소프트웨어 오류 자동 수정 기법

Hakjoo Oh

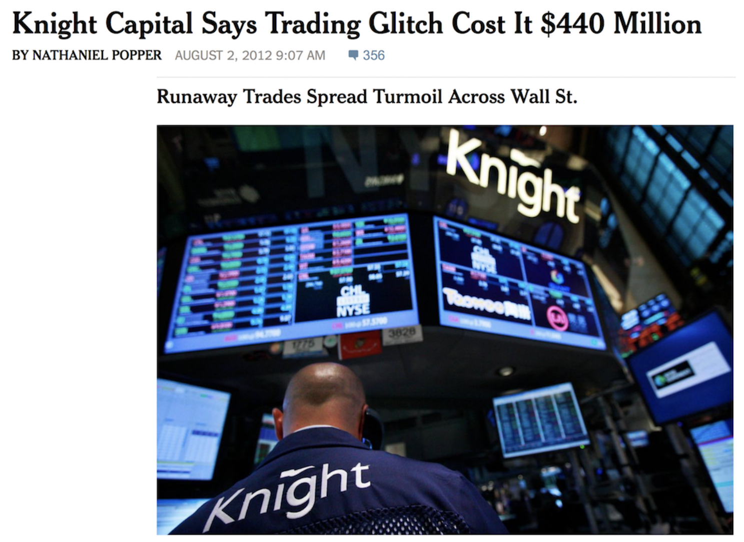
Korea University

Dec 4, 2018

(co-work with Junhee Lee and Seongjoon Hong)

소프트웨어 오류 문제

- 사회 각 영역에서 더욱 심각해지고 있는 소프트웨어 오류 문제

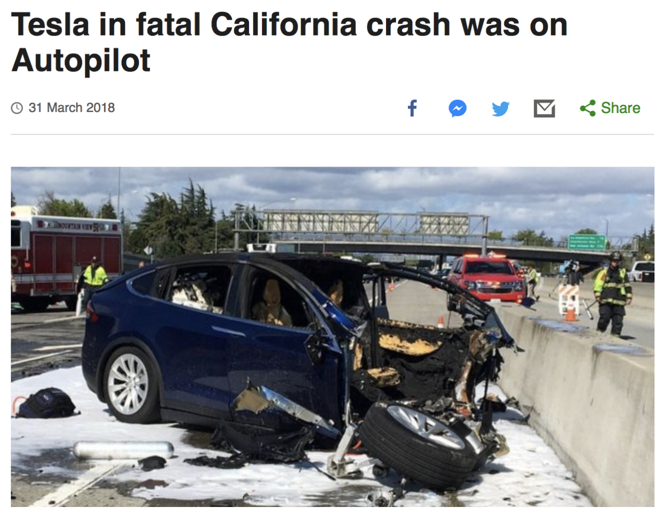


Knight Capital Says Trading Glitch Cost It $440 Million

BY NATHANIEL POPPER AUGUST 2, 2012 9:07 AM 356

Runaway Trades Spread Turmoil Across Wall St.

금융 소프트웨어 결함 (2012)

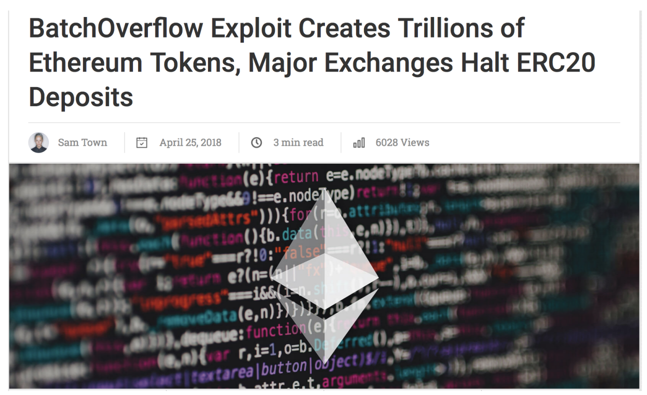


Tesla in fatal California crash was on Autopilot

31 March 2018

Share

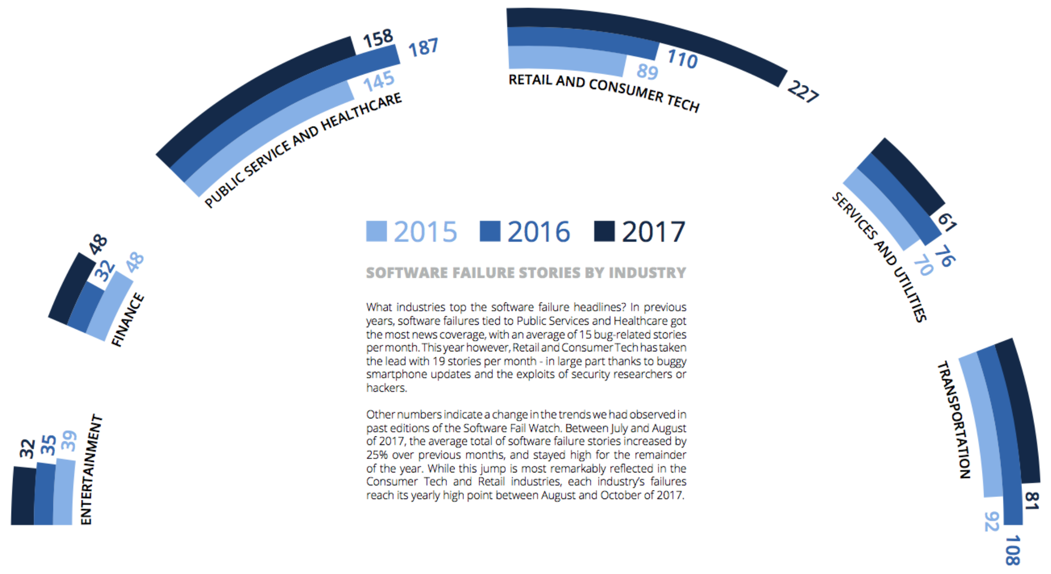
인공지능 소프트웨어 결함 (2017)



BatchOverflow Exploit Creates Trillions of Ethereum Tokens, Major Exchanges Halt ERC20 Deposits

Sam Town | April 25, 2018 | 3 min read | 6028 Views

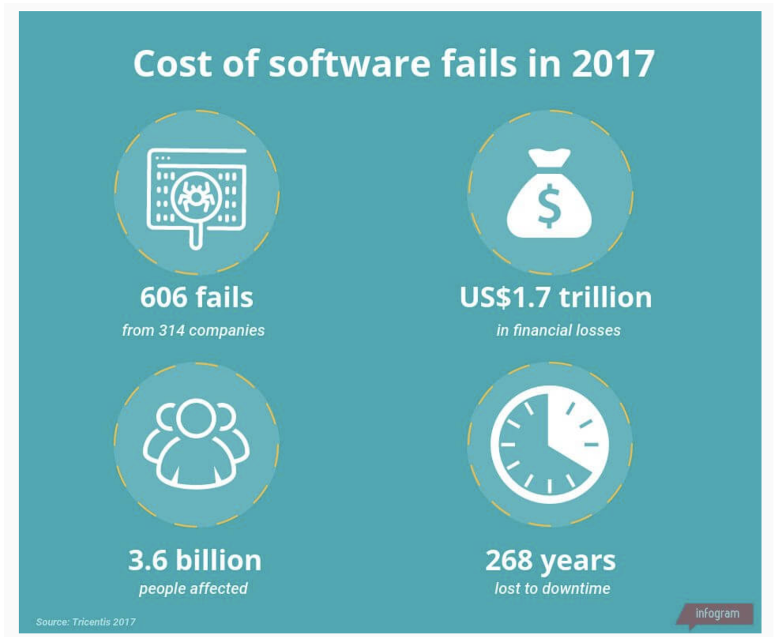
블록체인 소프트웨어 결함 (2018)



SOFTWARE FAILURE STORIES BY INDUSTRY

What industries top the software failure headlines? In previous years, software failures tied to Public Services and Healthcare got the most news coverage, with an average of 15 bug-related stories per month. This year however, Retail and Consumer Tech has taken the lead with 19 stories per month - in large part thanks to buggy smartphone updates and the exploits of security researchers or hackers.

Other numbers indicate a change in the trends we had observed in past editions of the Software Fail Watch. Between July and August of 2017, the average total of software failure stories increased by 25% over previous months, and stayed high for the remainder of the year. While this jump is most remarkably reflected in the Consumer Tech and Retail industries, each industry's failures reach its yearly high point between August and October of 2017.



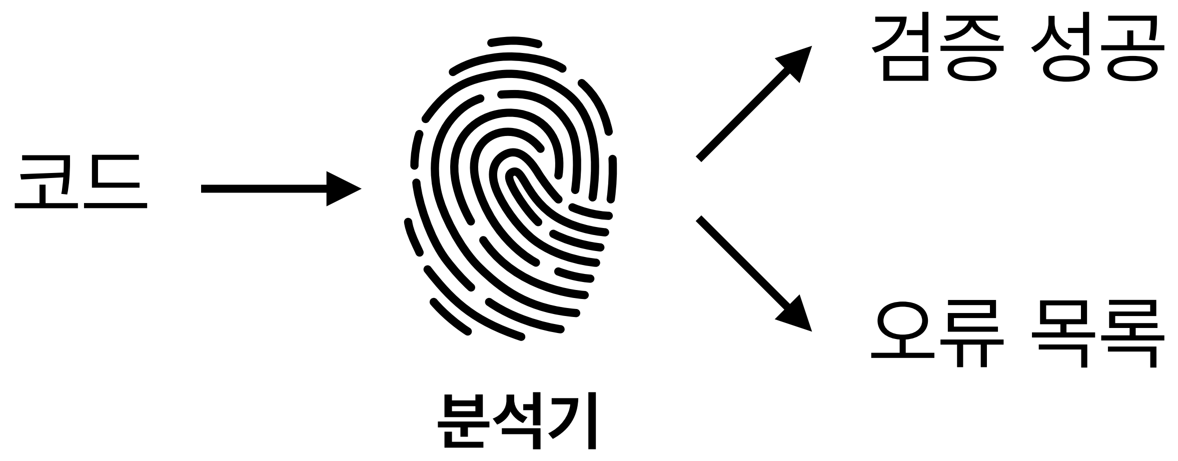
Software fail watch (5th ed) 2017

연구 방향

- Q) 어떻게 안전한 소프트웨어를 손쉽게 만들것인가?
- A) 소프트웨어 자동 **분석**, **패치**, **합성** 기술

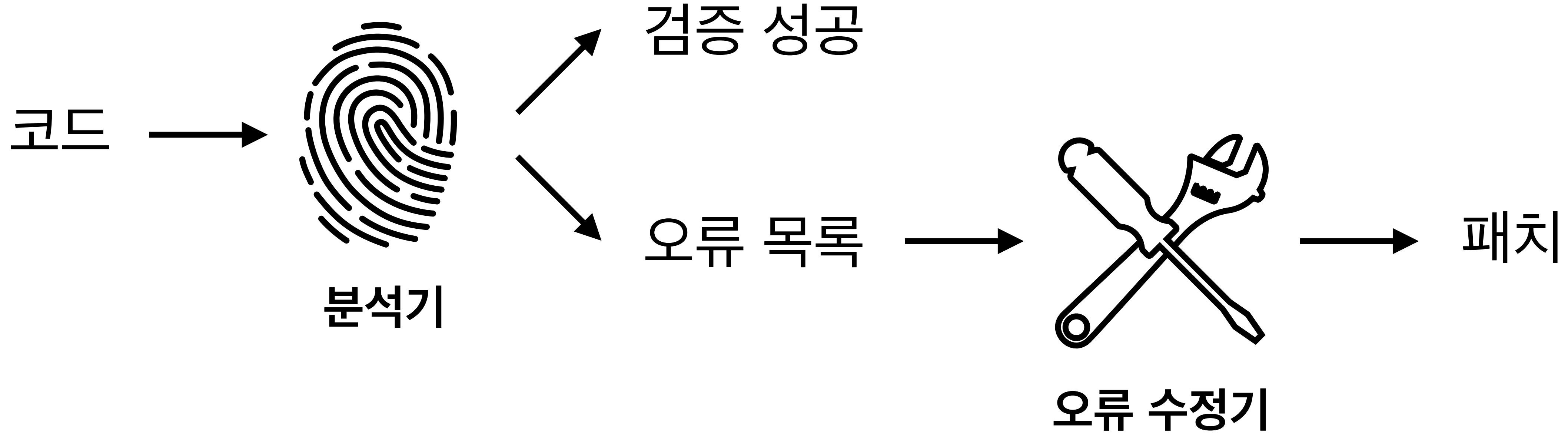
연구 방향

- Q) 어떻게 안전한 소프트웨어를 손쉽게 만들것인가?
- A) 소프트웨어 자동 **분석**, **패치**, **합성** 기술



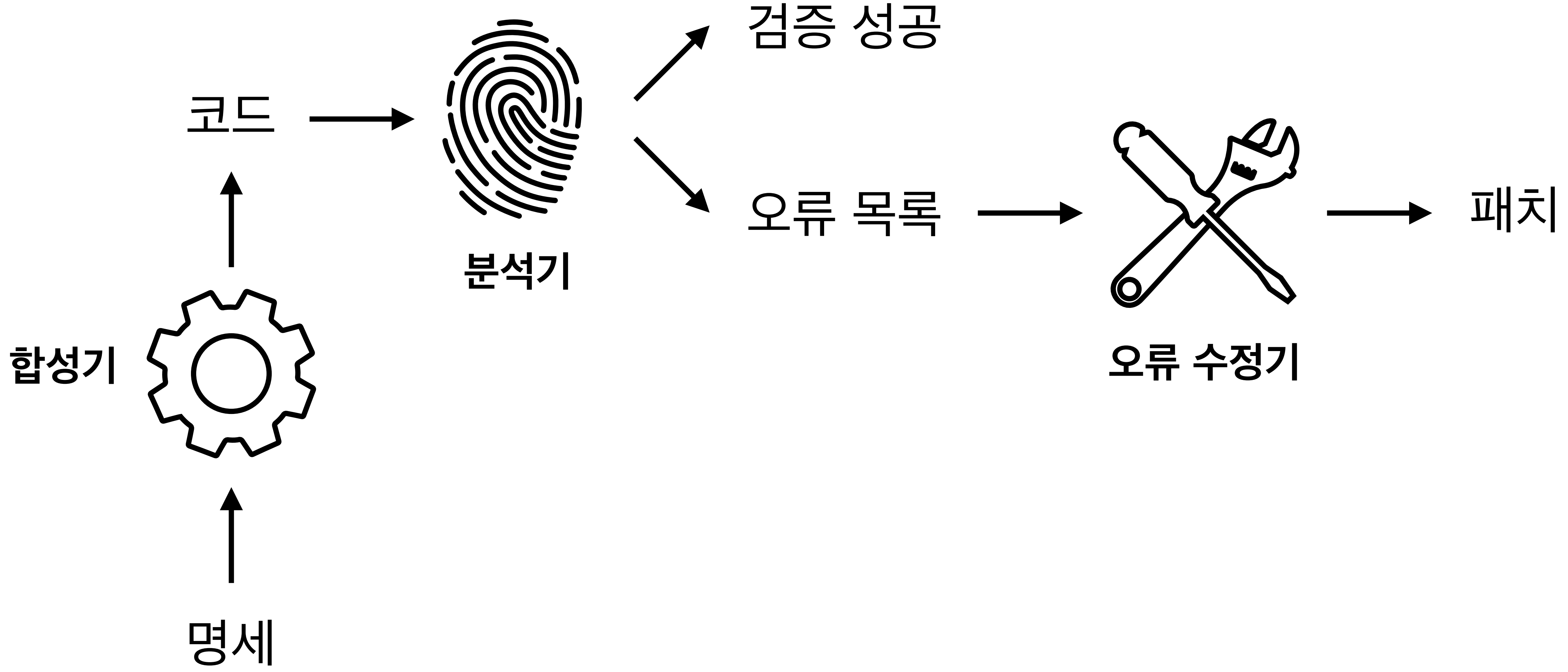
연구 방향

- Q) 어떻게 안전한 소프트웨어를 손쉽게 만들것인가?
- A) 소프트웨어 자동 **분석**, **패치**, **합성** 기술



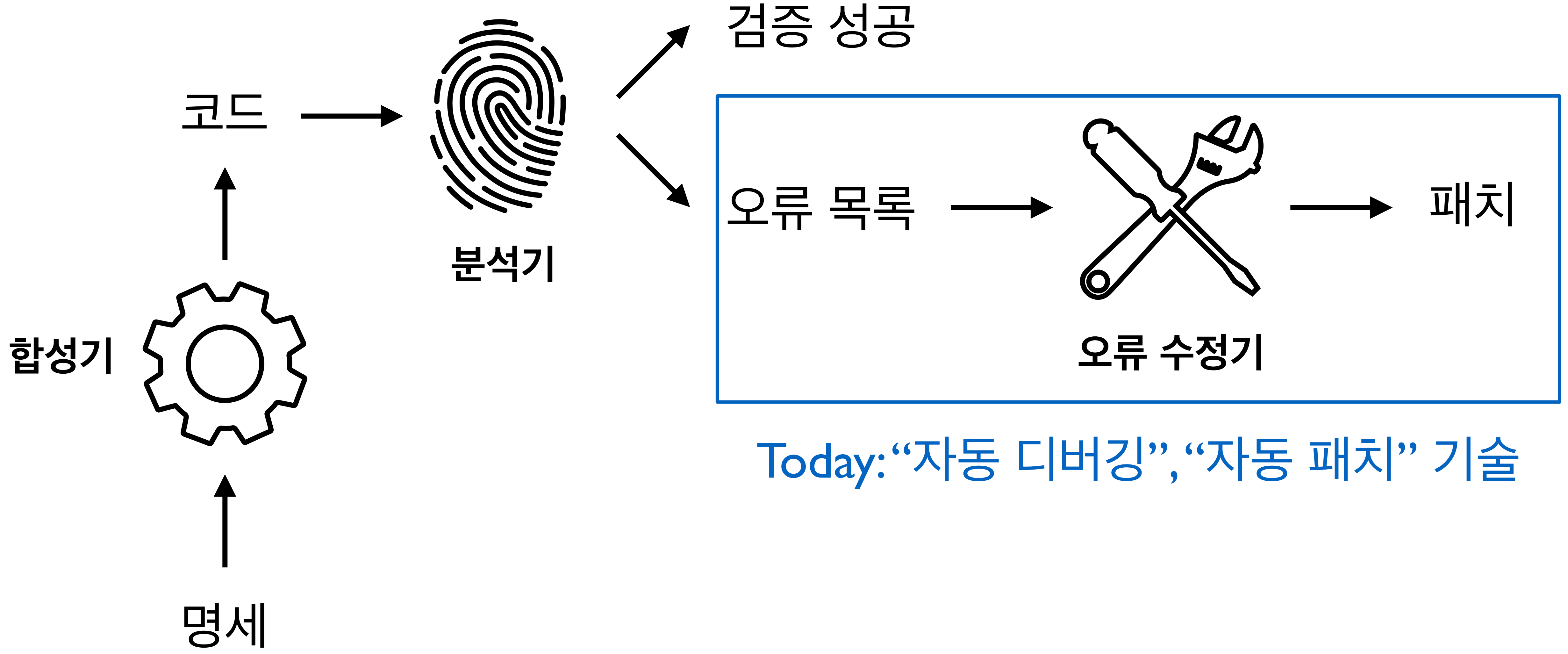
연구 방향

- Q) 어떻게 안전한 소프트웨어를 손쉽게 만들것인가?
- A) 소프트웨어 자동 **분석**, **패치**, **합성** 기술



연구 방향

- Q) 어떻게 안전한 소프트웨어를 손쉽게 만들것인가?
- A) 소프트웨어 자동 **분석**, **패치**, **합성** 기술



Today: "자동 디버깅", "자동 패치" 기술

왜 필요한가?

- 소프트웨어 개발에서 디버깅은 전체 시간의 절반을 차지
  - 상용 소프트웨어 오류 수정에 평균 200일 소요[1)]
  - 오류/취약점은 해마다 증가 개수: e.g., CVE 등록수 4,000('10년), 6,000('15년)
- 다른 개발 단계에 비해 자동화된 도구 지원이 가장 적음
  - cf) 소프트웨어 오류 탐지 분야는 지난 30여년간 눈부신 발전
  - 개발자에 전적으로 의존할수 밖에 없지만 가장 어렵고 부담스러운 단계

1) Kim and Whitehead. How long did it take to fix bugs? MSR 2006

실제 사례 (Linux Kernel)

```
in = malloc(1);
out = malloc(1);
... // use in, out
free(out);
free(in);

in = malloc(2);
if (in == NULL) {

  goto err;
}

out = malloc(2);
if (out == NULL) {
  free(in);

  goto err;
}
... // use in, out
err:
  free(in);
  free(out);
  return;
```

실제 사례 (Linux Kernel)

```
in = malloc(1);
out = malloc(1);
... // use in, out
free(out);
free(in);

in = malloc(2);
if (in == NULL) {

  goto err;
}

out = malloc(2);
if (out == NULL) {
  free(in);

  goto err;
}
... // use in, out
err:
  free(in);
  free(out);
  return;
```

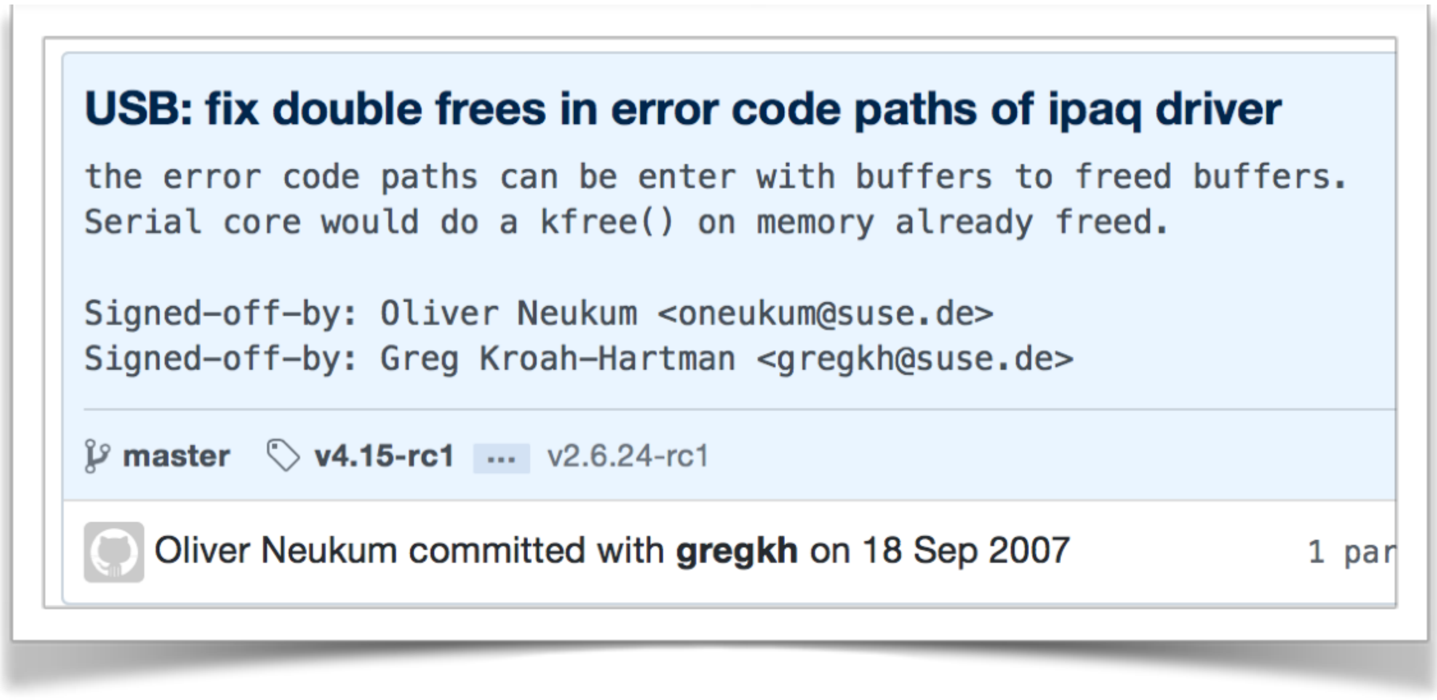
double-free

실제 사례 (Linux Kernel)

```
in = malloc(1);
out = malloc(1);
... // use in, out
free(out);
free(in);

in = malloc(2);
if (in == NULL) {

  goto err;
}

out = malloc(2);
if (out == NULL) {
  free(in);

  goto err;
}
... // use in, out
err:
  free(in);
  free(out);
  return;
```

double-free

실제 사례 (Linux Kernel)



USB: fix double frees in error code paths of ipaq driver

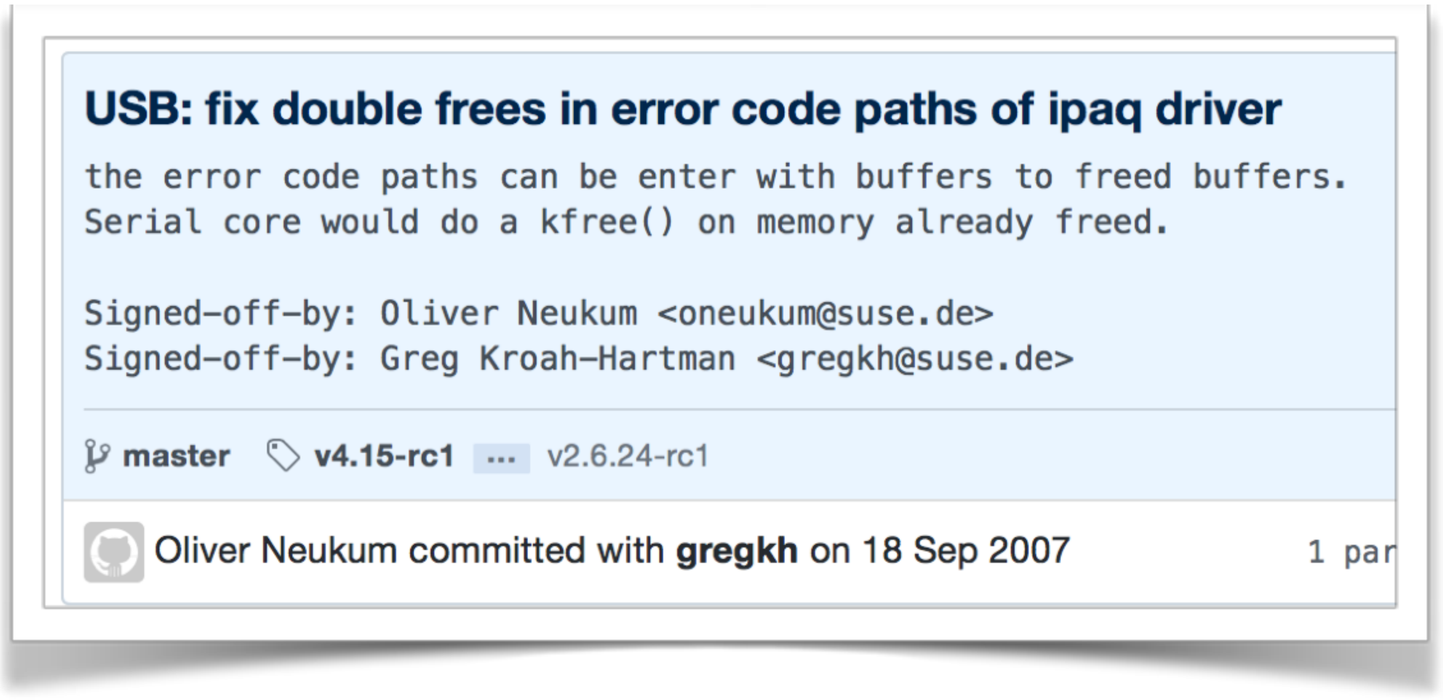
```
the error code paths can be enter with buffers to freed buffers.
Serial core would do a kfree() on memory already freed.

Signed-off-by: Oliver Neukum <oneukum@suse.de>
Signed-off-by: Greg Kroah-Hartman <gregkh@suse.de>
```

master v4.15-rc1 ... v2.6.24-rc1

Oliver Neukum committed with **gregkh** on 18 Sep 2007 1 par

```
in = malloc(1);
out = malloc(1);
... // use in, out
free(out);
free(in);

in = malloc(2);
if (in == NULL) {
  out = NULL;
  goto err;
}

out = malloc(2);
if (out == NULL) {
  free(in);
  in = NULL;
  goto err;
}
... // use in, out
err:
  free(in);
  free(out);
  return;
```

실제 사례 (Linux Kernel)



USB: fix double frees in error code paths of ipaq driver

```
the error code paths can be enter with buffers to freed buffers.
Serial core would do a kfree() on memory already freed.

Signed-off-by: Oliver Neukum <oneukum@suse.de>
Signed-off-by: Greg Kroah-Hartman <gregkh@suse.de>
```

master v4.15-rc1 … v2.6.24-rc1

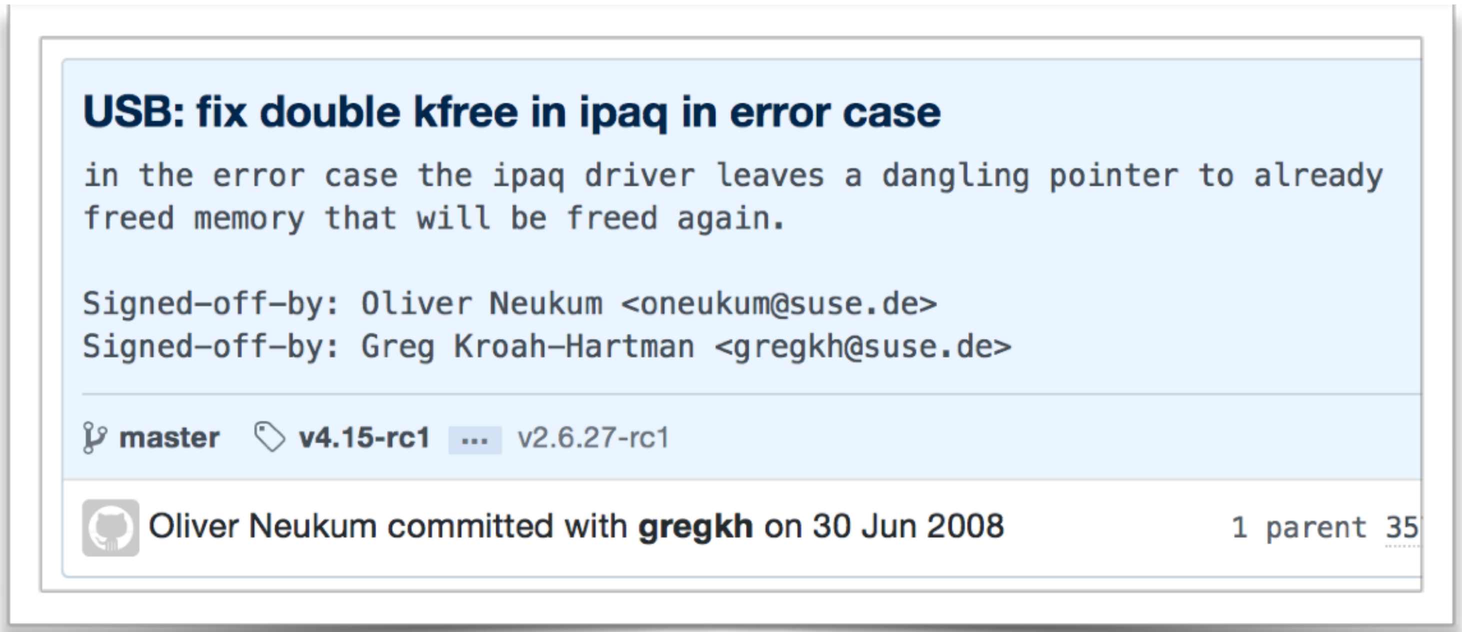
Oliver Neukum committed with **gregkh** on 18 Sep 2007 1 par

수동 디버깅의 문제 1:
오류가 사라졌는지 확신하기 어려움

```
in = malloc(1);
out = malloc(1);
... // use in, out
free(out);
free(in);

in = malloc(2);
if (in == NULL) {
  out = NULL;
  goto err;
}

out = malloc(2);
if (out == NULL) {
  free(in);
  in = NULL;
  goto err;
}
... // use in, out
err:
  free(in);
  free(out);
  return;
```

실제 사례 (Linux Kernel)

```
in = malloc(1);
out = malloc(1);
... // use in, out
// removed
free(in);

in = malloc(2);
if (in == NULL) {
  out = NULL;
  goto err;
}
free(out);
out = malloc(2);
if (out == NULL) {
  free(in);
  in = NULL;
  goto err;
}
... // use in, out
err:
  free(in);
  free(out);
  return;
```


USB: fix double kfree in ipaq in error case

```
in the error case the ipaq driver leaves a dangling pointer to already
freed memory that will be freed again.

Signed-off-by: Oliver Neukum <oneukum@suse.de>
Signed-off-by: Greg Kroah-Hartman <gregkh@suse.de>
```

master v4.15-rc1 ... v2.6.27-rc1

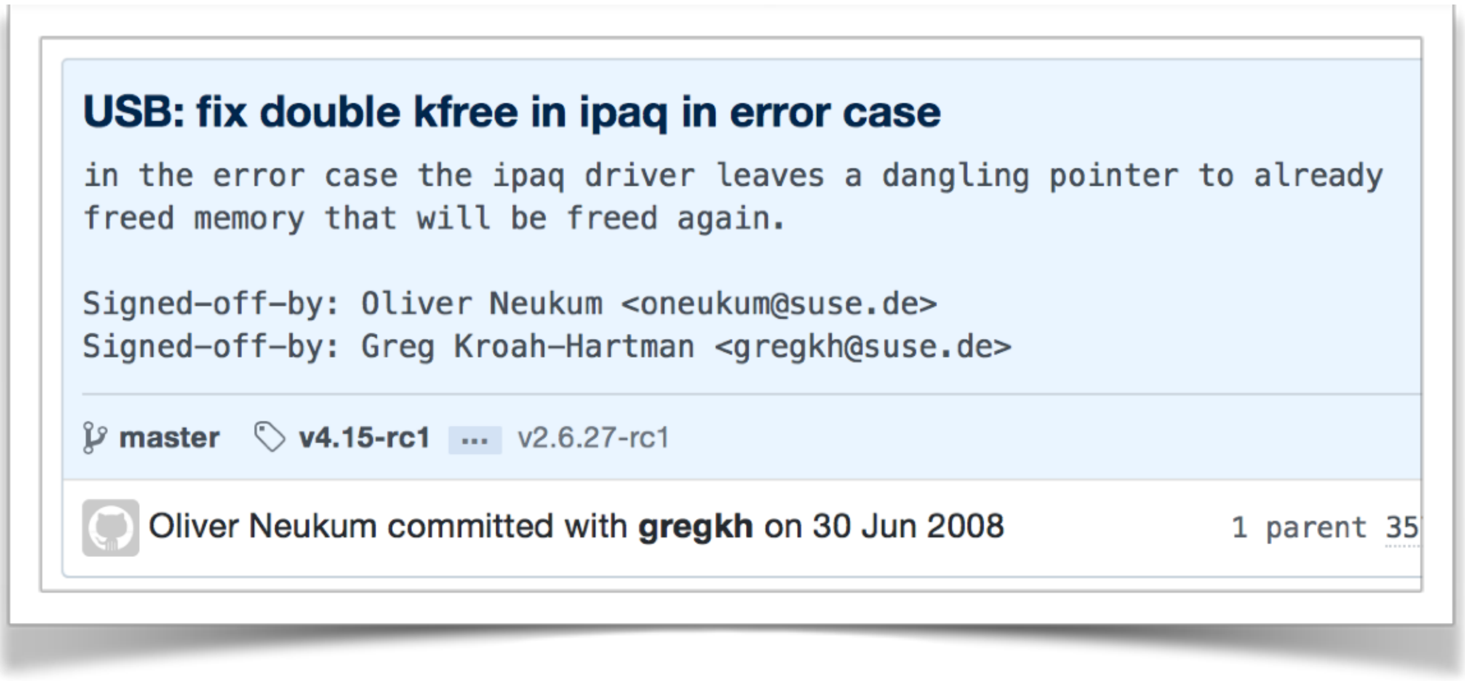
Oliver Neukum committed with **gregkh** on 30 Jun 2008 1 parent 35

실제 사례 (Linux Kernel)

수동 디버깅의 문제 2:
고치는 과정에서 새로운 오류가 발생

memory leak

```
in = malloc(1);
out = malloc(1);
... // use in, out
// removed
free(in);

in = malloc(2);
if (in == NULL) {
  out = NULL;
  goto err;
}
free(out);
out = malloc(2);
if (out == NULL) {
  free(in);
  in = NULL;
  goto err;
}
... // use in, out
err:
  free(in);
  free(out);
  return;
```



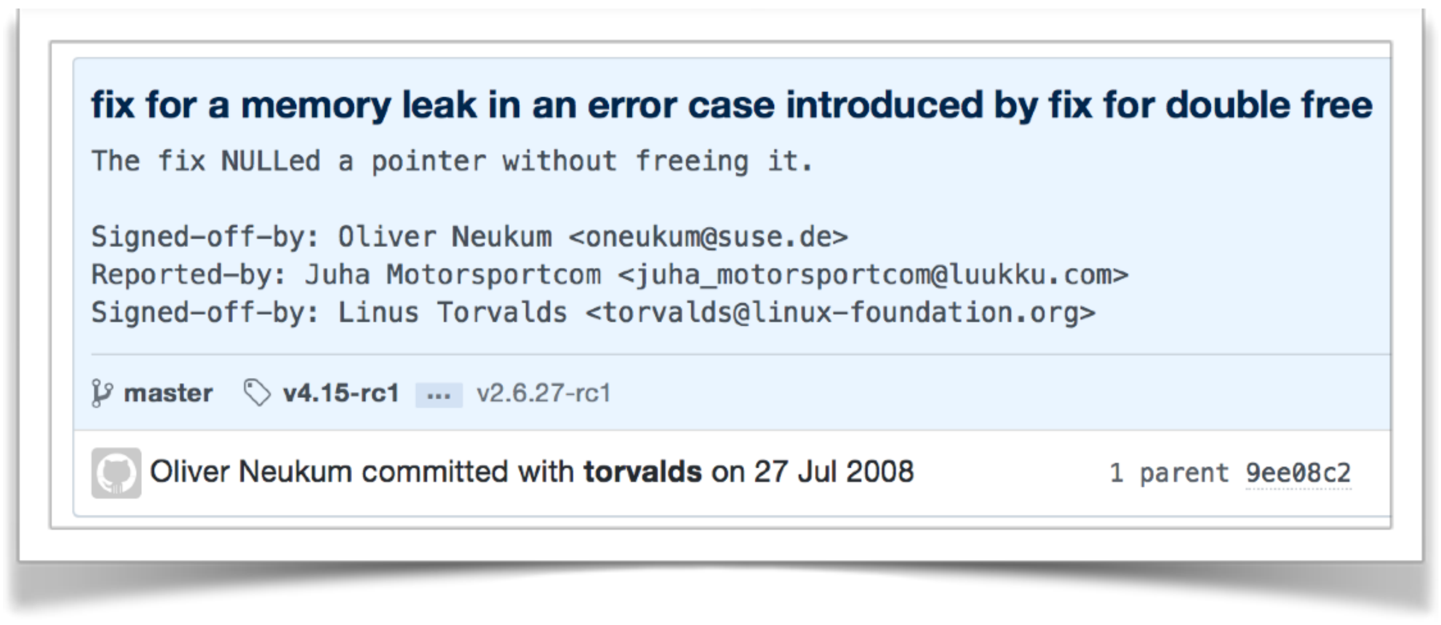
USB: fix double kfree in ipaq in error case

```
in the error case the ipaq driver leaves a dangling pointer to already
freed memory that will be freed again.

Signed-off-by: Oliver Neukum <oneukum@suse.de>
Signed-off-by: Greg Kroah-Hartman <gregkh@suse.de>
```

master v4.15-rc1 ... v2.6.27-rc1

Oliver Neukum committed with **gregkh** on 30 Jun 2008 1 parent 35

실제 사례 (Linux Kernel)


fix for a memory leak in an error case introduced by fix for double free

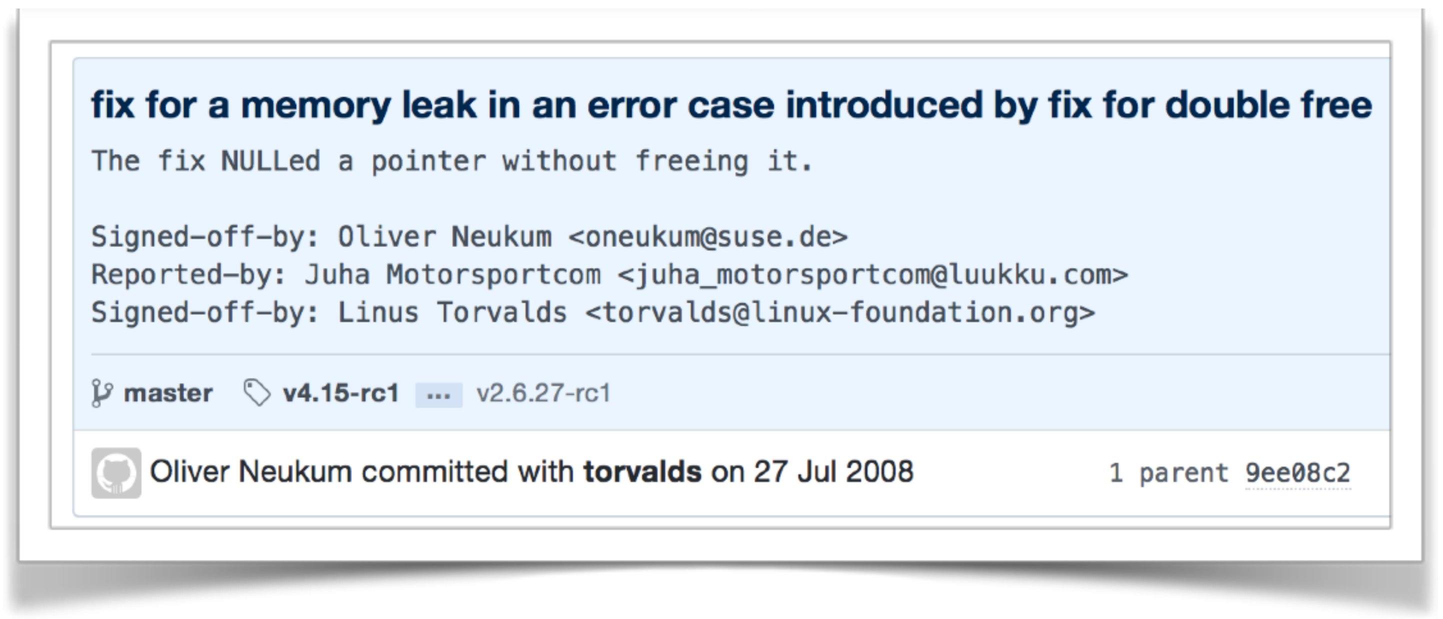
```
The fix NULLed a pointer without freeing it.

Signed-off-by: Oliver Neukum <oneukum@suse.de>
Reported-by: Juha Motorsportcom <juha_motorsportcom@luukku.com>
Signed-off-by: Linus Torvalds <torvalds@linux-foundation.org>
```

master v4.15-rc1 ... v2.6.27-rc1

Oliver Neukum committed with **torvalds** on 27 Jul 2008 1 parent 9ee08c2

```
in = malloc(1);
out = malloc(1);
... // use in, out
free(out);
free(in);
out = NULL;
in = malloc(2);
if (in == NULL) {
  out = NULL;
  goto err;
}
// removed
out = malloc(2);
if (out == NULL) {
  free(in);
  in = NULL;
  goto err;
}
... // use in, out
err:
  free(in);
  free(out);
  return;
```

실제 사례 (Linux Kernel)


fix for a memory leak in an error case introduced by fix for double free

```
The fix NULLed a pointer without freeing it.

Signed-off-by: Oliver Neukum <oneukum@suse.de>
Reported-by: Juha Motorsportcom <juha_motorsportcom@luukku.com>
Signed-off-by: Linus Torvalds <torvalds@linux-foundation.org>
```

master v4.15-rc1 … v2.6.27-rc1

Oliver Neukum committed with **torvalds** on 27 Jul 2008 1 parent 9ee08c2

수동 디버깅의 문제 3: 수정된 코드가 복잡

```
in = malloc(1);
out = malloc(1);
... // use in, out
free(out);
free(in);
out = NULL;
in = malloc(2);
if (in == NULL) {
  out = NULL;
  goto err;
}
// removed
out = malloc(2);
if (out == NULL) {
  free(in);
  in = NULL;
  goto err;
}
... // use in, out
err:
  free(in);
  free(out);
  return;
```

메모리 오류 자동 수정기

```
in = malloc(1);
out = malloc(1);
... // use in, out
free(out);
free(in);

in = malloc(2);
if (in == NULL) {

  goto err;
}

out = malloc(2);
if (out == NULL) {
  free(in);

  goto err;
}
... // use in, out
err:
  free(in);
  free(out);
  return;
```

패치 자동 생성

```
in = malloc(1);
out = malloc(1);
... // use in, out
// removed
free(in);

in = malloc(2);
if (in == NULL) {

  goto err;
}
free(out);
out = malloc(2);
if (out == NULL) {
  // removed

  goto err;
}
... // use in, out
err:
  free(in);
  free(out);
  return;
```

메모리 오류 자동 수정기

```
in = malloc(1);
out = malloc(1);
... // use in, out
free(out);
free(in);

in = malloc(2);
if (in == NULL) {

  goto err;
}

out = malloc(2);
if (out == NULL) {
  free(in);

  goto err;
}
... // use in, out
err:
  free(in);
  free(out);
  return;
```

패치 자동 생성

```
in = malloc(1);
out = malloc(1);
... // use in, out
// removed
free(in);

in = malloc(2);
if (in == NULL) {

  goto err;
}
free(out);
out = malloc(2);
if (out == NULL) {
  // removed

  goto err;
}
... // use in, out
err:
  free(in);
  free(out);
  return;
```

수동 디버깅의 문제 해결:
1. 대상 오류가 반드시 제거됨
2. 새로운 오류가 발생하지 않음
3. 간결한 패치 (최소한의 변경)

=> **수학적 보장.**
추가적인 리뷰 불필요.

메모리 해제 오류

- 메모리 관리를 수동으로 해야하는 언어(e.g., C/C++) 발생
  - Memory-leak (CWE-401): 메모리를 너무 늦게 해제
  - Use-after-free (CWE-416): 메모리를 너무 빨리 해제
  - Double-free (CWE-415): 메모리를 여러번 해제

Memory-Leak

```
p = malloc(1);
...
return;
```

Use-After-Free

```
p = malloc(1);
...
free(p);
...
use(p);
```

Double-Free

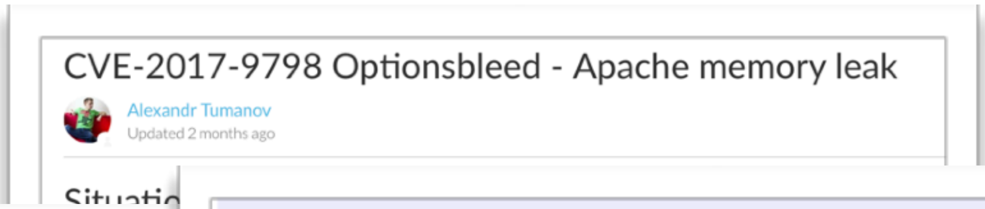
```
p = malloc(1);
...
free(p);
...
free(p);
```

메모리 해제 오류

- C/C++ 프로그램에서 가장 골칫거리중 하나

Repository	#commits	ML	DF	UAF	Total	*-overflow
linux	721,119	3,740	821	1,986	**6,363**	5,092
openssl	21,009	220	36	12	**264**	61
numpy	17,008	58	2	2	**59**	53
php	105,613	1,129	148	197	**1,449**	649
git	49,475	350	19	95	**442**	258

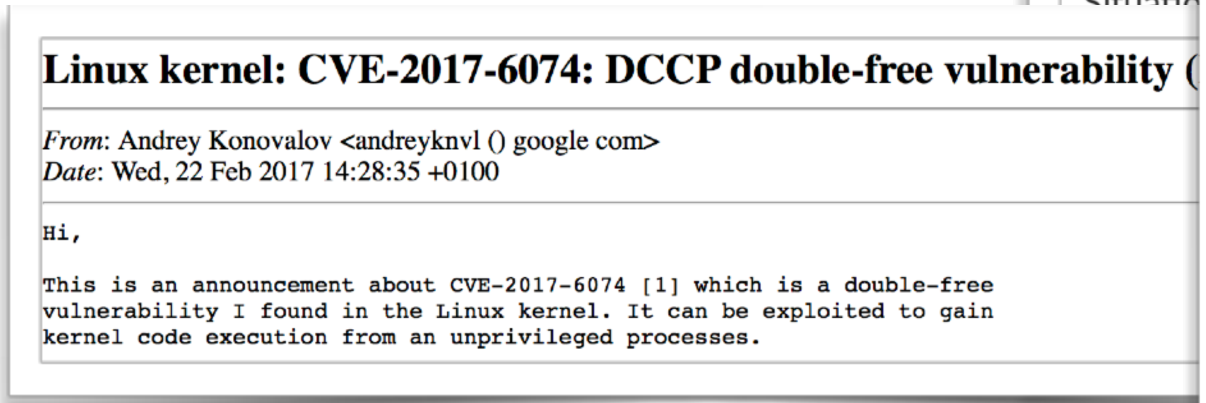
- 소프트웨어 결함의 주요 원인이지만 정확한 수정이 까다로움


CVE-2017-9798 Optionsbleed - Apache memory leak

Alexandr Tumanov

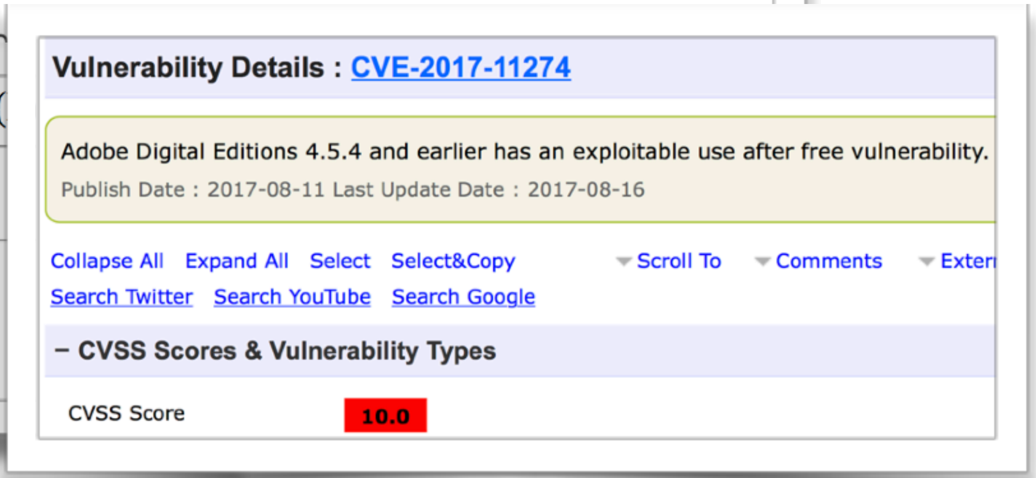
Updated 2 months ago

Situatio


Linux kernel: CVE-2017-6074: DCCP double-free vulnerability (

From: Andrey Konovalov <andreyknvl () google com>
Date: Wed, 22 Feb 2017 14:28:35 +0100

```
Hi,

This is an announcement about CVE-2017-6074 [1] which is a double-free
vulnerability I found in the Linux kernel. It can be exploited to gain
kernel code execution from an unprivileged processes.
```


Vulnerability Details : CVE-2017-11274

Adobe Digital Editions 4.5.4 and earlier has an exploitable use after free vulnerability.

Publish Date : 2017-08-11 Last Update Date : 2017-08-16

– CVSS Scores & Vulnerability Types

CVSS Score 10.0

FSE'18

MemFix

- Automatically repairs deallocation errors
  - memory-leak, double-free and use-after-free
- Key features
  - sound: generated patch is guaranteed to be correct
  - safe: no new errors are introduced
- Approach: Static Analysis + Exact Cover Problem

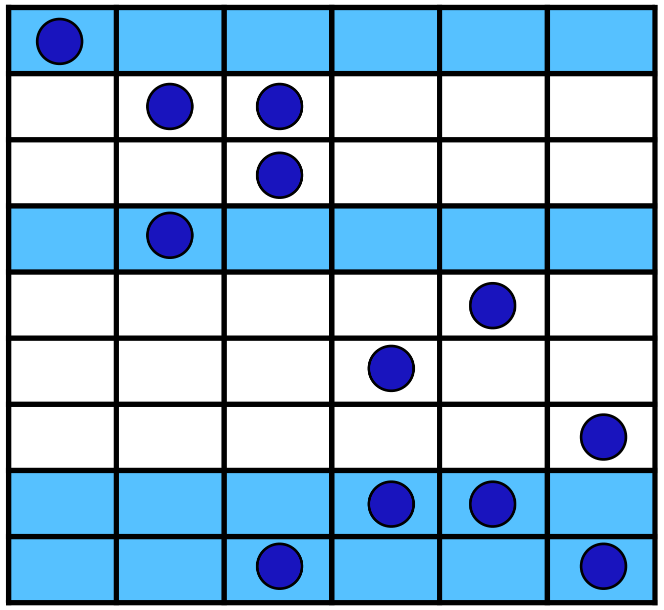
Key Insight

```c
out = malloc(1);
in = malloc(1);
… // use in, out
free(out);
free(in);

in = malloc(2);
if(in == NULL) {

    goto err;
}

out = malloc(2);
if(out == NULL) {
    free(in);

    goto err;
}
… // use in, out
err:
free(in);
free(out);
```

$\longrightarrow$

Find a set of free-statements

|||



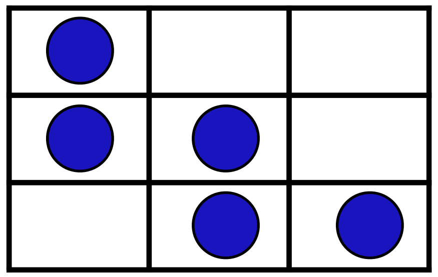
Solve an Exact Cover Problem

```c
out = malloc(1);
in = malloc(1);
… // use in, out
// -
free(in);

in = malloc(2);
if(in == NULL) {

    goto err;
}
free(out); // +
out = malloc(2);
if(out == NULL) {
    // -

    goto err;
}
… // use in, out
err:
free(in);
free(out);
```

Example: Double Free

```
p = malloc(1);
if(…) {
    q = malloc(2);

}
else
    q = p;
… // use q
free(p);
free(q);
```

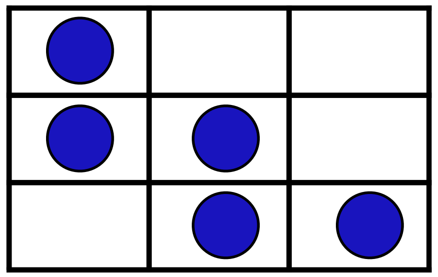
|||

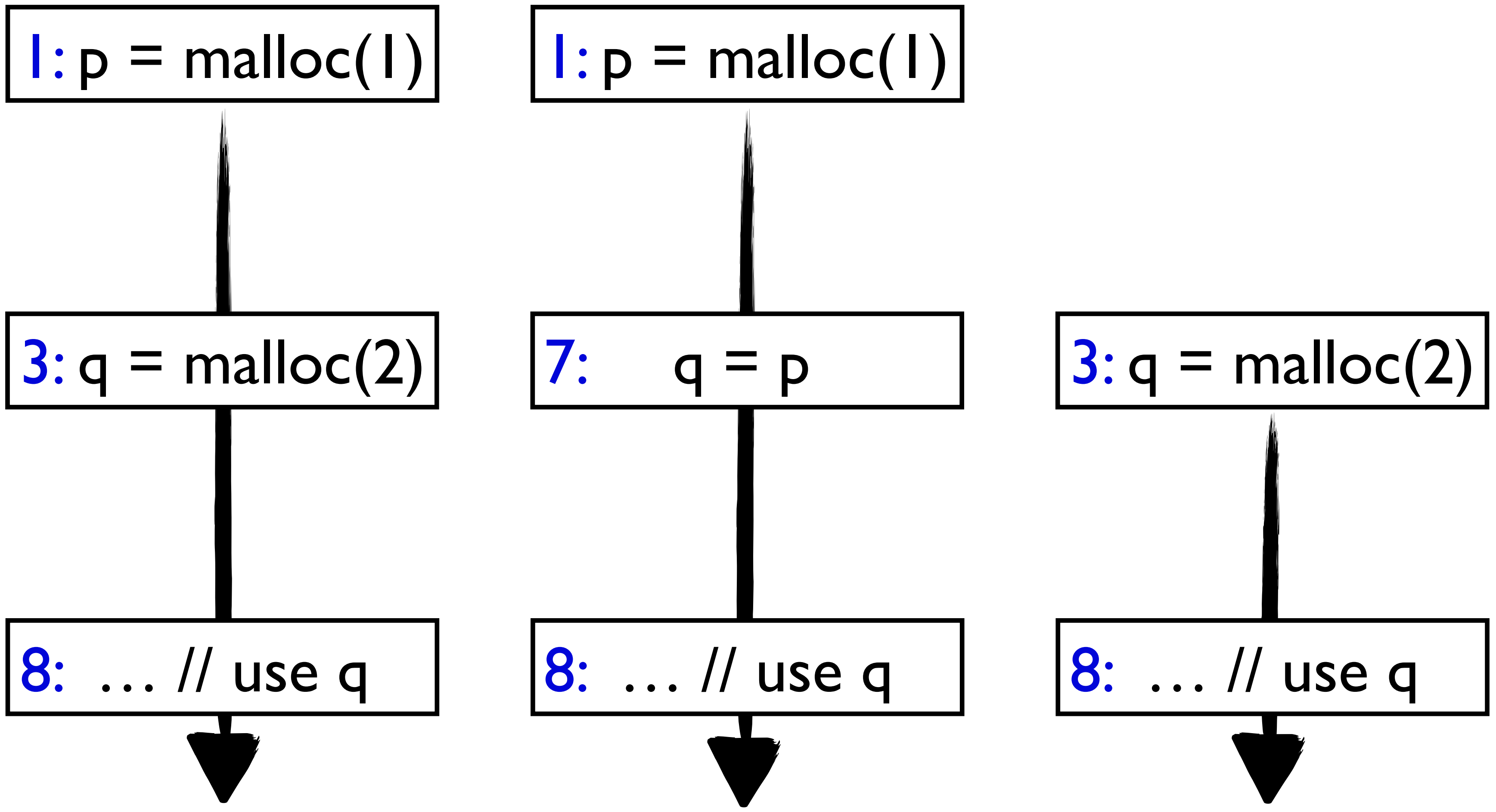


Enumerate All Object Traces

```
1   p = malloc(1);
2   if(…) {
3       q = malloc(2);
4
5   }
6   else
7       q = p;
8   … // use q
9   free(p);
10  free(q);
```

≡



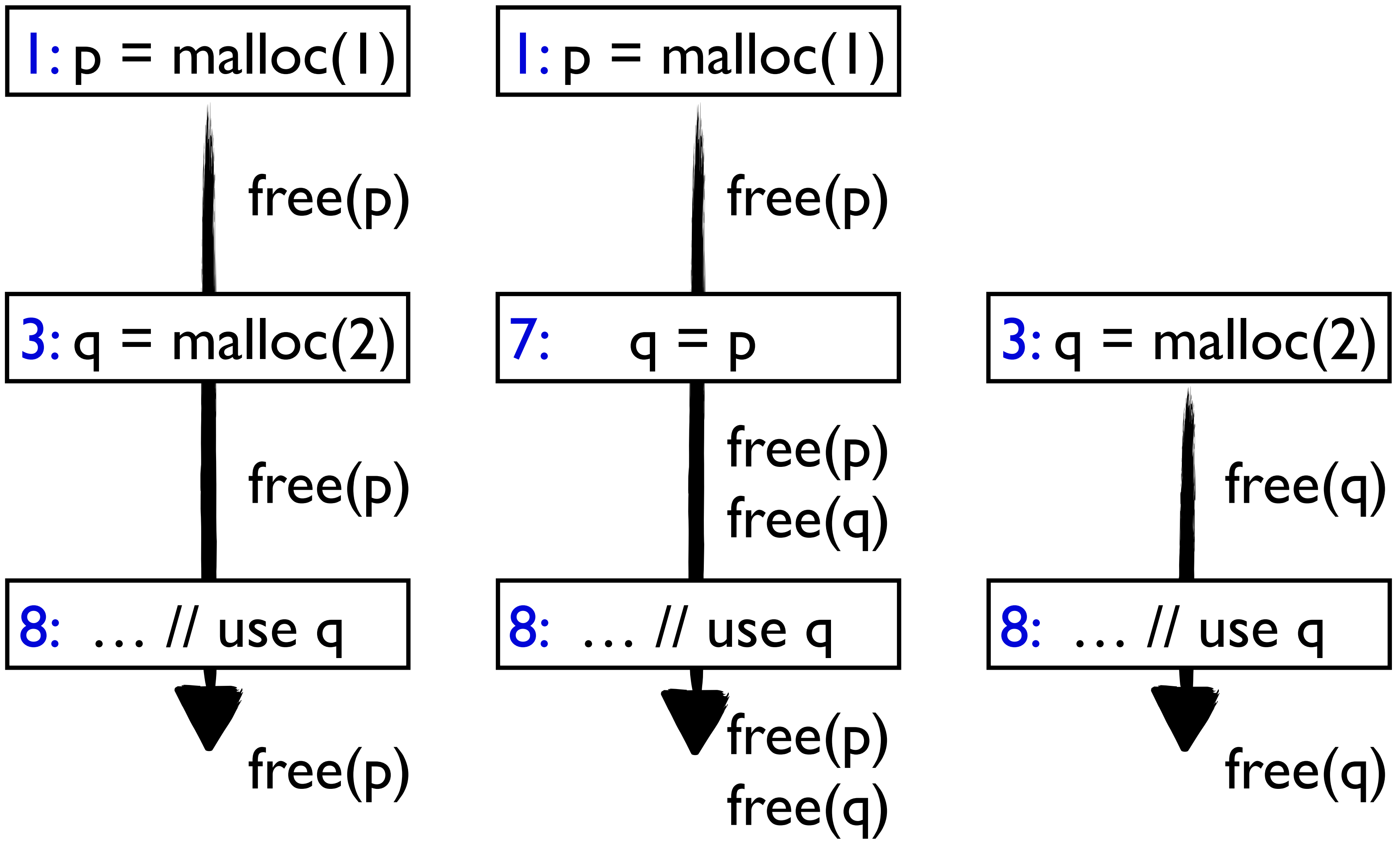


Object traces

Find Safe Patches for Each Trace

```
1   p = malloc(1);
2   if(…) {
3       q = malloc(2);
4
5   }
6   else
7       q = p;
8   … // use q
9   free(p);
10  free(q);
```

|||



Object traces

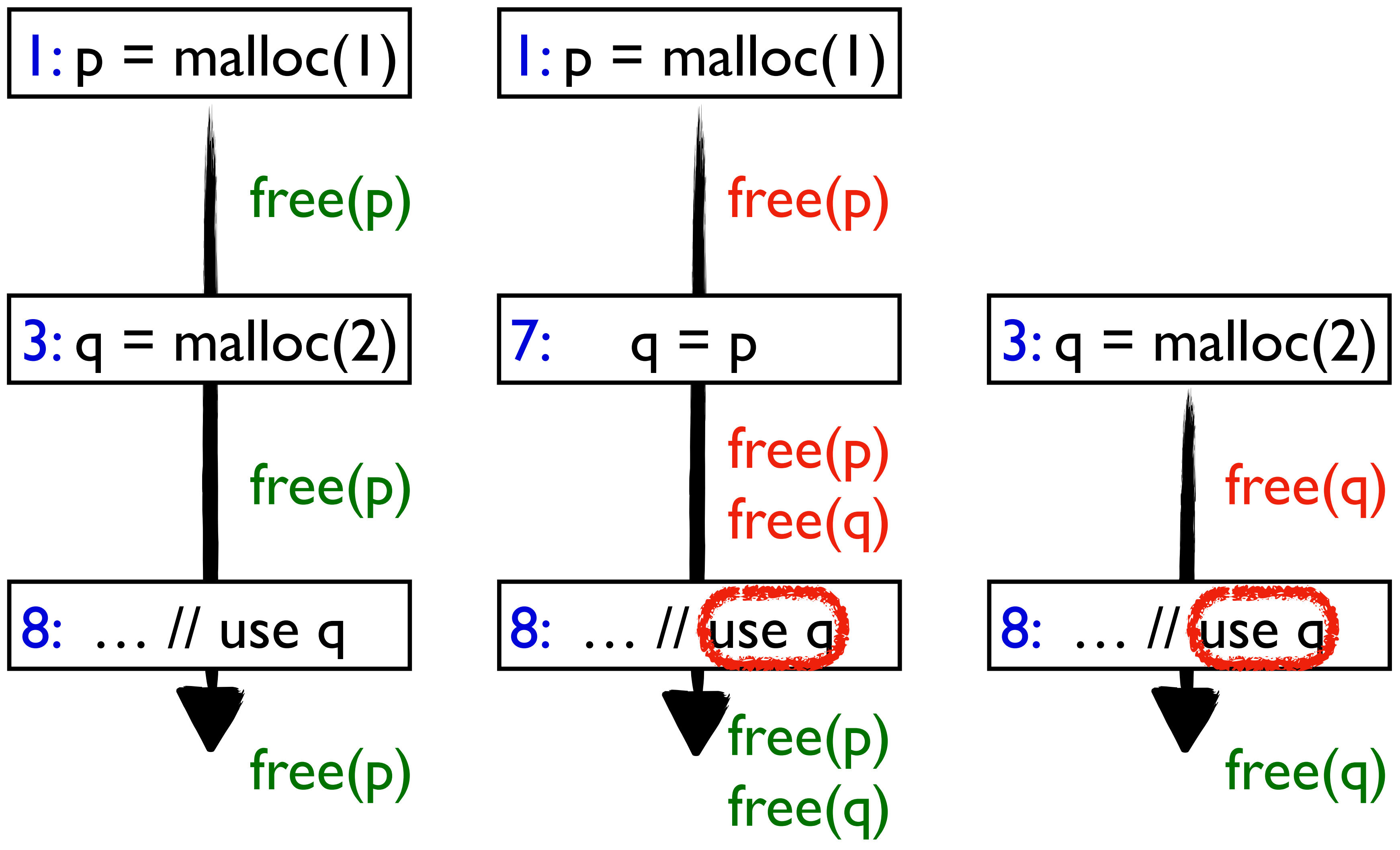
(3, p)	●		
(8, p)	●	●	
(8, q)		●	●

Find Safe Patches for Each Trace

```
1   p = malloc(1);
2   if(…) {
3       q = malloc(2);
4
5   }
6   else
7       q = p;
8   … // use q
9   free(p);
10  free(q);
```

|||

(3, p)	●		
(8, p)	●	●	
(8, q)		●	●



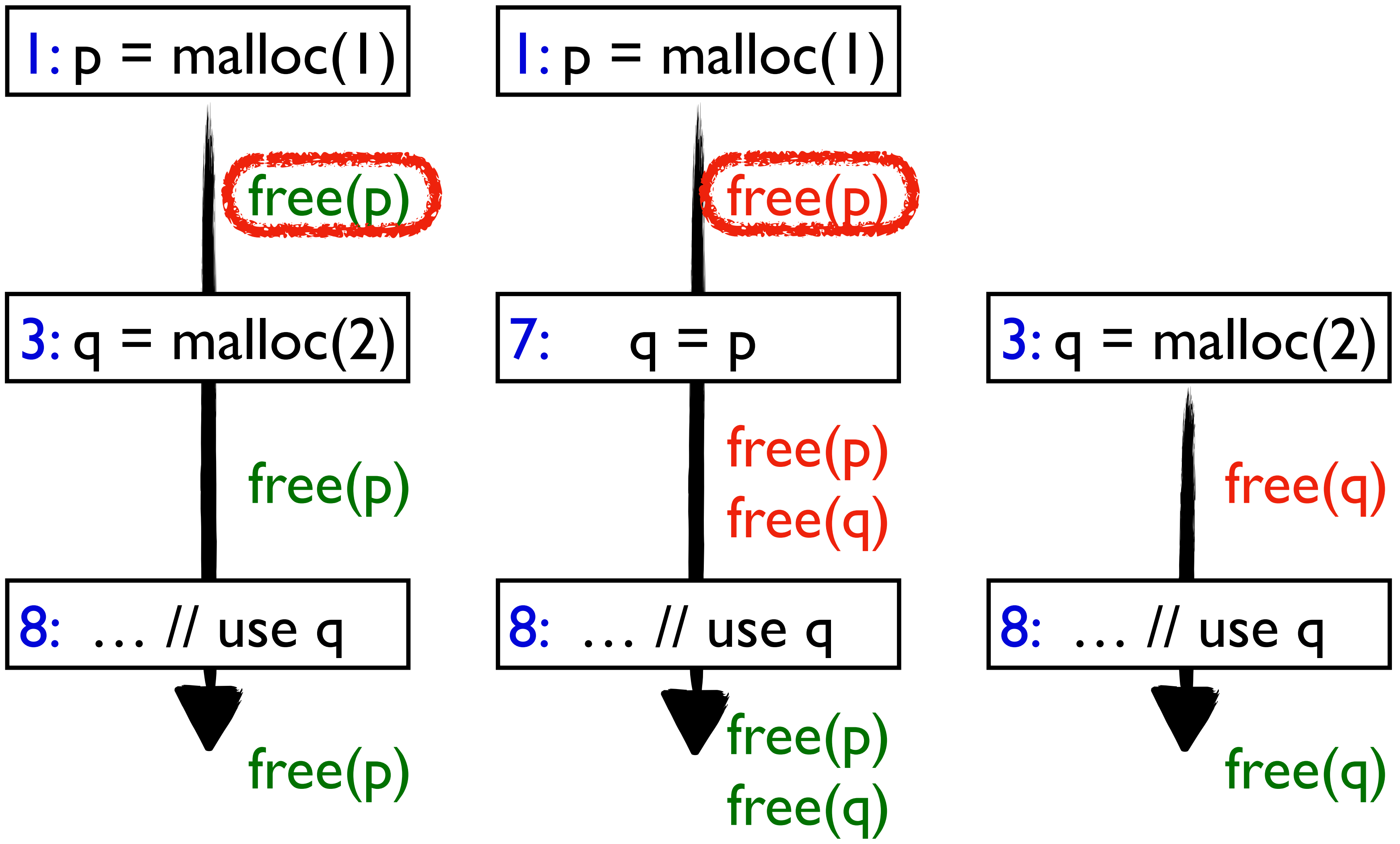
Object traces

Find Safe Patches for Each Trace

```
1  p = malloc(1);
2  if(…) {
3      q = malloc(2);
4
5  }
6  else
7      q = p;
8  … // use q
9  free(p);
10 free(q);
```

|||

(3, p)	●		
(8, p)	●	●	
(8, q)		●	●

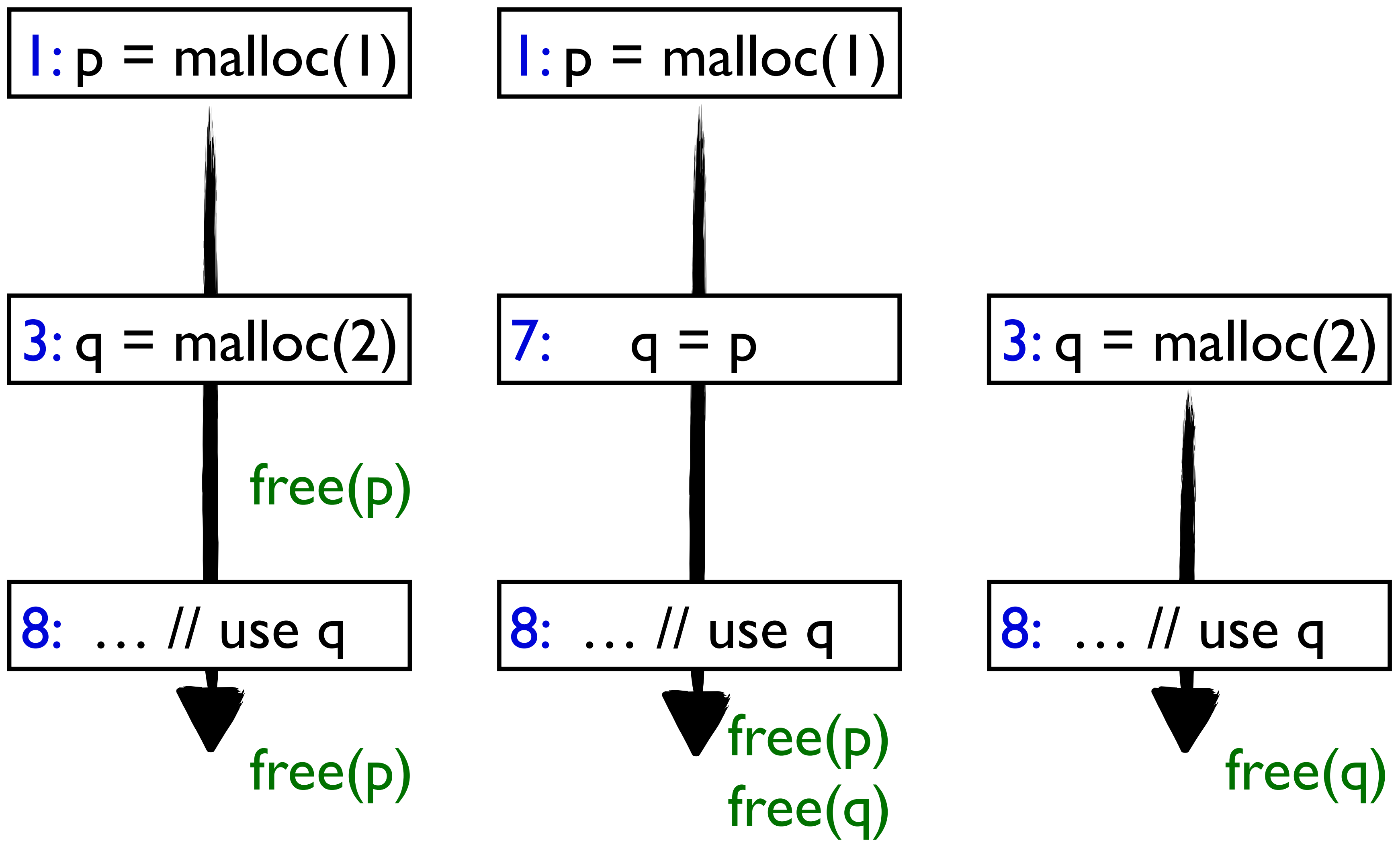


Object traces

Find Safe Patches for Each Trace

```
1   p = malloc(1);
2   if(…) {
3       q = malloc(2);
4
5   }
6   else
7       q = p;
8   … // use q
9   free(p);
10  free(q);
```

|||

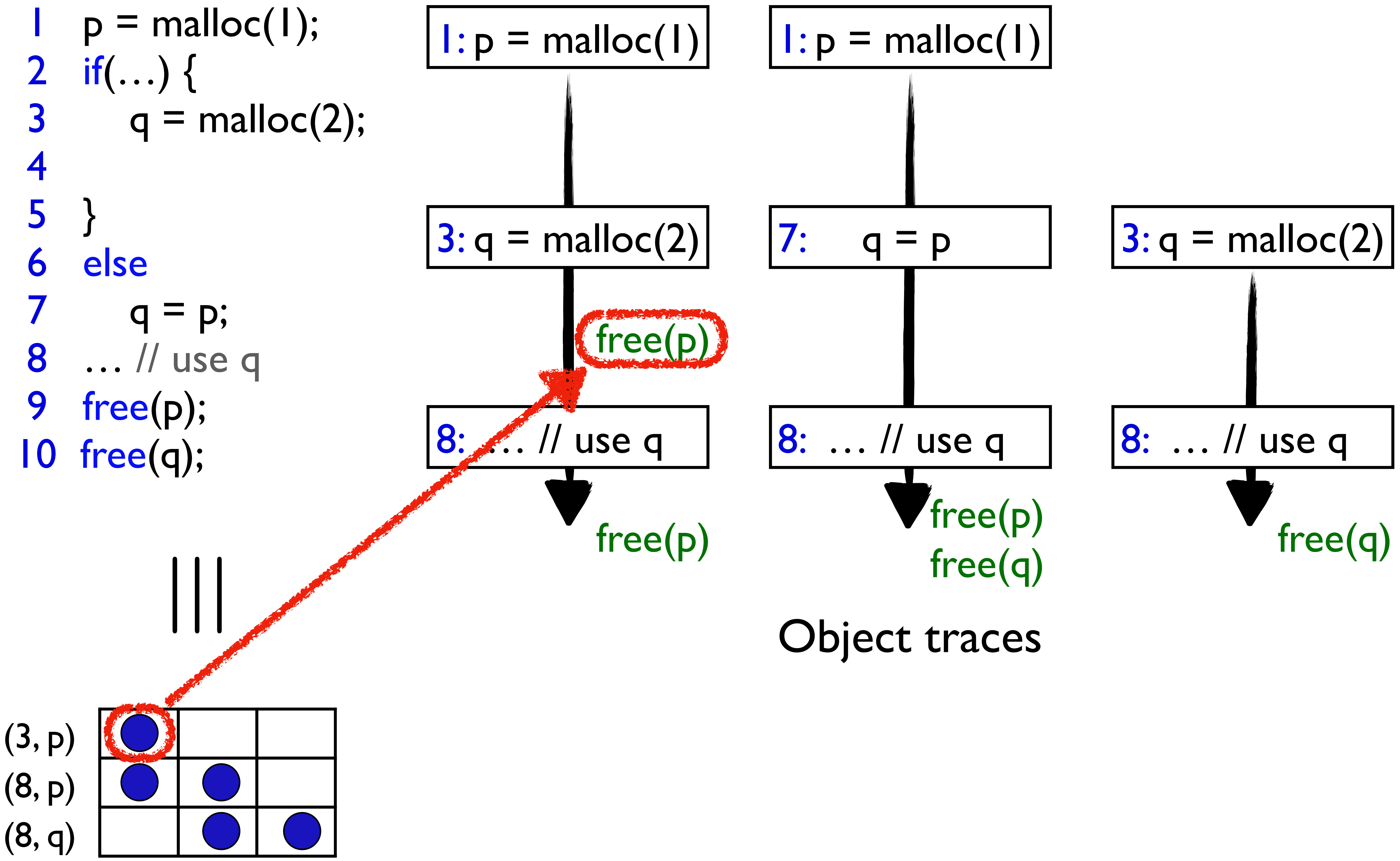


Object traces

(3, p)	●		
(8, p)	●	●	
(8, q)		●	●

Find Safe Patches for Each Trace

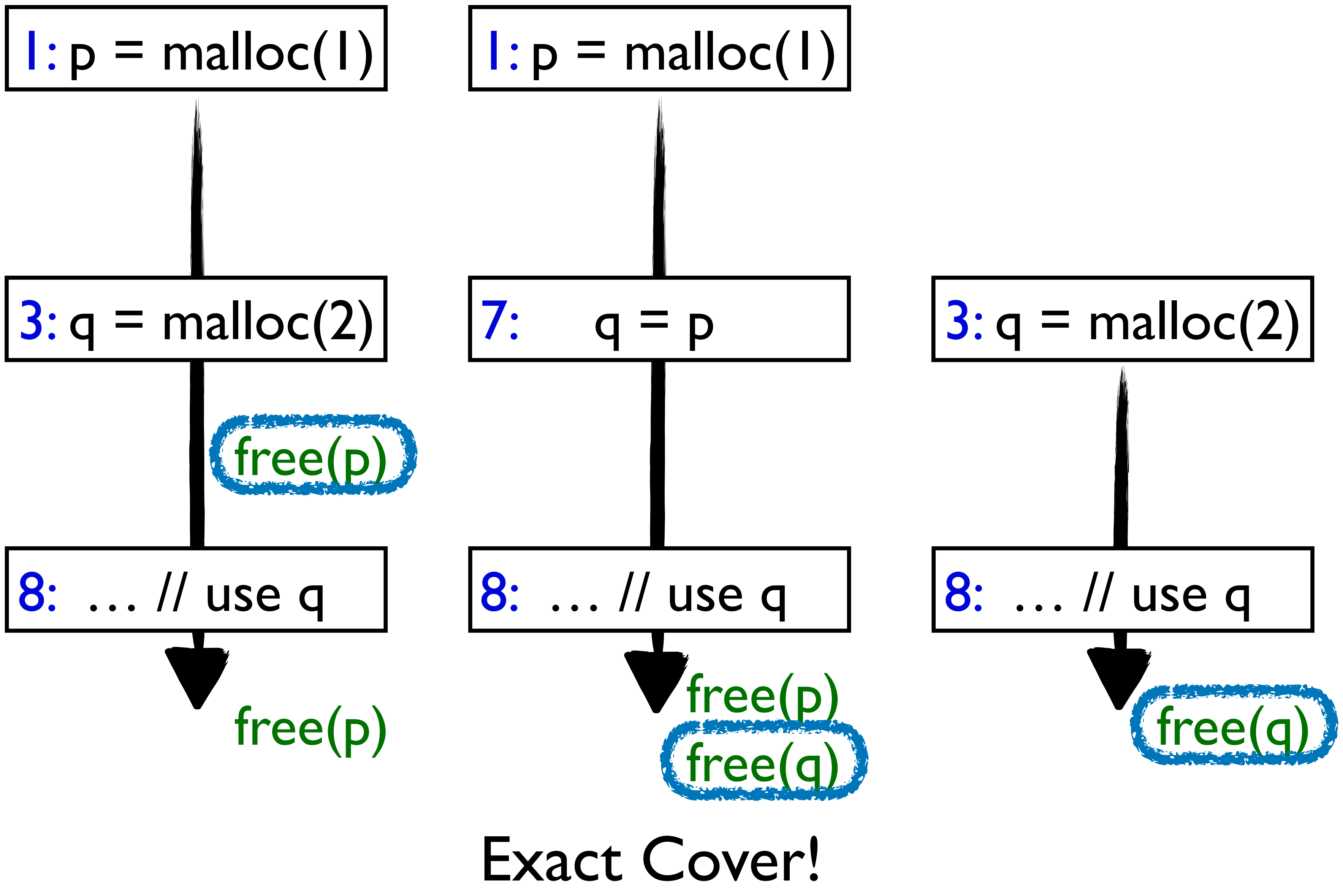
```
1   p = malloc(1);
2   if(…) {
3       q = malloc(2);
4
5   }
6   else
7       q = p;
8   … // use q
9   free(p);
10  free(q);
```



Object traces

Find Safe Patches for Each Trace

```
1   p = malloc(1);
2   if(…) {
3       q = malloc(2);
4
5   }
6   else
7       q = p;
8   … // use q
9   free(p);
10  free(q);
```

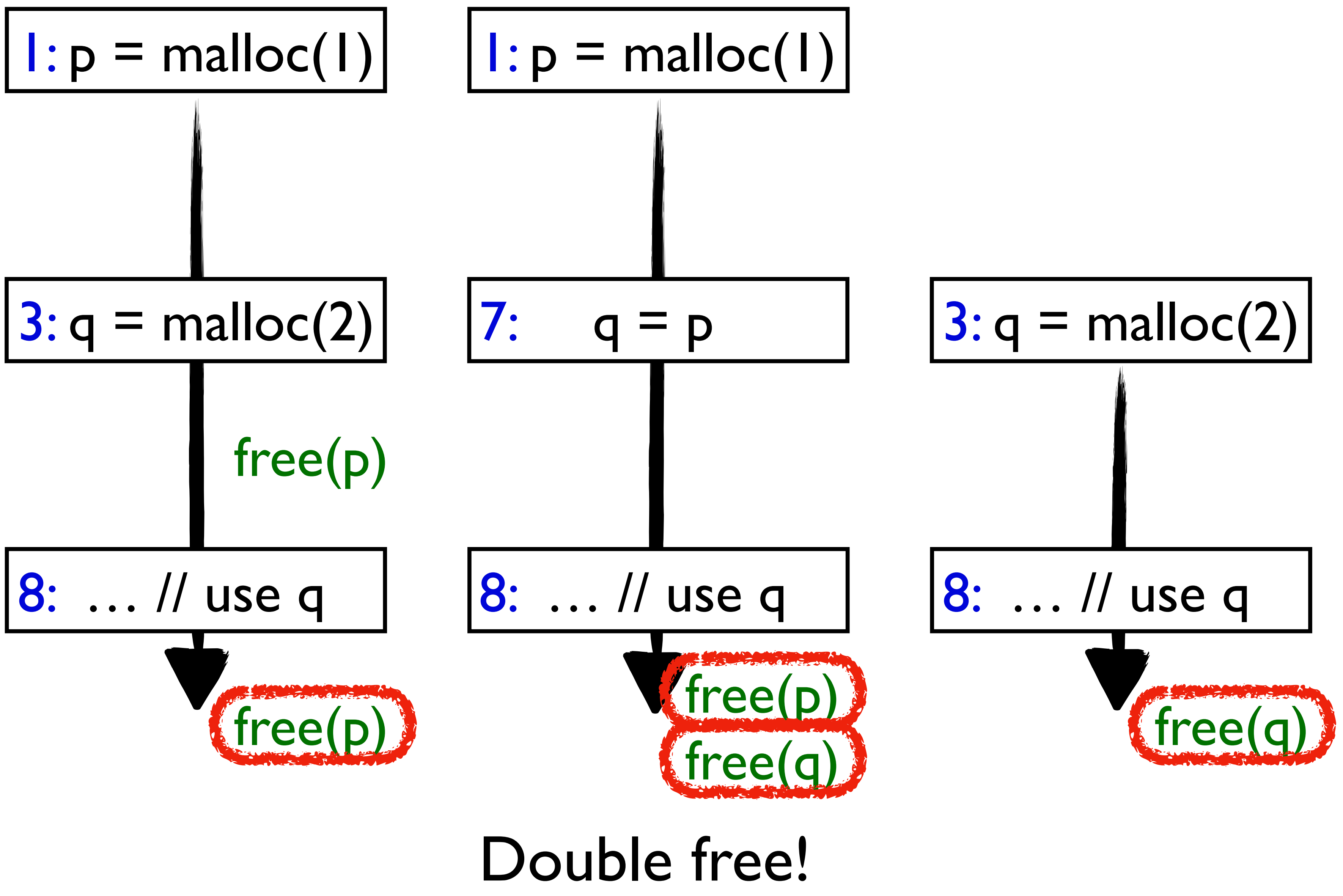


|||

(3, p)	●		
(8, p)	●	●	
(8, q)		●	●

Non-Exact Cover

```
1   p = malloc(1);
2   if(…) {
3       q = malloc(2);
4
5   }
6   else
7       q = p;
8   … // use q
9   free(p);
10  free(q);
```



|||

(3, p)	●		
(8, p)	●	●	
(8, q)		●	●

Applying Generated Patches

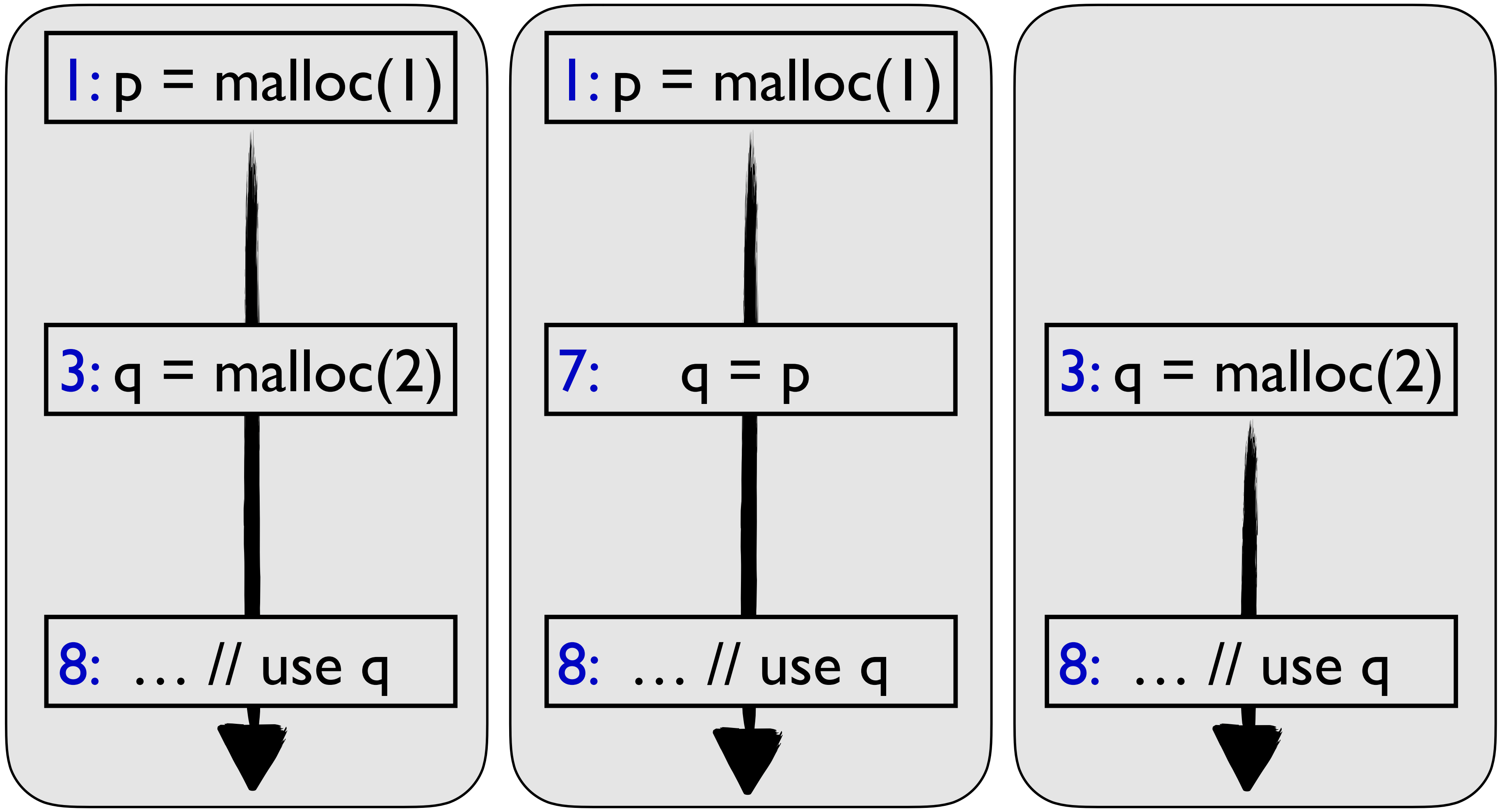
```
1   p = malloc(1);
2   if(…) {
3       q = malloc(2);
4
5   }
6   else
7       q = p;
8   … // use q
9   free(p);
10  free(q);
```

Apply the patch (3, p), (8, q)

```
1   p = malloc(1);
2   if(…) {
3       q = malloc(2);
4       free(p); // +
5   }
6   else
7       q = p;
8   … // use q
9
10  free(q); // -
```

Hurdle 1: Unbounded Traces

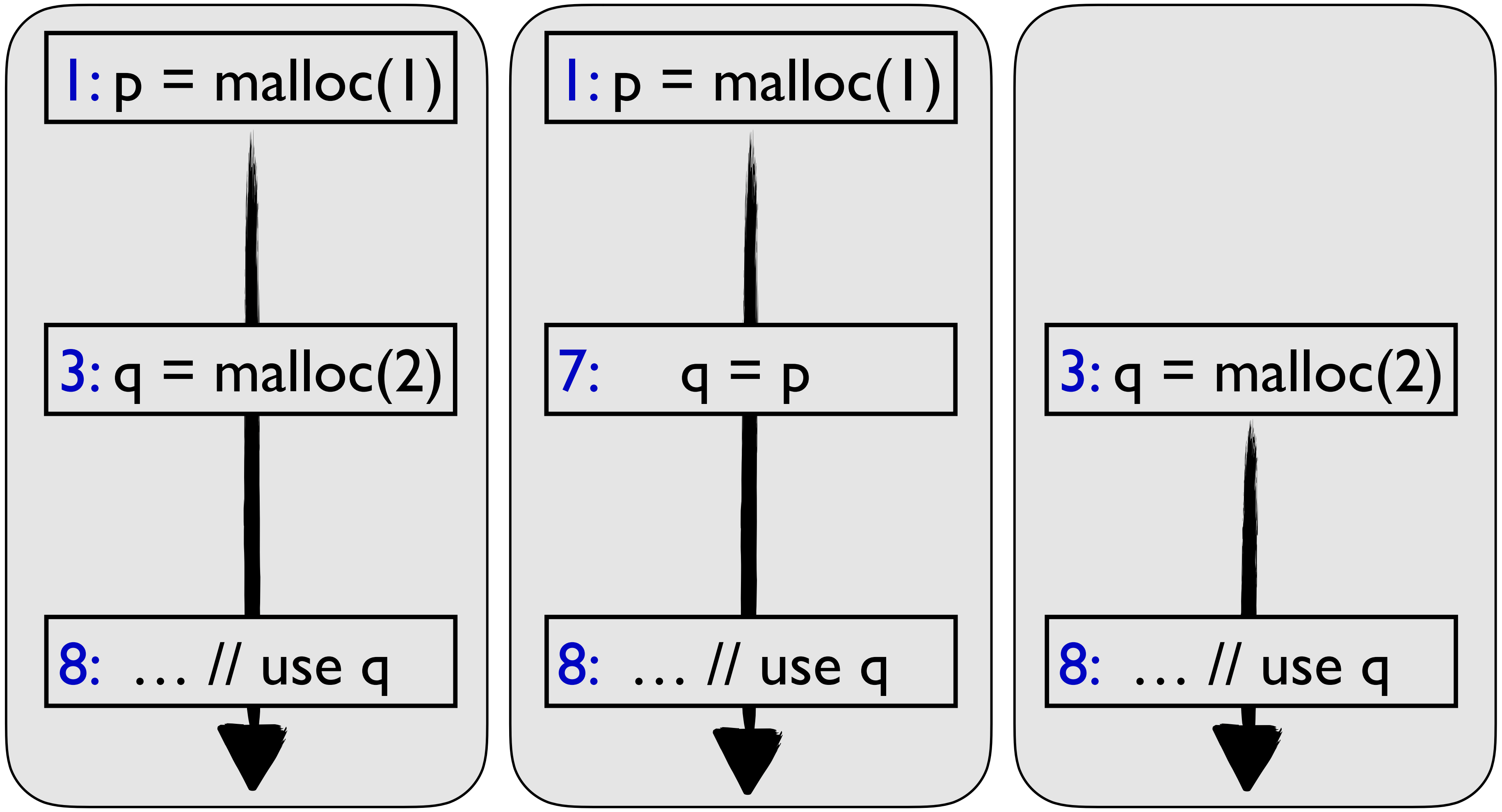
```
1   p = malloc(1);
2   if(…) {
3       q = malloc(2);
4
5   }
6   else
7       q = p;
8   … // use q
9   free(p);
10  free(q);
```



Unbounded number of object traces

Hurdle 2: Finding Exact Cover

```
1   p = malloc(1);
2   if(…) {
3       q = malloc(2);
4
5   }
6   else
7       q = p;
8   … // use q
9   free(p);
10  free(q);
```



Unbounded number of object traces

(3, p)	●		
(8, p)	●	●	
(8, q)		●	●

Well-known NP-complete problem

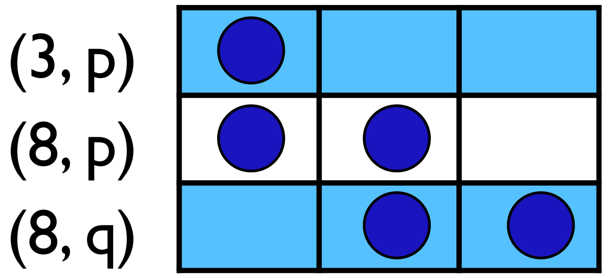
MemFix Algorithm

Approx. by Static Analysis

MaxSAT Encoding

```
1  p = malloc(1);
2  if(…) {
3      q = malloc(2);
4
5  }
6  else
7      q = p;
8  … // use q
9  free(p);
10 free(q);
```

Buggy Program



"Approximated"
Exact Cover Problem

```
1  p = malloc(1);
2  if(…) {
3      q = malloc(2);
4      free(p); // +
5  }
6  else
7      q = p;
8  … // use q
9
10 free(q); // -
```

Patched Program

Static Analysis by Abstract Interpretation

Abstract Domain $\mathbb{D} : \mathbb{C} \rightarrow \mathbb{S}$

<AllocSite, Must, MustNot, Patch, PatchNot> $\in \mathbb{S}$

- Use-after-free
- Double-free
- **Uncertain**

```
1   while(…) {
2       p = malloc(1);
3       … // use p
4   }
5   … // use p
```

Infinite number of object traces

- 2→3→1→5
- 2→3→1→2→3→1→5
- 2→3→1→2→3→1→2→3→1→5
- …

Static Analysis by Abstract Interpretation

Abstract Domain $\mathbb{D} : \mathbb{C} \rightarrow \mathbb{S}$

<AllocSite, Must, MustNot, Patch, PatchNot> $\in \mathbb{S}$

- Use-after-free
- Double-free
- **Uncertain**

```
1  while(…) {
2      p = malloc(1);
3      … // use p
4  }
5  … // use p
```

AllocSite	Must	MustNot	Patch	PatchNot
2	p	{}	(5, p)	…
2	{}	p	(1, p)	…

Experiments

ICSE'15

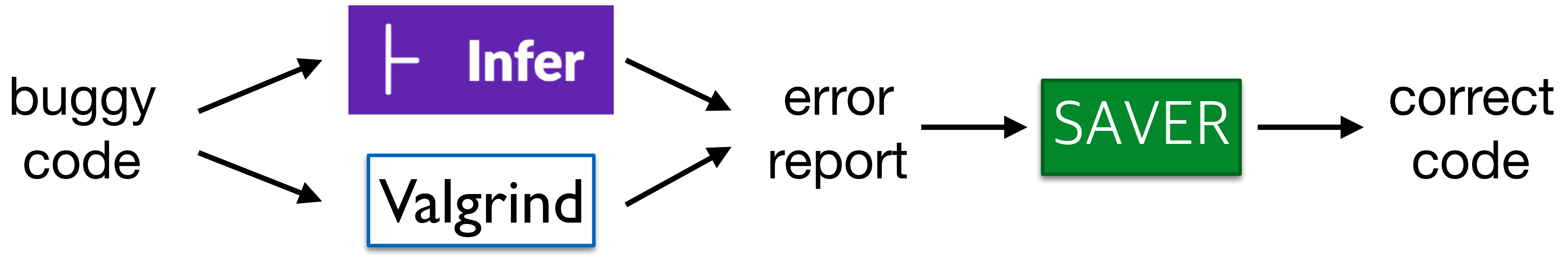
- 실험 1) Core utils
- 실험 2) Open-sources

Repo.	ML	DF	UAF	Total
	Fix/#Pgm.	Fix/#Pgm.	Fix/#Pgm.	Fix/#Pgm.
Binutils	4/10	1/5	2/5	7/20 (35%)
Git	1/10	1/4	2/6	4/20 (20%)
OpenSSH	6/10	5/7	1/3	12/20 (60%)
OpenSSL	5/10	3/5	1/5	9/20 (45%)
Tmux	5/10	0/3	0/7	5/20 (25%)
Total	21/50 (42%)	10/24 (42%)	6/26 (23%)	37/100 (37%)

			MemFIx		LeakFix	
Programs	LoC	#Al.	#Ins.	sec	#Ins.	sec
yes	553	1	1	< 1.0	✗	< 1.0
users	577	1	1	< 1.0	✗	< 1.0
unexpand	707	1	1	< 1.0	✗	< 1.0
tee	779	1	1	< 1.0	1	< 1.0
mktemp	794	4	✗	1.3	✗	< 1.0
tsort	920	3	✗	1.4	✗	< 1.0
paste	982	3	3	2.4	△/3	< 1.0
date	1,054	1	1	3.5	✗	< 1.0
cut	1,056	1	✗	2.0	✗	< 1.0
nl	1,063	4	4	4.0	✗	< 1.0
pinky	1,120	3	4	5.2	✗	< 1.0
cat	1,209	3	✗	9.3	✗	< 1.0
ln	1,258	2	✗	5.2	✗	< 1.0
printf	1,288	1	1	3.0	✗	< 1.0
stdbuf	1,605	3	3	1.3	✗	< 1.0
wc	1,669	1	1	7.3	△/2	< 1.0
shred	1,822	5	✗	31.1	✗	< 1.0
cp	1,926	8	✗	430.7	✗	< 1.0
install	2,076	1	✗	13.4	✗	< 1.0
who	2,156	8	✗	36.8	✗	< 1.0
tr	2,304	10	✗	20.0	✗	< 1.0
expr	2,378	9	✗	13.0	✗	< 1.0
stat	2,439	10	6	130.3	✗	< 1.0
dd	3,475	2	✗	52.2	✗	< 1.0

후속 연구

- SAVER (**S**calable, **A**utomatic, and **V**erified **E**rror **R**epair)
  - Scalability: 수십만 라인 코드에 적용 가능
  - Verifiability: 패치의 안전성 보장
  - Repairability: 평균 60% 이상 패치 성공
- 정적/동적 오류 탐지 도구(e.g., Infer, Valgrind)와 연동



Application to Intelligent Tutoring System

- 오류 수정 기술을 프로그래밍 교육에 적용 가능
- 현재 코딩 교육 자동 도구들의 한계: 개인화된 피드백 제공 못함

```
let rec diff : aexp * string -> aexp
= fun (e, x) ->
   match e with
   | Const n -> Const 0
   | Var a -> if (a <> x) then Const 0 else Const 1
   | Power (a, n) -> if (a <> x) then Const 0 else Times [Const n; Power (a, n-1)]
   | Times l ->
      begin
      match l with
      | [] -> Const 0
      | hd::tl -> Sum [Times ((diff (hd, x))::tl); Times [hd; diff (Times tl, x)]]
      end
   | Sum l -> Sum (List.map (fun e -> diff (e,x)) l)
```

제공된 솔루션

```
type aexp =
   |CONST of int
   | VAR of string
   | POWER of string * int
   | TIMES of aexp list
   | SUM of aexp list

type env = (string * int * int) list

let diff : aexp * string -> aexp
= fun (aexp, x) ->

   let rec deployEnv : env -> int -> aexp list
   = fun env flag ->
   match env with
   | hd::tl ->
   (
    match hd with
    |(x, c, p) ->
      if (flag = 0 && c = 0) then deployEnv tl flag
      else if (x = "const" && flag = 1 && c = 1) then deployEnv tl flag
      else if (p = 0) then (CONST c)::(deployEnv tl flag)
      else if (c = 1 && p = 1) then (VAR x)::(deployEnv tl flag)
      else if (p = 1) then TIMES[CONST c; VAR x]::(deployEnv tl flag)
      else if (c = 1) then POWER(x, p)::(deployEnv tl flag)
      else TIMES [CONST c; POWER(x, p)]::(deployEnv tl flag)
)
| [] -> []
in

let rec updateEnv : (string * int * int) -> env -> int -> env
= fun elem env flag ->
match env with
| (hd::tl) ->
   (
   match hd with
   | (x, c, p) ->
     (
     match elem with
     |(x2, c2, p2) ->
       if (flag = 0) then
         if (x = x2 && p = p2) then (x, (c + c2), p)::tl
         else hd::(updateEnv elem tl flag)
       else
         if (x = x2) then (x, (c*c2), (p + p2))::tl
         else hd::(updateEnv elem tl flag)
     )
   )
| [] -> elem::[]
in

let rec doDiff : aexp * string -> aexp
= fun (aexp, x) ->
match aexp with
| CONST _ -> CONST 0
| VAR v ->
   if (x = v) then CONST 1
   else CONST 0
| POWER (v, p) ->
   if (p = 0) then CONST 0
   else if (x = v) then TIMES ((CONST p)::POWER (v, p-1)::[])
   else CONST 0
| TIMES lst ->
   (
     match lst with
       (
         match (hd, diff_hd, tl, diff_tl) with
         | (CONST p, CONST s, [CONST r], CONST q) -> CONST (p*q + r*s)
         | (CONST p, _, _, CONST q) ->
           if (diff_hd = CONST 0 || tl = [CONST 0]) then CONST (p*q)
           else SUM [CONST(p*q); TIMES(diff_hd::tl)]
         | (_, CONST s, [CONST r], _) ->
           if (hd = CONST 0 || diff_tl = CONST 0) then CONST (r*s)
           else  SUM [TIMES [hd; diff_tl]; CONST(r*s)]
         | _ ->
           if (hd = CONST 0 || diff_tl = CONST 0) then TIMES(diff_hd::tl)
           else if (tl = [CONST 0] || diff_hd = CONST 0) then TIMES [hd; diff_tl]
           else SUM [TIMES [hd; diff_tl]; TIMES (diff_hd::tl)]
       )
     | [] -> CONST 0
   )
| SUM lst -> SUM(List.map (fun aexp -> doDiff(aexp, x)) lst)
in

let rec simplify : aexp -> env -> int -> aexp list
= fun aexp env flag ->
match aexp with
| SUM lst ->
   (
     match lst with
     | (CONST c)::tl -> simplify (SUM tl) (updateEnv ("const", c, 0) env 0) 0
     | (VAR x)::tl -> simplify (SUM tl) (updateEnv (x, 1, 1) env 0) 0
     | (POWER (x, p))::tl -> simplify (SUM tl) (updateEnv (x, 1, p) env 0) 0
     | (SUM lst)::tl -> simplify (SUM (List.append lst tl)) env 0
     | (TIMES lst)::tl ->
       (
         let l = simplify (TIMES lst) [] 1 in
         match l with
         | h::t ->
           if (t = []) then List.append l (simplify (SUM tl) env 0)
           else List.append (TIMES l::[]) (simplify (SUM tl) env 0)
         | [] -> []
       )
     | [] -> deployEnv env 0
   )
| TIMES lst ->
   (
     match lst with
     | (CONST c)::tl -> simplify (TIMES tl) (updateEnv ("const", c, 0) env 1) 1
     | (VAR x)::tl -> simplify (TIMES tl) (updateEnv (x, 1, 1) env 1) 1
     | (POWER (x, p))::tl -> simplify (TIMES tl) (updateEnv (x, 1, p) env 1) 1
     | (SUM lst)::tl ->
       (
         let l = simplify (SUM lst) [] 0 in
         match l with
         | h::t ->
           if (t = []) then List.append l (simplify (TIMES tl) env 1)
            else List.append (SUM l::[]) (simplify (TIMES tl) env 1)
          | [] -> []  (* Feedback : Replace [] by ((Sum lst) :: tl) *)
        )
      | (TIMES lst)::tl -> simplify (TIMES (List.append lst tl)) env 1
      | [] -> deployEnv env 1
   )
in

let result = doDiff (aexp, x) in
match result with
| SUM _ -> SUM (simplify result [] 0)
| TIMES _ -> TIMES (simplify result [] 1)
| _ -> result
```

학생 제출 답안

Application to Intelligent Tutoring System

- 오류 수정 기술을 프로그래밍 교육에 적용 가능
- 현재 코딩 교육 자동 도구들의 한계: 개인화된 피드백 제공 못함

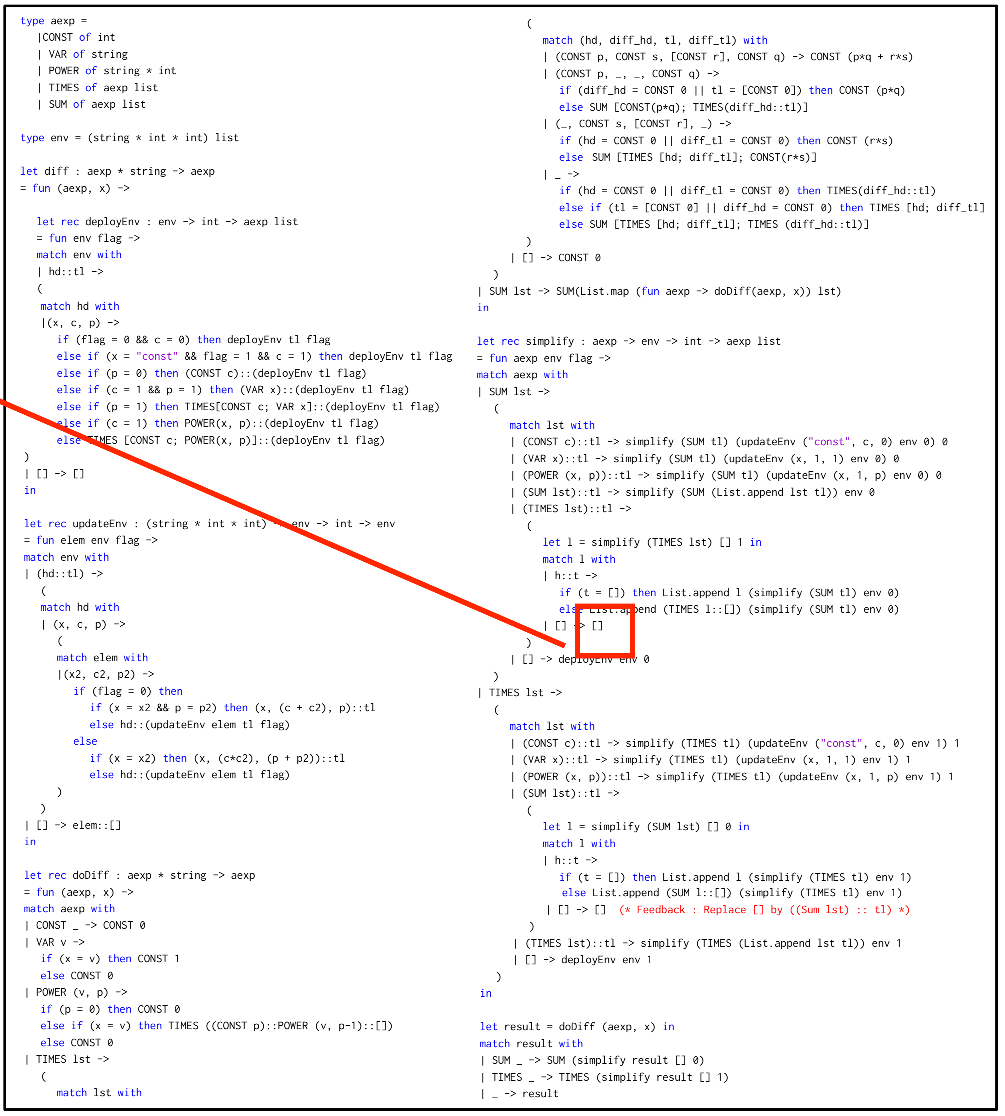
CodeOnWeb oncoder

OOPSLA'18

FixML-generated feedback: ((Sum lst)::tl)

```
let rec diff : aexp * string -> aexp
= fun (e, x) ->
   match e with
   | Const n -> Const 0
   | Var a -> if (a <> x) then Const 0 else Const 1
   | Power (a, n) -> if (a <> x) then Const 0 else Times [Const n; Power (a, n-1)]
   | Times l ->
      begin
      match l with
      | [] -> Const 0
      | hd::tl -> Sum [Times ((diff (hd, x))::tl); Times [hd; diff (Times tl, x)]]
      end
   | Sum l -> Sum (List.map (fun e -> diff (e,x)) l)
```

제공된 솔루션

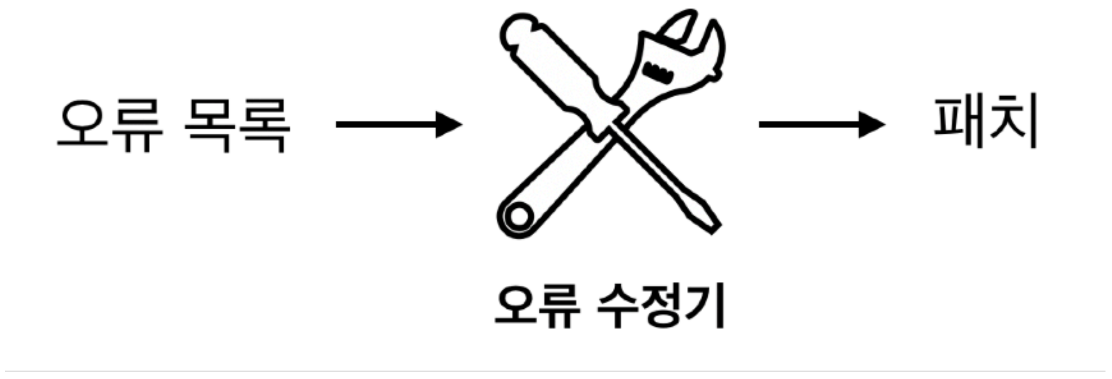


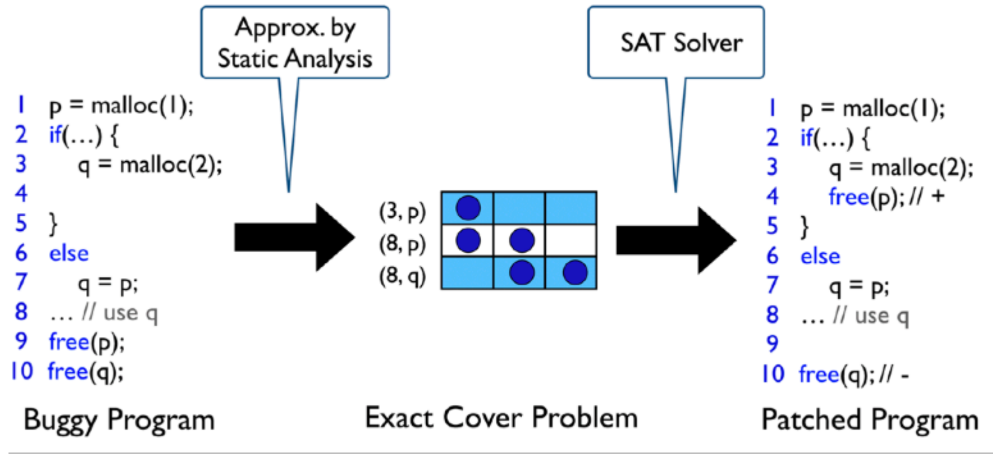
```
type aexp =
  |CONST of int
  | VAR of string
  | POWER of string * int
  | TIMES of aexp list
  | SUM of aexp list

type env = (string * int * int) list

let diff : aexp * string -> aexp
= fun (aexp, x) ->

  let rec deployEnv : env -> int -> aexp list
  = fun env flag ->
  match env with
  | hd::tl ->
  (
   match hd with
   |(x, c, p) ->
     if (flag = 0 && c = 0) then deployEnv tl flag
     else if (x = "const" && flag = 1 && c = 1) then deployEnv tl flag
     else if (p = 0) then (CONST c)::(deployEnv tl flag)
     else if (c = 1 && p = 1) then (VAR x)::(deployEnv tl flag)
     else if (p = 1) then TIMES[CONST c; VAR x]::(deployEnv tl flag)
     else if (c = 1) then POWER(x, p)::(deployEnv tl flag)
     else TIMES [CONST c; POWER(x, p)]::(deployEnv tl flag)
)
| [] -> []
in

let rec updateEnv : (string * int * int) -> env -> int -> env
= fun elem env flag ->
match env with
| (hd::tl) ->
  (
  match hd with
  | (x, c, p) ->
    (
    match elem with
    |(x2, c2, p2) ->
      if (flag = 0) then
        if (x = x2 && p = p2) then (x, (c + c2), p)::tl
        else hd::(updateEnv elem tl flag)
      else
        if (x = x2) then (x, (c*c2), (p + p2))::tl
        else hd::(updateEnv elem tl flag)
    )
  )
| [] -> elem::[]
in

let rec doDiff : aexp * string -> aexp
= fun (aexp, x) ->
match aexp with
| CONST _ -> CONST 0
| VAR v ->
  if (x = v) then CONST 1
  else CONST 0
| POWER (v, p) ->
  if (p = 0) then CONST 0
  else if (x = v) then TIMES ((CONST p)::POWER (v, p-1)::[])
  else CONST 0
| TIMES lst ->
  (
    match lst with
        (
          match (hd, diff_hd, tl, diff_tl) with
          | (CONST p, CONST s, [CONST r], CONST q) -> CONST (p*q + r*s)
          | (CONST p, _, _, CONST q) ->
            if (diff_hd = CONST 0 || tl = [CONST 0]) then CONST (p*q)
            else SUM [CONST(p*q); TIMES(diff_hd::tl)]
          | (_, CONST s, [CONST r], _) ->
            if (hd = CONST 0 || diff_tl = CONST 0) then CONST (r*s)
            else  SUM [TIMES [hd; diff_tl]; CONST(r*s)]
          | _ ->
            if (hd = CONST 0 || diff_tl = CONST 0) then TIMES(diff_hd::tl)
            else if (tl = [CONST 0] || diff_hd = CONST 0) then TIMES [hd; diff_tl]
            else SUM [TIMES [hd; diff_tl]; TIMES (diff_hd::tl)]
        )
      | [] -> CONST 0
    )
| SUM lst -> SUM(List.map (fun aexp -> doDiff(aexp, x)) lst)
in

let rec simplify : aexp -> env -> int -> aexp list
= fun aexp env flag ->
match aexp with
| SUM lst ->
  (
    match lst with
    | (CONST c)::tl -> simplify (SUM tl) (updateEnv ("const", c, 0) env 0) 0
    | (VAR x)::tl -> simplify (SUM tl) (updateEnv (x, 1, 1) env 0) 0
    | (POWER (x, p))::tl -> simplify (SUM tl) (updateEnv (x, 1, p) env 0) 0
    | (SUM lst)::tl -> simplify (SUM (List.append lst tl)) env 0
    | (TIMES lst)::tl ->
      (
        let l = simplify (TIMES lst) [] 1 in
        match l with
        | h::t ->
          if (t = []) then List.append l (simplify (SUM tl) env 0)
          else List.append (TIMES l::[]) (simplify (SUM tl) env 0)
        | [] -> []
      )
    | [] -> deployEnv env 0
  )
| TIMES lst ->
  (
    match lst with
    | (CONST c)::tl -> simplify (TIMES tl) (updateEnv ("const", c, 0) env 1) 1
    | (VAR x)::tl -> simplify (TIMES tl) (updateEnv (x, 1, 1) env 1) 1
    | (POWER (x, p))::tl -> simplify (TIMES tl) (updateEnv (x, 1, p) env 1) 1
    | (SUM lst)::tl ->
      (
        let l = simplify (SUM lst) [] 0 in
        match l with
        | h::t ->
          if (t = []) then List.append l (simplify (TIMES tl) env 1)
           else List.append (SUM l::[]) (simplify (TIMES tl) env 1)
         | [] -> []  (* Feedback : Replace [] by ((Sum lst) :: tl) *)
       )
     | (TIMES lst)::tl -> simplify (TIMES (List.append lst tl)) env 1
     | [] -> deployEnv env 1
  )
in

let result = doDiff (aexp, x) in
match result with
| SUM _ -> SUM (simplify result [] 0)
| TIMES _ -> TIMES (simplify result [] 1)
| _ -> result
```

학생 제출 답안

Summary

- Technology for automatic software repair
- MemFix focuses on memory deallocation errors
- Very exciting and new research area!



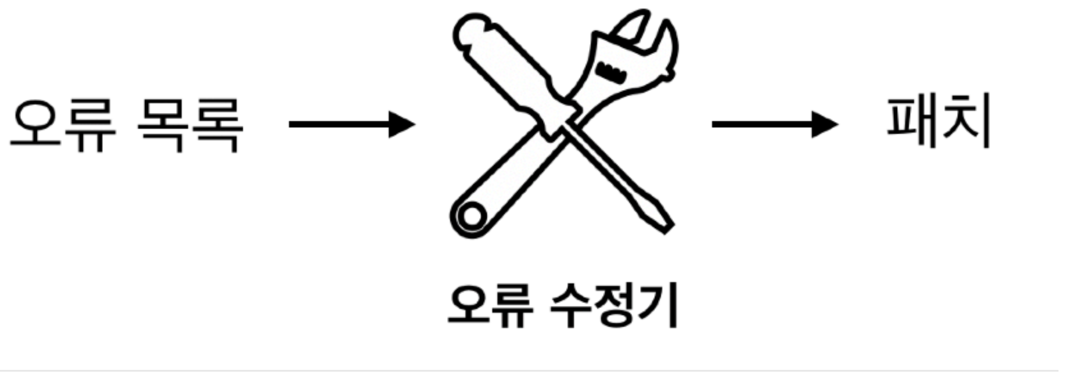


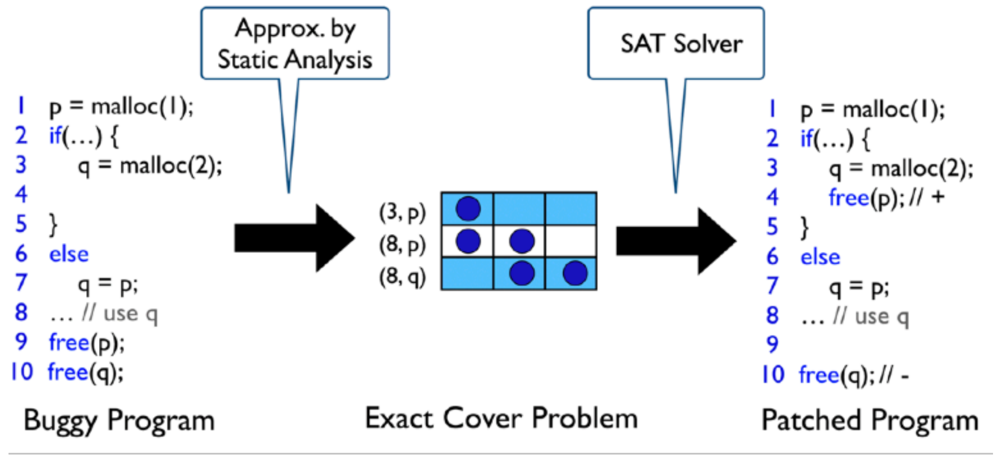
```
1  p = malloc(1);
2  if(...) {
3      q = malloc(2);
4
5  }
6  else
7      q = p;
8  ... // use q
9  free(p);
10 free(q);
```

```
1  p = malloc(1);
2  if(...) {
3      q = malloc(2);
4      free(p); // +
5  }
6  else
7      q = p;
8  ... // use q
9
10 free(q); // -
```

Summary

- Technology for automatic software repair
- MemFix focuses on memory deallocation errors
- Very exciting and new research area!





```
1  p = malloc(1);
2  if(...) {
3      q = malloc(2);
4
5  }
6  else
7      q = p;
8  ... // use q
9  free(p);
10 free(q);
```

```
1  p = malloc(1);
2  if(...) {
3      q = malloc(2);
4      free(p); // +
5  }
6  else
7      q = p;
8  ... // use q
9
10 free(q); // -
```

Thank you