Effective Program Debloating via Reinforcement Learning

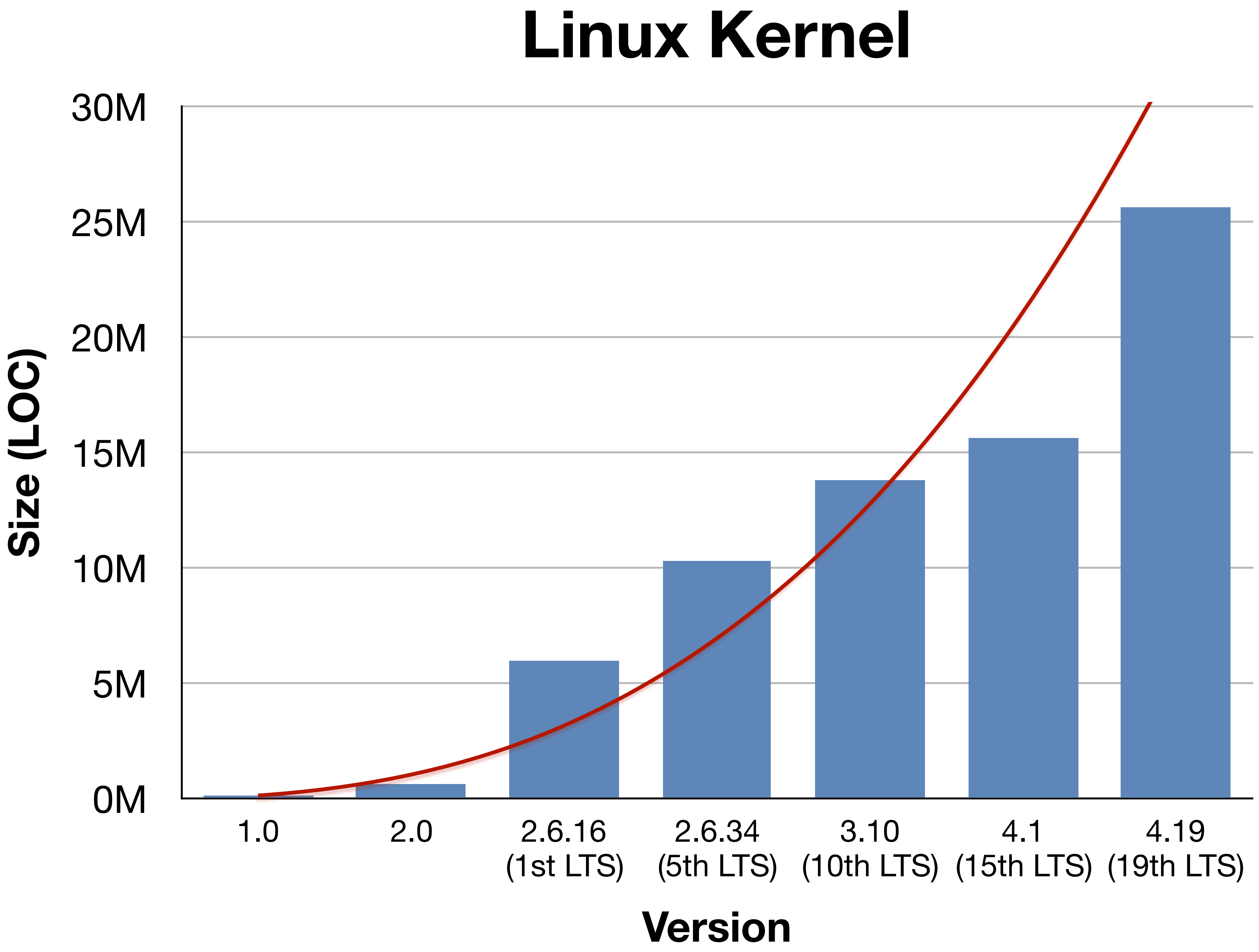
Kihong Heo[1]
Pardis Pashakhanloo[1]

Woosuk Lee[1,2]
Mayur Naik[1]

University of Pennsylvania[1]
Hanyang University[2]

CCS 2018

Growth of SW Complexity



*https://www.linuxcounter.net

Consequences of SW Bloat

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

Consequences of SW Bloat

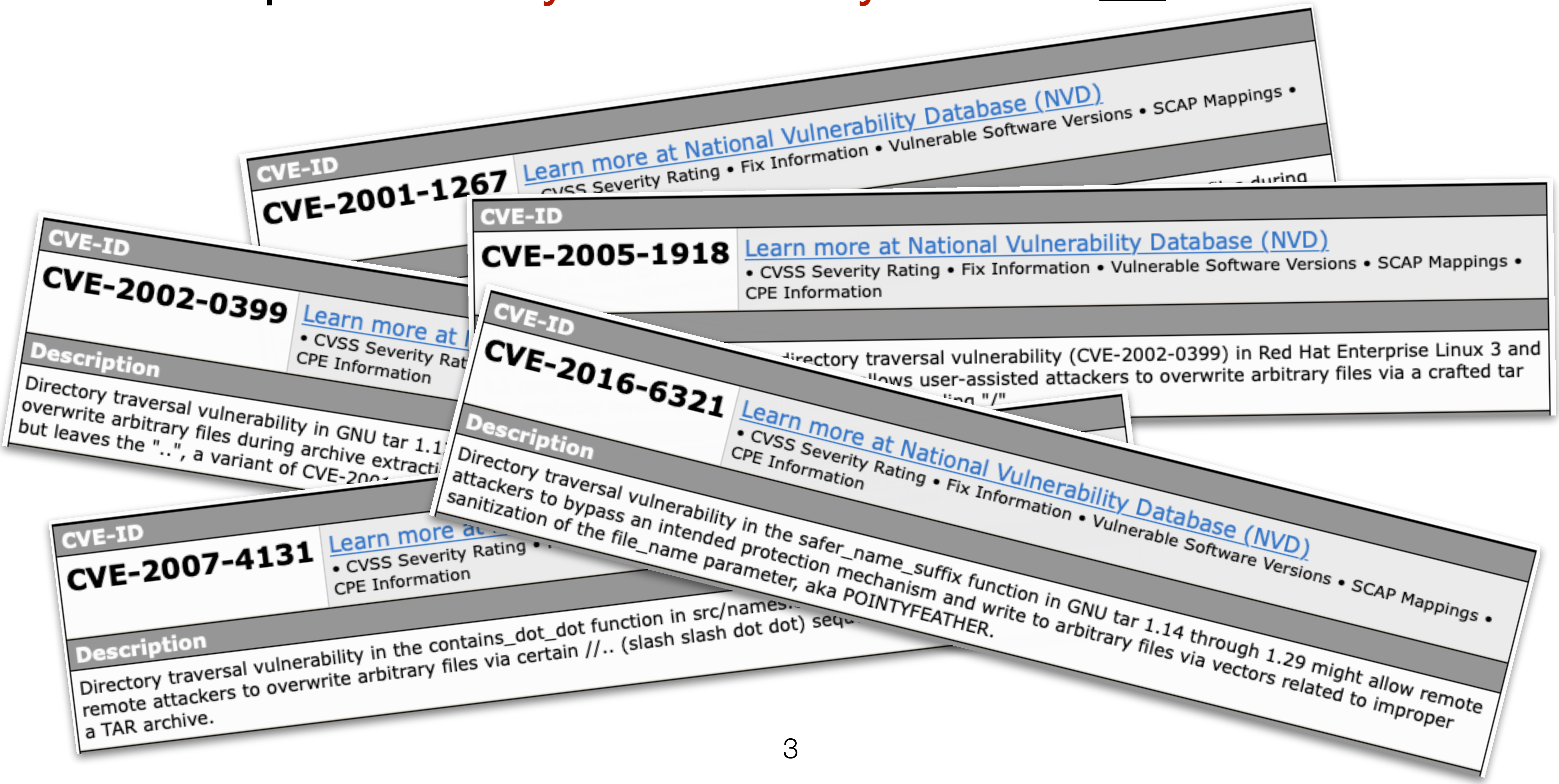
Performance	Maintainability	Security

- Example: security vulnerability in GNU tar

Consequences of SW Bloat

Performance	Maintainability	Security

- Example: security vulnerability in GNU tar



CVE-ID

CVE-2001-1267

Learn more at National Vulnerability Database (NVD)

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

iles during

CVE-ID

CVE-2005-1918

Learn more at National Vulnerability Database (NVD)

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

irectory traversal vulnerability (CVE-2002-0399) in Red Hat Enterprise Linux 3 and llows user-assisted attackers to overwrite arbitrary files via a crafted tar ing "/"

CVE-ID

CVE-2002-0399

Learn more at N

• CVSS Severity Rat

CPE Information

Description

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

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.

CVE-ID

CVE-2016-6321

Learn more at National Vulnerability Database (NVD)

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

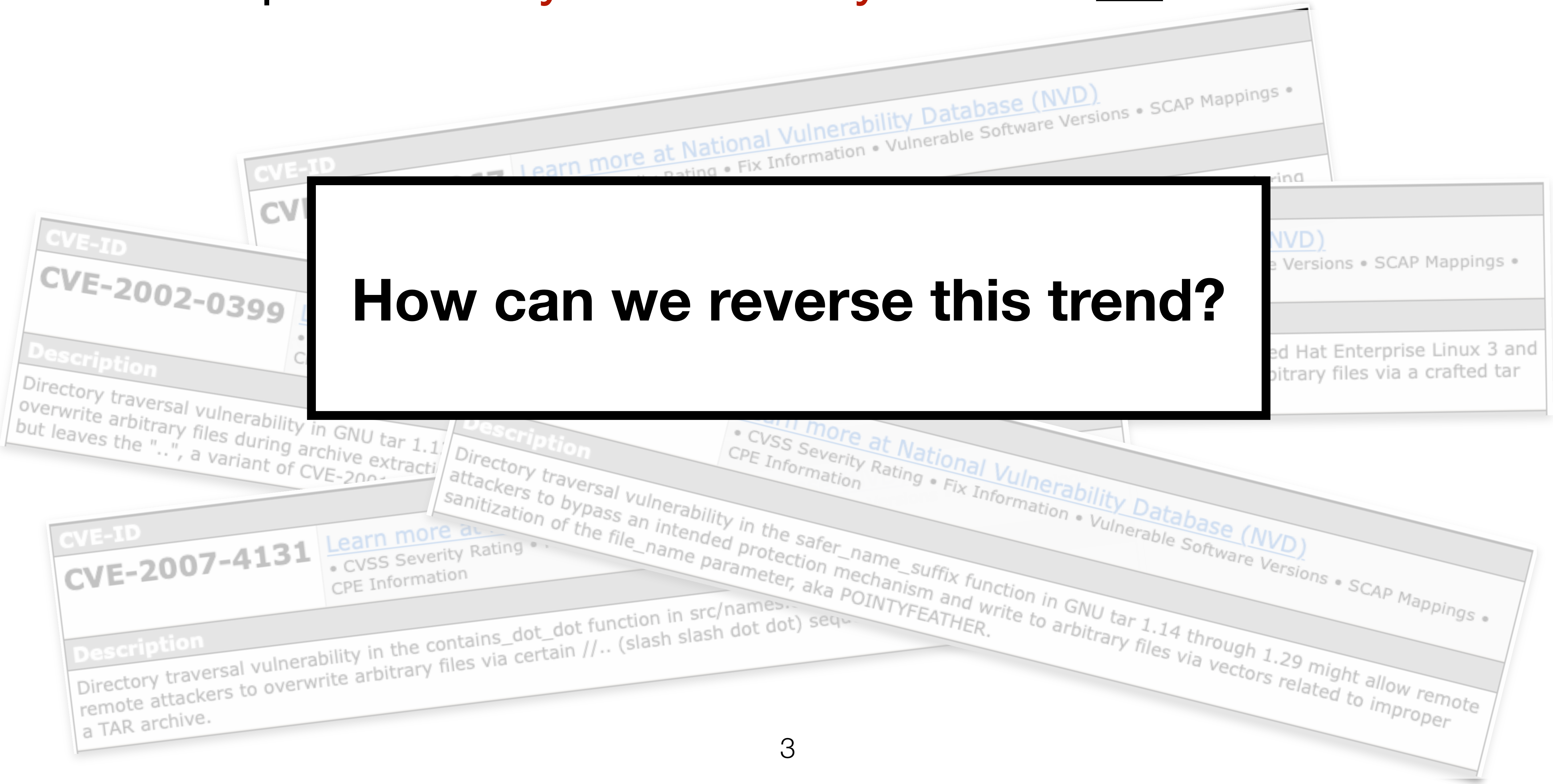
Description

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.

Consequences of SW Bloat

Performance **Maintainability** **Security**

- Example: security vulnerability in GNU tar



CVE-ID

CV

Learn more at National Vulnerability Database (NVD)

• CVSS Severity 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)

e 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.

Learn more at National Vulnerability Database (NVD)

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

CPE Information

Description

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?

State-of-the-Practice

General-purpose tar

- Out-of-the-box Linux

Customized tar

- BusyBox Utility Package*

*https://busybox.net

State-of-the-Practice

General-purpose tar

- Out-of-the-box Linux
- 97 cmd line options

Customized tar

- BusyBox Utility Package*
- 8 cmd line options

*https://busybox.net

State-of-the-Practice

General-purpose tar

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

Customized tar

- BusyBox Utility Package*
- 8 cmd line options
- 3,287 LOC
- 403 statements

*https://busybox.net

State-of-the-Practice

General-purpose tar

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

Customized tar

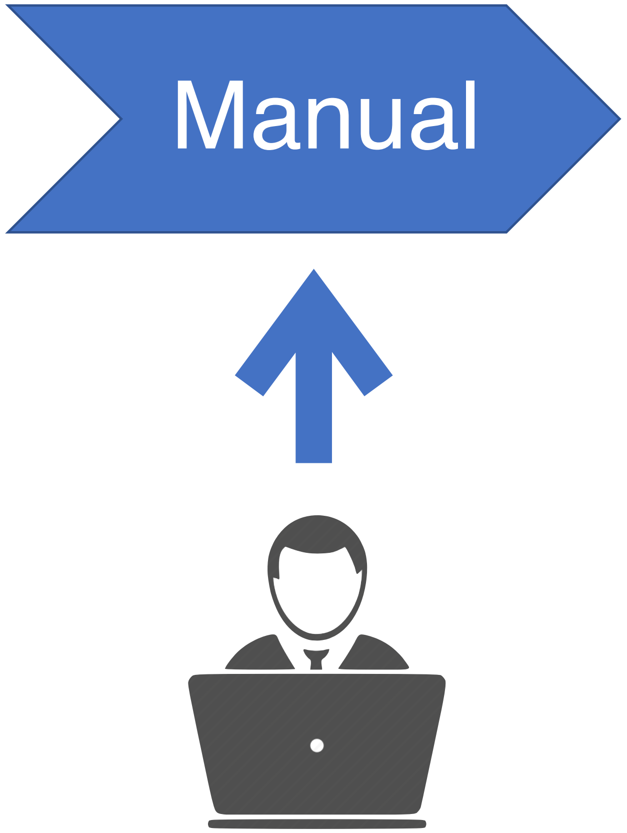
- BusyBox Utility Package*
- 8 cmd line options
- 3,287 LOC
- 403 statements
- No known CVEs

*https://busybox.net

State-of-the-Practice

General-purpose tar

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



Customized tar

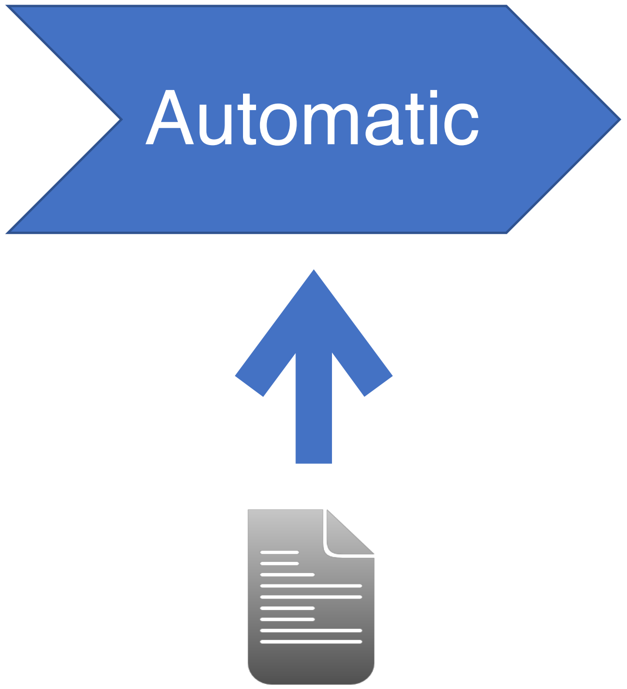
- BusyBox Utility Package*
- 8 cmd line options
- 3,287 LOC
- 403 statements
- No known CVEs

*https://busybox.net

Our Goal

General-purpose tar

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



Automatic

High-level Spec

Customized tar

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

*https://busybox.net

Our Contribution

Chisel: an automated program debloating system

Our Contribution

Chisel: an automated program debloating system

- **minimality**: trim code as aggressively as possible

Our Contribution

Chisel: an automated program debloating system

- **minimality**: trim code as aggressively as possible
- **efficiency**: scale to large programs

Our Contribution

Chisel: an automated program debloating system

- **minimality**: trim code as aggressively as possible
- **efficiency**: scale to large programs
- **robustness**: avoid introducing new vulnerabilities

Our Contribution

Chisel: an automated program debloating system

- **minimality**: trim code as aggressively as possible
- **efficiency**: scale to large programs
- **robustness**: avoid introducing new vulnerabilities
- **naturalness**: produce maintainable code

Our Contribution

Chisel: an automated program debloating system

- **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

Example: tar-1.14

```
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() { ... }
```

```
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;
}
```

Code containing CVE removed

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

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

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;
        }
    }
    ...
}
```

Unnecessary functionalities removed

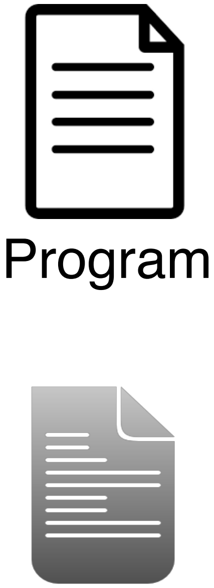
```
/* Supports all options: -x, -t, -P, -i, ... */
int main(int argc, char **argv) {
    int optchar;
    while (optchar = getopt_long(argc, argv) != -1) {
        switch(optchar) {
        case 'x': read_and(&extract_archive); break;
        case 't': read_and(&list_archive); break;
        case 'P':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

Unsupported options removed

Talk Outline

- Motivation
- **System Architecture**
- Evaluation
- Conclusion

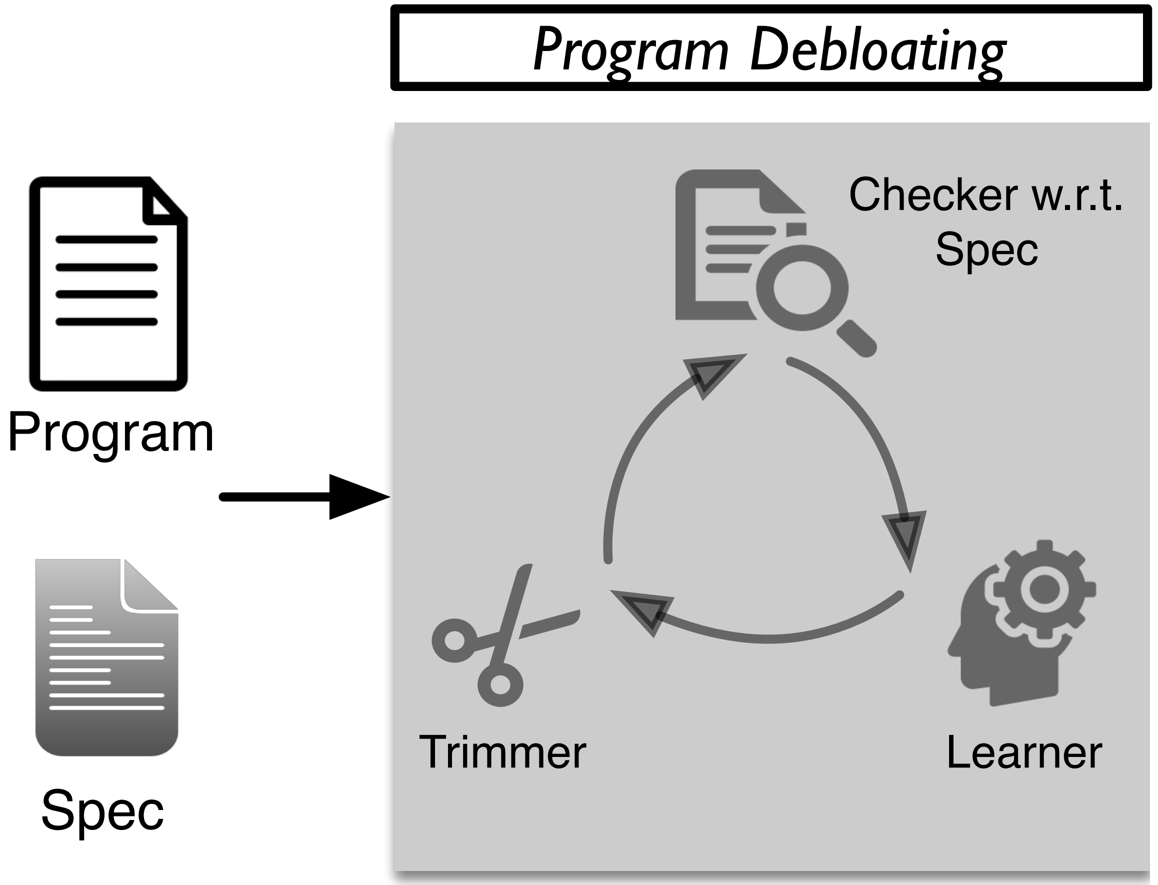
System Architecture



Program

Spec

System Architecture



Program

Spec

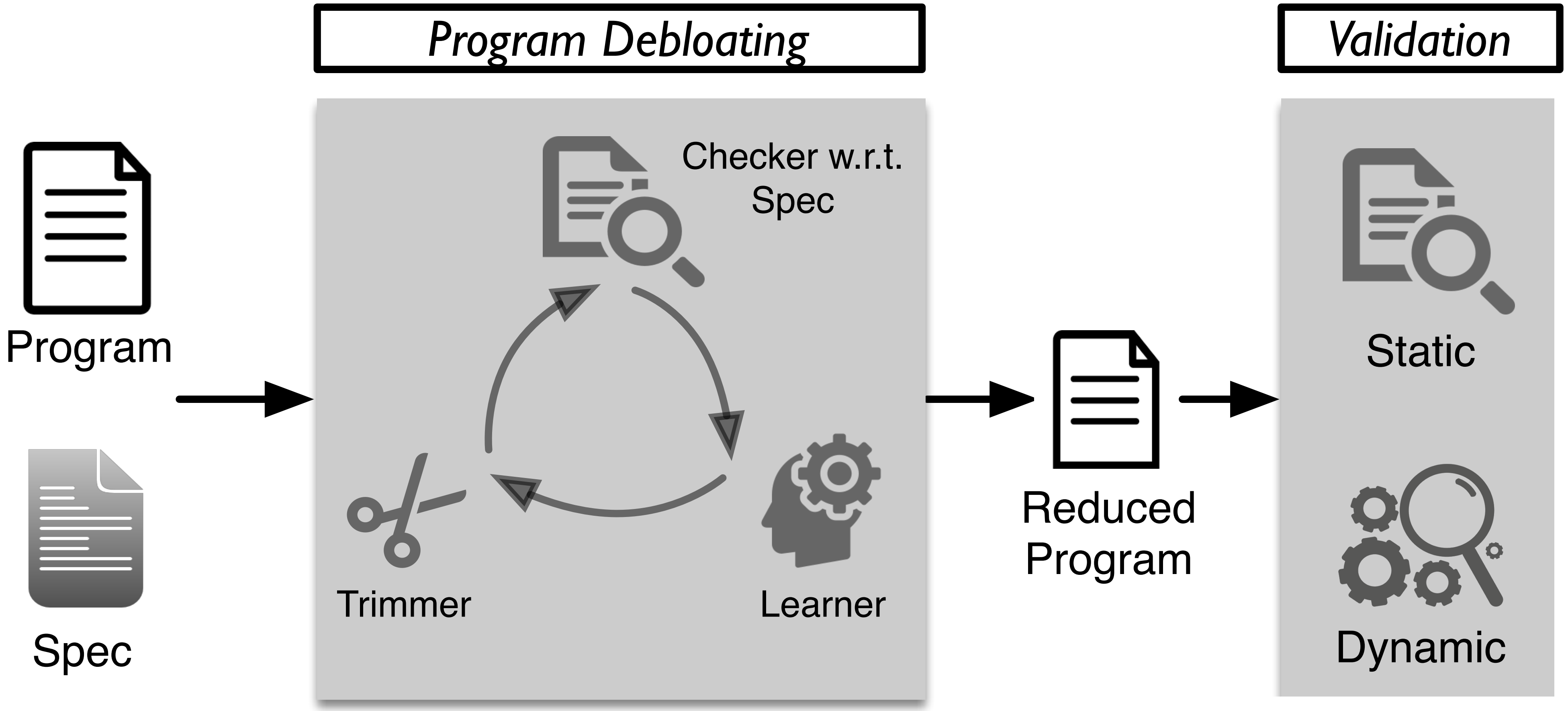
Program Debloating

Checker w.r.t. Spec

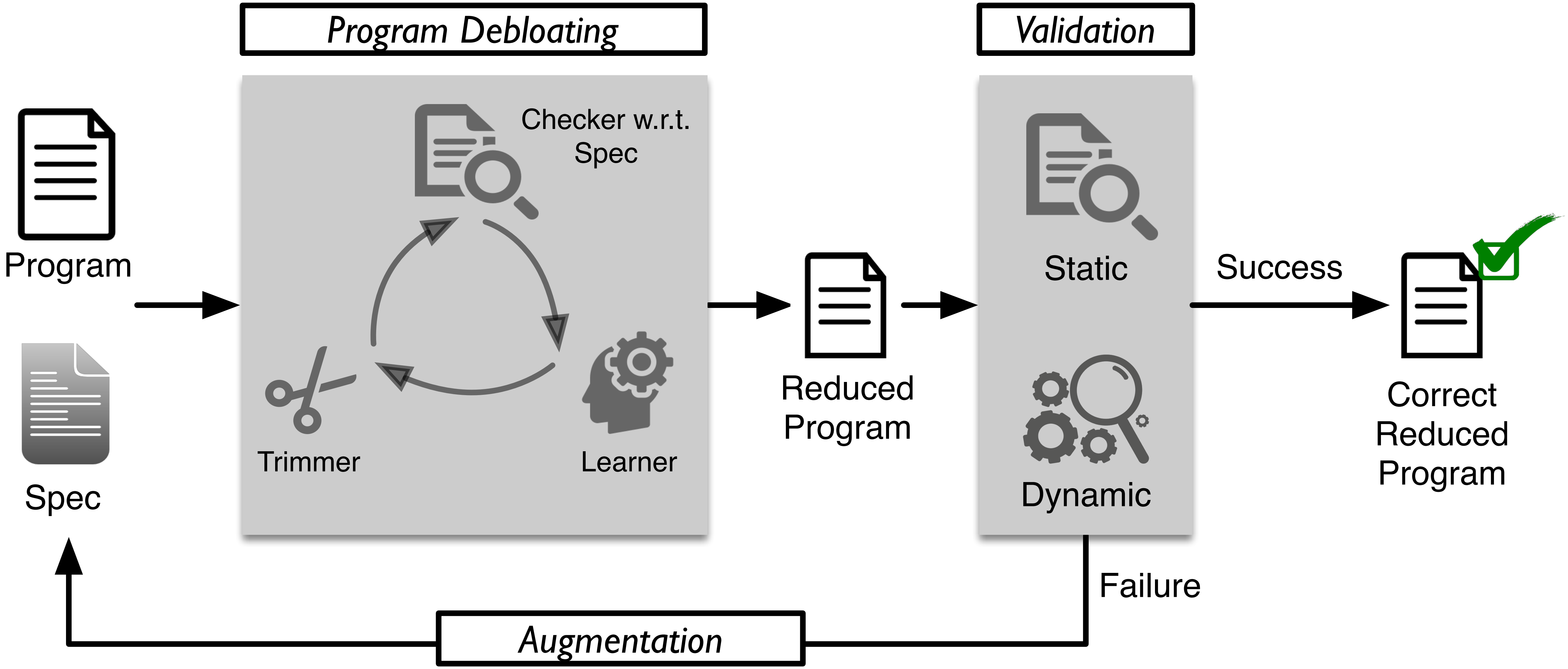
Trimmer

Learner

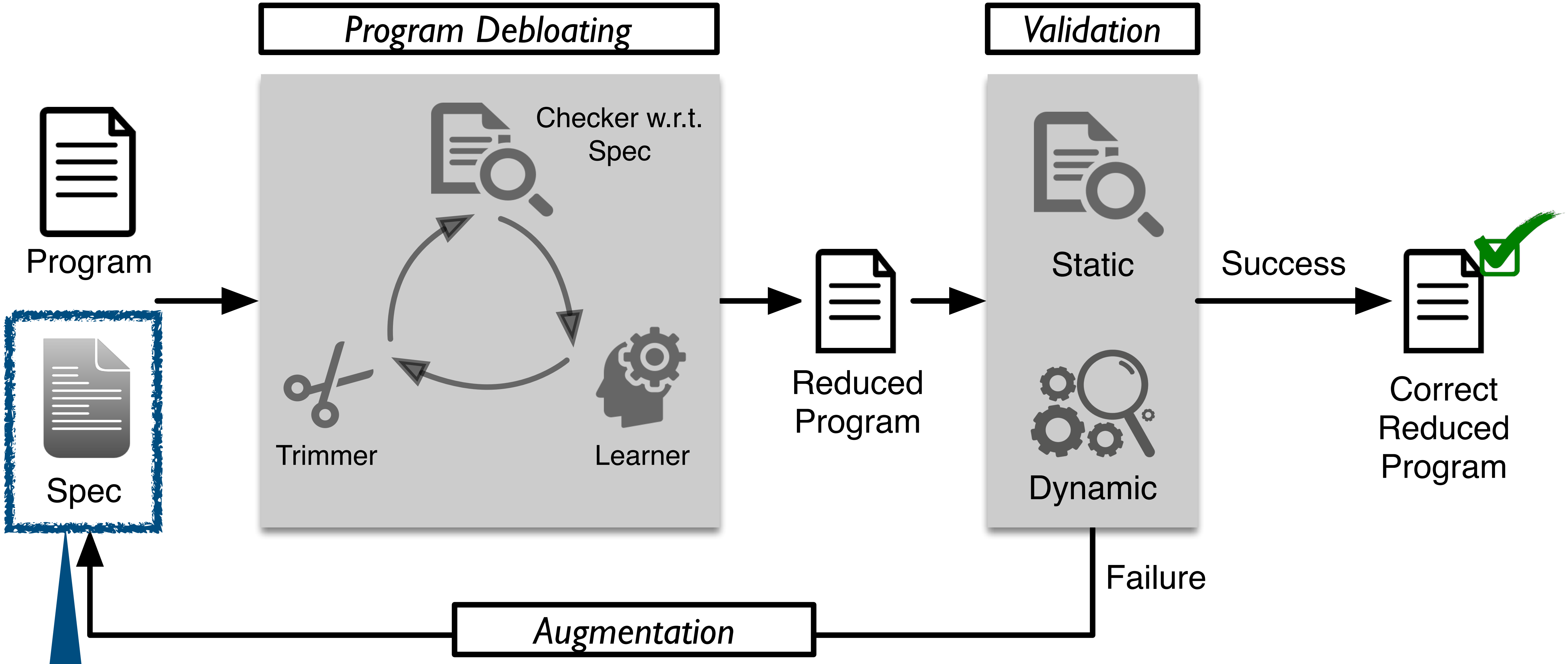
System Architecture



System Architecture



Key Questions



Program

Spec

Program Debloating

Checker w.r.t. Spec

Trimmer

Learner

Reduced Program

Validation

Static

Dynamic

Success

Correct Reduced Program

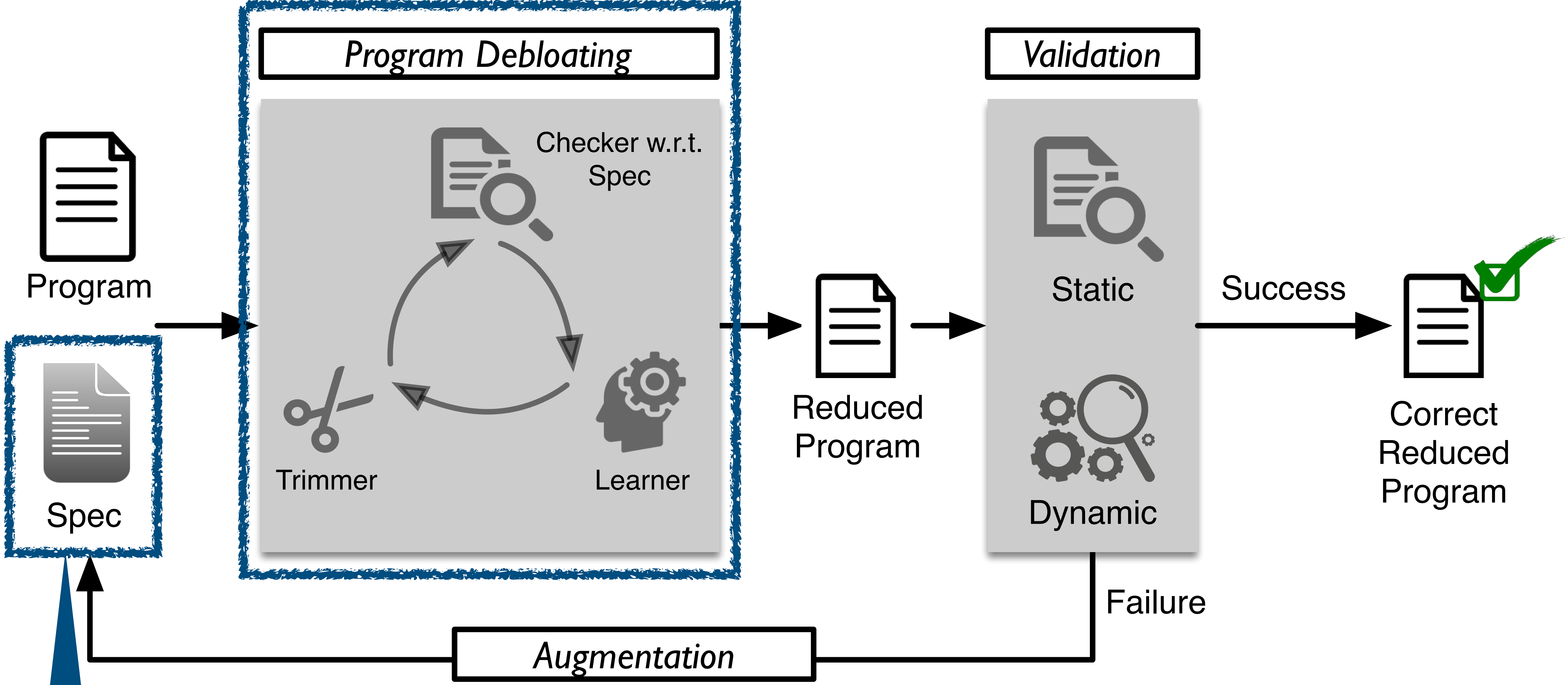
Failure

Augmentation

1. How to provide high-level specification?

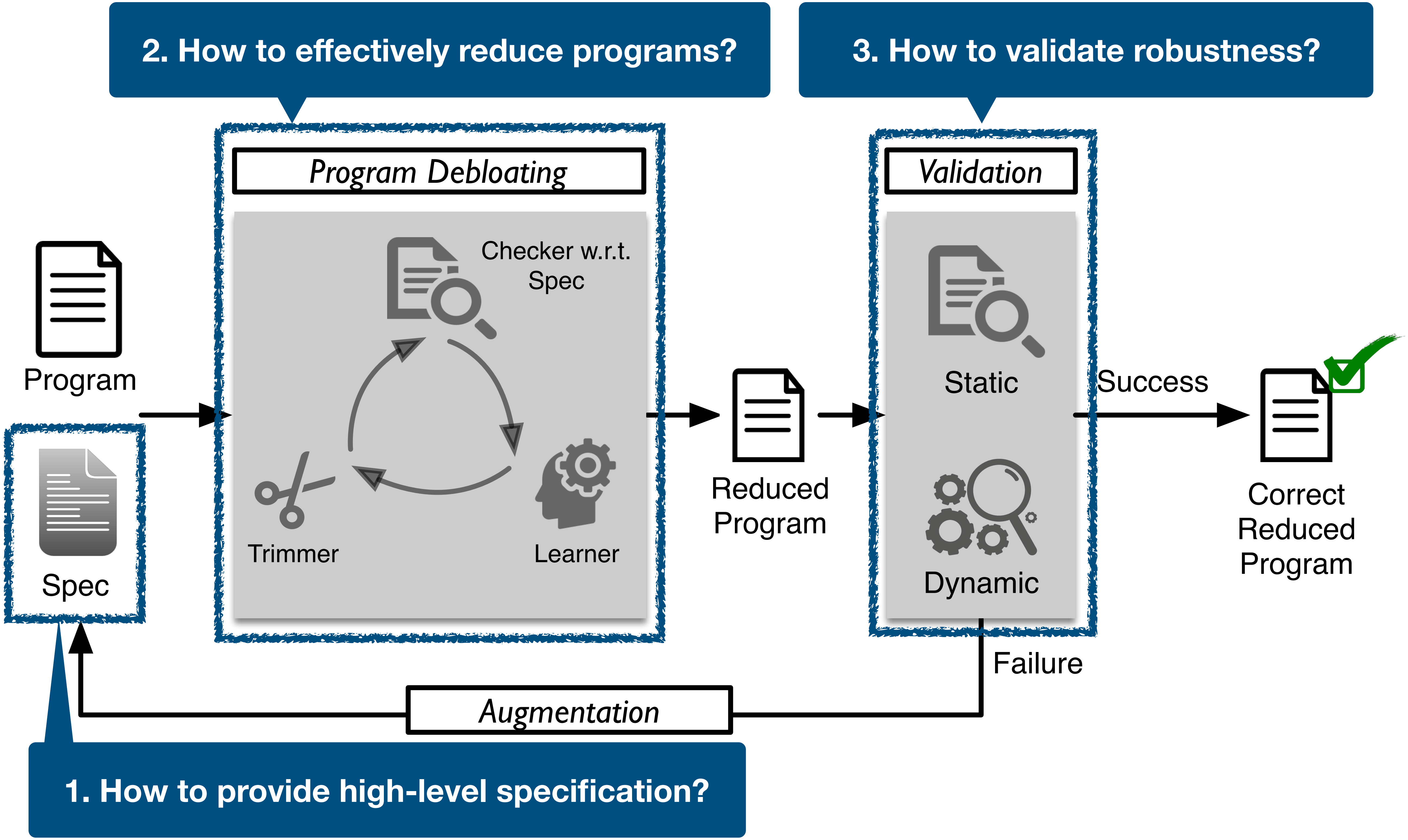
Key Questions

2. How to effectively reduce programs?



1. How to provide high-level specification?

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
core || exit 1
non_core || exit 1
```

High-level Specification

```
#!/bin/bash

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

1. The program is compilable.

```
# 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
core || exit 1
non_core || 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
core || exit 1
non_core || exit 1
```

High-level Specification

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

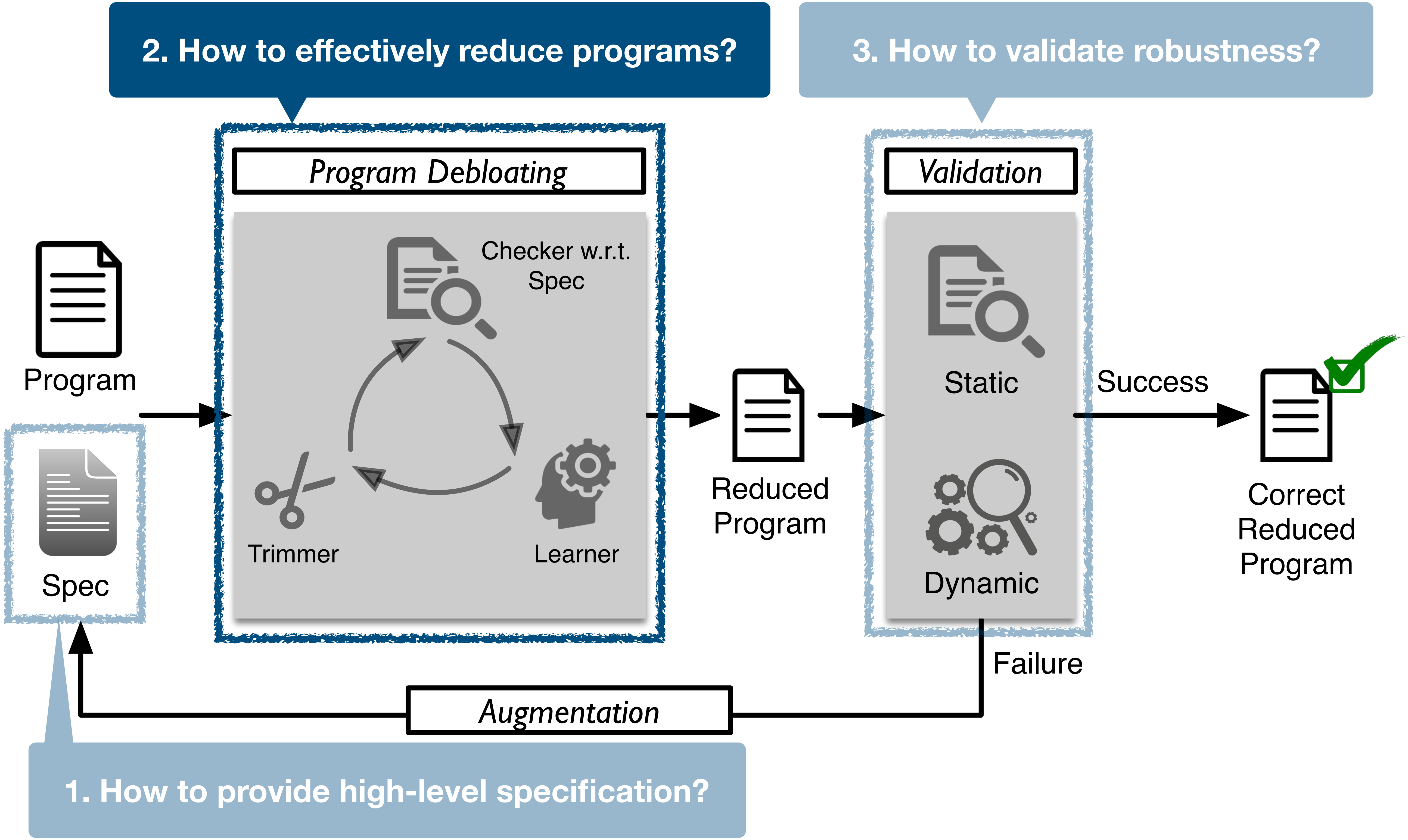
3. The program does not crash with the undesired functionalities. (e.g., using Clang sanitizers)

```
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
core || exit 1
non_core || exit 1
```

Key Questions



2. How to effectively reduce programs?

3. How to validate robustness?

Program Debloating

Checker w.r.t. Spec

Trimmer

Learner

Program

Spec

Reduced Program

Validation

Static

Dynamic

Success

Correct Reduced Program

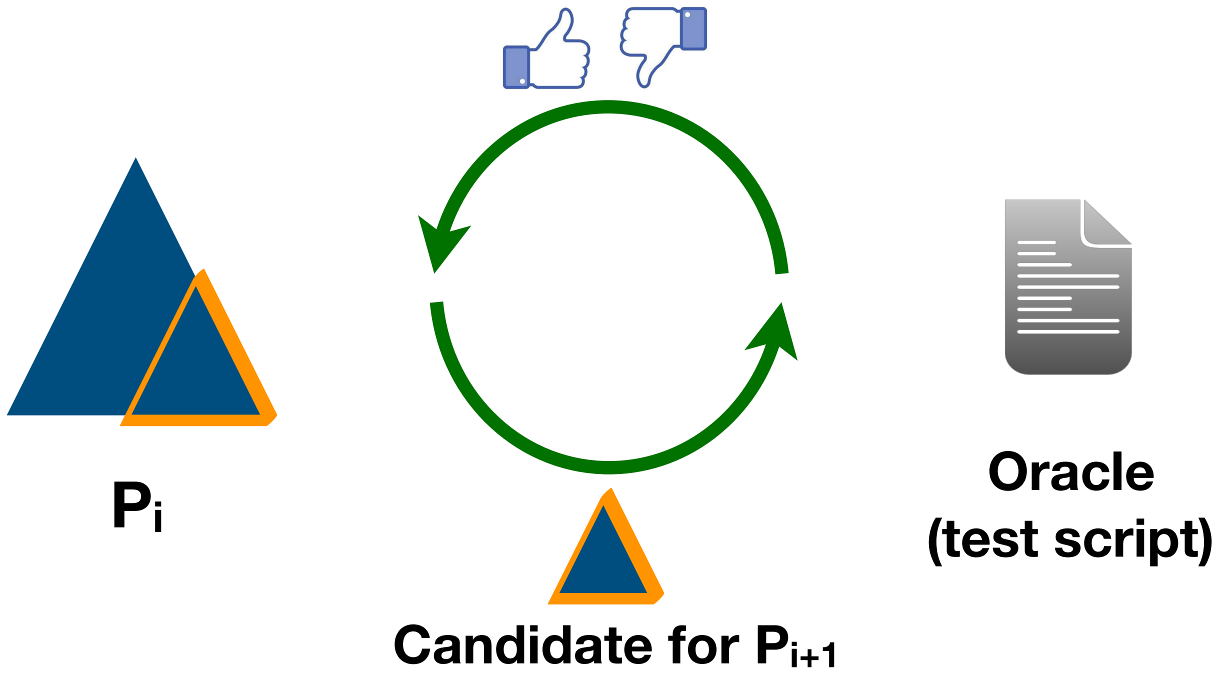
Failure

Augmentation

1. How to provide high-level specification?

Delta Debugging (DD)

[Zeller and Hildebrandt, 2002]



P_i

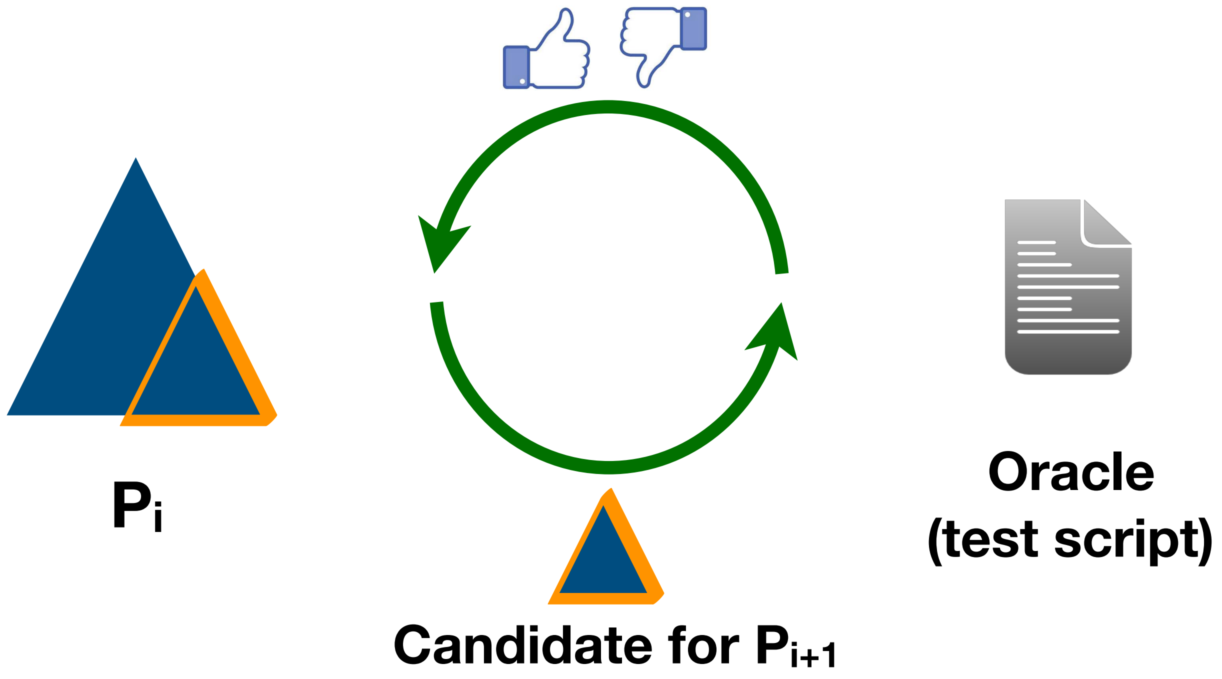
Candidate for P_{i+1}

Oracle
(test script)

Delta Debugging (DD)

[Zeller and Hildebrandt, 2002]

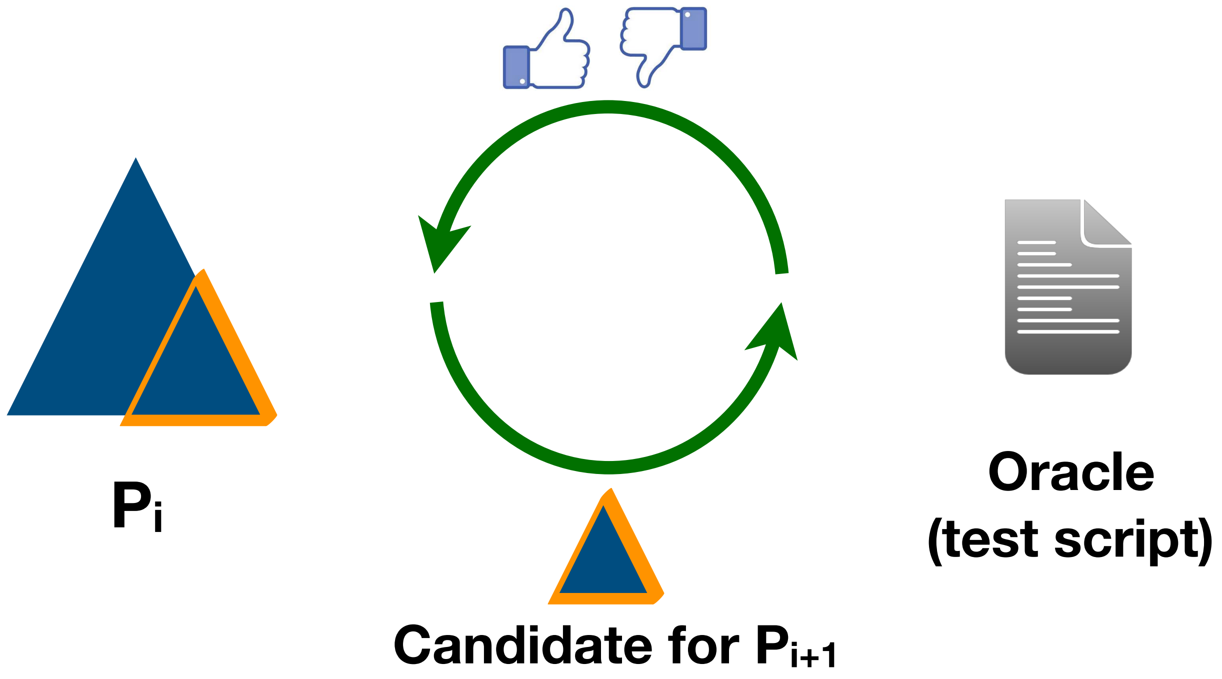
- Oracle O takes a program and returns Pass or Fail



Delta Debugging (DD)

[Zeller and Hildebrandt, 2002]

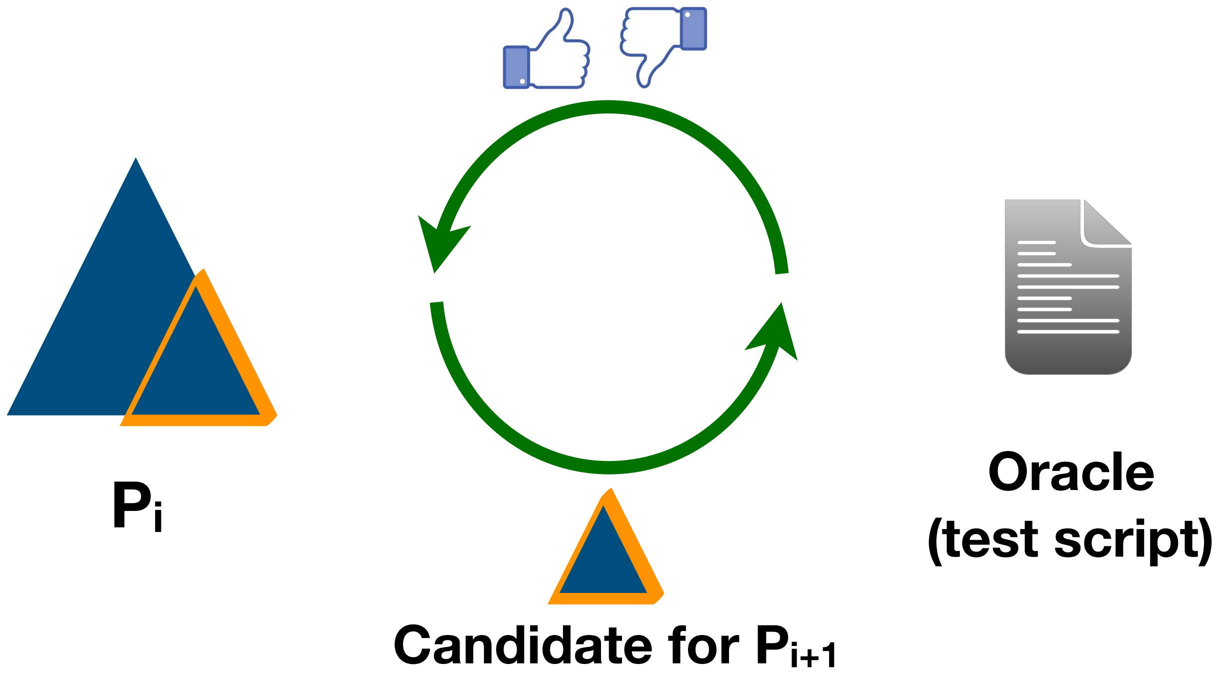
- Oracle O takes a program and returns Pass or Fail
- Given a program $\boldsymbol{P}$, find a **1-minimal** $\boldsymbol{P^*}$ such that $O(\boldsymbol{P^*})$ = Pass
- **1-minimal:** removing any element of $\boldsymbol{P^*}$ does not pass O



Delta Debugging (DD)

[Zeller and Hildebrandt, 2002]

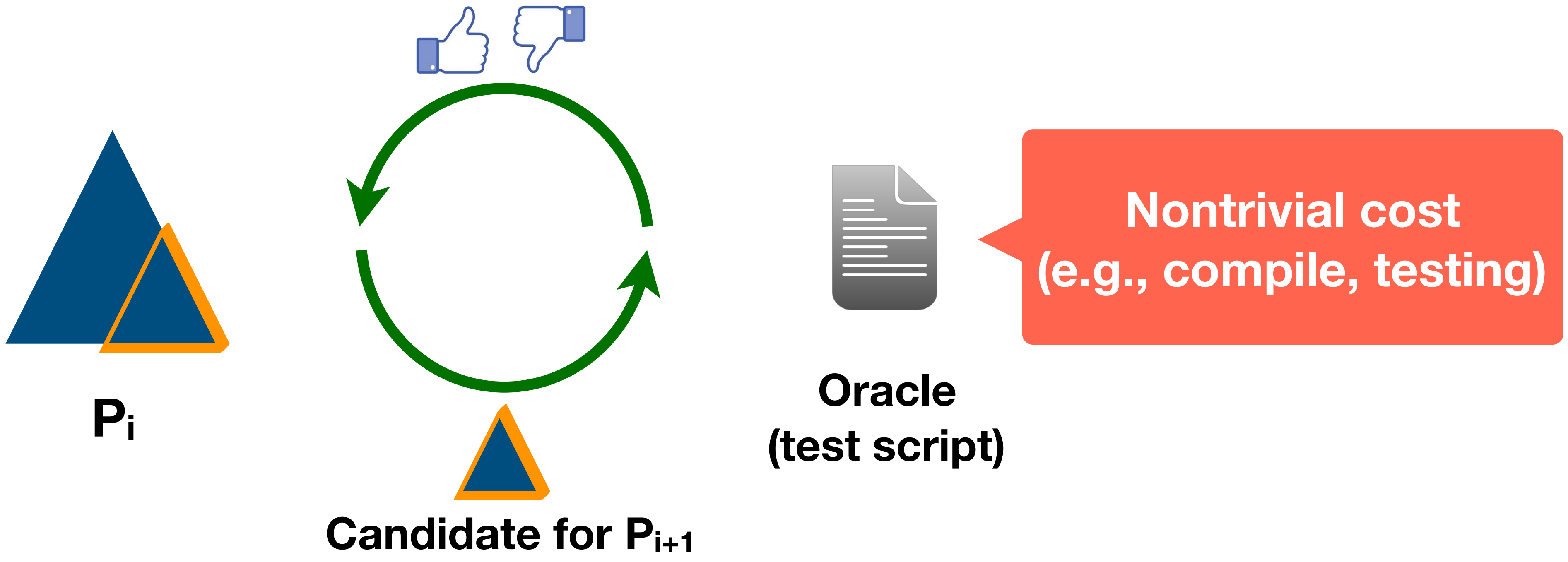
- Oracle $\mathcal{O}$ takes a program and returns Pass or Fail
- Given a program ***P***, find a **1-minimal** ***P****** such that $\mathcal{O}$(***P****) = Pass
- **1-minimal:** removing any element of ***P**** does not pass $\mathcal{O}$
- Time complexity: O(|***P***|^2)



DD: Key Challenges

[Zeller and Hildebrandt, 2002]

- Oracle $\mathcal{O}$ takes a program and returns Pass or Fail
- Given a program ***P***, find a **1-minimal** ***P**** such that $\mathcal{O}$(***P****) = Pass
- **1-minimal:** removing any element of ***P**** does not pass $\mathcal{O}$
- Time complexity: O(|***P***|^2)

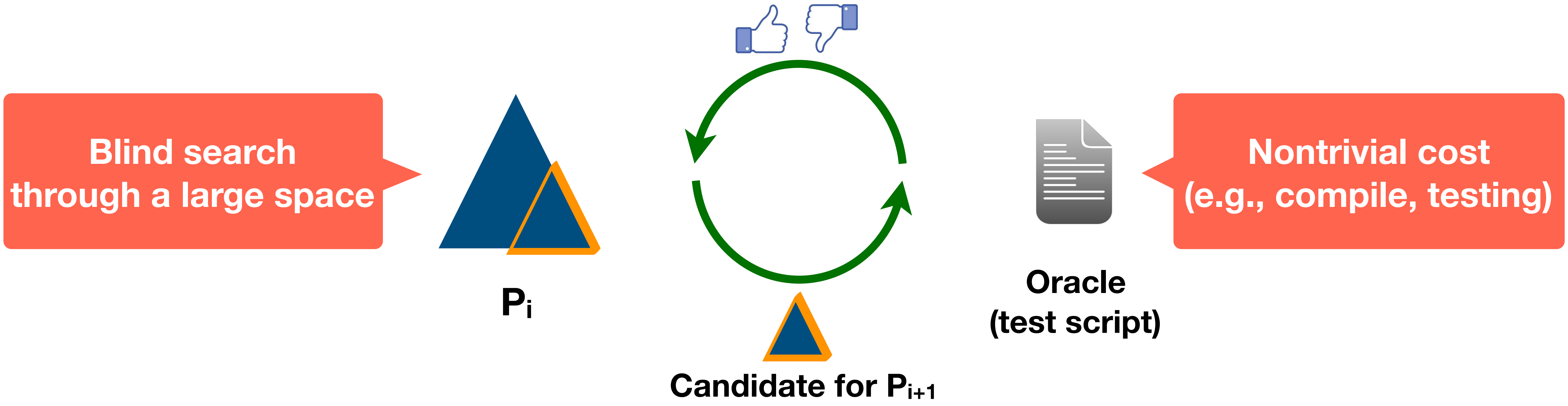


Nontrivial cost (e.g., compile, testing)

DD: Key Challenges

[Zeller and Hildebrandt, 2002]

- 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 element of ***P**** does not pass O
- Time complexity: O(|***P***|^2)



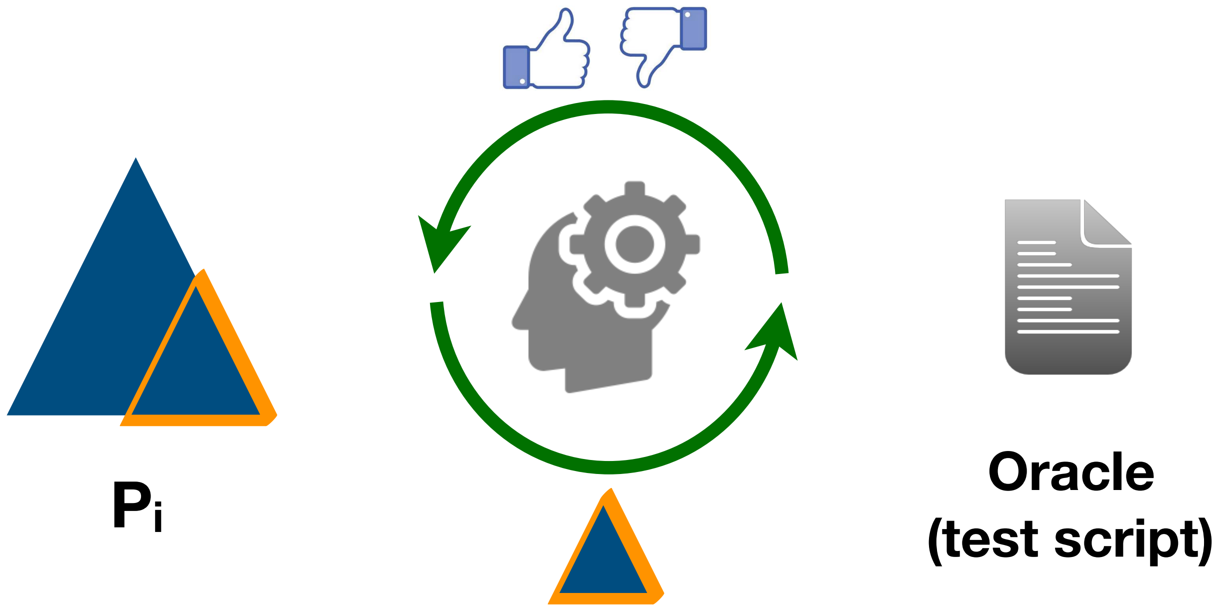
Blind search through a large space

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

Our Solution: Learning-guided DD

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

Data



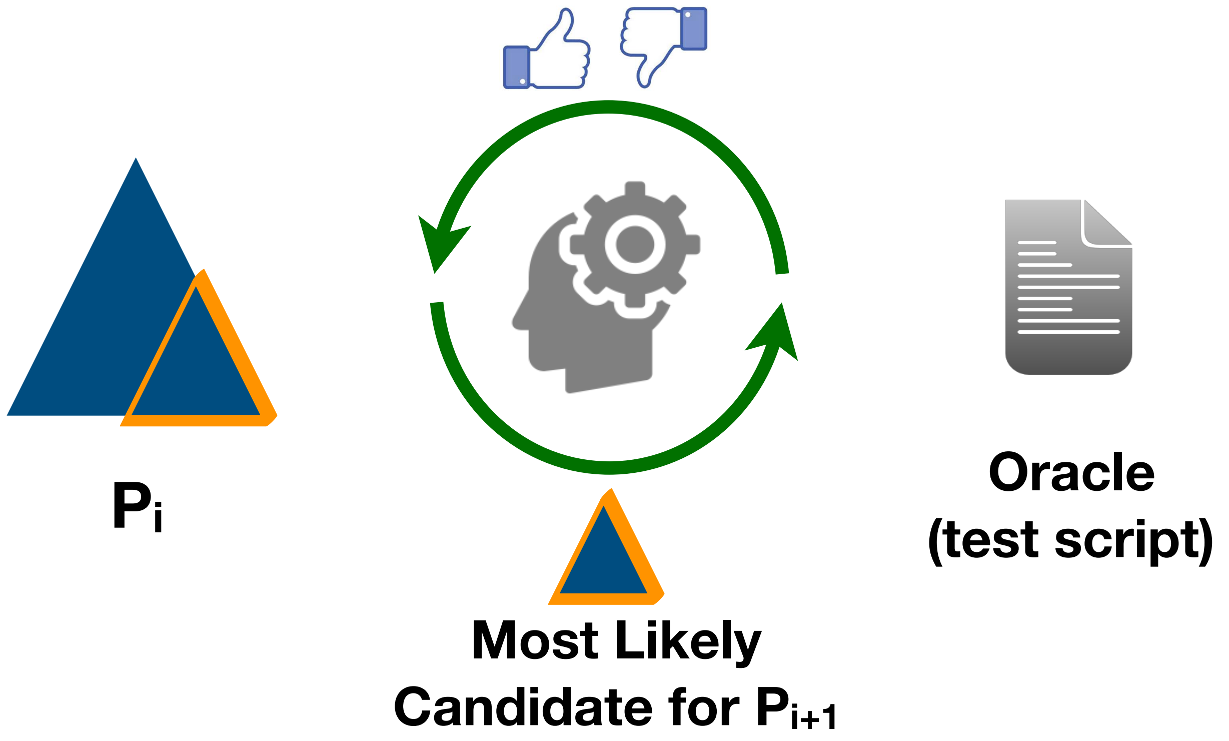
Most Likely Candidate for P_{i+1}

Our Solution: Learning-guided DD

- **Learn a policy** for DD using reinforcement learning (RL)

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

Data

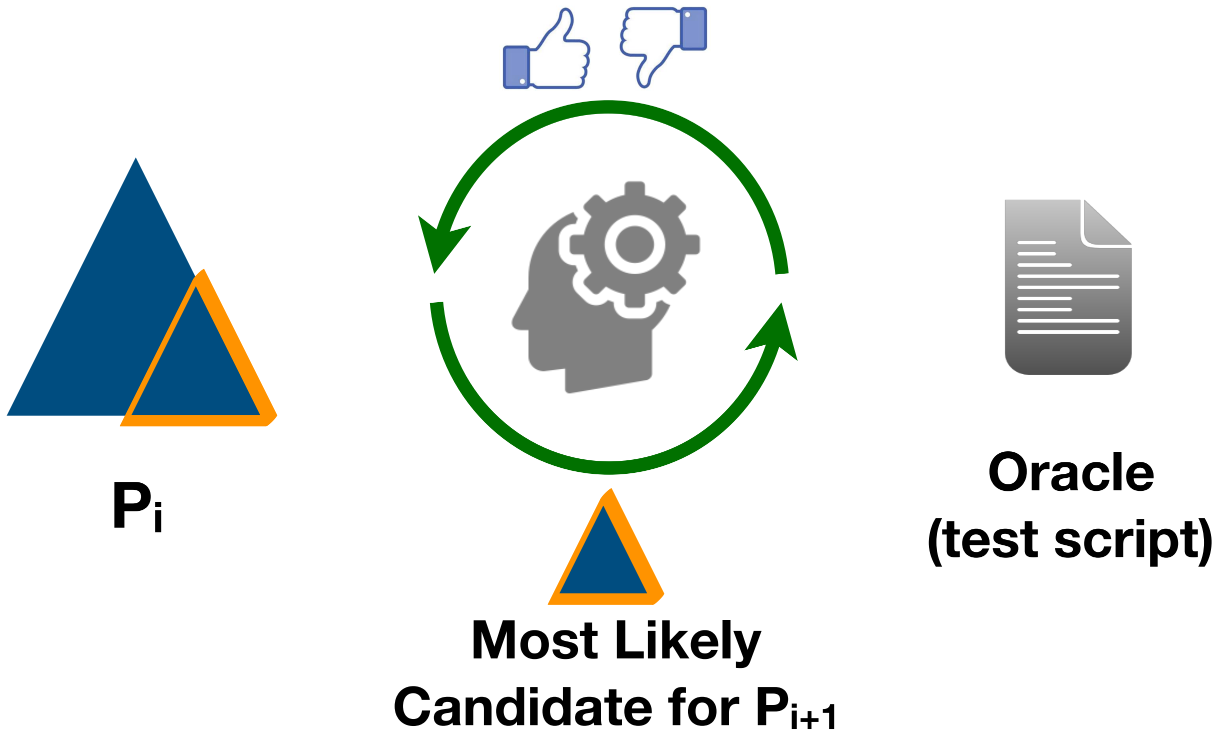


Our Solution:
Learning-guided DD

- **Learn a policy** for DD using reinforcement learning (RL)
- **Guide the search** based on the prediction of the learned policy

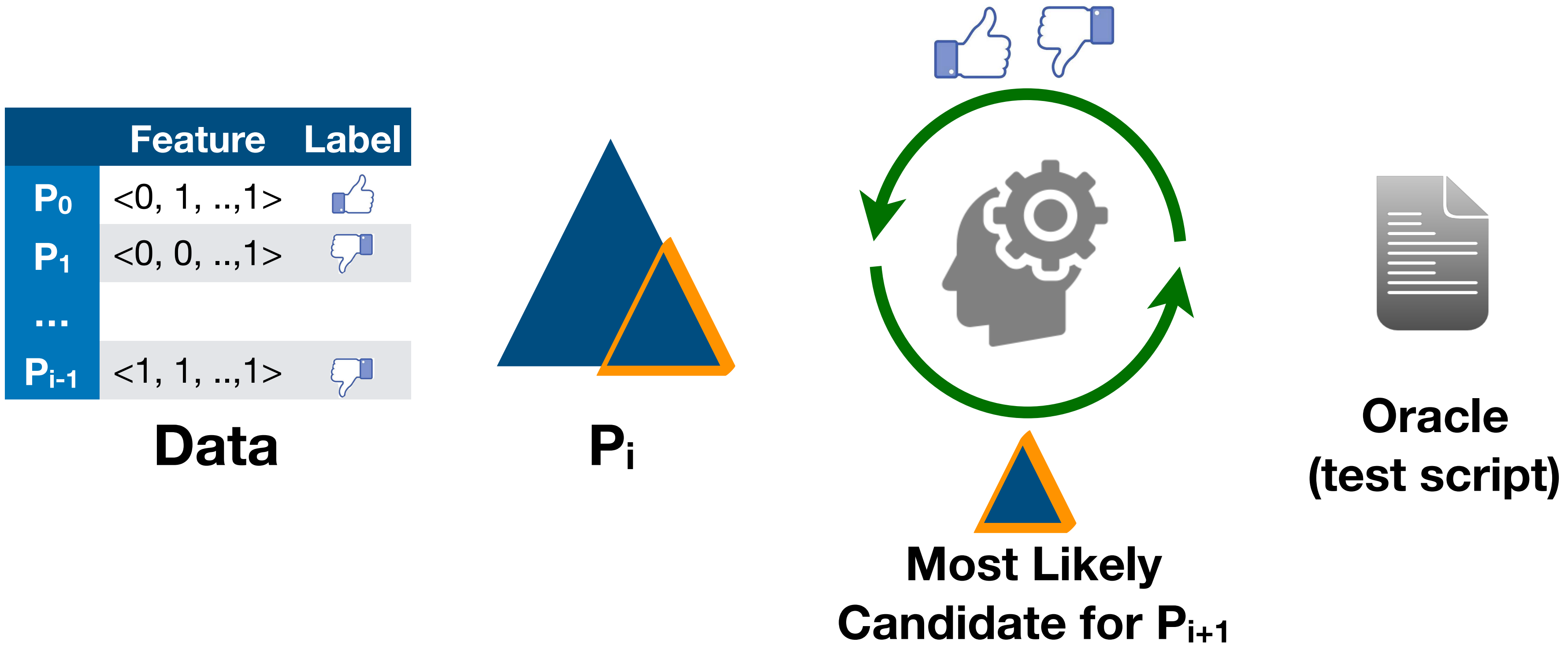
	Feature	Label
P_0	<0, 1, ..,1>	👍
P_1	<0, 0, ..,1>	👎
…		
P_{i-1}	<1, 1, ..,1>	👎

Data



Our Solution: Learning-guided DD

- **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

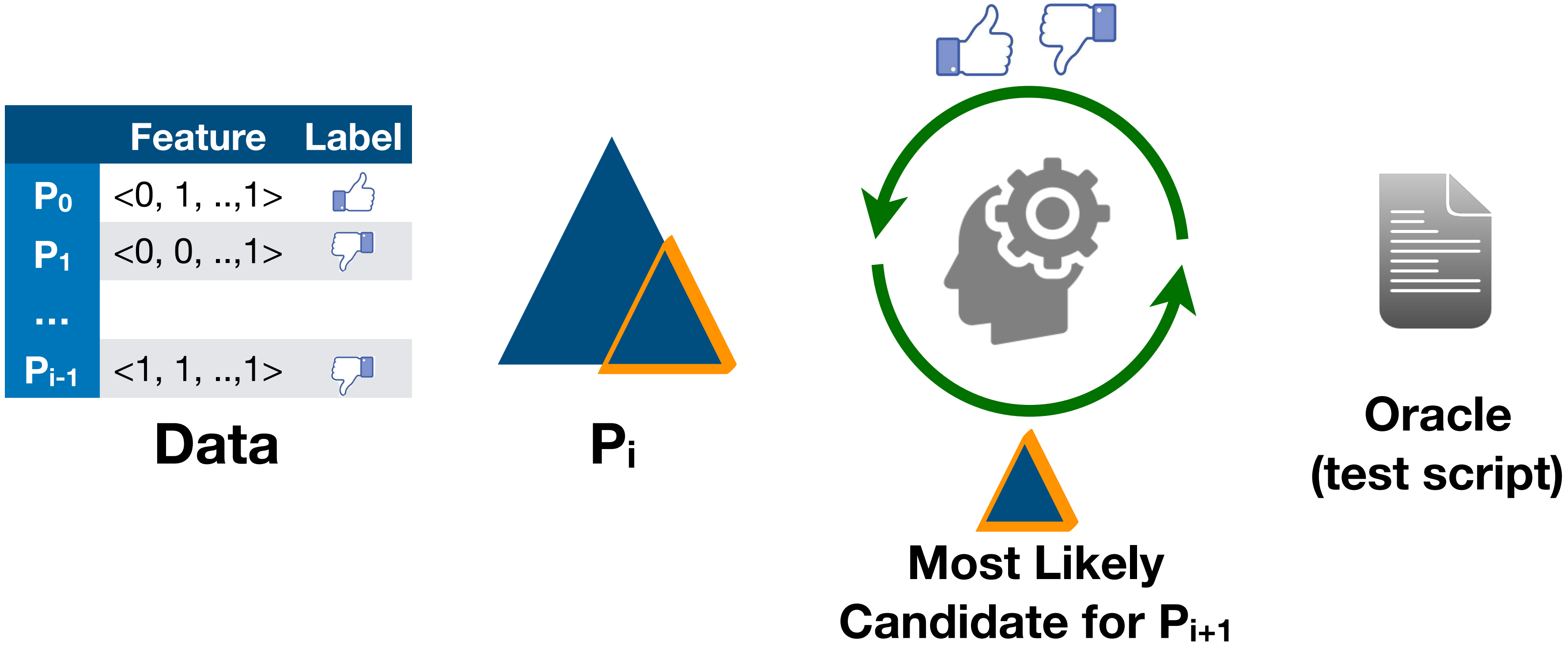
P_i

Most Likely Candidate for P_{i+1}

Oracle (test script)

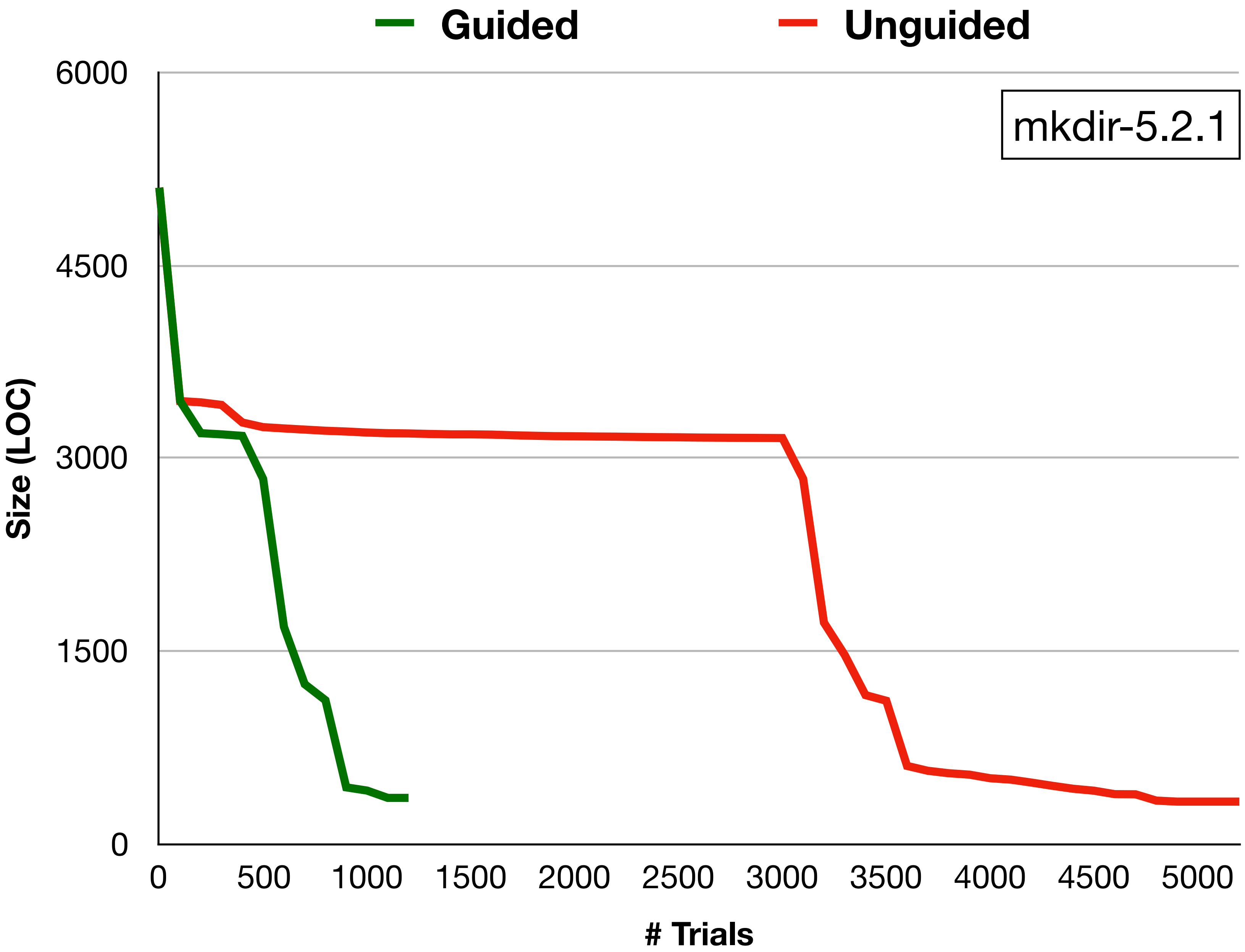
Our Solution: Learning-guided DD

- **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**
- Discard nonsensical programs upfront
  (e.g., invalid syntax, no main, uninitialized variables, etc)



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

Our Solution: Learning-guided DD



Example

```c
/* 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

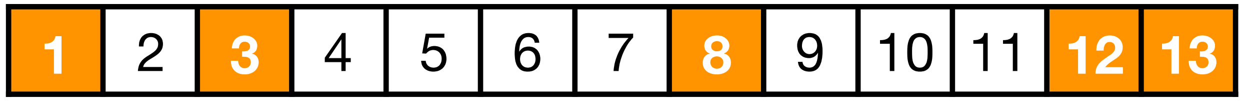
Minimal Desired Program:

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

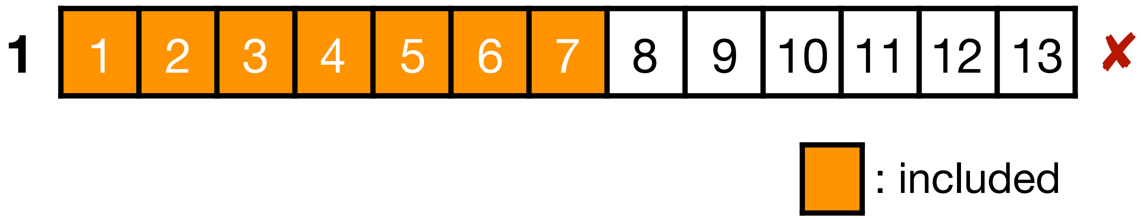
Guided Delta-Debugging



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

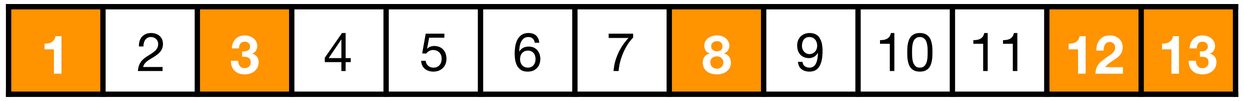
Unguided Delta-Debugging

Guided Delta-Debugging



1	1	2	3	4	5	6	7	8	9	10	11	12	13	✘

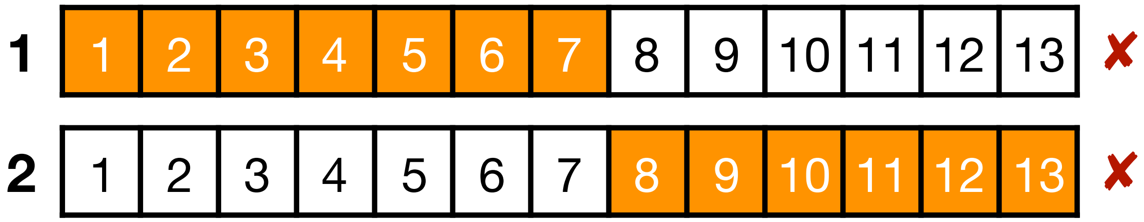
🟧 : included



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

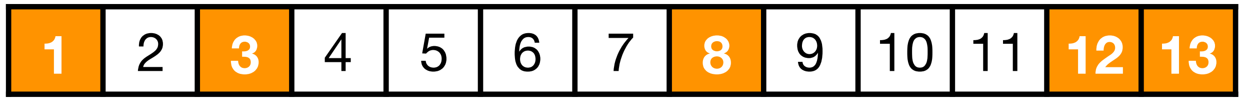
Unguided Delta-Debugging

Guided Delta-Debugging



1 | 1 2 3 4 5 6 7 | 8 9 10 11 12 13 ✘

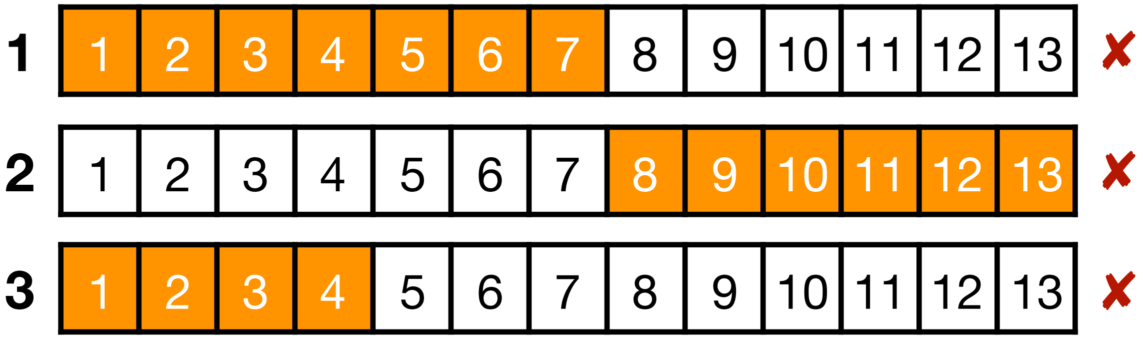
2 | 1 2 3 4 5 6 7 | 8 9 10 11 12 13 ✘



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

Unguided Delta-Debugging

Guided Delta-Debugging

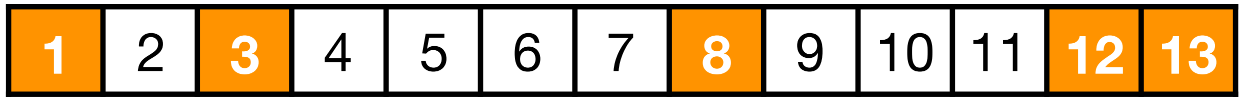


1 1 2 3 4 5 6 7 8 9 10 11 12 13 ✘

2 1 2 3 4 5 6 7 8 9 10 11 12 13 ✘

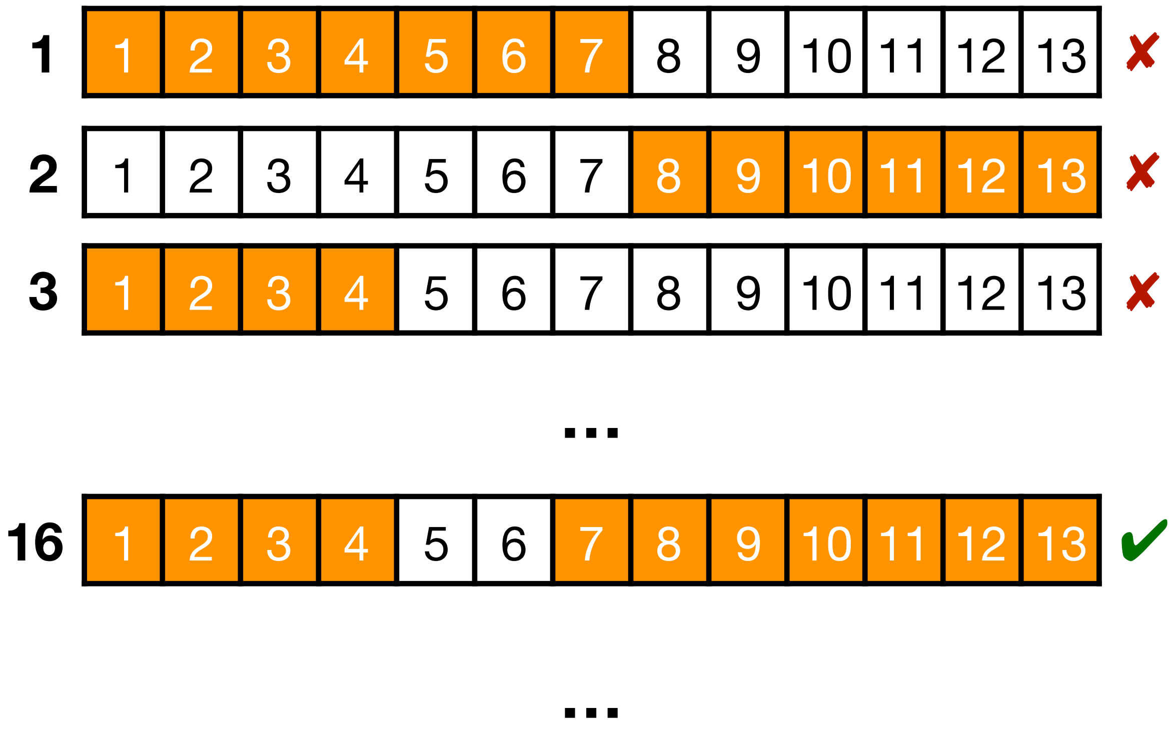
3 1 2 3 4 5 6 7 8 9 10 11 12 13 ✘

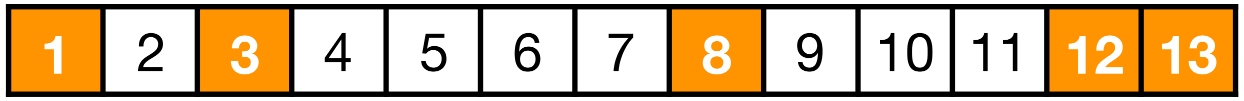
...



Unguided Delta-Debugging

Guided Delta-Debugging

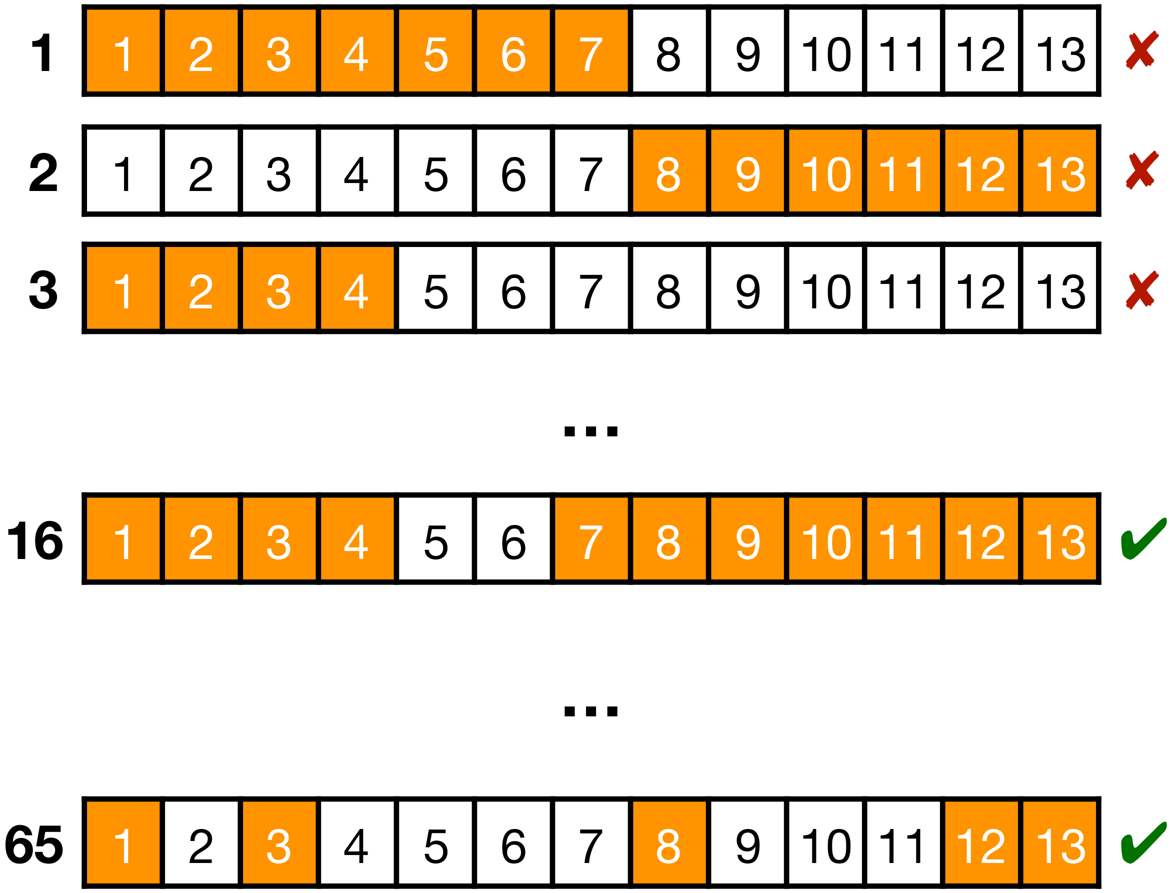




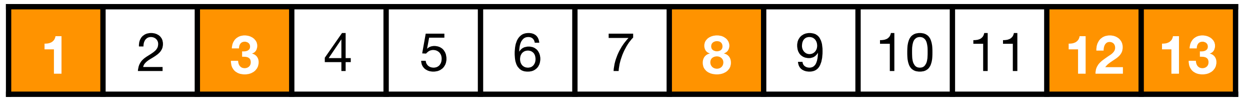
1	2	**3**	4	5	6	7	**8**	9	10	11	**12**	**13**

Unguided Delta-Debugging

Guided Delta-Debugging



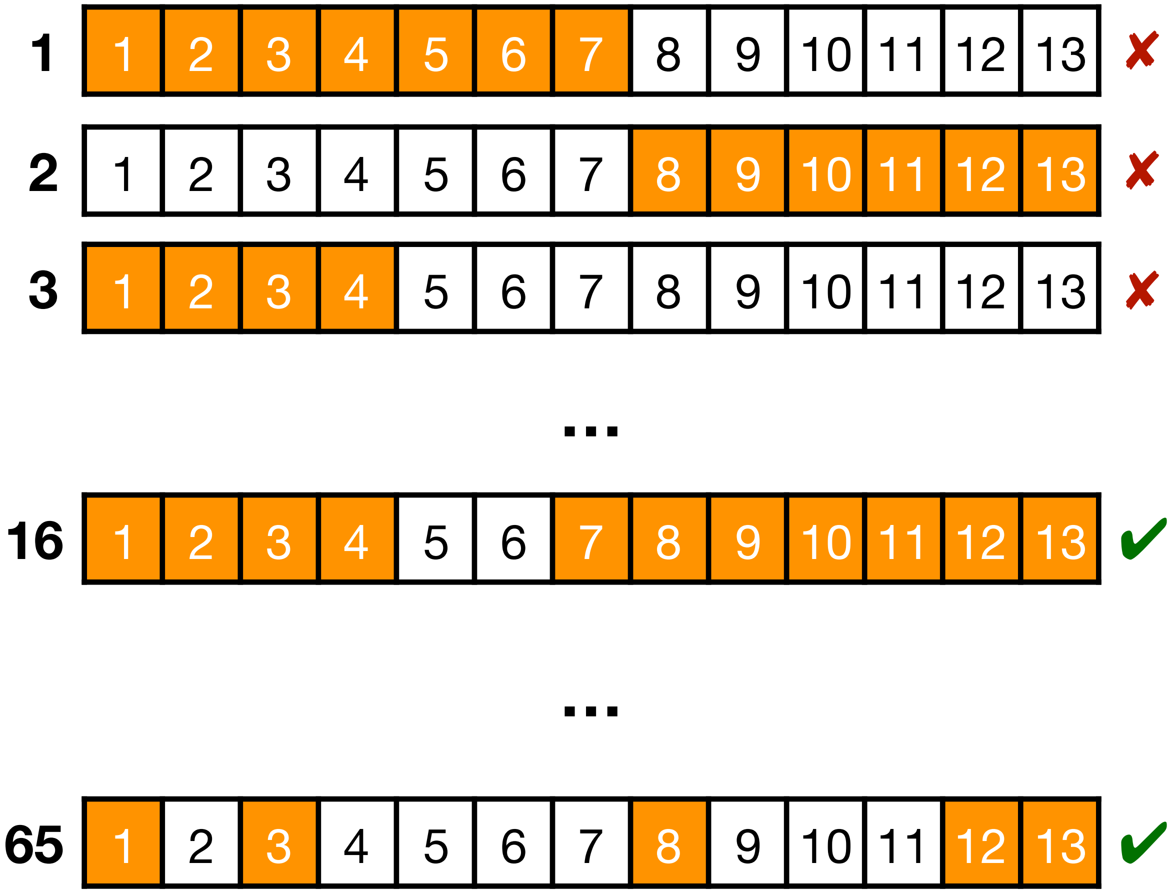
1	1	2	3	4	5	6	7	8	9	10	11	12	13	✗
2	1	2	3	4	5	6	7	8	9	10	11	12	13	✗
3	1	2	3	4	5	6	7	8	9	10	11	12	13	✗
							...							
16	1	2	3	4	5	6	7	8	9	10	11	12	13	✓
							...							
65	1	2	3	4	5	6	7	8	9	10	11	12	13	✓



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

Unguided Delta-Debugging

Guided Delta-Debugging



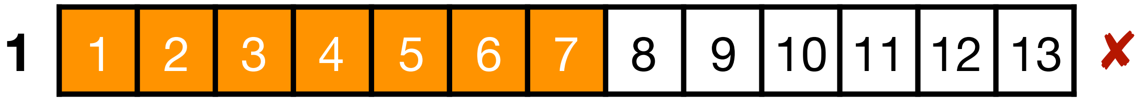
1	1	2	3	4	5	6	7	8	9	10	11	12	13	✗
2	1	2	3	4	5	6	7	8	9	10	11	12	13	✗
3	1	2	3	4	5	6	7	8	9	10	11	12	13	✗

...

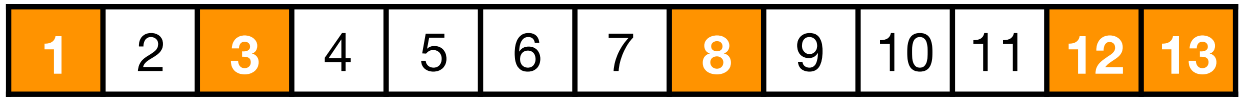
16	1	2	3	4	5	6	7	8	9	10	11	12	13	✓

...

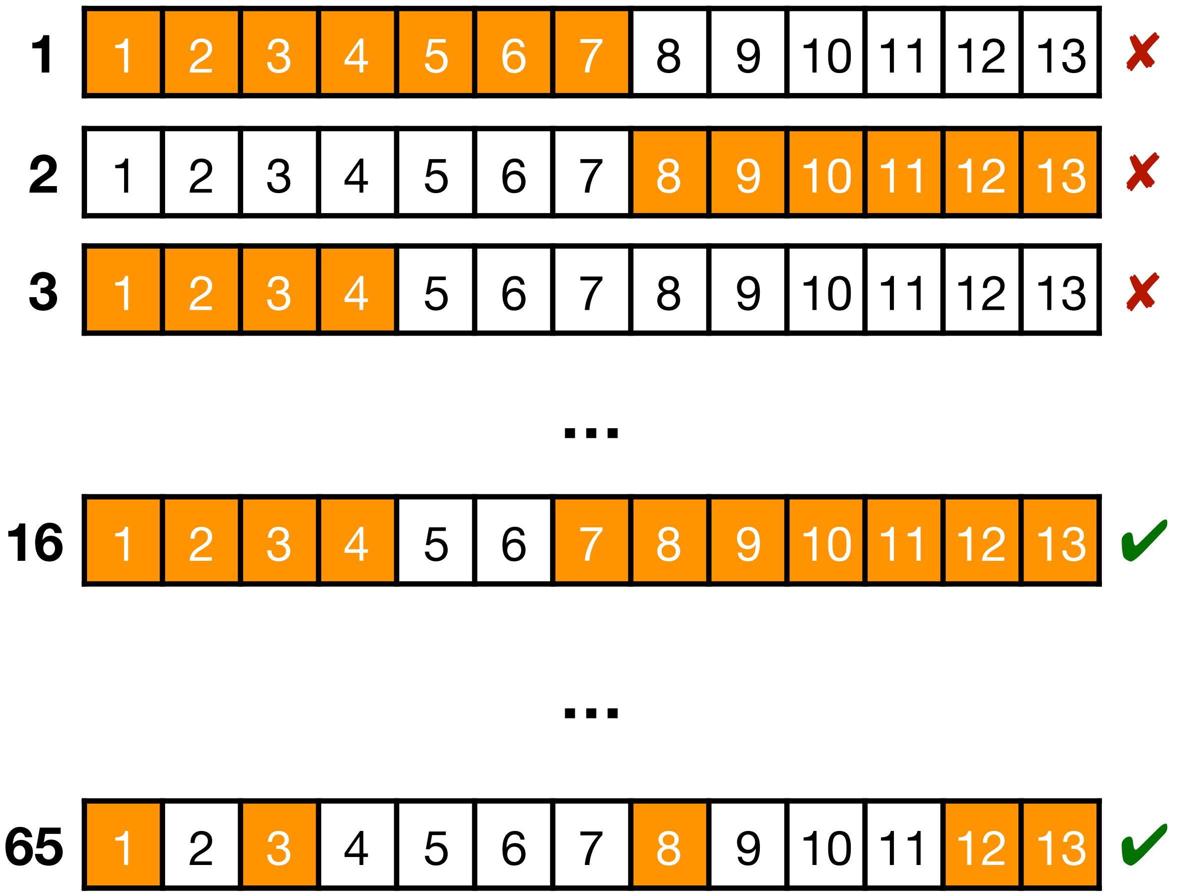
65	1	2	3	4	5	6	7	8	9	10	11	12	13	✓



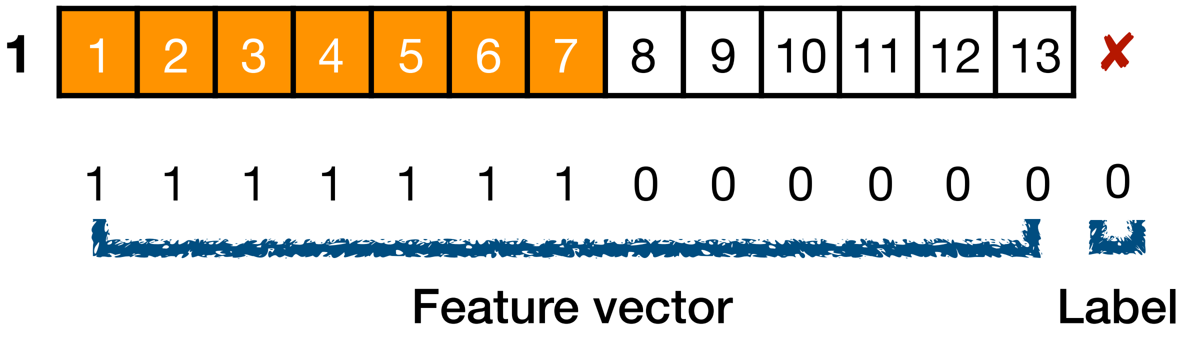
1	1	2	3	4	5	6	7	8	9	10	11	12	13	✗



Unguided Delta-Debugging



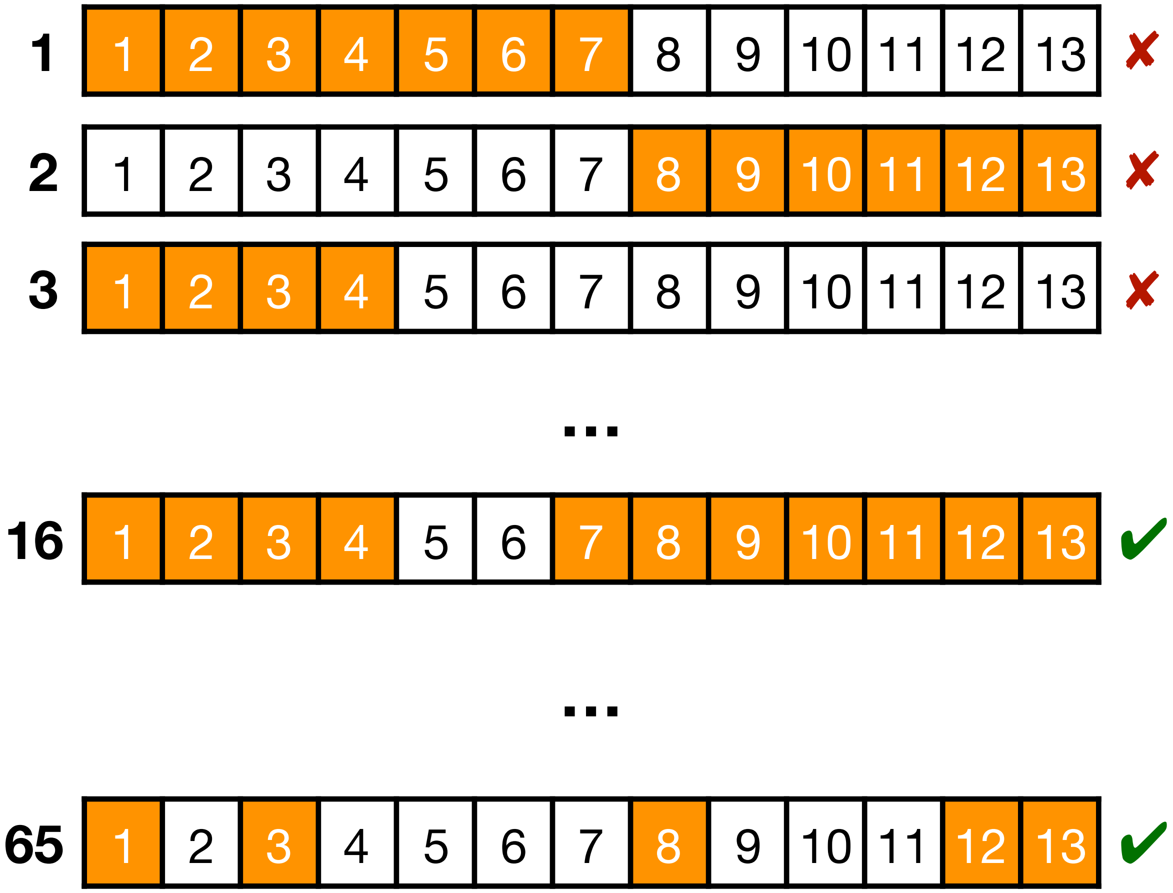
Guided Delta-Debugging



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

Unguided Delta-Debugging

Guided Delta-Debugging



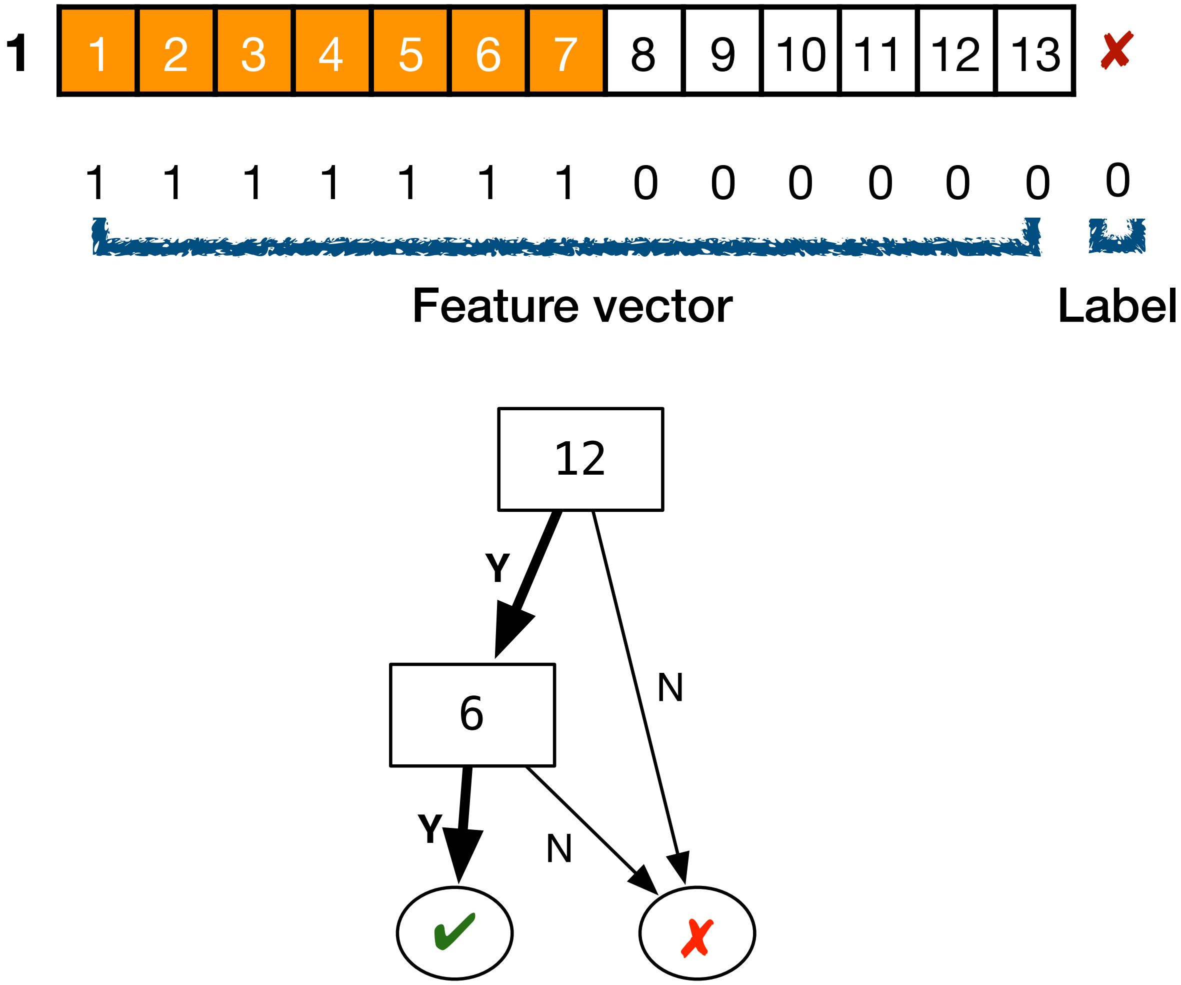
1	1	2	3	4	5	6	7	8	9	10	11	12	13	✗
2	1	2	3	4	5	6	7	8	9	10	11	12	13	✗
3	1	2	3	4	5	6	7	8	9	10	11	12	13	✗

...

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	✓

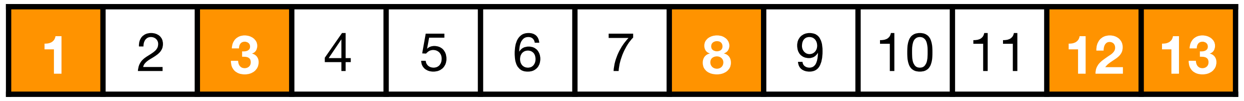


1	1	2	3	4	5	6	7	8	9	10	11	12	13	✗
	1	1	1	1	1	1	1	0	0	0	0	0	0	0

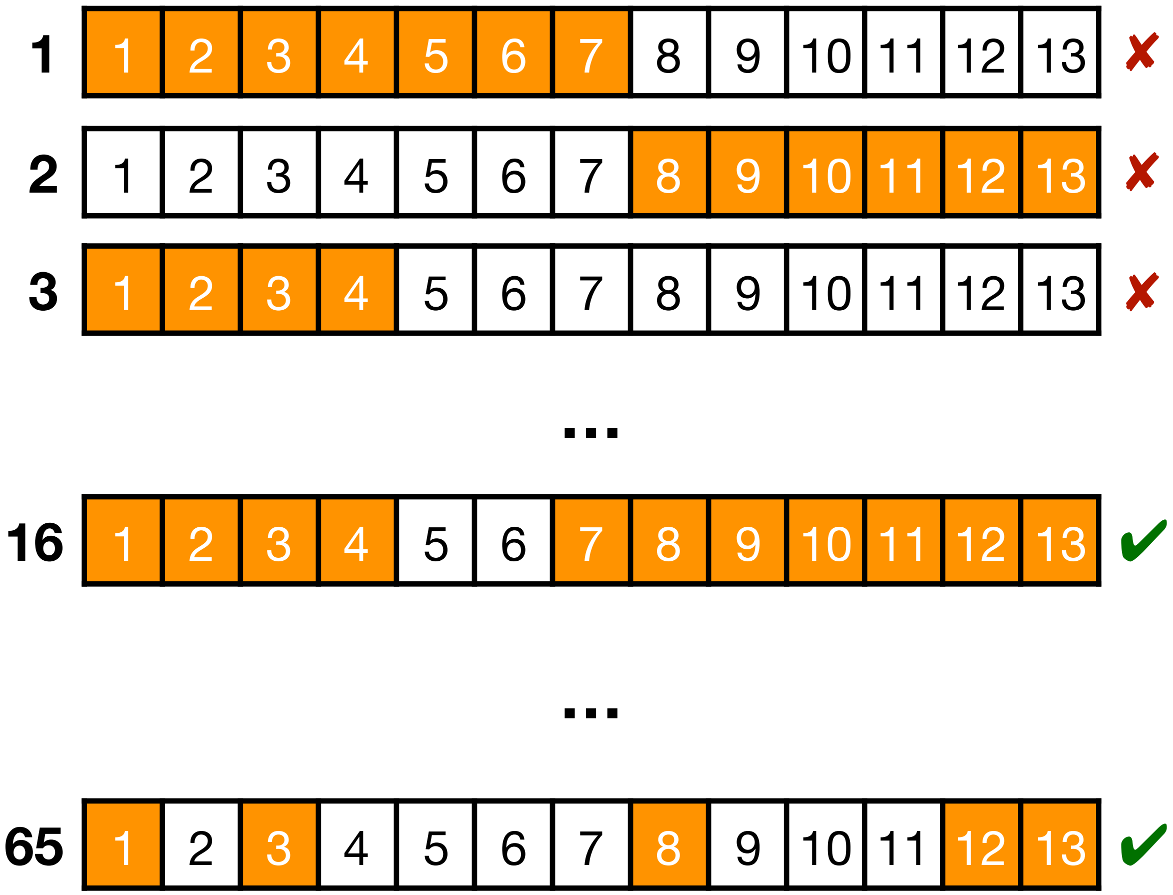
Feature vector

Label

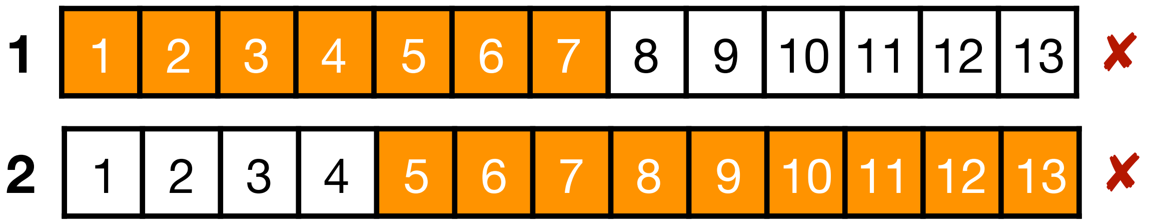
*P** should include 6 and 12

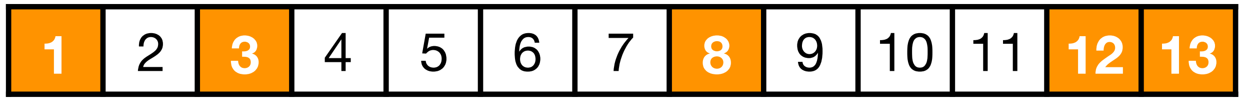


Unguided Delta-Debugging

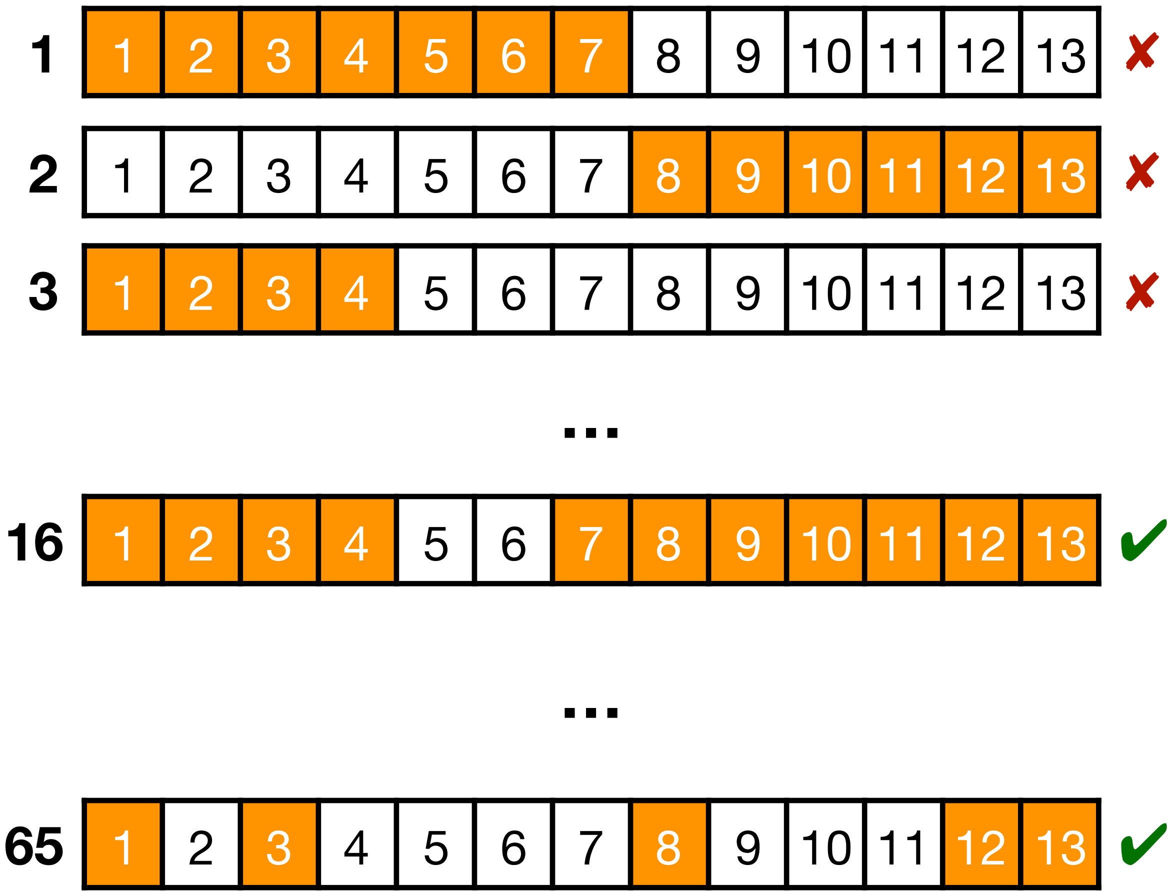


Guided Delta-Debugging

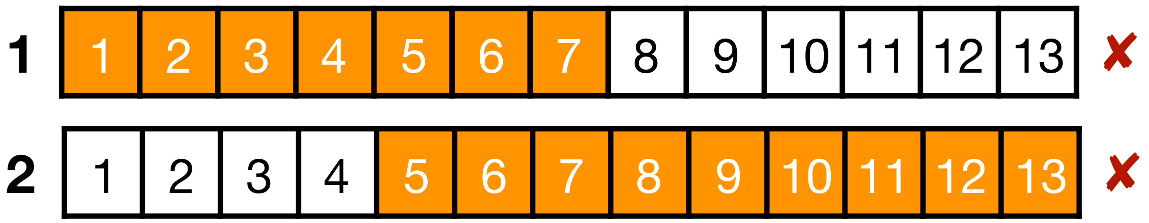


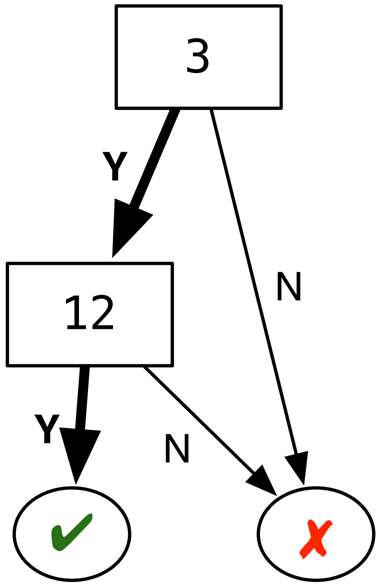


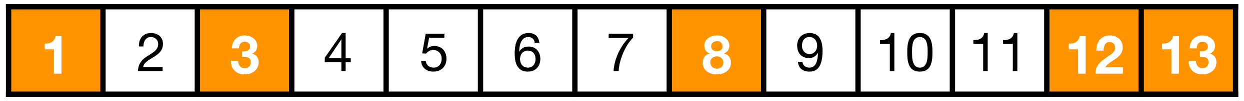
Unguided Delta-Debugging



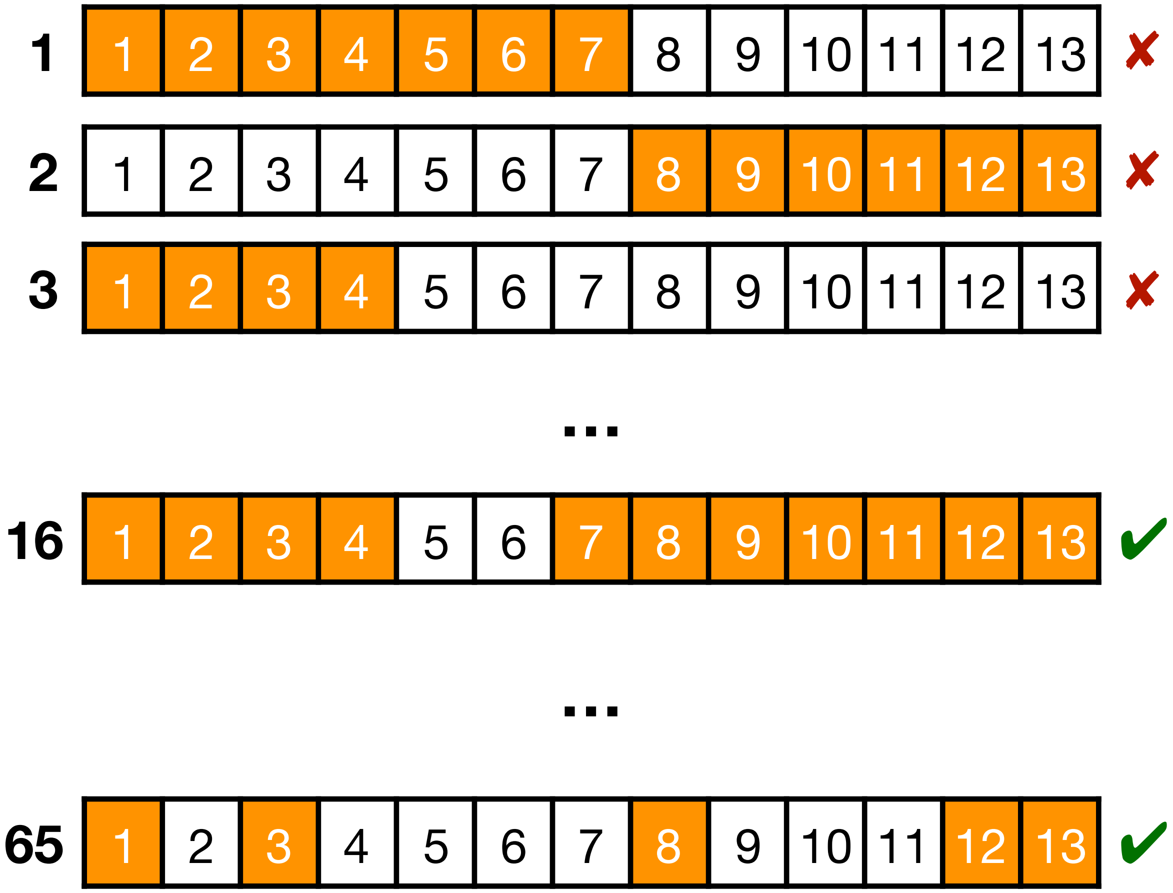
Guided Delta-Debugging



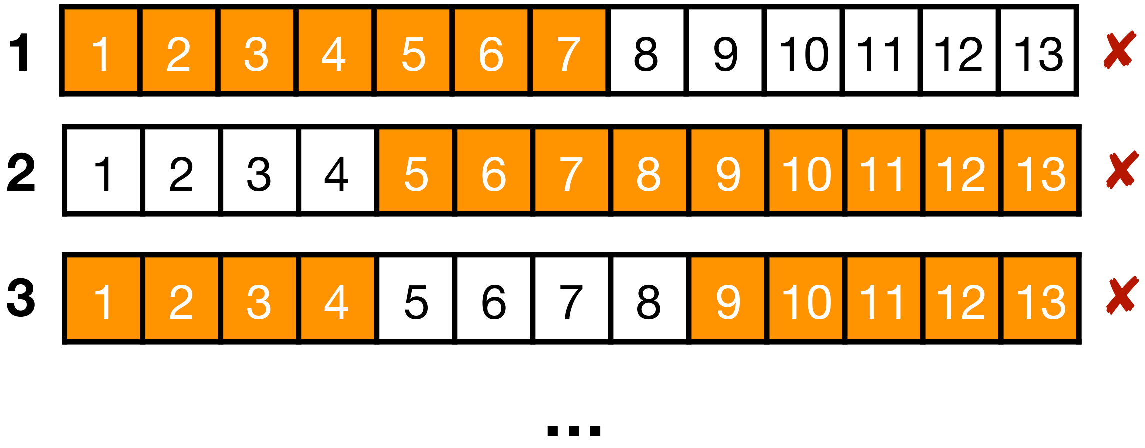


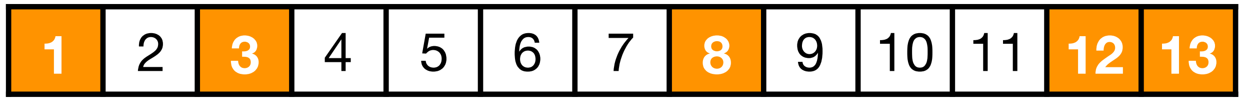


Unguided Delta-Debugging



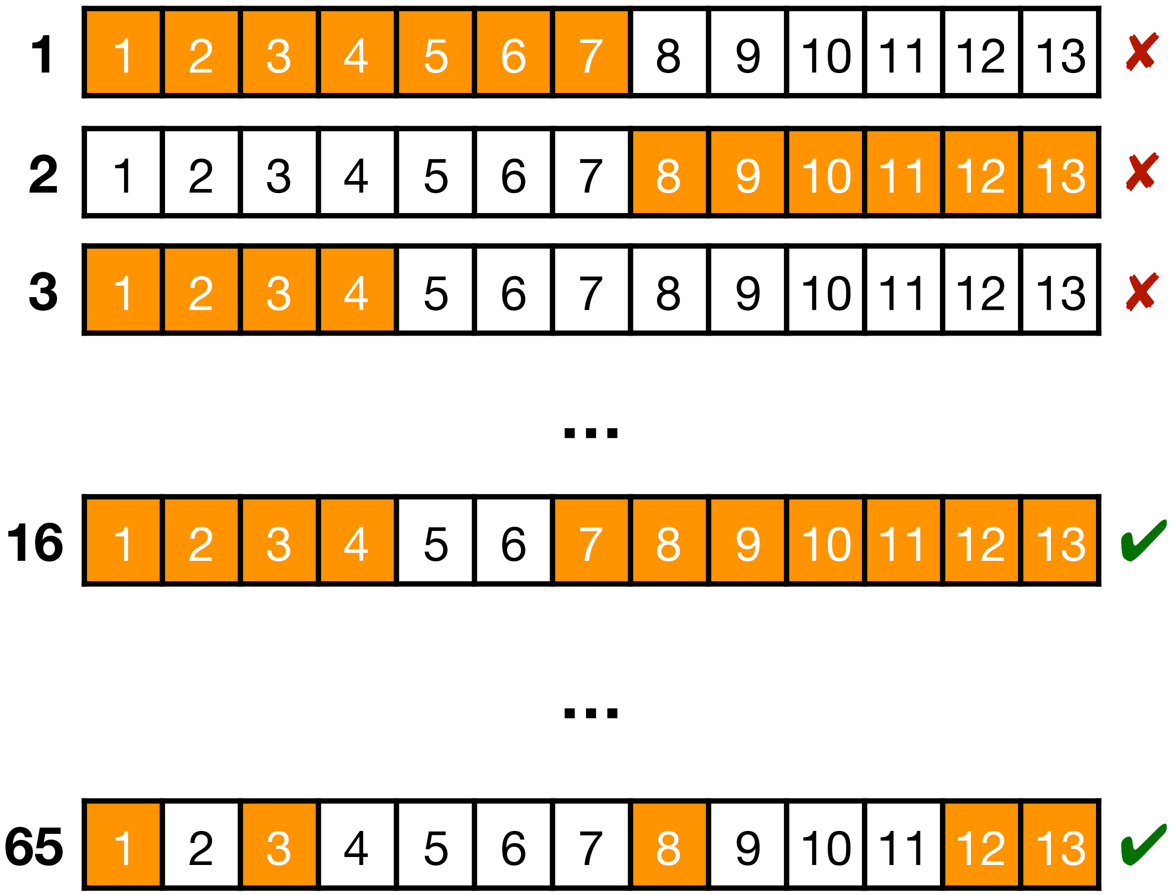
Guided Delta-Debugging

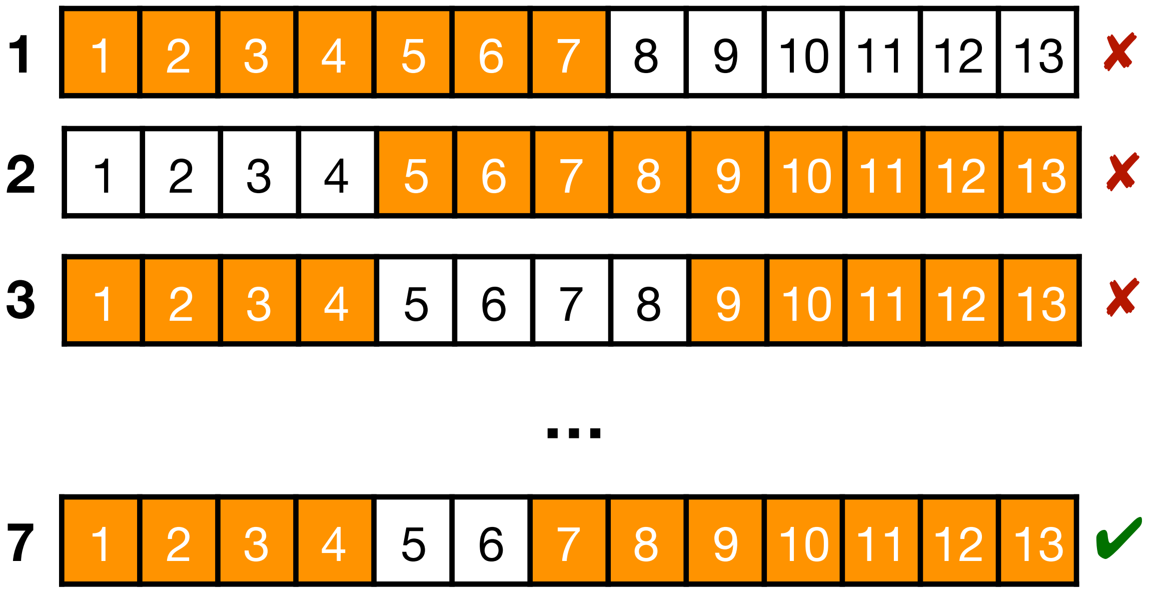


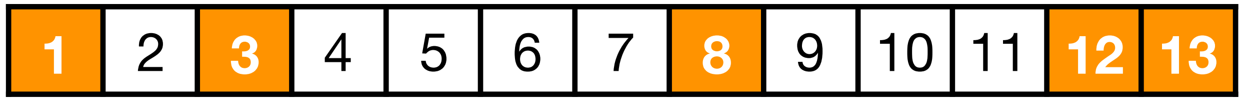


Unguided Delta-Debugging

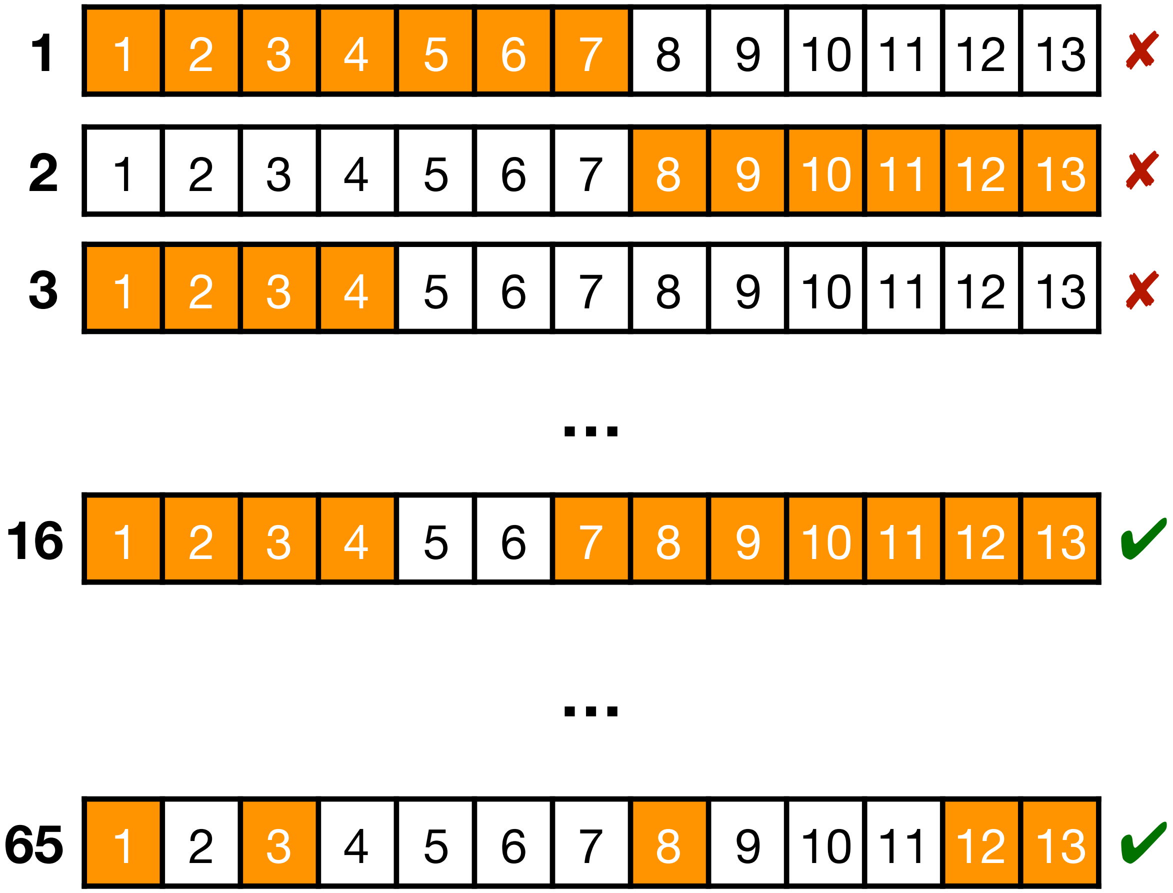
Guided Delta-Debugging



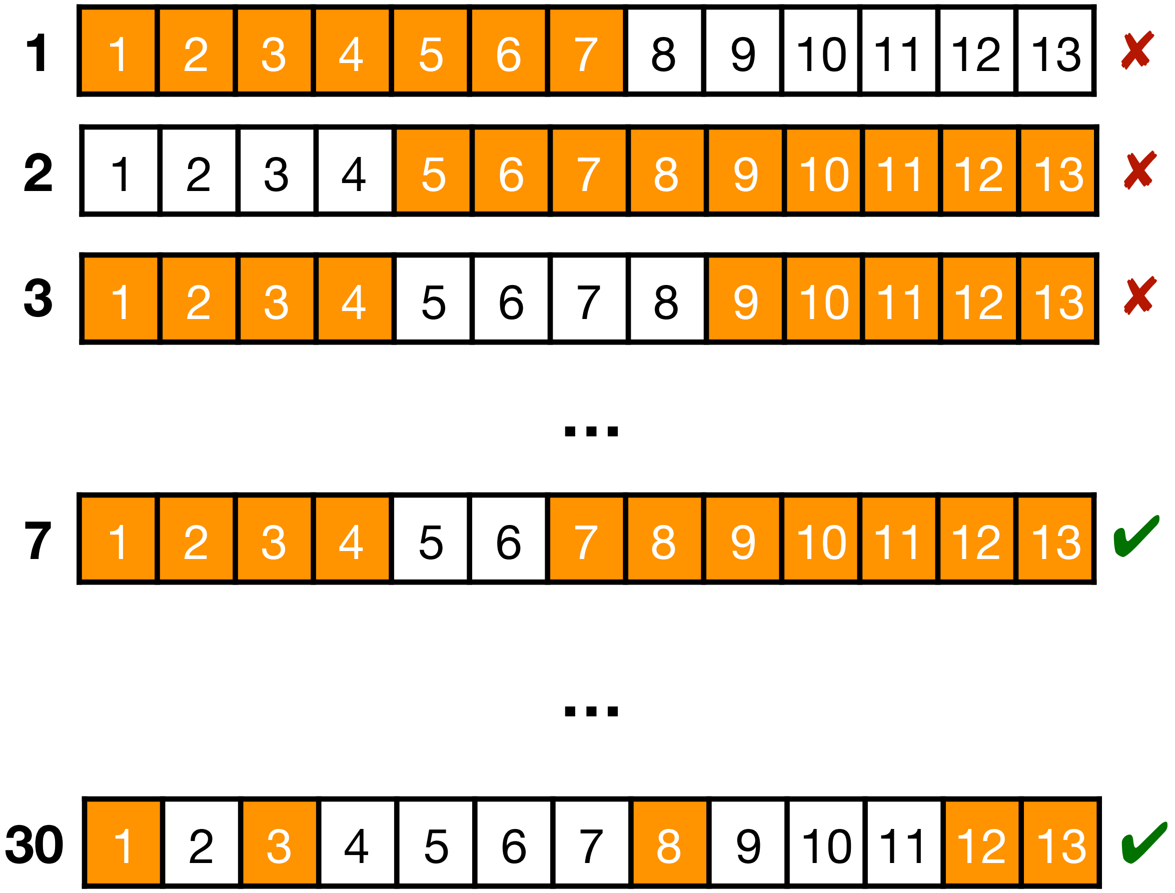


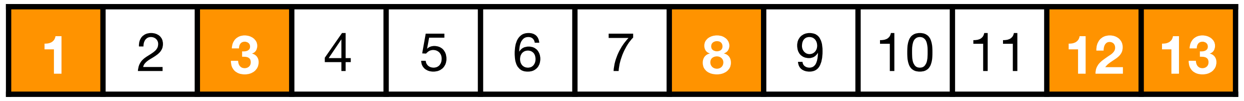


Unguided Delta-Debugging



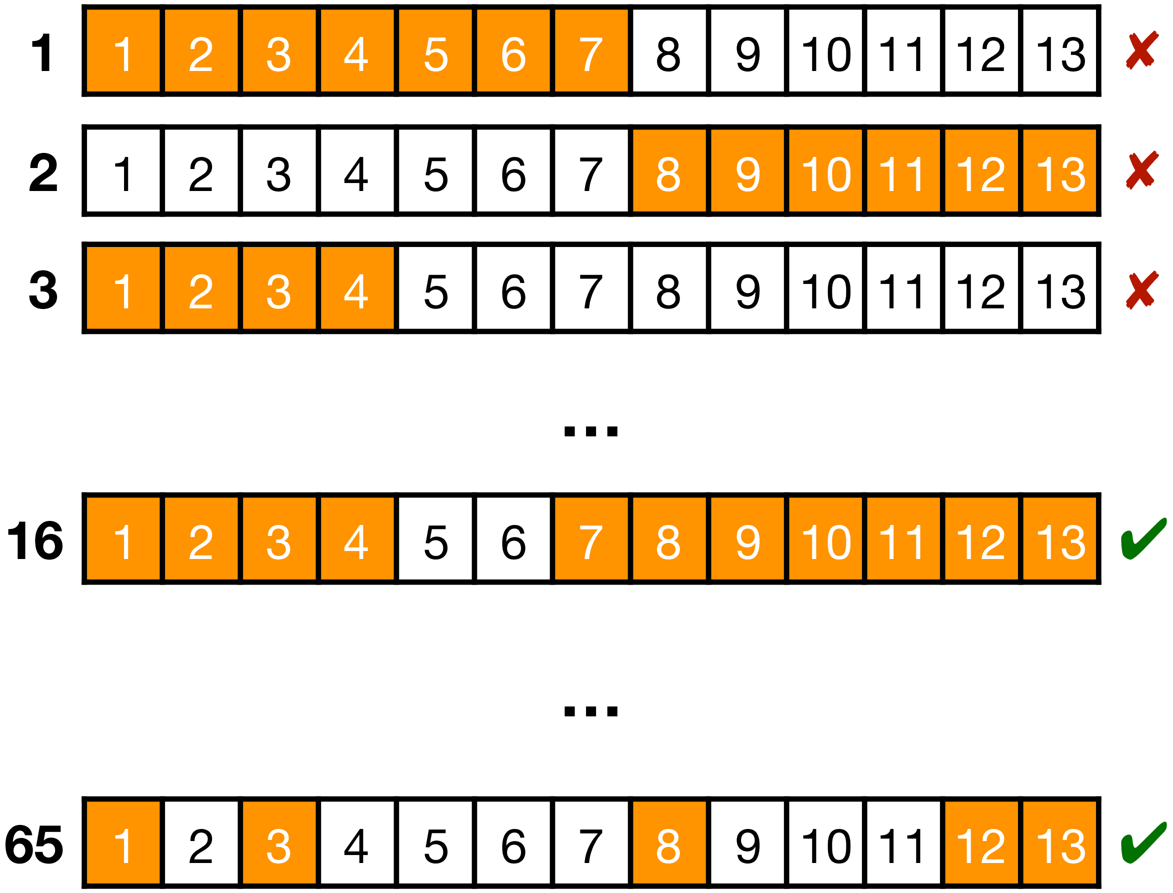
Guided Delta-Debugging





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 ✗

...

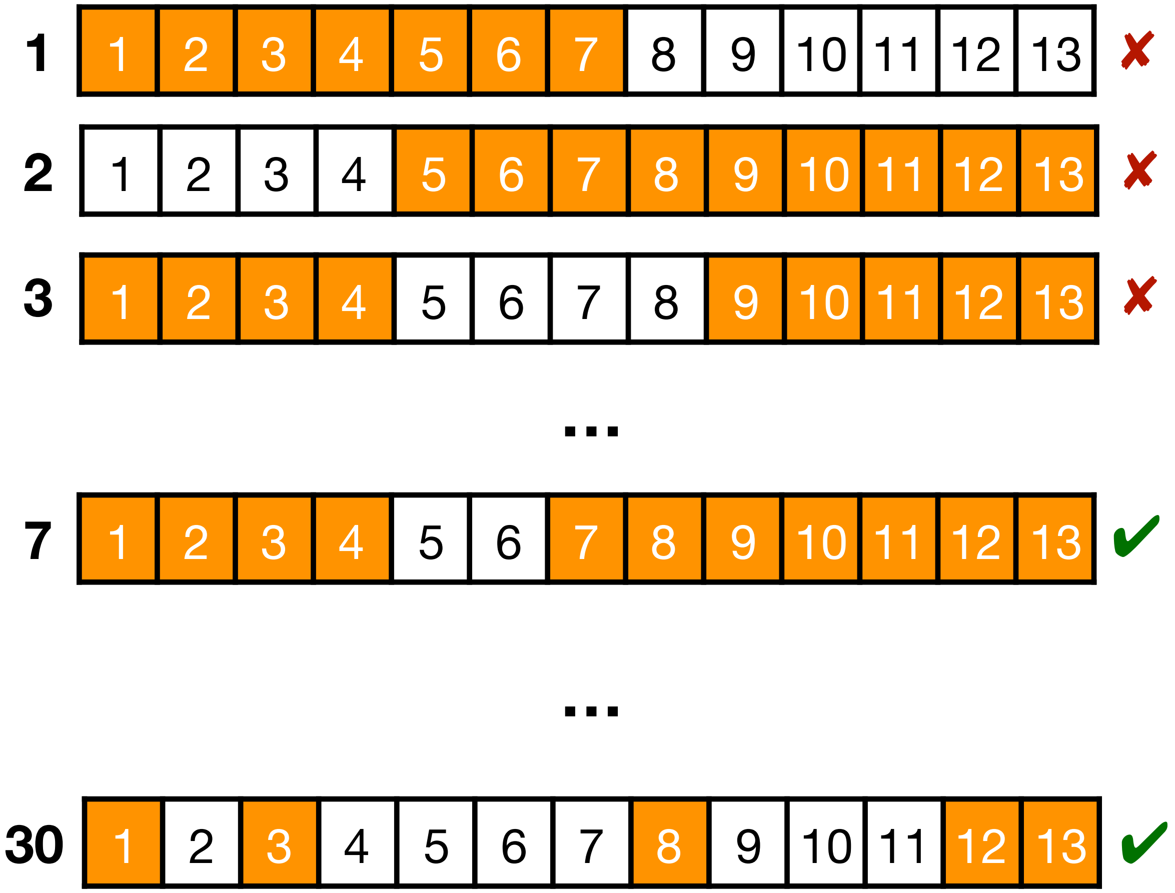
16 | 1 2 3 4 5 6 7 8 9 10 11 12 13 ✓

...

65 | 1 2 3 4 5 6 7 8 9 10 11 12 13 ✓

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 ✓

...

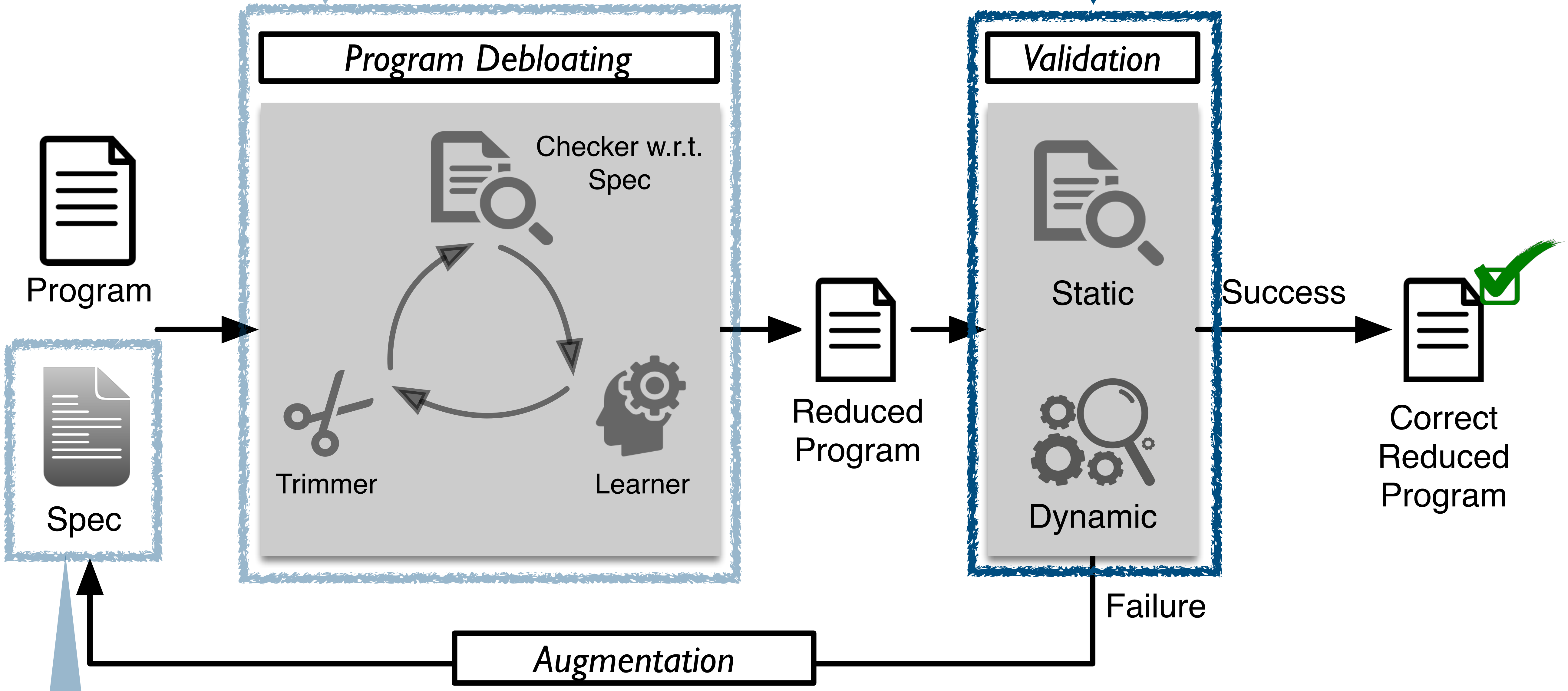
30 | 1 2 3 4 5 6 7 8 9 10 11 12 13 ✓

1,174 trials (901 failures)

Key Questions

2. How to effectively reduce programs?

3. How to validate robustness?



Program Debloating

Checker w.r.t. Spec

Trimmer

Learner

Validation

Static

Dynamic

Program

Spec

Reduced Program

Success

Correct Reduced Program

Failure

Augmentation

1. How to provide high-level specification?

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;
}
```

Talk Outline

- Motivation
- System Architecture
- **Evaluation**
- Conclusion

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	3,470	73	89
sort-8.16	7,206	379	89
tar-1.14	12,780	538	403
uniq-8.16	1,923	192	51
Total	55,848	6,111	4,729

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	3,470	73	89
sort-8.16	[illegible]	379	89
tar-1.14	[illegible]	538	403
uniq-8.16	1,923	192	51
Total	55,848	6,111	4,729

Reachable code by static analysis

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	3,470	73	89
sort-8.16	[illegible]	[illegible]	89
tar-1.14	[illegible]	[illegible]	403
uniq-8.16	1,923	[illegible]	51
Total	55,848	6,111	4,729

Reachable code by static analysis

Chisel reduced 89%

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	3,470	73	89
sort-8.16	[illegible]	[illegible]	[illegible]
tar-1.14	[illegible]	[illegible]	[illegible]
uniq-8.16	1,923	[illegible]	[illegible]
Total	55,848	6,111	4,729

Reachable code by static analysis

Chisel reduced 89%

Comparable to hand-written versions

Security Hardening

		#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	✔	885	210	(76%)	673	5	(99%)
tar-1.14	✔	1,528	303	(80%)	1,290	19	(99%)
uniq-8.16	✘	349	109	(69%)	60	1	(98%)
Total		6,752	2,285	(66%)	5,298	243	(95%)

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	✔	885	210	(76%)	673	5	(99%)
tar-1.14	✔	1,528	303	(80%)	1,290	19	(99%)
uniq-8.16	✘	349	109	(69%)	60	1	(98%)
Total		6,752	2,285	(66%)	5,298	243	(95%)

Security Hardening

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

Program	CVE	#ROP Gadgets			#Alarms		
		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	✓	[illegible]	[illegible]	[illegible]	673	5	(99%)
tar-1.14	✓	[illegible]	[illegible]	[illegible]	1,290	19	(99%)
uniq-8.16	✗	349	[illegible]	(69%)	60	1	(98%)
Total		6,752	2,285	(66%)	5,298	243	(95%)

Reduced potential attack surface

Security Hardening

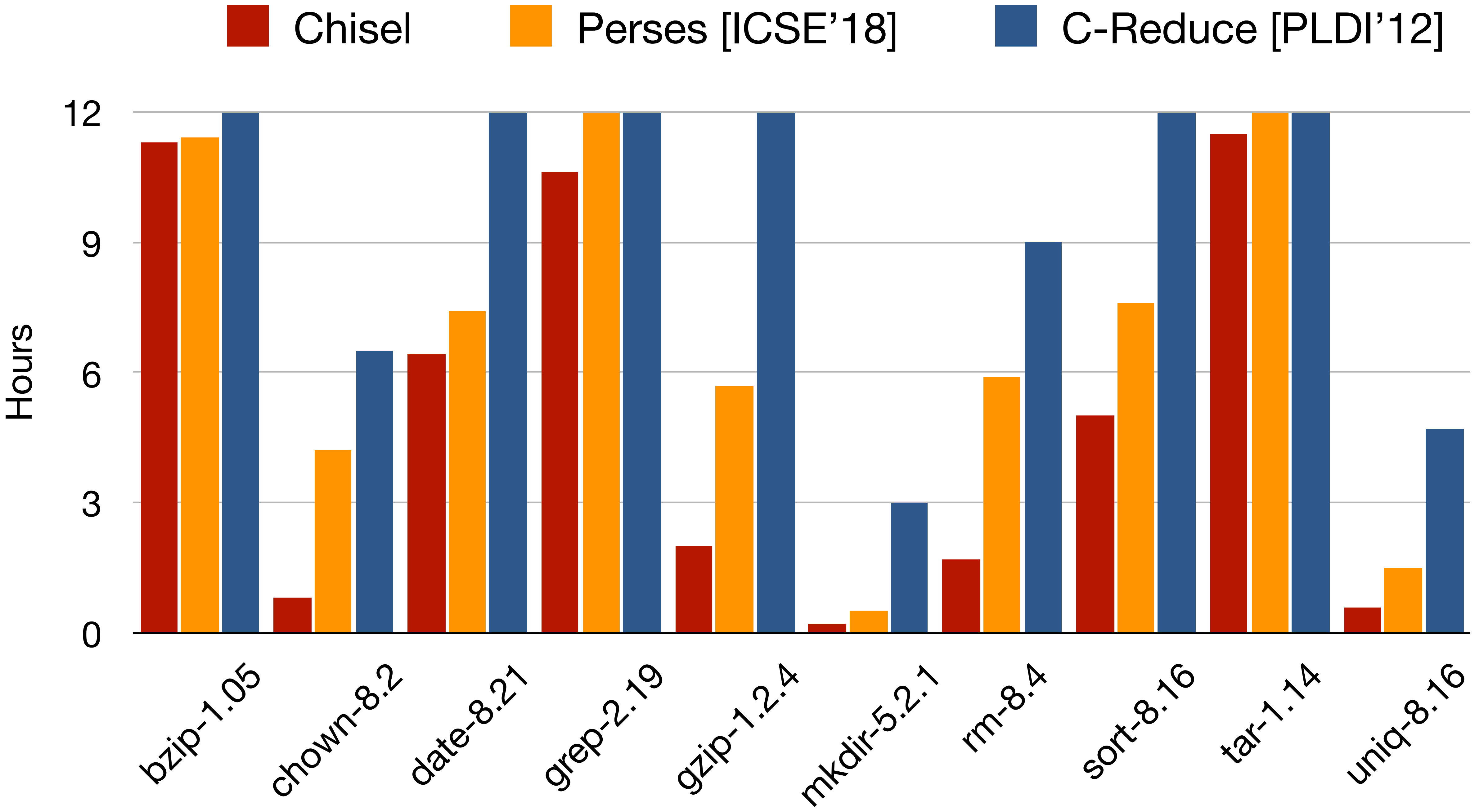
Remove 4 and 2 CVEs in undesired and desired functionalities.
4 CVEs are not easily fixable by reduction (e.g., race condition).

		#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	✓	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
tar-1.14	✓	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
uniq-8.16	✗	349	[illegible]	(69%)	60	1	(98%)
Total		6,752	2,285	(66%)	5,298	243	(95%)

Reduced potential attack surface

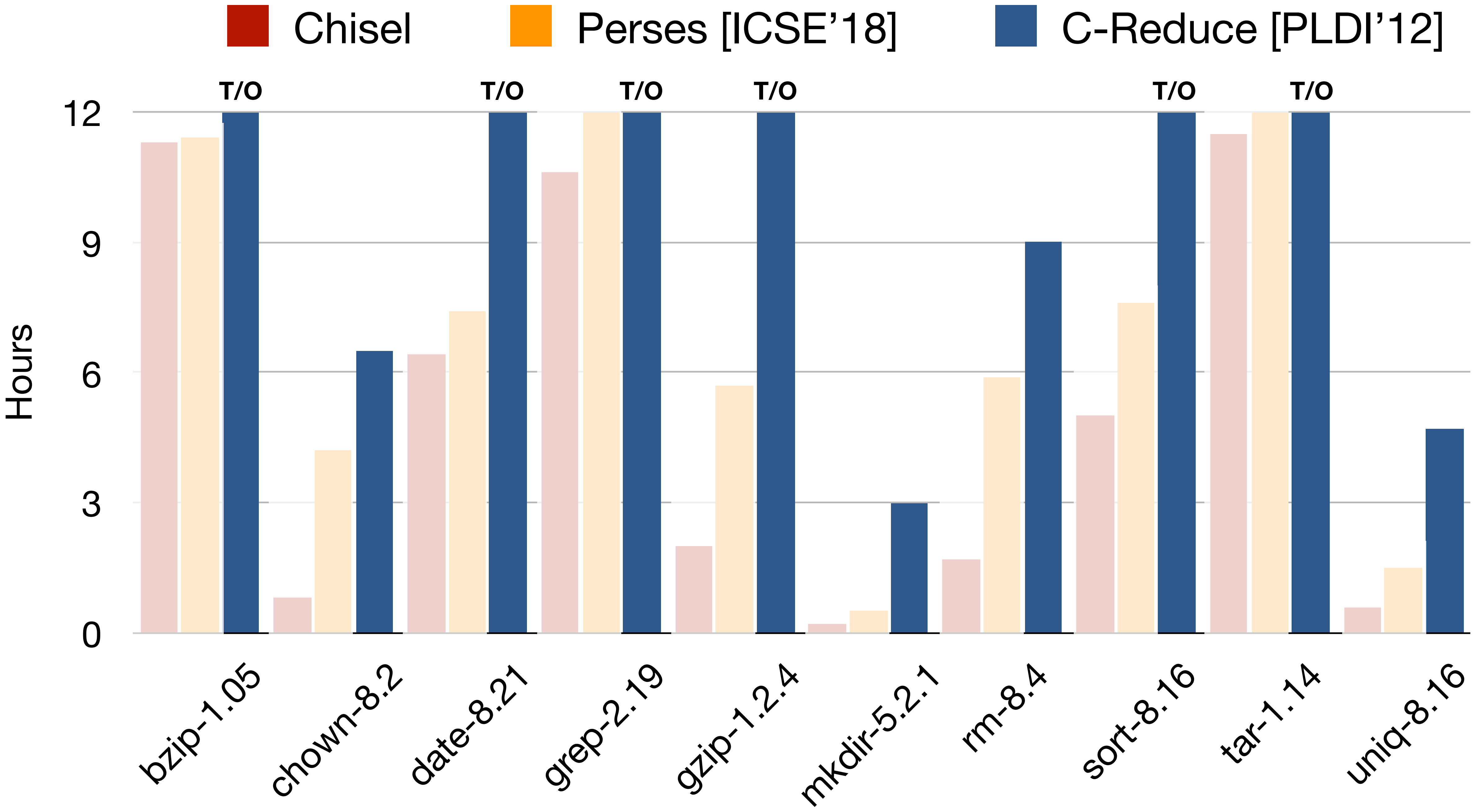
Make it feasible for manual alarm inspection

Reduction Time



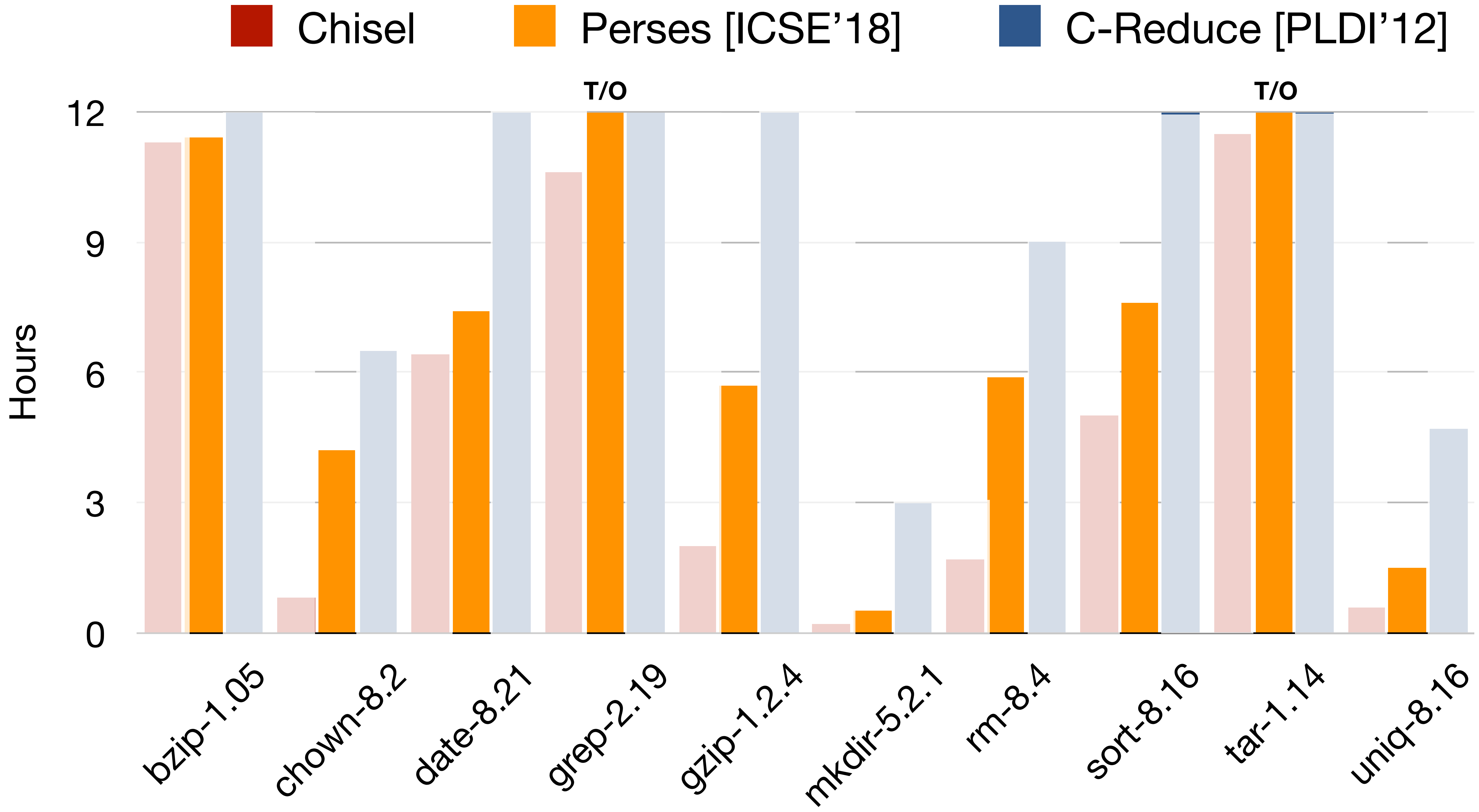
Reduction Time

Line-based reducer ran out of time for 6 programs



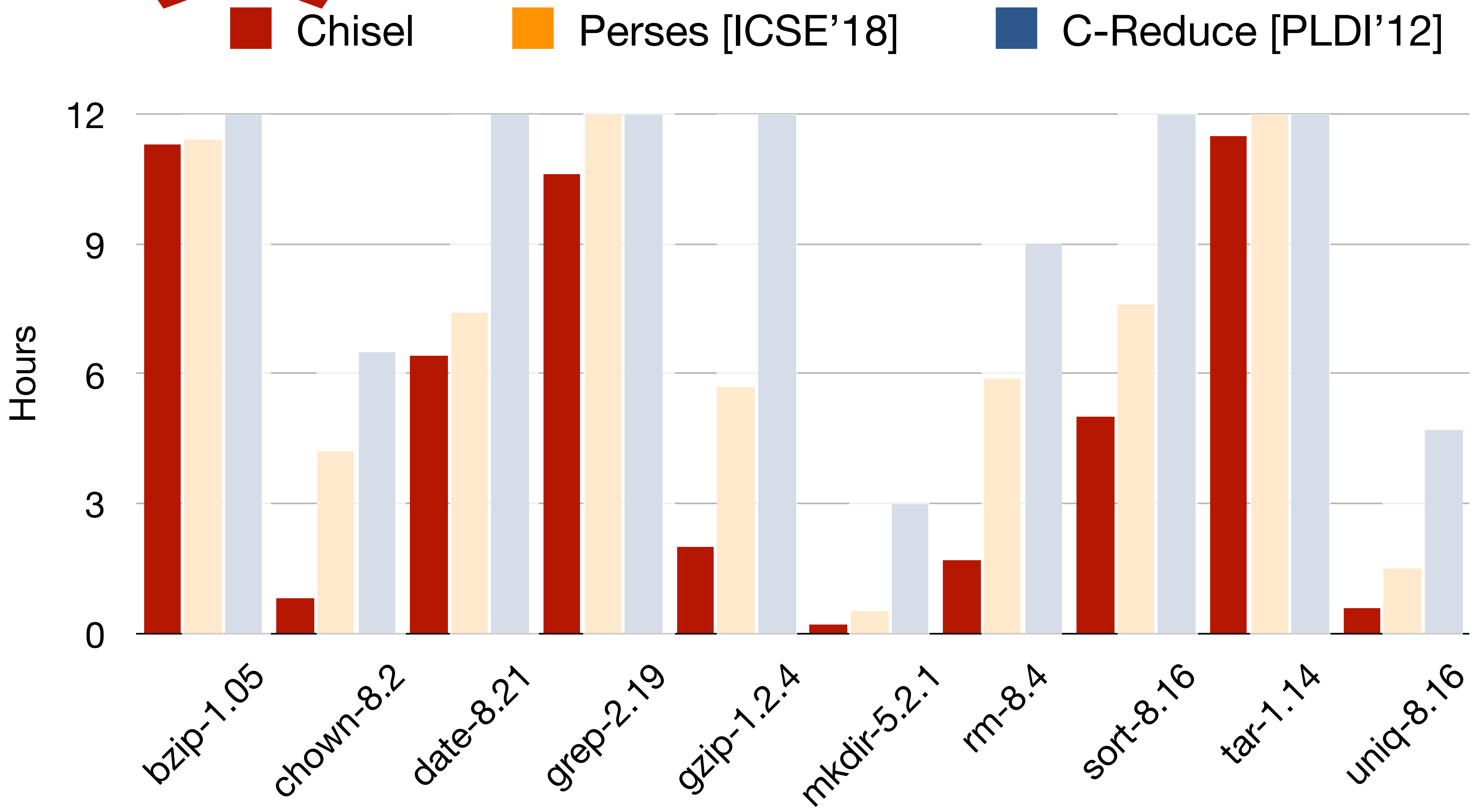
Reduction Time

Grammar-based reducer ran out of time for 2 programs



Reduction Time

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



Conclusion

- **Chisel**: automated software debloating system
  - **tractable search** via learning-guided delta debugging
  - **security hardening** by removing undesired features
  - **robustness** via static & dynamic analyses
  - https://github.com/aspire-project/chisel
- **In the paper**,
  - reduction algorithm details
  - learning a debloating policy
  - engineering issues and design choices

Acknowledgment: Total Platform Cyber Protection (TPCP)