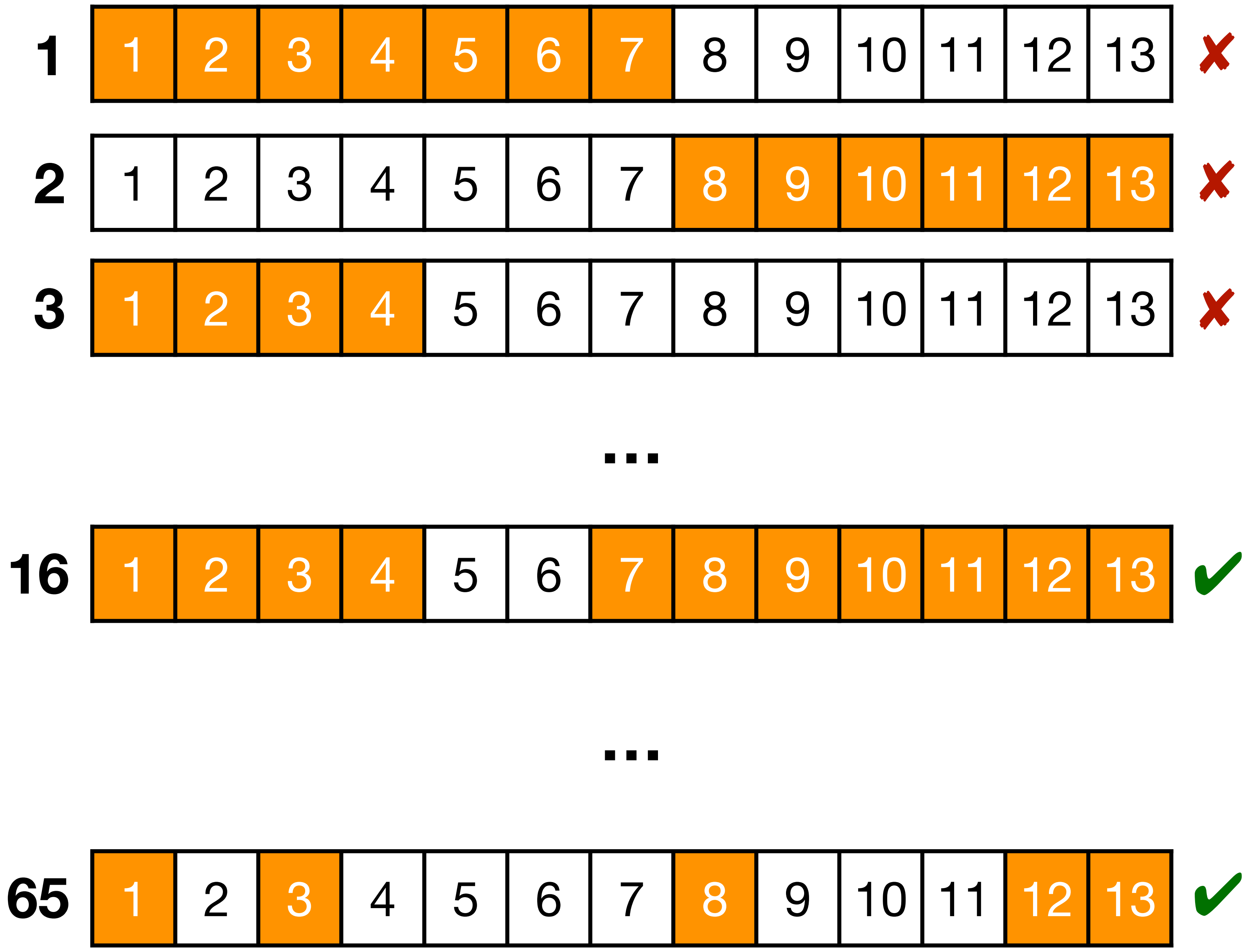
Guided Delta-Debugging

1	**1**	**2**	**3**	**4**	**5**	**6**	**7**	8	9	10	11	12	13	✗
2	1	2	3	4	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	✗
3	**1**	**2**	**3**	**4**	5	6	7	8	**9**	**10**	**11**	**12**	**13**	✗
							...							
7	**1**	**2**	**3**	**4**	5	6	**7**	**8**	**9**	**10**	**11**	**12**	**13**	✓
							...							
30	**1**	2	**3**	4	5	6	7	**8**	9	10	11	**12**	**13**	✓



1	2	**3**	4	5	6	7	**8**	9	10	11	**12**	**13**

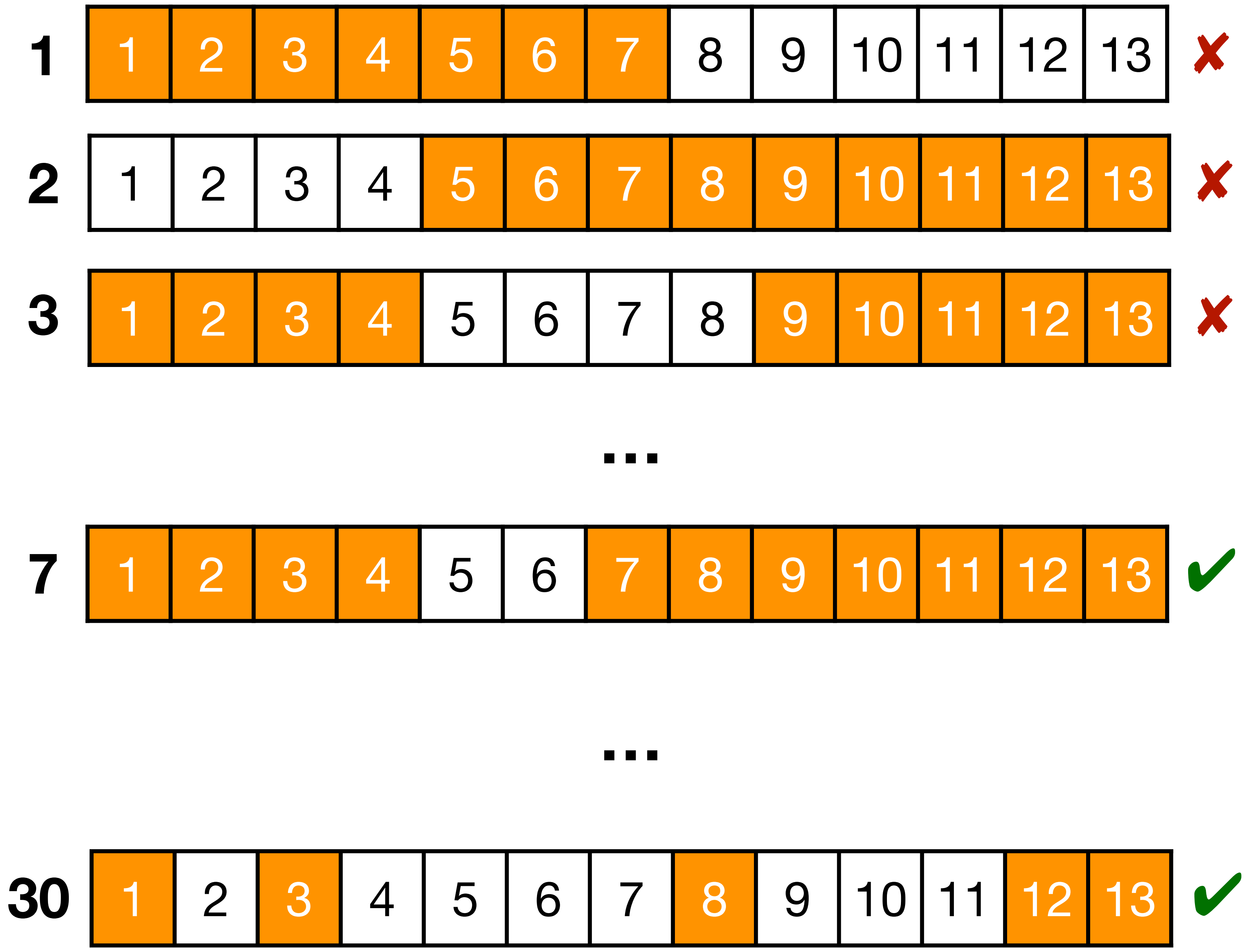
Unguided Delta-Debugging



1	**1**	**2**	**3**	**4**	**5**	**6**	**7**	8	9	10	11	12	13	✗
2	1	2	3	4	5	6	7	**8**	**9**	**10**	**11**	**12**	**13**	✗
3	**1**	**2**	**3**	**4**	5	6	7	8	9	10	11	12	13	✗
							...							
16	**1**	**2**	**3**	**4**	5	6	**7**	**8**	**9**	**10**	**11**	**12**	**13**	✓
							...							
65	**1**	2	**3**	4	5	6	7	**8**	9	10	11	**12**	**13**	✓

5,169 trials (4,872 failures)

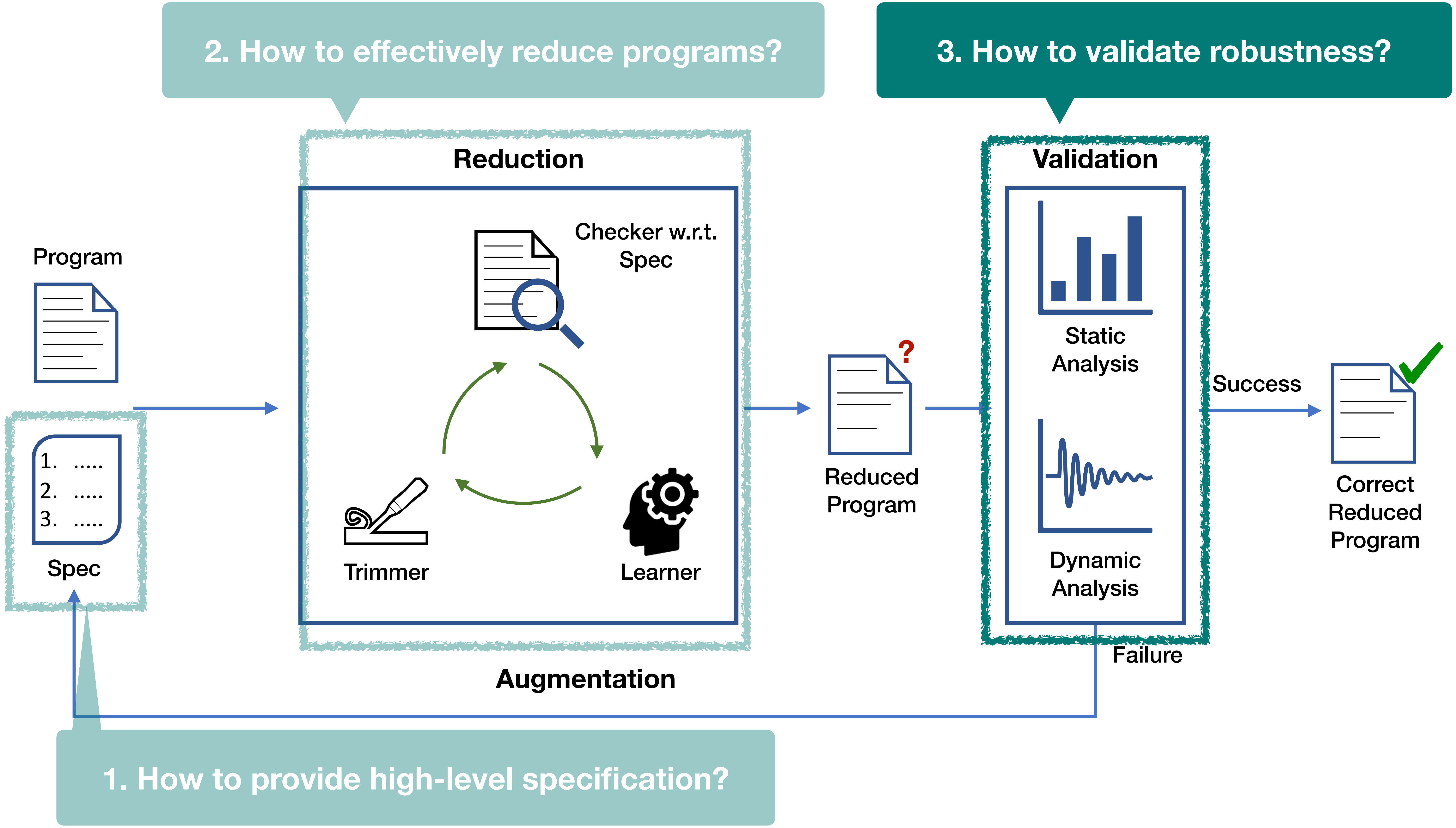
Guided Delta-Debugging



1	**1**	**2**	**3**	**4**	**5**	**6**	**7**	8	9	10	11	12	13	✗
2	1	2	3	4	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	✗
3	**1**	**2**	**3**	**4**	5	6	7	8	**9**	**10**	**11**	**12**	**13**	✗
							...							
7	**1**	**2**	**3**	**4**	5	6	**7**	**8**	**9**	**10**	**11**	**12**	**13**	✓
							...							
30	**1**	2	**3**	4	5	6	7	**8**	9	10	11	**12**	**13**	✓

1,174 trials (901 failures)

Key Questions

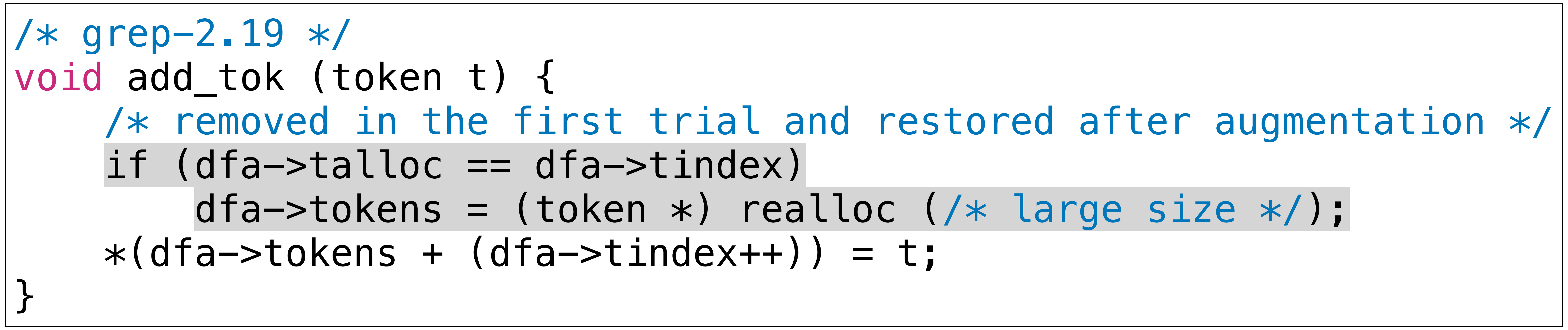


Validation

- Check the **robustness** of the reduced program
  - preventing newly introduced security holes
- Sound static buffer overflow analyzer (Sparrow)
  - #alarms in tar: 1,290 → 19 (feasible for manual inspection)
- Random fuzzer (AFL)
  - no crashing input found in 3 days for tar

Augmentation

- Augment the test script with crashing inputs by AFL
- Typically converges in up to 3 iterations in practice
- But, may be incomplete



```
/* grep-2.19 */
void add_tok (token t) {
    /* removed in the first trial and restored after augmentation */
    if (dfa->talloc == dfa->tindex)
        dfa->tokens = (token *) realloc (/* large size */);
    *(dfa->tokens + (dfa->tindex++)) = t;
}
```

Experimental Setup

- 10 widely used **UNIX utility programs** (13—90 KLOC)
    - each program has a **known CVE**
    - **remove unreachable code** by static analysis upfront
- Specification:
    - supporting **the same cmd line options** as BusyBox
    - with the **test suites** by the original developers

Size of Reduced Program

#Statement

Program	Original	Chisel	Hand-written
bzip-1.05	6,316	1,575	2,342
chown-8.2	3,422	186	141
date-8.21	4,100	913	107
grep-2.19	10,816	1,071	355
gzip-1.2.4	4,069	1,042	1,058
mkdir-5.2.1	1,746	142	94
rm-8.4	[illegible]	[illegible]	[illegible]
sort-8.16	[illegible]	[illegible]	[illegible]
tar-1.14	[illegible]	[illegible]	[illegible]
uniq-8.16	1,923	192	51
Total	55,848	6,111	4,729

Reachable code by static analysis

Chisel reduced 89%

Comparable to hand-written versions

Security Hardening

Remove 4 and 2 CVEs in undesired and desired functionalities.
4 CVEs are not easily fixable by reduction (e.g., race condition).

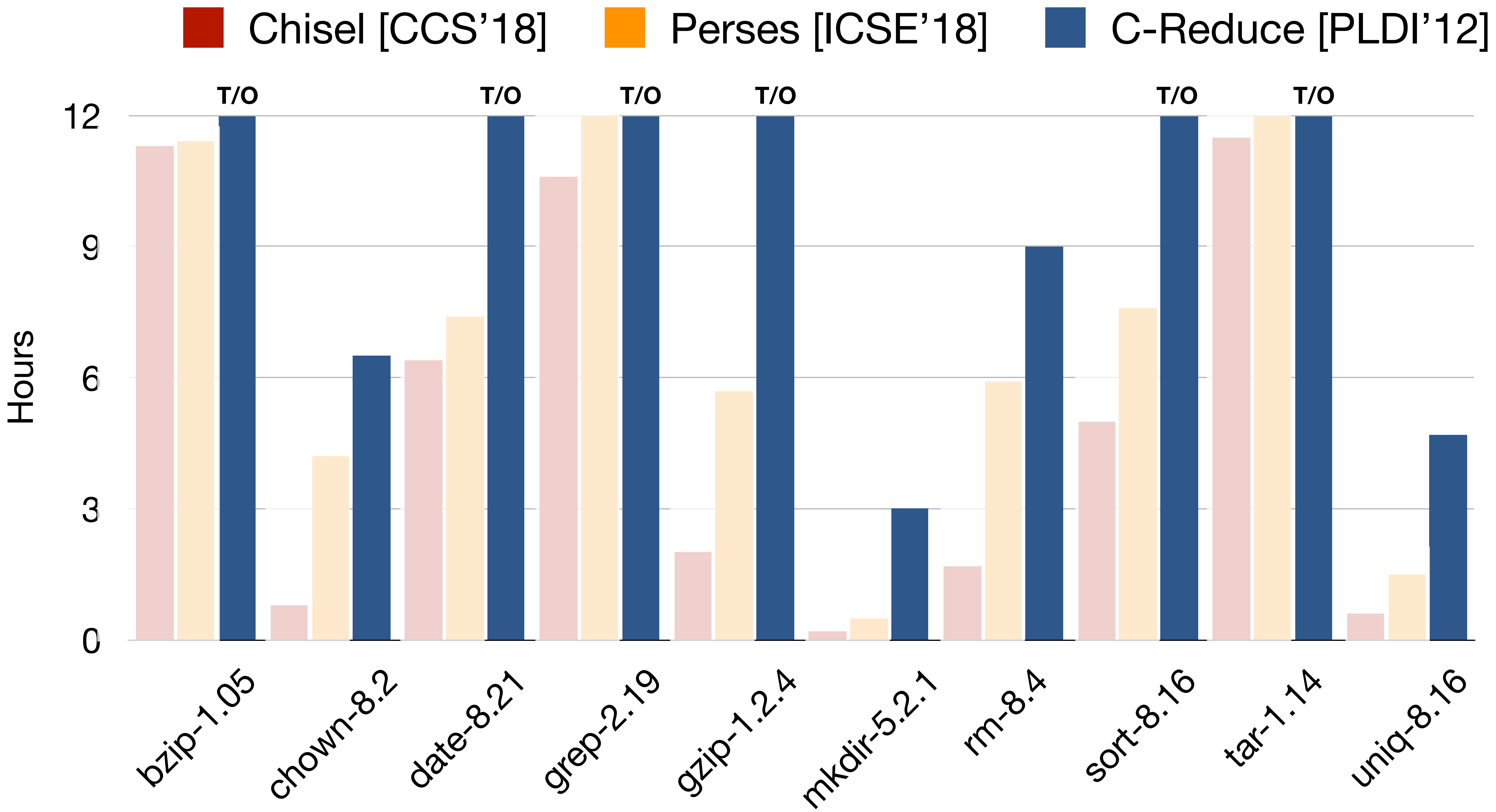
		#ROP Gadgets			#Alarms		
Program	CVE	Original	Reduced		Original	Reduced	
bzip-1.05	✗	662	298	(55%)	1,991	33	(98%)
chown-8.2	✓	534	162	(70%)	47	1	(98%)
date-8.21	✓	479	233	(51%)	201	23	(89%)
grep-2.19	✓	1,065	411	(61%)	619	31	(95%)
gzip-1.2.4	✓	456	340	(25%)	326	128	(61%)
mkdir-5.2.1	✗	229	124	(46%)	43	2	(95%)
rm-8.4	✗	565	95	(83%)	48	0	(100%)
sort-8.16	✓						
tar-1.14	✓						
uniq-8.16	✗	349	[illegible]	(69%)	60	1	(98%)
Total		6,752	2,285	(66%)	5,298	243	(95%)

Reduced potential attack surface

Make it feasible for manual alarm inspection

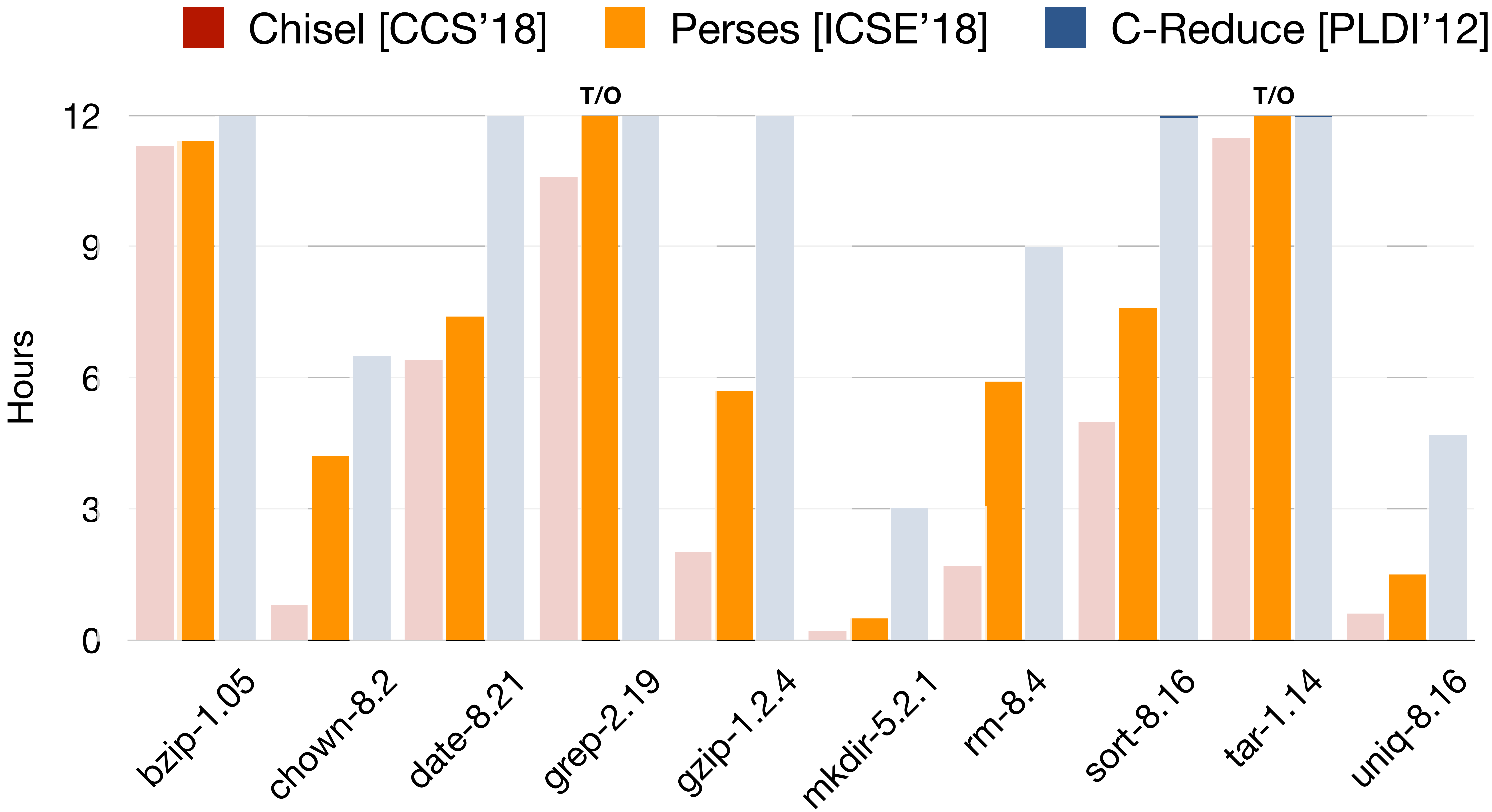
Reduction Time

Line-based reducer ran out of time for 6 programs



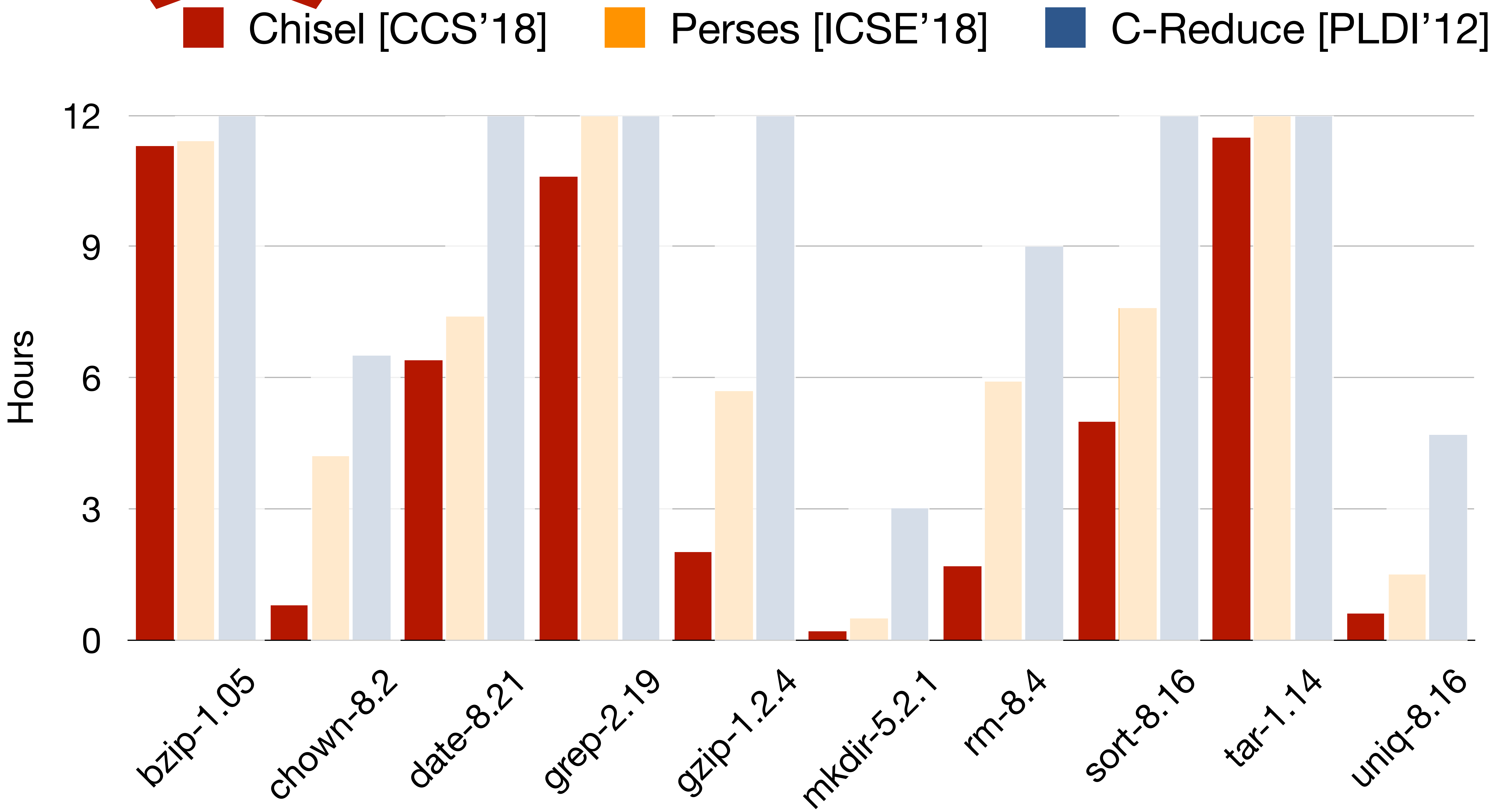
Reduction Time

Grammar-based reducer ran out of time for 2 programs



Reduction Time

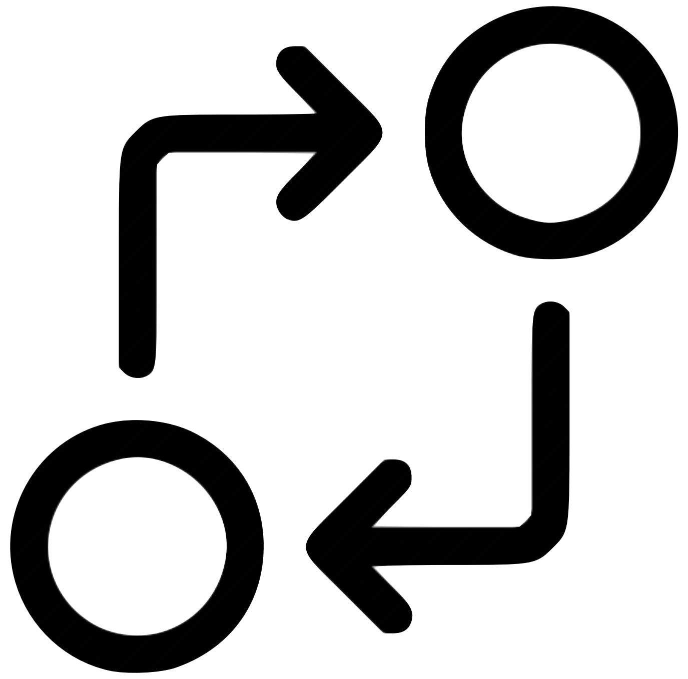
7x and 4x faster than
C-Reduce and Perses



Summary

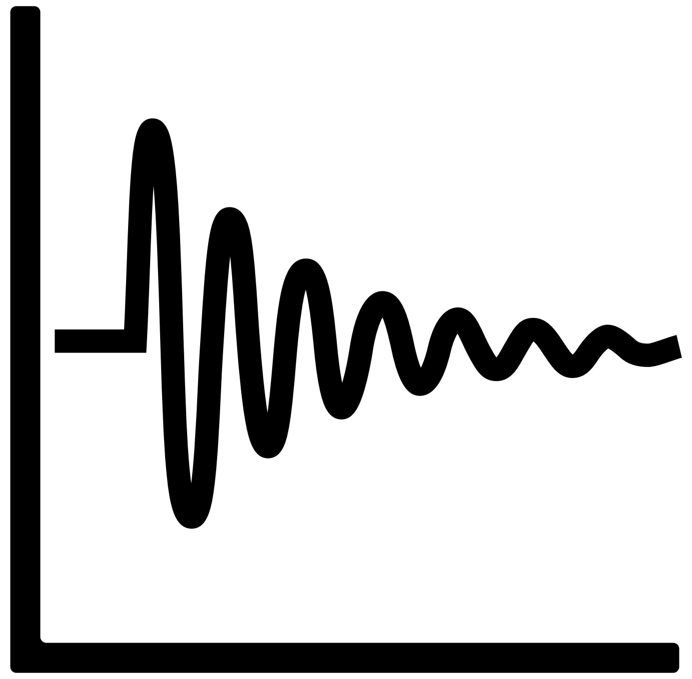
- Program debloating: **simplifying and hardening** large & complex SW
- Chisel: automated software debloating system
  - **tractable search** via learning-guided delta debugging
  - **security hardening** by removing undesired features
  - **robustness** via static & dynamic analyses
- Need a lot more research on efficiency and effectiveness
  - E.g., advanced learning techniques, system-level debloating (inter-program)

세 가지 아이디어



연관성 [CCS’18]

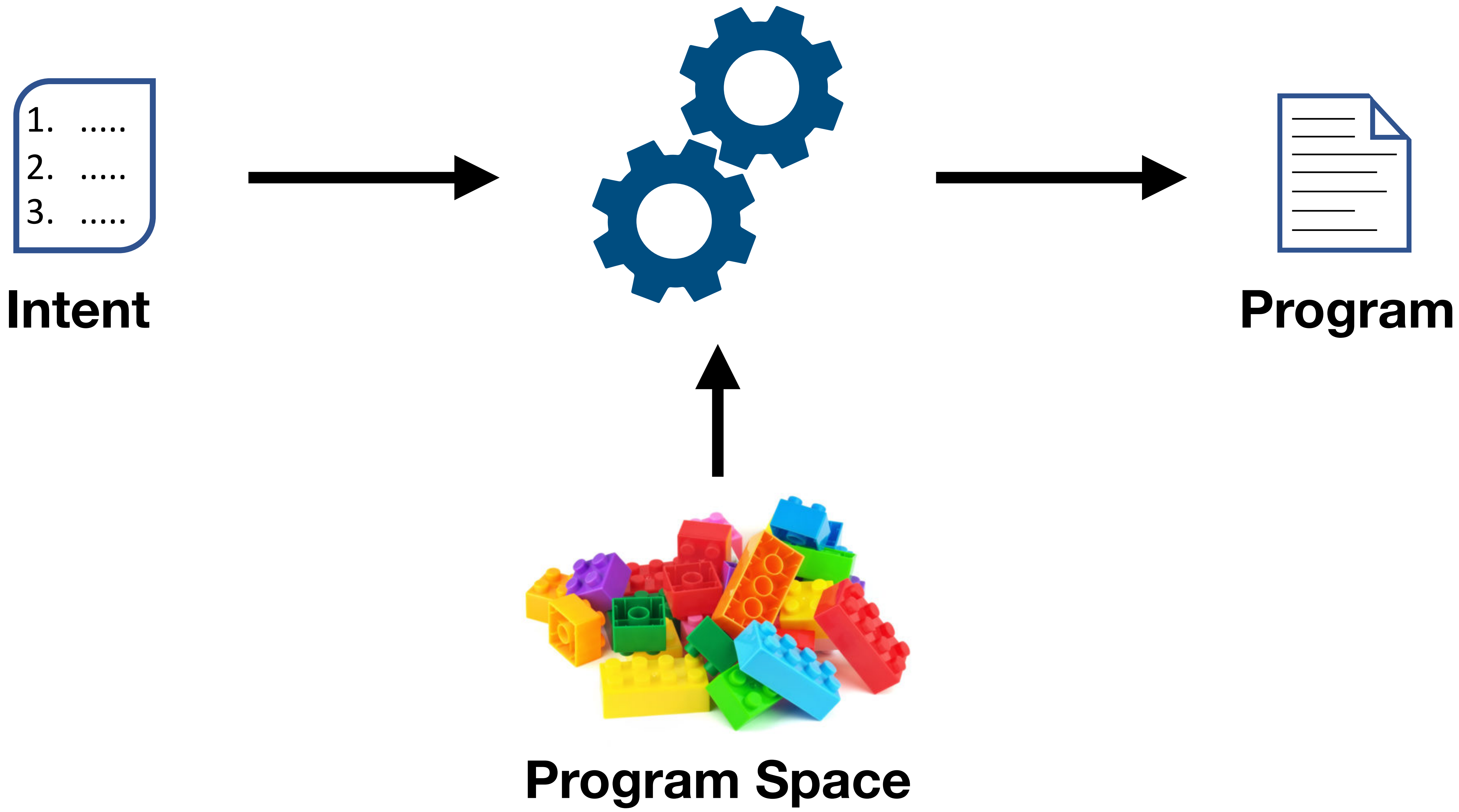
규칙성 [PLDI’18]



연속성 [IJCAI’19]

프로그램 합성

- 사용자에 **의도**에 맞는 프로그램을 주어진 **탐색 공간**에서 **자동**으로 찾기



예제

Specification

Find a function $f(x, y)$ where $f(3,1) = 3 \ \wedge \ f(1,2) = 3$

Grammar

$$S \rightarrow x \mid y \mid S + S \mid S - S \mid \texttt{if} \ B \ S \ S$$
$$B \rightarrow S \leq S \mid S = S$$

나열식 탐색 (Enumerative Search)

- 원하는 정답을 찾을 때까지 프로그램 후보를 하나하나 나열
- 단순 무식하지만 가장 일반적이어서 널리 쓰이는 방식
  - 여러가지 탐색 공간 최적화와 더불어서 쓰임
- 기존 나열 방식: 프로그램 크기 순 (작은 프로그램 먼저 탐색)
  - 왜? 오캄의 면도날

예제

Specification

Find a function $f(x, y)$ where $f(3,1) = 3 \ \wedge \ f(1,2) = 3$

Grammar

$$S \rightarrow x \mid y \mid S + S \mid S - S \mid \texttt{if}\ B\ S\ S$$
$$B \rightarrow S \leq S \mid S = S$$

Enumeration

iter 1	x	y		
iter 2	$x + y$	$x - y$	$x \leq y$	$x = y$
iter 3	$x + x + y$	$x + x - y$	…	if $(x \leq y)\ y\ x$
iter 4	$x + x + x + y$		…	if $(x \leq y)\ (y + x)\ x$
…				

단순한 나열식 탐색의 단점

- 단순 나열: 프로그램 크기 순
- 두 가지 문제
  - 속도: 가능한 프로그램 후보가 기하급수적으로 증가
  - 품질: 주어진 입출력 예제에만 우연히 들어맞는 프로그램 생성 가능성 (overfitting)
- 예: $f(-1,0) = 0 \ \wedge \ f(0,-1) = 0$

iter 0	x	y				
iter 1	$x + x$	$x - x$	$x + y$	$\dots$	$x \leq y$	$\dots$
iter 2	$x + x + y$	$x + x - y$	$\dots$	if $(x \leq y)$ y x		$\dots$
iter 3	$x + x + x + y$		$\dots$	if $(x \leq y)$ $(y + x)$ x		

But which one is more likely to be a solution?

$x - x$ **vs.** `if` $(x \leq y)$ y x

프로그램 구조의 통계적 규칙성

강조되고 반복되는 소리는 강아지를 불안하게 해요!

프로그램 구조의 통계적 규칙성

- 프로그램: **반복되고 예측가능한** 패턴이 자주 등장

```
for (i = 0; i < 100; ??)
```

- **통계적** 프로그램 모델: 프로그램 구조 상의 확률 분포
  - E.g., n-gram, probabilistic context-free gramma (PCFG), etc

```
Pr(?? → i++ | for (i = 0; i < 100; ??)) = 0.85
Pr(?? → i-- | for (i = 0; i < 100; ??)) = 0.01
```

- 응용: 코드 채우기, 역난독화, 자동 패치 생성

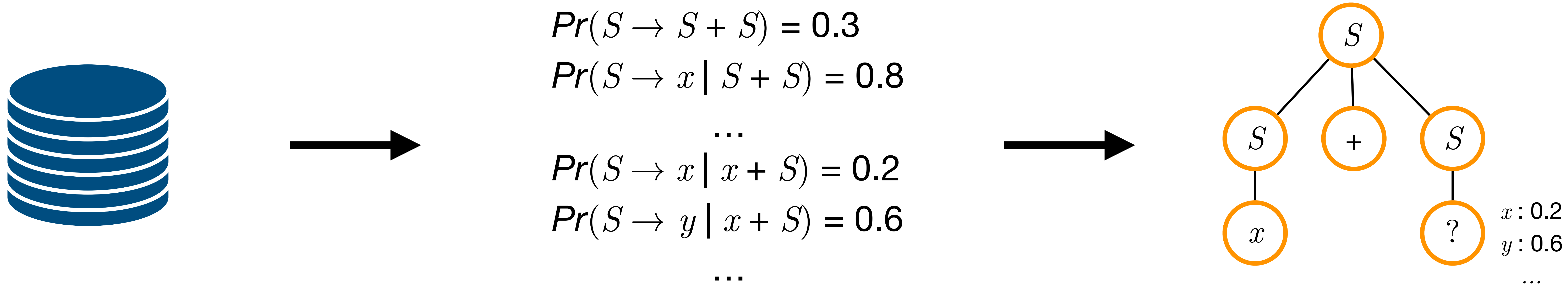
해결책: Euphony*

- 나열방식: 크기가 아닌 **개연성 (likelihood)**
- 확률 모델: 프로그램의 그럴싸함을 계산해주는 함수
  - “이 프로그램이 얼마나 그럴싸한가?”
- **가장 그럴싸한** 프로그램 (확률이 제일 높은) 먼저 시도

*Lee et al., Accelerating Search-Based Synthesis Using Learned Probabilistic Models, PLDI, 2018

확률 모델

- Learn a probabilistic model of programs from a corpus of programs
  - Human-written or auto-generated programs by other synthesizers
- A wide range of models is applicable



$Pr(S \to S + S) = 0.3$

$Pr(S \to x \mid S + S) = 0.8$

$\ldots$

$Pr(S \to x \mid x + S) = 0.2$

$Pr(S \to y \mid x + S) = 0.6$

$\ldots$

Program Corpus **Learned Probabilistic Model** **Probability of Programs**

프로그램의 개연성

- For a CFG $\langle N, \Sigma, R, S \rangle$,
- Given a context, provide the prob. of each production rule: Pr(rule | context)
  - Context: sentential form $\in (N \cup \Sigma)^*$
- Ultimately assign a probability to each program
- Example:

CFG $S \rightarrow x \mid 1 \mid S + S$

Probability of "x + 1"

$$S \rightarrow S + S \rightarrow x + S \rightarrow x + 1$$

$$\Pr(x + 1) = \boxed{\Pr(S \rightarrow S + S \mid S)} \times \boxed{\Pr(S \rightarrow x + S \mid S + S)} \times \boxed{\Pr(S \rightarrow x + 1 \mid x + S)}$$

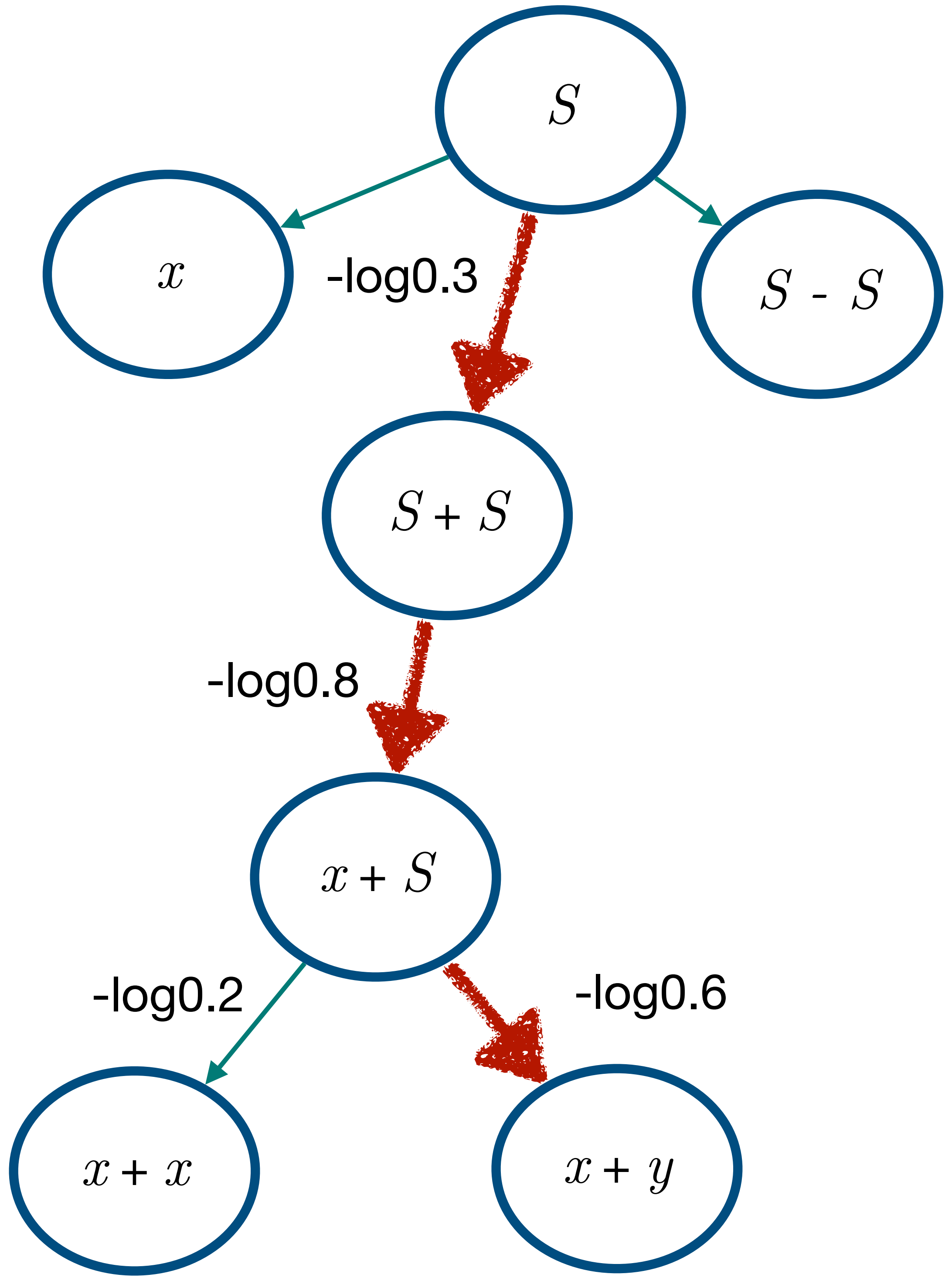
Example: PCFG

- Probabilistic Context Free Grammar (PCFG)
- One of the simplest form of probabilistic language model: ignore context
- Provide a probability to each production rule

$A \to \beta$	P
$S \to 0$	0.1
$S \to 1$	0.2
$S \to x$	0.3
$S \to S + S$	0.3
$S \to S - S$	0.1

Guided Enumeration by Probabilistic Model

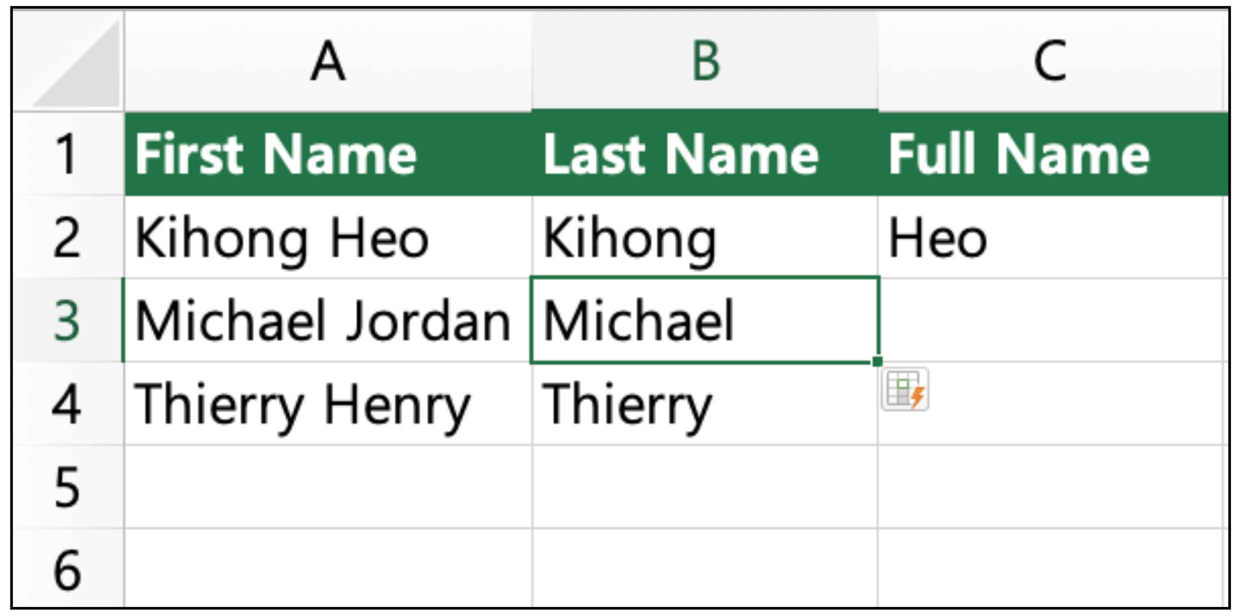
- Given a model, construct a directed graph
  - Node: sentential forms
  - Weight: negative log probability of a production rule
- Compute the shortest path
  - starting from the start symbol to the program
  - E.g., Dijkstra's, A*, etc



$Pr(S \rightarrow S + S) = 0.3$
$Pr(S \rightarrow x \mid S + S) = 0.8$
…
$Pr(S \rightarrow x \mid x + S) = 0.2$
$Pr(S \rightarrow y \mid x + S) = 0.6$
…

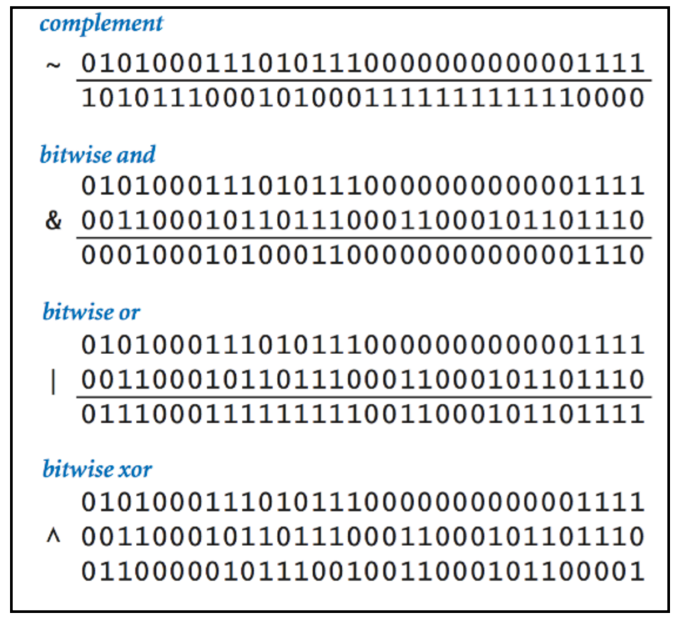
Experimental Setup

- 1167 tasks from 3 different domains



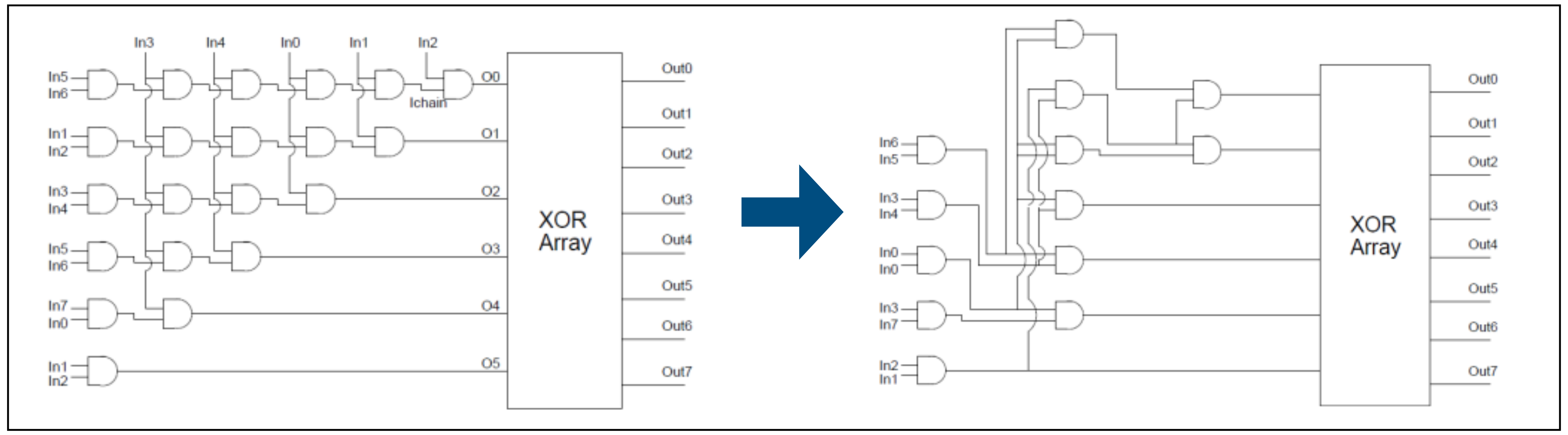
	A	B	C
1	First Name	Last Name	Full Name
2	Kihong Heo	Kihong	Heo
3	Michael Jordan	Michael	
4	Thierry Henry	Thierry	
5			
6			

STRING: End-user programming for string manipulations
(205 tasks)



```
complement
~ 01010001110101110000000000001111
  10101110001010001111111111110000

bitwise and
  01010001110101110000000000001111
& 00110001011011100011000101101110
  00010001010001100000000000001110

bitwise or
  01010001110101110000000000001111
| 00110001011011100011000101101110
  01110001111111110011000101101111

bitwise xor
  01010001110101110000000000001111
^ 00110001011011100011000101101110
  01100000101110010011000101100001
```

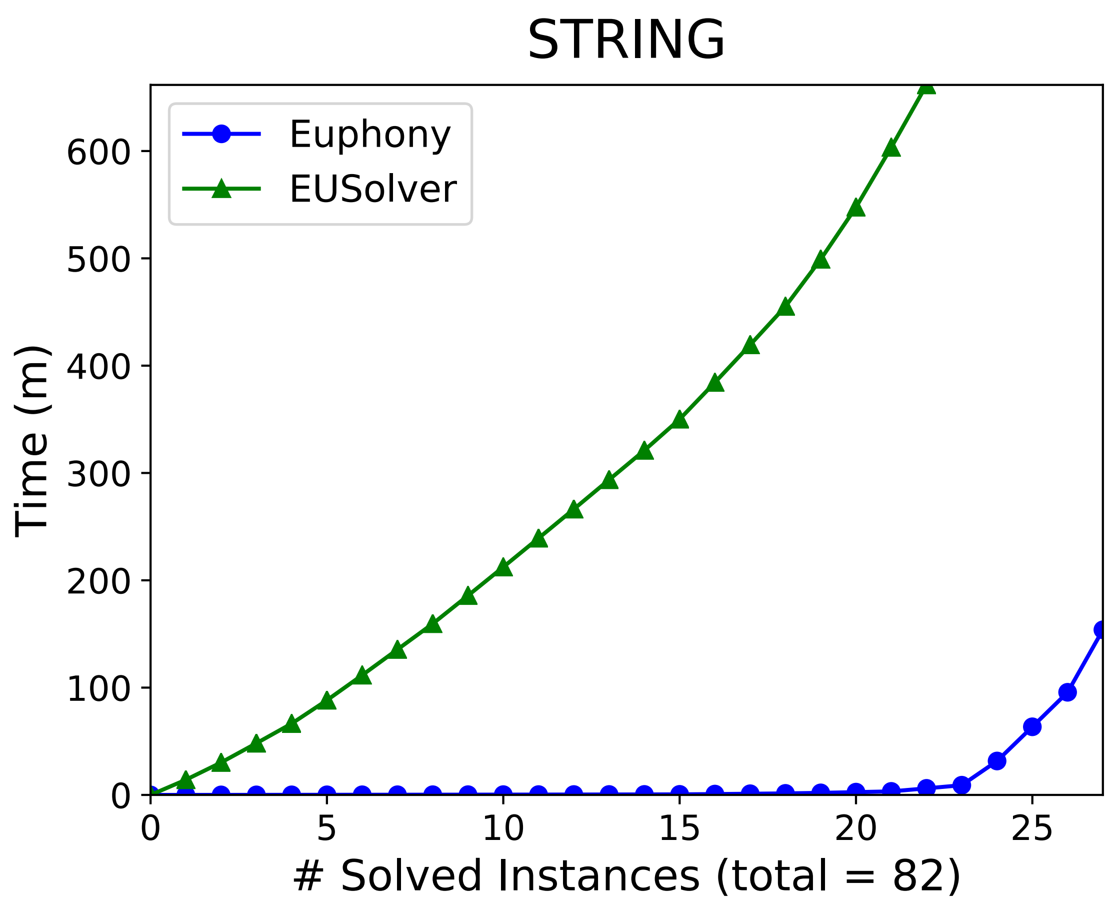
BITVEC: Efficient low-level algorithms
(750 tasks)

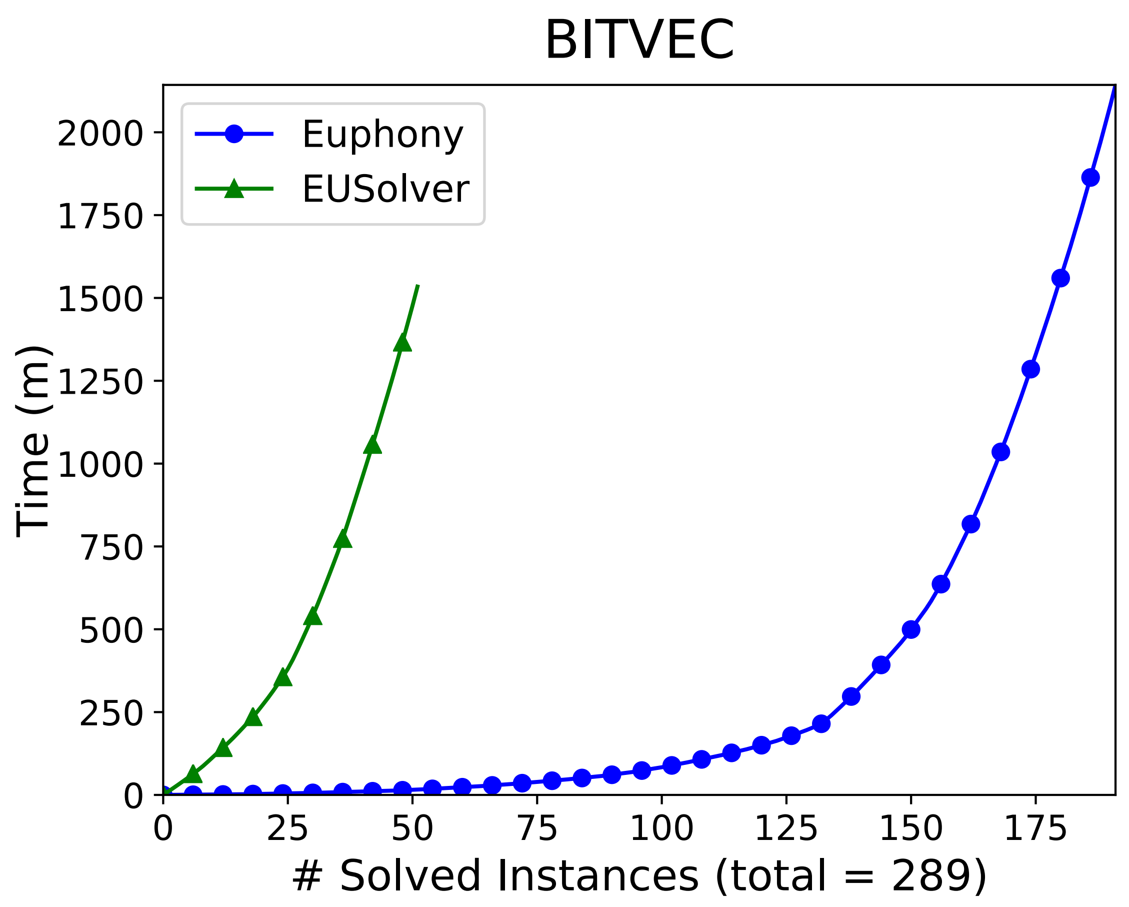


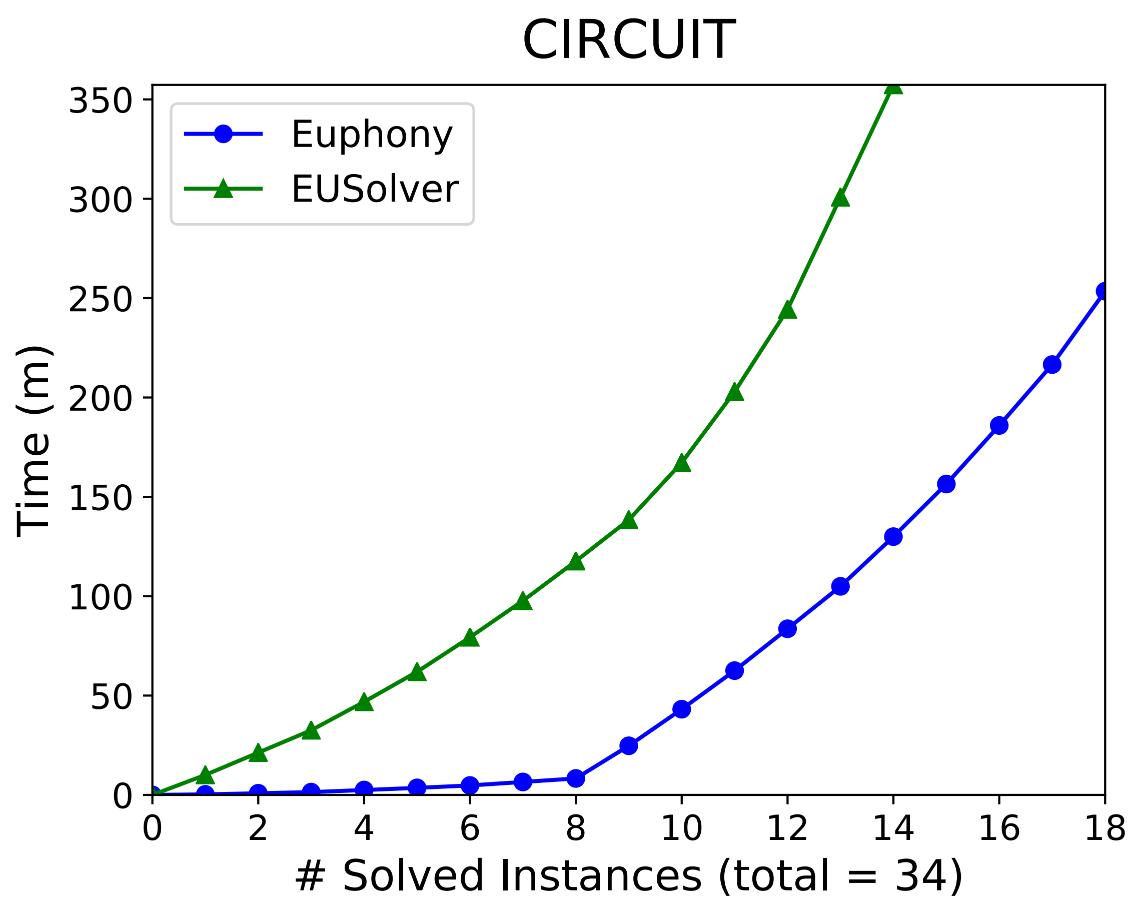
CIRCUIT: Attack-resistant crypto circuits generations
(212 tasks)

Effectiveness

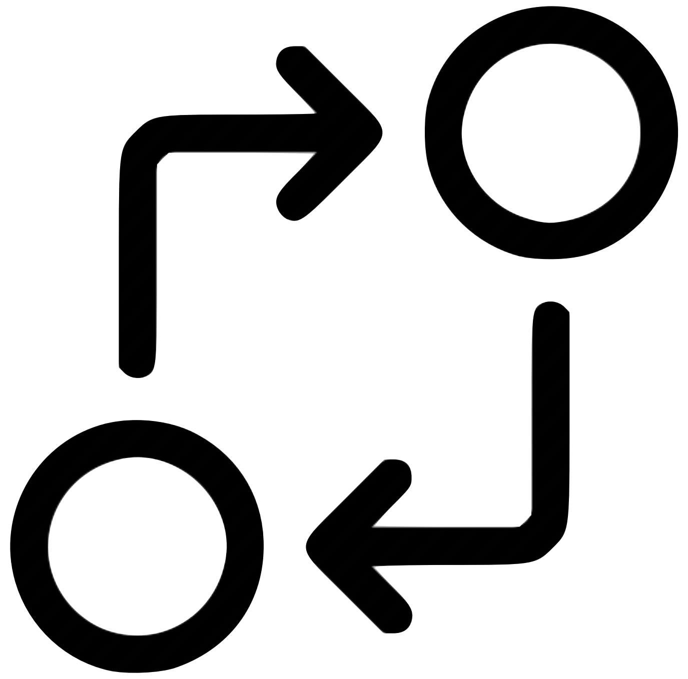
- Comparison to EUSolver (a program synthesizer without prob. guidance)
  - Training: 762 tasks solved by EUSolver in 10 minutes
  - Testing: 405 (timeout: 1 hour)





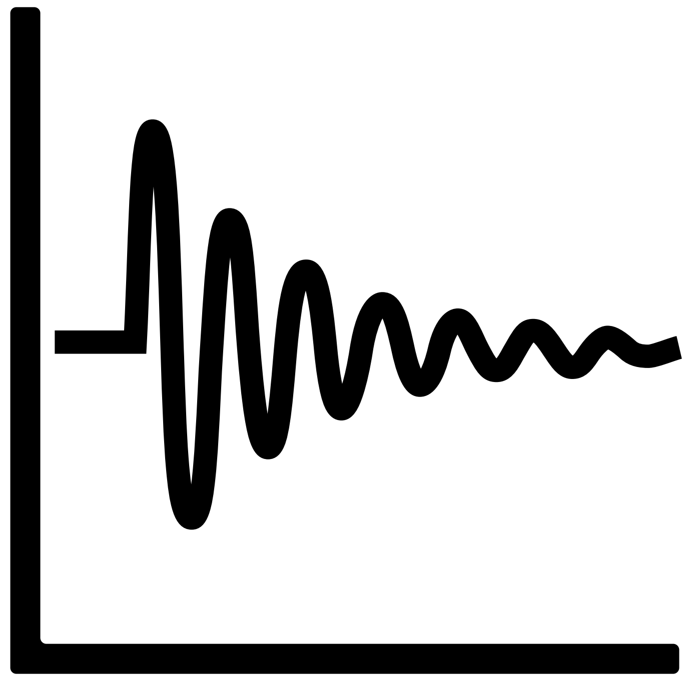


세 가지 아이디어



연관성 [CCS’18]

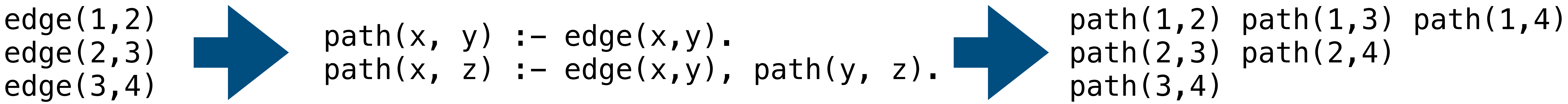
규칙성 [PLDI’18]



연속성 [IJCAI’19]

Datalog 프로그램

- Horn clause 형태로 표현된 1차 논리 규칙의 집합
  - 입력 튜플을 받아서 출력 튜플을 만들어 냄
- 응용: 빅데이터 분석, 네트워크 프로토콜, 프로그램 분석
- 예제: transitive closure



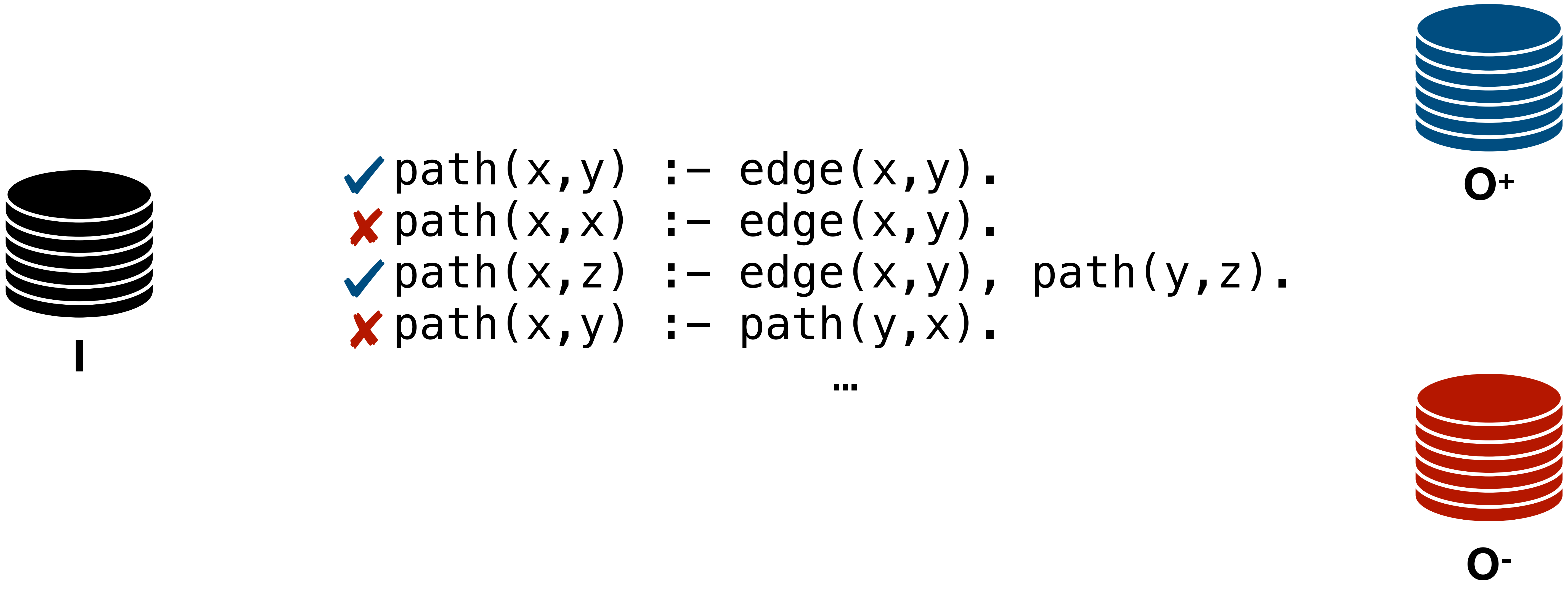
```
edge(1,2)
edge(2,3)
edge(3,4)
```

```
path(x, y) :- edge(x,y).
path(x, z) :- edge(x,y), path(y, z).
```

```
path(1,2) path(1,3) path(1,4)
path(2,3) path(2,4)
path(3,4)
```

Datalog 프로그램 합성

- 가능한 모든 규칙 중에서 주어진 입출력 예제와 아귀가 맞는 부분 집합 선택 문제
    - 전형인 조합 최적화 문제 (combinatorial optimization), 즉 NP-hard



```
✓path(x,y) :- edge(x,y).
✗path(x,x) :- edge(x,y).
✓path(x,z) :- edge(x,y), path(y,z).
✗path(x,y) :- path(y,x).
                    …
```

연속적 의미로 변환

- Datalog 프로그램 (불연속 함수) 를 연속 함수로 해석
    - 튜플의 존재 여부 {0, 1} → 튜플의 가중치 [0, 1]
- 그러면, 조합 최적화 문제 → 수치 최적화 문제

Parameters: $\vec{W}$

```
0.7  path(x,y) :- edge(x,y).
0.9  path(x,x) :- edge(x,y).
0.1  path(x,z) :- edge(x,y), path(y,z).
0.3  path(x,y) :- path(y,x).
```

최적화문제: 다음 식을 최소화하는 $\vec{W}$ **를 찾으시오**

$$loss = \Sigma_{t \in pos}(1 - v_t)^2 + \Sigma_{t \in neg}(0 - v_t)^2$$

Input	edge(1,2)	edge(2,3)
Weight	1.0	1.0

Output	path(1,2)	path(2,3)	path(1,3)	path(1,1)	path(2,1)
Weight (v_t)	0.7	0.7	0.63	0.1	0.21
Expectation	1	1	1	0	0

연속적 실행 의미

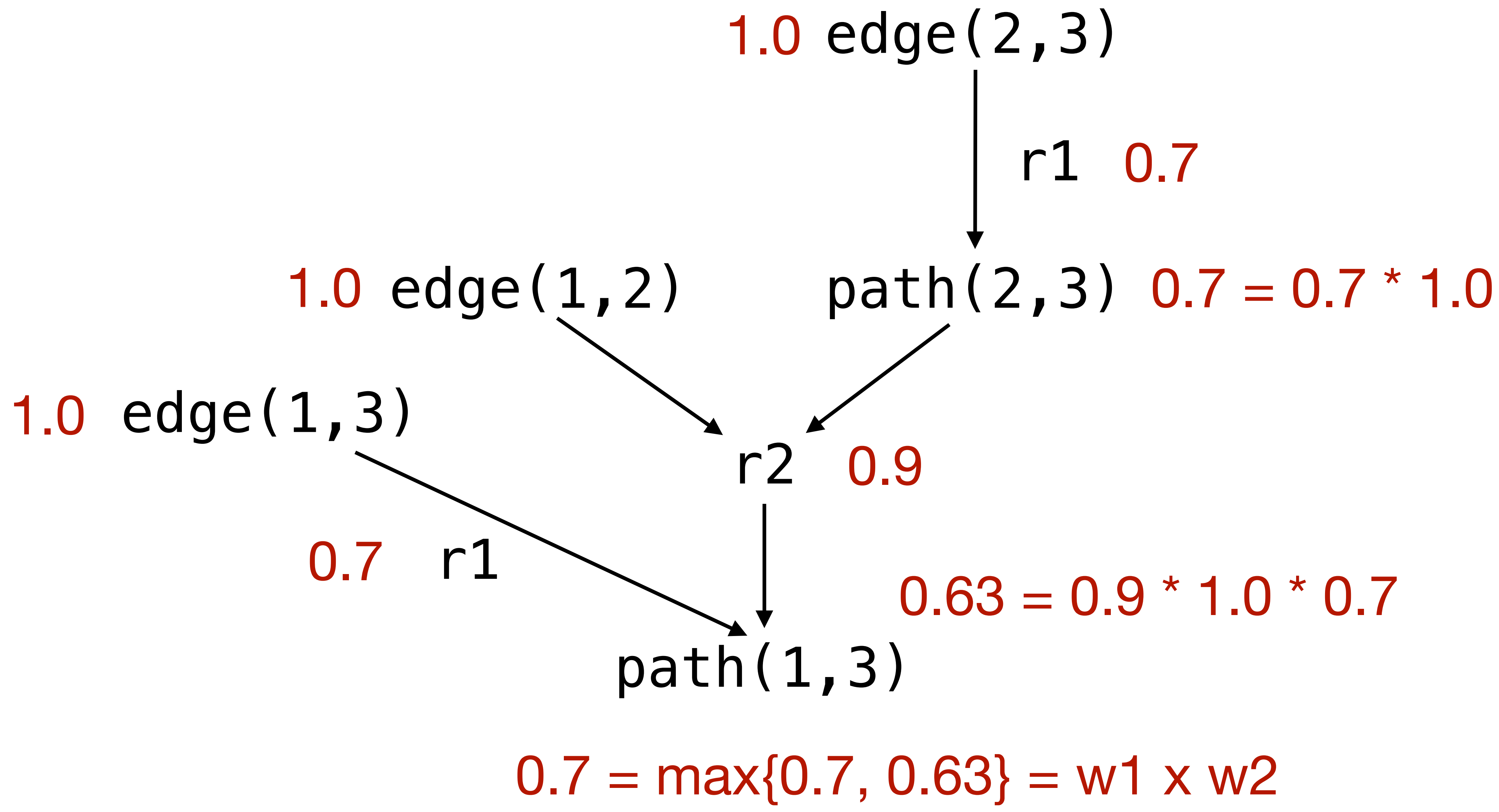
```
0.7 r1:  path(x,y) :- edge(x,y).
0.9 r2:  path(x,z) :- edge(x,y), path(y,z).
```

원래 실행 의미 (boolean semiring)

$$v_t = \bigvee_g (v_{a_1} \wedge v_{a_2} \wedge \cdots \wedge v_{a_k})$$

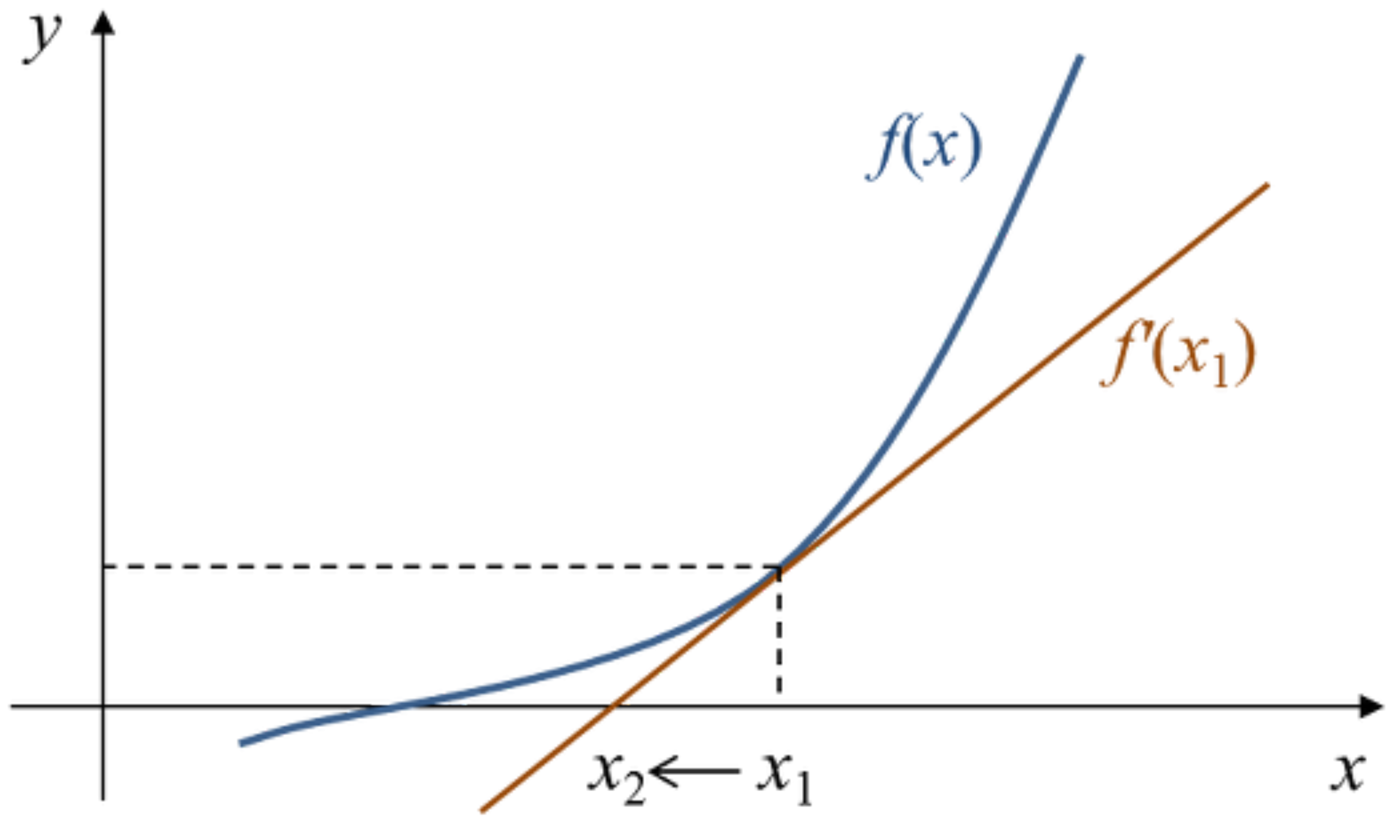
연속적 실행 의미 (Viterbi semiring)

$$v_t = \max_g (w_g \times v_{a_1} \times v_{a_2} \times \cdots \times v_{a_k})$$



최적화

- 연속 의미로 실행하고 나면, 각 v_t 는 w_r 에 대한 다항식 (미분가능)
- 따라서, 다음 손실 함수도 w_r 에 대한 다항식 (미분가능): $loss = \Sigma_{t \in pos}(1 - v_t)^2 + \Sigma_{t \in neg}(0 - v_t)^2$
- 잘 알려진 연속 최적화 알고리즘을 이용 (예: Newton's method)
  - 손실이 0 인 지점 = 원하는 프로그램 = 모든 입출력이 의도에 부합

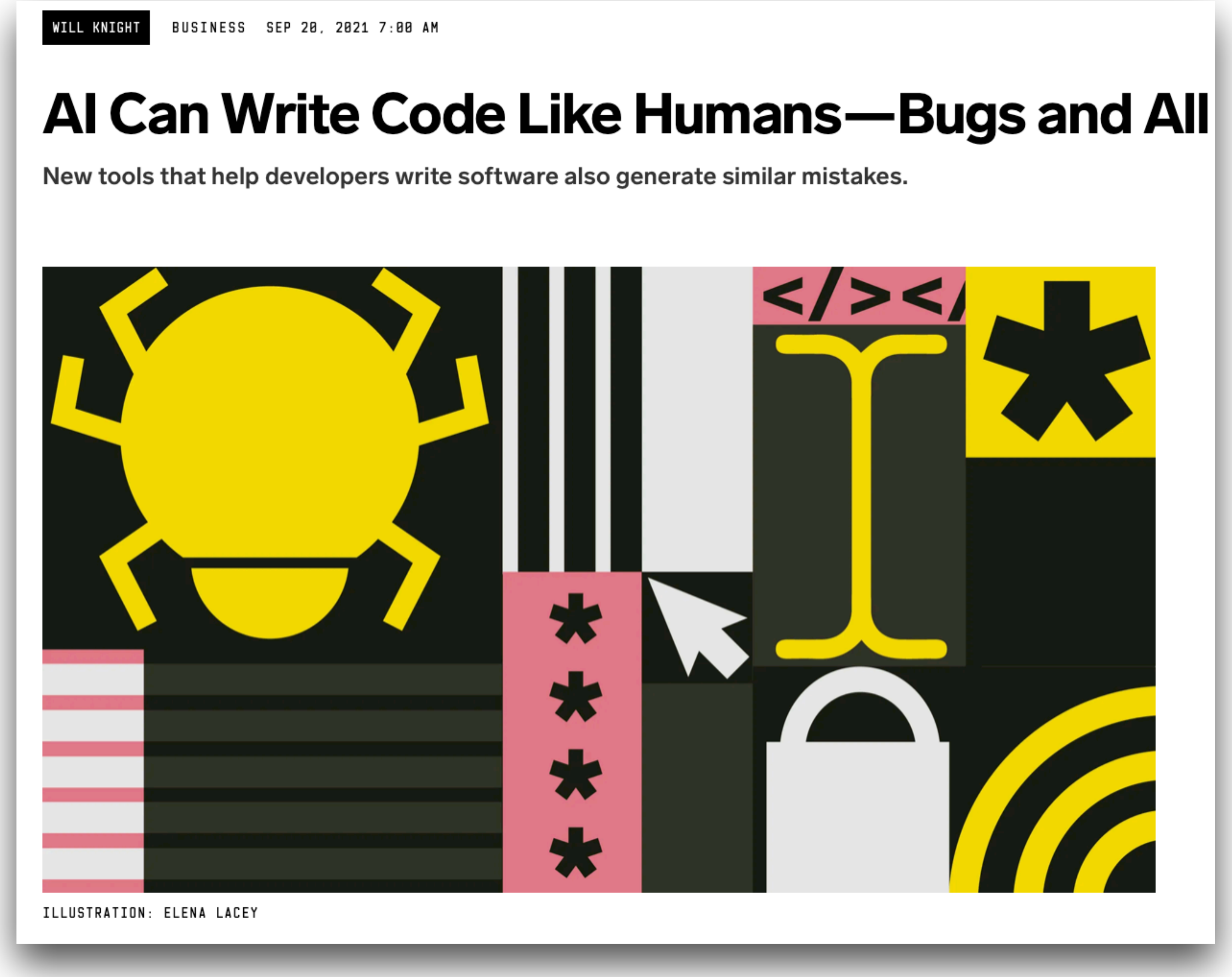


성능

Benchmark	#relations	#candidate rules	Difflog	ALPS
polysite	6	552	27	84
downcast	9	1267	30	1646
rv-check	5	335	22	195
andersen	5	175	4	27
1-call-site	9	173	4	106
2-call-site	9	122	53	676
1-object	11	46	3	345
1-type	12	70	4	13
escape	10	140	1	5
modref	13	129	1	2836

마무리

- AI/ML 기반 프로그래밍은 득인가 실인가?
  - 성배? 좀 더 편리한 Stackoverflow? 오류 자동 생성기?
- 인간적인 AI 프로그래머?
  - 기계가 잘하던 일: 계산, 논리, 기억
  - 인간이 잘하던 일: 직관, 통찰, 영감
  - 프로그래밍은?
- 대상 영역에 따라? (예: 안전필수 SW)



WILL KNIGHT BUSINESS SEP 20, 2021 7:00 AM

AI Can Write Code Like Humans—Bugs and All

New tools that help developers write software also generate similar mistakes.

ILLUSTRATION: ELENA LACEY