Introduction to Software Analysis Research @Korea Univ.

Hakjoo Oh

Korea University

2018.12

소프트웨어 분야의 현재 수준

- 안전하고 신뢰할 수 있는 소프트웨어를 만드는 좋은 방법이 있는가?
- 다른 분야와의 비교





Newton (1642-1726)

vs.


Turing (1912-1954)

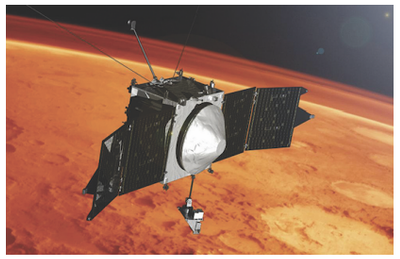

Einstein (1879-1955)

vs.

소프트웨어 결함 문제

아리안 5 로켓 폭발 (1996). 개발기간 10년, 개발 비용 $80억.

NASA 화성 탐사선 실종 (1998).



금융 거래 소프트웨어 오류 (2012)

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.

테슬라 자율주행차 소프트웨어 결함 (2017).

Tesla in fatal California crash was on Autopilot

31 March 2018 Share

SmartMesh (2018)

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

Sam Town April 25, 2018 3 min read 6028 Views

SmartMesh 사례 (2018)

- SmartMesh 토큰 스마트 컨트랙트의 정수 오버플로우 취약점 (CVE-2018-10376)을 이용하여 천문학적 금액의 토큰을 생성

5499035 (1348012 Block Confirmations)

227 days 10 hrs ago (Apr-24-2018 07:16:19 PM +UTC)

0xd6a09bdb29e1eafa92a30373c44b09e2e2e0651e

Contract 0x55f93985431fc9304077687a35a1ba103dc1e081 (SmartMeshICO)

From 0xdf31a499a5a8358... To 0xdf31a499a5a8358... for
65,133,050,195,990,400,000,000,000,000,000,000,000,000,000,000,000,000,000,000.891004451135422463
($398,308,806,220,652,000,000,000,000,000,000,000,000,000,000,000,000,000,000.00) ERC-20 (SMT)

From 0xdf31a499a5a8358... To 0xd6a09bdb29e1ea... for
50,659,039,041,325,800,000,000,000,000,000,000,000,000,000,000,000,000,000,000.693003461994217473
($309,795,738,171,618,000,000,000,000,000,000,000,000,000,000,000,000,000,000.00) ERC-20 (SMT)

0 Ether ($0.00)

https://etherscan.io/tx/0x1abab4c8db9a30e703114528e31dee129a3a758f7f8abc3b6494aad3d304e43f

SmartMesh 사례 (2018)

- 정수 오버플로우 (integer overflow) 취약점
- 방어적으로 코드를 작성했음에도 문제가 된 경우

```
function transferProxy (address from, address to, uint
      value, uint fee) public returns (bool) {
  if (balance[from] < fee + value)
    revert();
  if (balance[to] + value < balance[to] ||
      balance[msg.sender] + fee < balance[msg.sender])
    revert();
  balance[to] += value;
  balance[msg.sender] += fee;
  balance[from] -= value + fee;
  return true;
}
```

SmartMesh 사례 (2018)

- 정수 오버플로우 (integer overflow) 취약점
- 방어적으로 코드를 작성했음에도 문제가 된 경우

```
function transferProxy (address from, address to, uint
      value, uint fee) public returns (bool) {
  if (balance[from] < fee + value)
    revert();
  if (balance[to] + value < balance[to] ||
      balance[msg.sender] + fee < balance[msg.sender])
    revert();
  balance[to] += value;
  balance[msg.sender] += fee;
  balance[from] -= value + fee;
  return true;
}
```

송금

SmartMesh 사례 (2018)

- 정수 오버플로우 (integer overflow) 취약점
- 방어적으로 코드를 작성했음에도 문제가 된 경우

```
function transferProxy (address from, address to, uint
    value, uint fee) public returns
  if (balance[from] < fee + value)
    revert();
  if (balance[to] + value < balance[to] ||
      balance[msg.sender] + fee < balance[msg.sender])
    revert();
  balance[to] += value;
  balance[msg.sender] += fee;
  balance[from] -= value + fee;
  return true;
}
```

보내는 사람의 잔고가 충분한지 체크

송금

SmartMesh 사례 (2018)

- 정수 오버플로우 (integer overflow) 취약점
- 방어적으로 코드를 작성했음에도 문제가 된 경우

```
function transferProxy (address from, address to, uint
    value, uint fee) public returns
  if (balance[from] < fee + value)
    revert();
  if (balance[to] + value < balance[to] ||
      balance[msg.sender] + fee < balance[msg.sender])
    revert();
  balance[to] += value;
  balance[msg.sender] += fee;
  balance[from] -= value + fee;
  return true;
}
```

보내는 사람의 잔고가 충분한지 체크

송금

오버플로우 체크

SmartMesh 사례 (2018)

- 정수 오버플로우 (integer overflow) 취약점
- 방어적으로 코드를 작성했음에도 문제가 된 경우

```
function transferProxy (address from, address to, uint
    value, uint fee) public returns
  if (balance[from] < fee + value)
    revert();
  if (balance[to] + value < balance[to] ||
      balance[msg.sender] + fee < balance[msg.sender])
    revert();
  balance[to] += value;
  balance[msg.sender] += fee;
  balance[from] -= value + fee;
  return true;
}
```

보내는 사람의 잔고가 충분한지 체크

오버플로우 체크

송금

오버플로우/언더플로우 발생하지 않음

SmartMesh 사례 (2018)

```
function transferProxy (address from, address to, uint
    value, uint fee) public returns (bool) {
  if (balance[from] < fee + value)
    revert();
  if (balance[to] + value < balance[to] ||
      balance[msg.sender] + fee < balance[msg.sender])
    revert();
  balance[to] += value;
  balance[msg.sender] += fee;
  balance[from] -= value + fee;
  return true;
}
```

SmartMesh 사례 (2018)

```
balance[from] = balance[to] = balance[msg.sender] = 0
```

```
function transferProxy (address from, address to, uint
    value, uint fee) public returns (bool) {
  if (balance[from] < fee + value)
    revert();
  if (balance[to] + value < balance[to] ||
      balance[msg.sender] + fee < balance[msg.sender])
    revert();
  balance[to] += value;
  balance[msg.sender] += fee;
  balance[from] -= value + fee;
  return true;
}
```

SmartMesh 사례 (2018)

```
balance[from] = balance[to] = balance[msg.sender] = 0

value: 8fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fee  : 7000000000000000000000000000000000000000000000000000000000000001
```

```
function transferProxy (address from, address to, uint
      value, uint fee) public returns (bool) {
  if (balance[from] < fee + value)
    revert();
  if (balance[to] + value < balance[to] ||
      balance[msg.sender] + fee < balance[msg.sender])
    revert();
  balance[to] += value;
  balance[msg.sender] += fee;
  balance[from] -= value + fee;
  return true;
}
```

SmartMesh 사례 (2018)

```
balance[from] = balance[to] = balance[msg.sender] = 0

value: 8fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fee  : 7000000000000000000000000000000000000000000000000000000000000001
```

```
function transferProxy (address from, address to, uint
      value, uint fee) public returns (bool) {
  if (balance[from] < fee + value)
    revert();
  if (balance[to] + value < balance[to] ||
      balance[msg.sender] + fee < balance[msg.sender])
    revert();
  balance[to] += value;
  balance[msg.sender] += fee;
  balance[from] -= value + fee;
  return true;
}
```

O!

SmartMesh 사례 (2018)

```
balance[from] = balance[to] = balance[msg.sender] = 0

value: 8fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fee  : 7000000000000000000000000000000000000000000000000000000000000001
```

```
function transferProxy (address from, address to, uint
      value, uint fee) public returns (bool) {
    (balance[from] < fee + value)
      revert();
    if (balance[to] + value < balance[to] ||
        balance[msg.sender] + fee < balance[msg.sender])
      revert();
    balance[to] += value;
    balance[msg.sender] += fee;
    balance[from] -= value + fee;
    return true;
}
```

false

0!

SmartMesh 사례 (2018)

```
balance[from] = balance[to] = balance[msg.sender] = 0

value: 8fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fee  : 7000000000000000000000000000000000000000000000000000000000000001
```

```
function transferProxy (address from, address to, uint
      value, uint fee) public returns (bool) {
    (balance[from] < fee + value)
      revert();
    (balance[to] + value < balance[to] ||
     balance[msg.sender] + fee < balance[msg.sender])
      revert();
  balance[to] += value;
  balance[msg.sender] += fee;
  balance[from] -= value + fee;
  return true;
}
```

false

0!

false

SmartMesh 사례 (2018)

```
balance[from] = balance[to] = balance[msg.sender] = 0
```

```
value: 8fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fee  : 7000000000000000000000000000000000000000000000000000000000000001
```

```
function transferProxy (address from, address to, uint
      value, uint fee) public returns (bool) {
    (balance[from] < fee + value)
      revert();
    (balance[to] + value < balance[to] ||
     balance[msg.sender] + fee < balance[msg.sender])
      revert();
  balance[to] += value;
  balance[msg.sender] += fee;
  balance[from] -= value + fee;
  return true;
}
```

false

0!

false

8fffff…ff

SmartMesh 사례 (2018)

```
balance[from] = balance[to] = balance[msg.sender] = 0
```

```
value: 8fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fee  : 7000000000000000000000000000000000000000000000000000000000000001
```

```
function transferProxy (address from, address to, uint
      value, uint fee) public returns (bool) {
    (balance[from] < fee + value)
      revert();
    (balance[to] + value < balance[to] ||
     balance[msg.sender] + fee < balance[msg.sender])
      revert();
  balance[to] += value;
  balance[msg.sender] += fee;
  balance[from] -= value + fee;
  return true;
}
```

false

0!

false

8fffff…ff

700…00

SmartMesh 사례 (2018)

```
balance[from] = balance[to] = balance[msg.sender] = 0

value: 8fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fee  : 7000000000000000000000000000000000000000000000000000000000000001
```

```
function transferProxy (address from, address to, uint
      value, uint fee) public returns (bool) {
  (balance[from] < fee + value)
      revert();
  (balance[to] + value < balance[to] ||
   balance[msg.sender] + fee < balance[msg.sender])
      revert();
    balance[to] += value;
    balance[msg.sender] += fee;
    balance[from] -= value + fee;
    return true;
}
```

false

false

0!

8fffff…ff

700…00

0!

소프트웨어 결함 문제

- 2017년 소프트웨어 결함으로 인한 사회적 비용은 1.7조 달러로 추정 (Software failure watch, 2017)

연구 방향

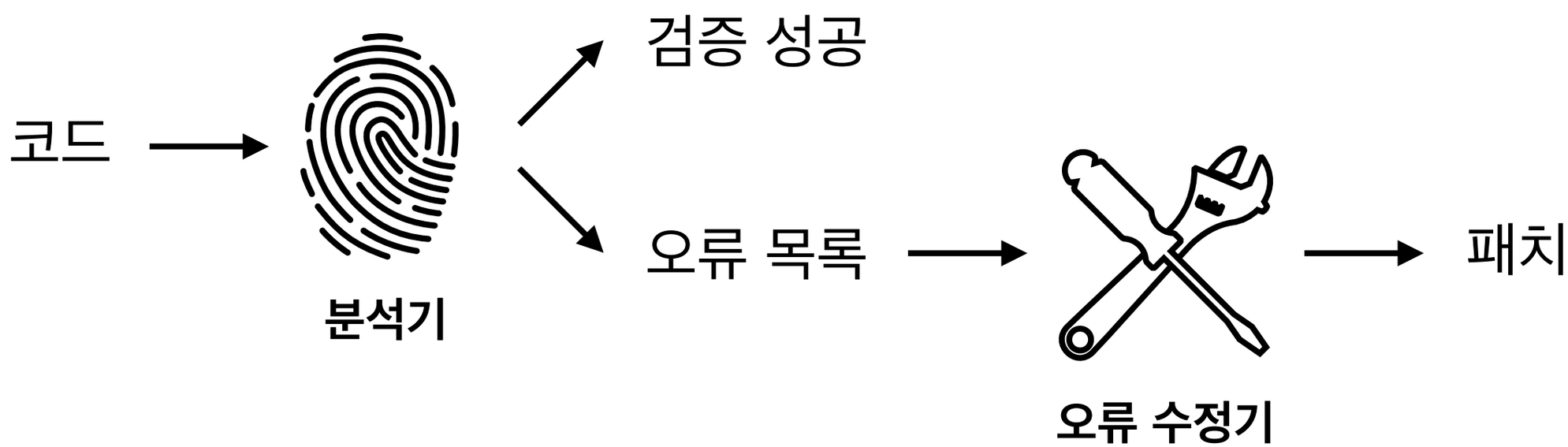
- Q) 어떻게 안전한 소프트웨어를 손쉽게 만들것인가?
- A) 소프트웨어 자동 **분석**, **패치**, **합성** 기술

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

소프트웨어 자동 분석 (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;
```

소프트웨어 자동 분석 (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

소프트웨어 자동 분석 (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

소프트웨어 자동 분석 기법

- 소프트웨어 테스팅
- 소프트웨어 정적 분석
- 소프트웨어 증명

랜덤 테스팅 (퍼징)

- 무작위로 입력을 생성하여 테스팅

```
int double (int v) {
  return 2*v;
}

void testme(int x, int y) {

  z := double (y);

  if (z==x) {

    if (x>y+10) {
      Error;
    }
  }
}
```

Probability of the error? ($0 \leq x,y \leq 100$)

랜덤 테스팅 (퍼징)

- 무작위로 입력을 생성하여 테스팅

```
int double (int v) {
  return 2*v;
}

void testme(int x, int y) {

  z := double (y);

  if (z==x) {

    if (x>y+10) {
      Error;
    }
  }
}
```

Probability of the error? ($0 \le x,y \le 100$)

< 0.4%

기호 실행 (Symbolic Execution)

- 프로그램을 실제값이 아닌 기호를 이용하여 실행

```
int double (int v) {
  return 2*v;
}

void testme(int x, int y) {
  z := double (y);

  if (z==x) {

    if (x>y+10) {
      Error;
    }
  }
}
```

$x=\alpha, y=\beta$

true

기호 실행 (Symbolic Execution)

```
int double (int v) {
  return 2*v;
}

void testme(int x, int y) {

  z := double (y);
```

$x=\alpha, y=\beta, z=2*\beta$

```
  if (z==x) {
```

true

```
    if (x>y+10) {
      Error;
    }
  }
}
```

기호 실행 (Symbolic Execution)

```
int double (int v) {
  return 2*v;
}

void testme(int x, int y) {

  z := double (y);

  if (z==x) {

    if (x>y+10) {
      Error;
    }
  }
}
```

$x=\alpha, y=\beta, z=2*\beta$

$2*\beta = \alpha$

기호 실행 (Symbolic Execution)

```
int double (int v) {
  return 2*v;
}

void testme(int x, int y) {

  z := double (y);

  if (z==x) {
    if (x>y+10) {
      Error;
    }
  }
}
```

$x=\alpha, y=\beta, z=2*\beta$

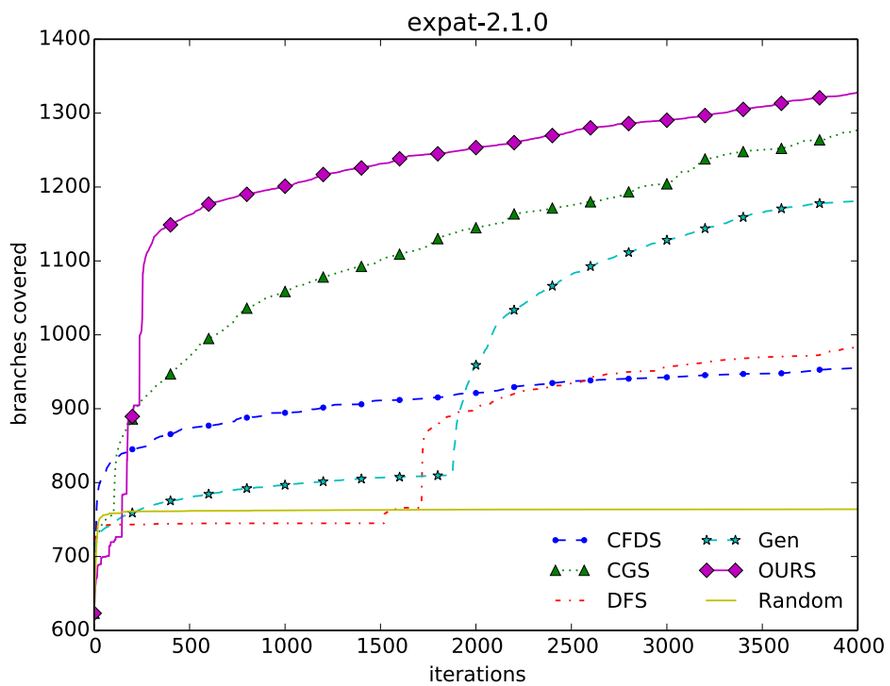
$2*\beta = \alpha$

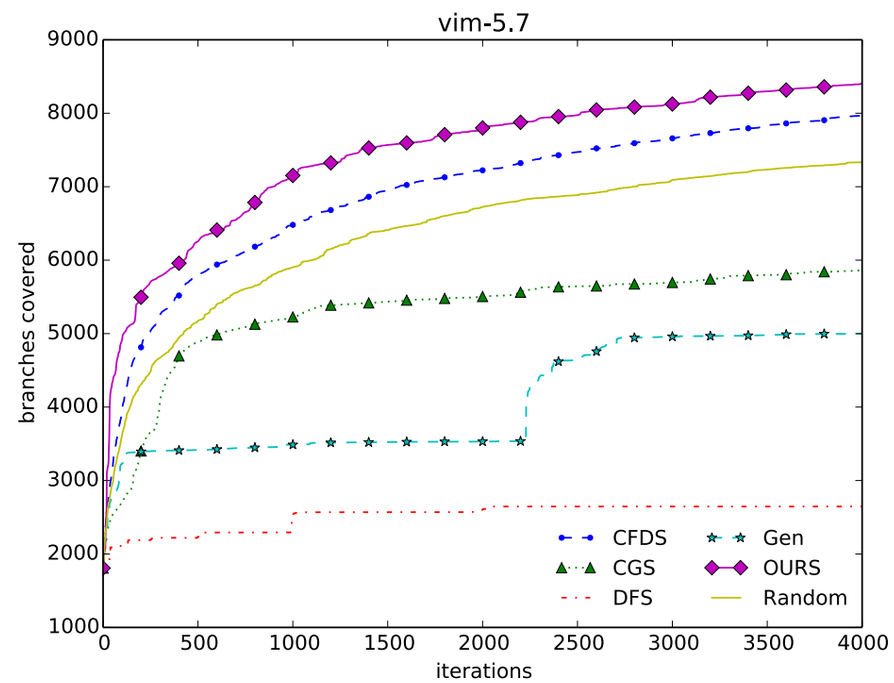
Challenge: Path explosion

ICSE'18

State-of-the-art Symbolic Execution

- Developed "data-driven symbolic execution"
  - considerable increase in code coverage





- dramatic increase in bug-finding capability

	OURS	CFDS	CGS	Random	Gen	DFS
gawk-3.0.3	**100/100**	0/100	0/100	0/100	0/100	0/100
grep-2.2	**47/100**	0/100	5/100	0/100	0/100	0/100

	Phenomenons	**Bug-Triggering Inputs**	**Version**
sed	Memory Exhaustion	**'H g ;D'**	4.4(latest)
sed	Infinite File Write	**'H w {- x; D'**	4.4(latest)
grep	Segmentation Fault	**'\(\)\1\+**'**	3.1(latest)
grep	Non-Terminating	**'?(^(|^+*)*\+\{8957\}'**	3.1(latest)
gawk	Memory Exhaustion	**'$6672467e2=E7'**	4.21(latest)

기호 실행 (Symbolic Execution)

```
int double (int v) {
  return 2*v;
}

void testme(int x, int y) {

  z := double (y);

  if (z==x) {

    if (x>y+10) {
      Error;
    }
  }
}
```

$x=\alpha, y=\beta, z=2*\beta$

$2*\beta = \alpha \wedge$
$\alpha > \beta+10$

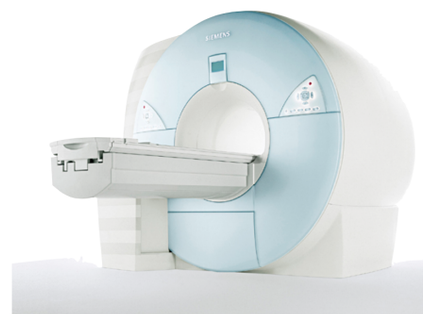
SMT solver

$x=30, y=15$

error-triggering input

Static Program Analysis

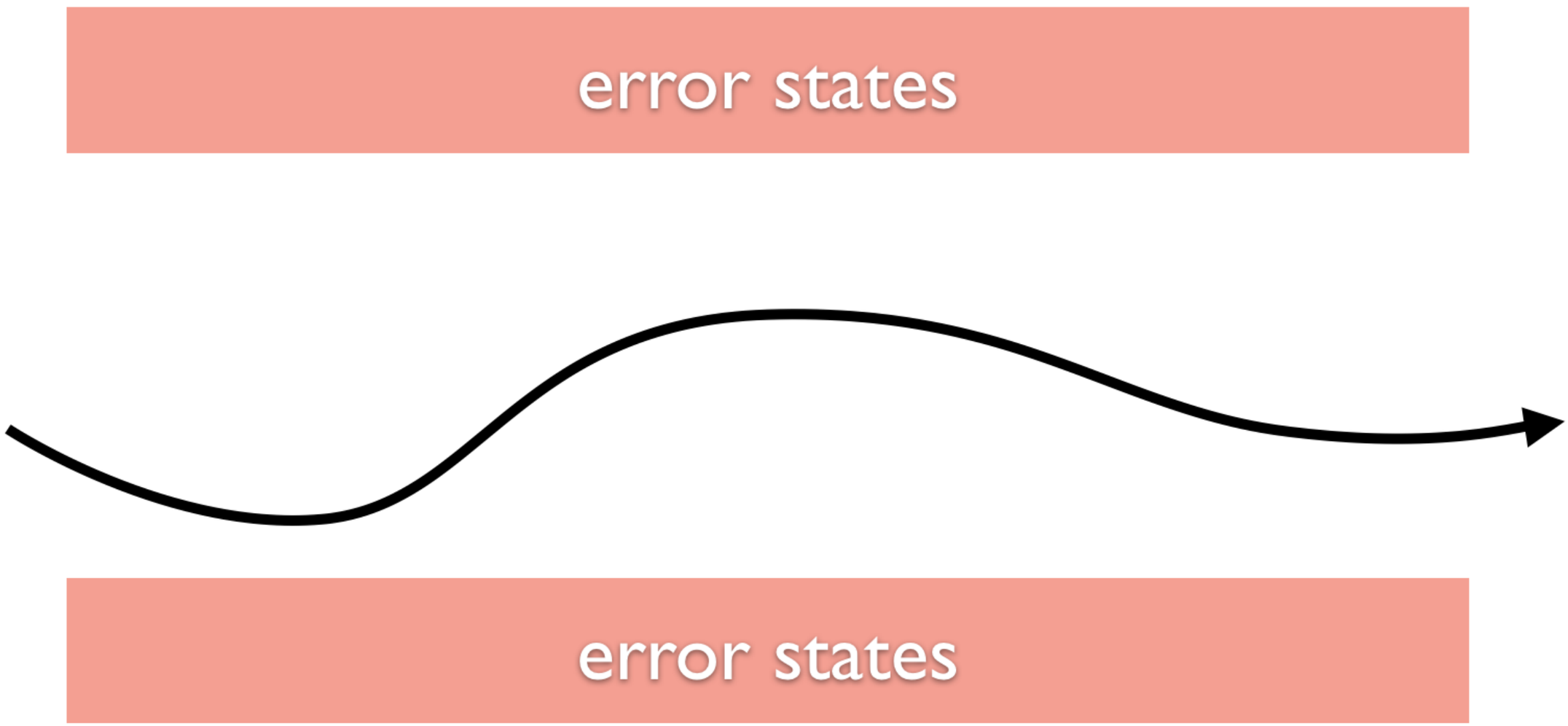
Technology for "software MRI"

Code →  → Safe / Error

- Predict software behavior statically and automatically
  - static: analyzing program text without execution
  - automatic: sw is analyzed by sw ("static analyzer")
- Next-generation software testing technology
  - finding bugs early / full automation / all bugs found
- Being widely used in sw industry

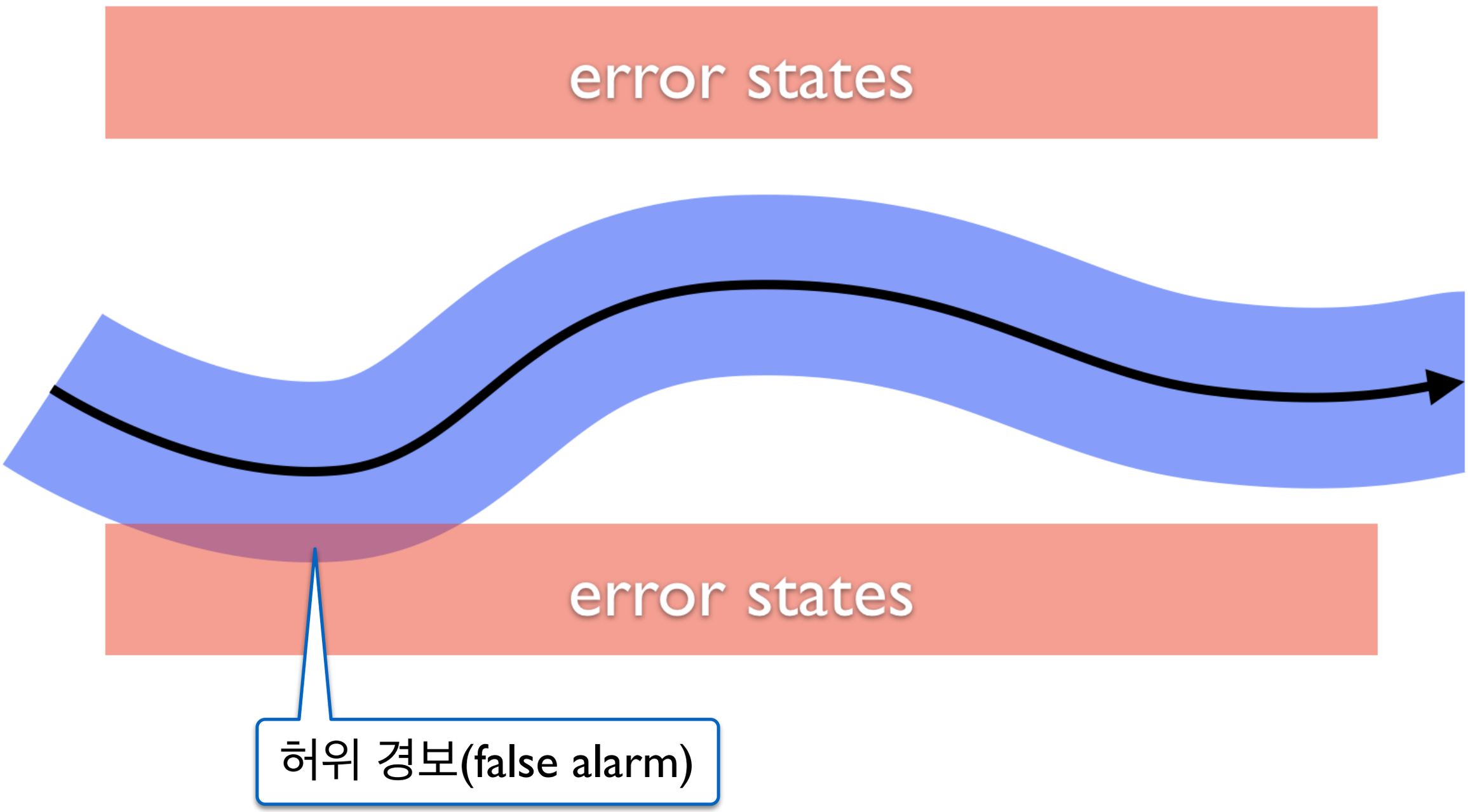
정적 분석

- 프로그램의 실제실행을 요약(abstraction)하여 분석



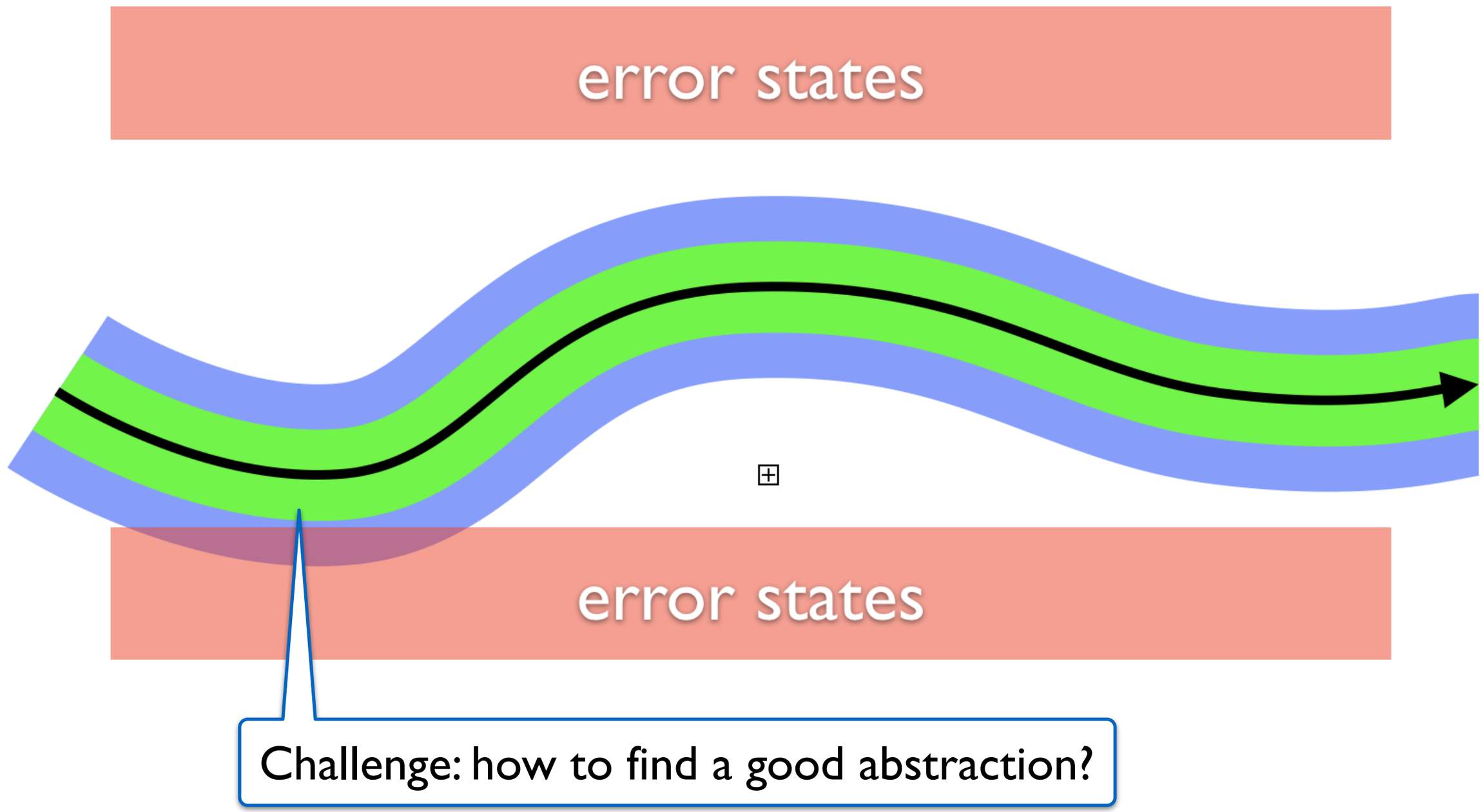
정적 분석

- 프로그램의 실제실행을 요약(abstraction)하여 분석



정적 분석

- 프로그램의 실제실행을 요약(abstraction)하여 분석



Data-Driven Program Analysis

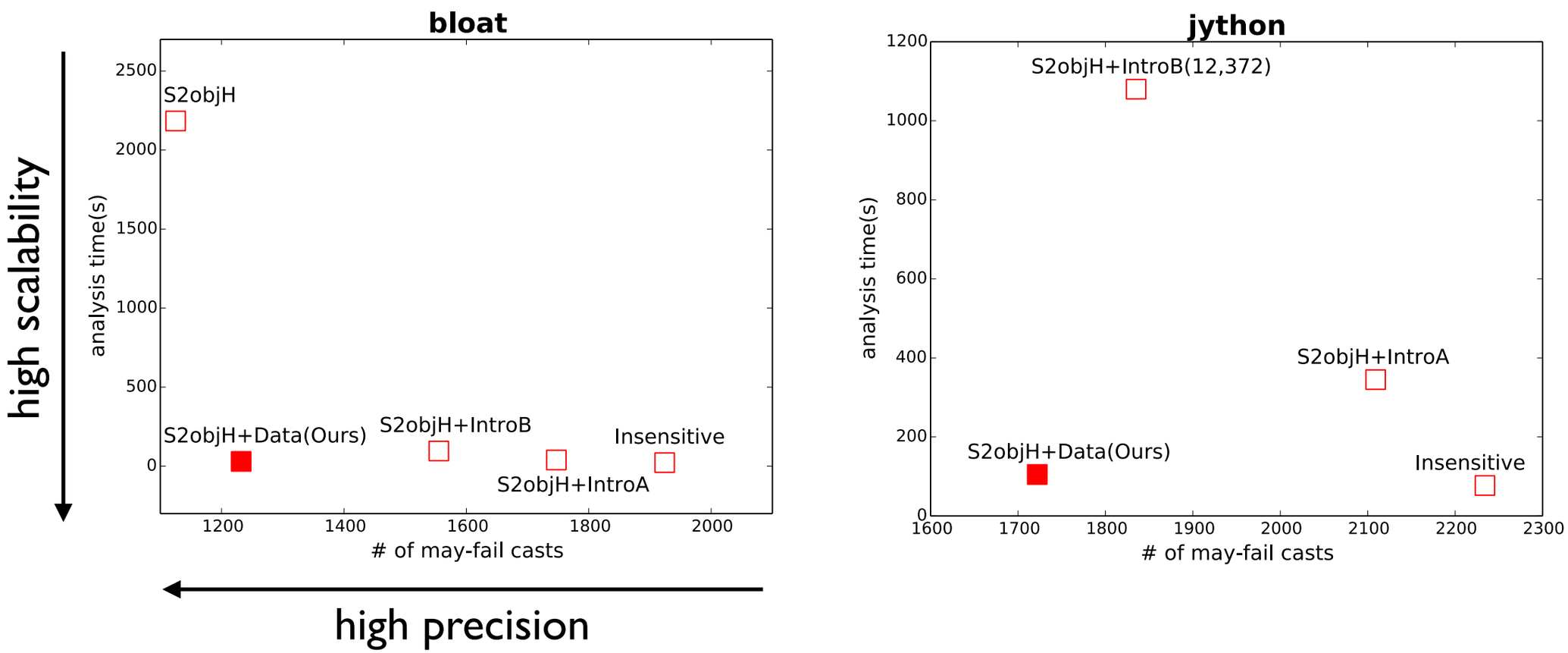
- Learning algorithms for data-driven program analysis
  - learning models [OOPSLA'17a]
  - optimization algorithms [TOPLAS'19]
  - feature engineering [OOPSLA'17b]
- State-of-the-art program analyses enabled by algorithms
  - interval / pointer analysis [OOPSLA'18a, TOPLAS'18]
  - symbolic analysis / execution [ICSE'18, ASE'18]
  - others program analyses [FSE'18, OOPSLA'18b]

9 papers in top-tier PL/SE conferences and journals

OOPSLA'17

State-of-the-art Pointer Analysis

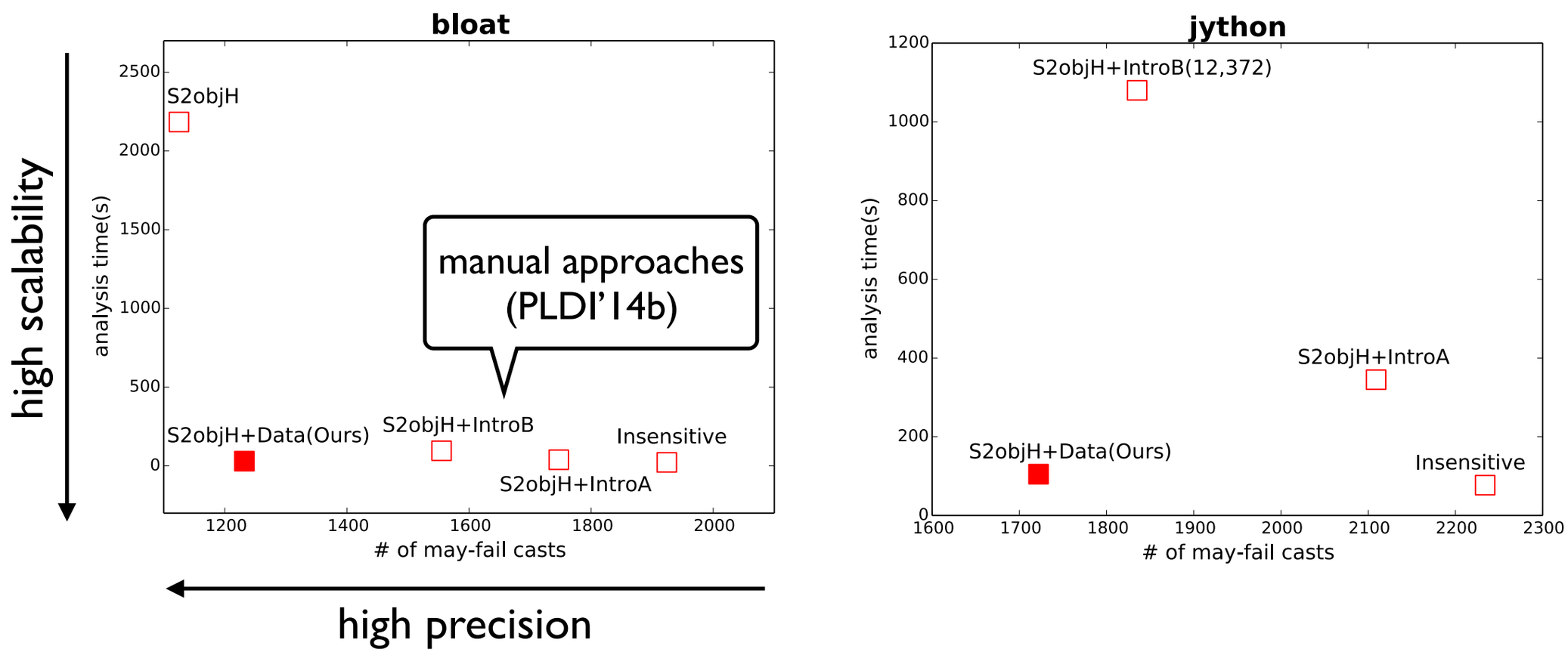
- Achieved state-of-the-art pointer analysis for Java
  - foundational static analysis for bug-finders, verifiers, etc
- Trained with 5 small programs from the DaCapo benchmark and tested with 5 remaining large programs



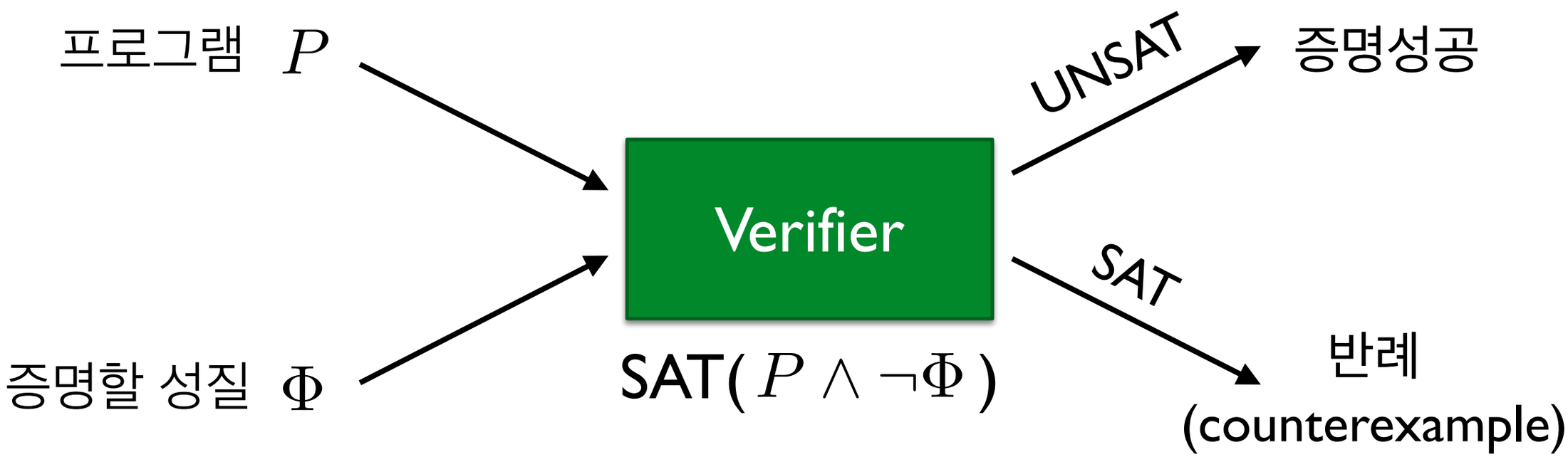
OOPSLA'17

State-of-the-art Pointer Analysis

- Achieved state-of-the-art pointer analysis for Java
  - foundational static analysis for bug-finders, verifiers, etc
- Trained with 5 small programs from the DaCapo benchmark and tested with 5 remaining large programs



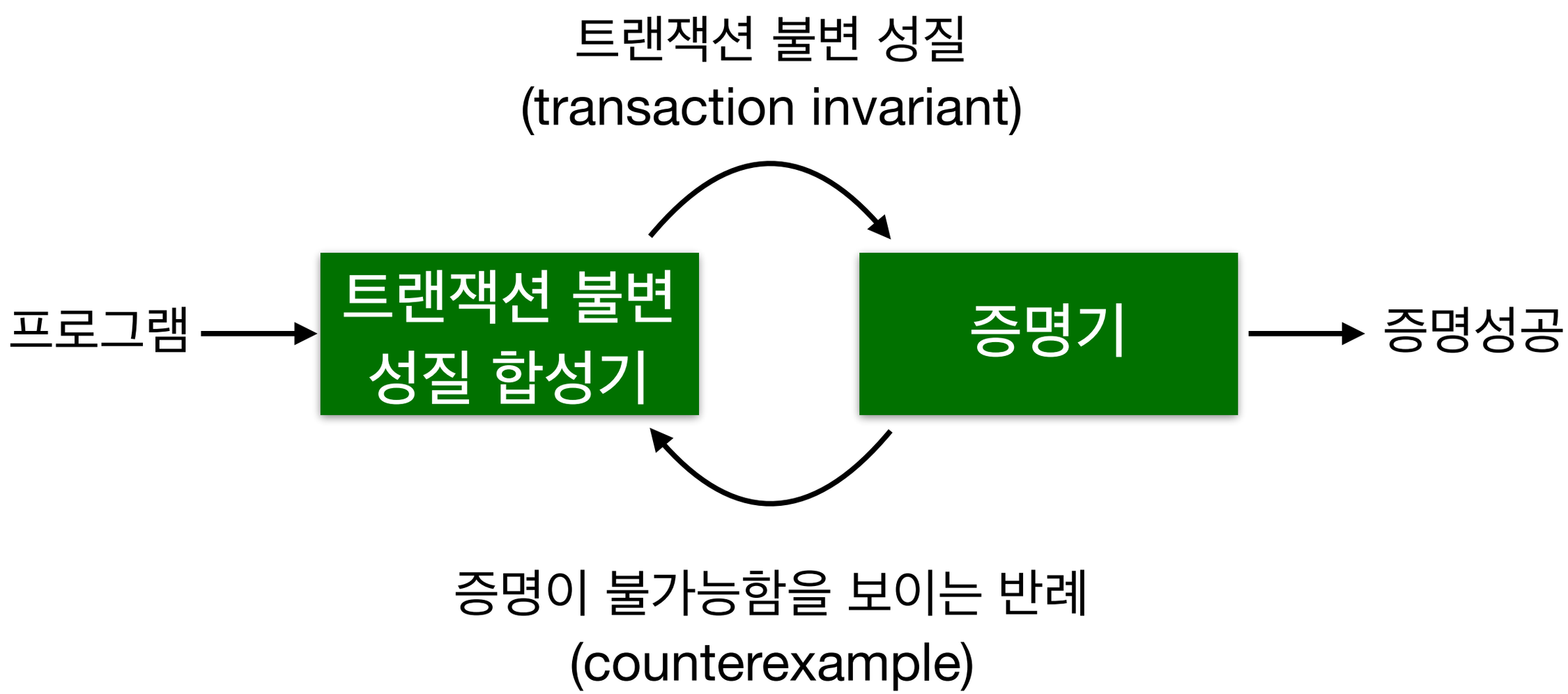
소프트웨어 증명 (Software Verification)



- 프로그램과 증명할 성질을 일차 논리식으로 표현
- 논리식의 satisfiability 여부를 판별

VeriSmart 검증 알고리즘

- 프로그램 증명과 불변 성질 합성을 동시에 진행



VeriSmart 검증 성능

- ZEUS[NDSS'18] 가 검증에 실패했던 13개 프로그램에 대해 예비 실험

프로그램	증명 대상 개수 (#queries)	Zeus	증명 쿼리 갯수 (트랜잭션 불변식 O)
zeus1	3	2	3
zeus2	3	2	3
zeus3	7	5	7
zeus4	6	3	6
zeus5	7	5	7
zeus6	7	5	7
zeus7	7	5	7
zeus8	7	5	7
zeus9	7	5	7
zeus10	5	2	5
zeus11	7	5	7
zeus12	3	2	3
zeus13	3	2	3
전체	**72**	**48**	**72**

Zeus가 증명에 실패한 13개 프로그램에 대해 모두 증명 성공

자동 패치 필요성 (Linux Kernel)


```
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;
```

자동 패치 필요성 (Linux Kernel)

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;
```

자동 패치 필요성 (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;
```

자동 패치 필요성 (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;
```

자동 패치 필요성 (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;
```

자동 패치 필요성 (Linux Kernel)

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;
```

소프트웨어 오류 자동 수정

```
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;
```

소프트웨어 오류 자동 수정

```
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

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

=> **수학적 보장.**
추가적인 리뷰 불필요.

```
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;
```

FSE'18

MemFix

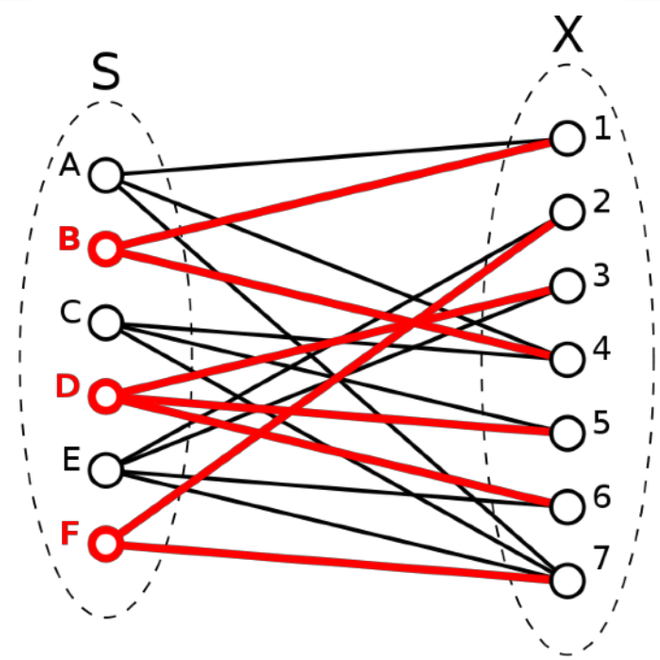
```
in = malloc(2);
if (in == NULL) {

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

  goto err;
}
```

Fixing memory errors
(undecidable)

approx. by
static analysis



Exact cover problem
(NP-complete)

SAT encoding

$$\varphi_1 = \bigwedge_{j=1}^{m} \bigvee_{i=1}^{n} T_{ij} \wedge S_i$$

$$\varphi_2 = \bigwedge_{j=1}^{m} \bigwedge_{i_1=1}^{n} \bigwedge_{i_2=1}^{n} \big((i_1 \neq i_2)$$

Boolean satisfiability
(NP-complete)

Automatic Feedback Generation for Programming Assignments

- In my programming language course,
  - students hardly receive personalized feedback, and
  - instructor's solutions are not very helpful.

모범 답안

```
let rec map f (l,var) =
    match l with
    | [] -> []
    | hd::tl -> (f (hd,var))::(map f (tl,var))
...
| Sum lst -> Sum (map diff (lst,var))
...
```

오답 코드

```
...
| Sum plus ->
 (match plus with
   []     -> Const 0
   | [hd]   -> diff( hd, var)
   | hd::tl -> Sum [diff(hd, var); diff(Times tl, var)]
 ) ...
```

Sum

학생 제출 답안

```
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
```

모범답안

```
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)
```

학생 제출 답안

```
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
```

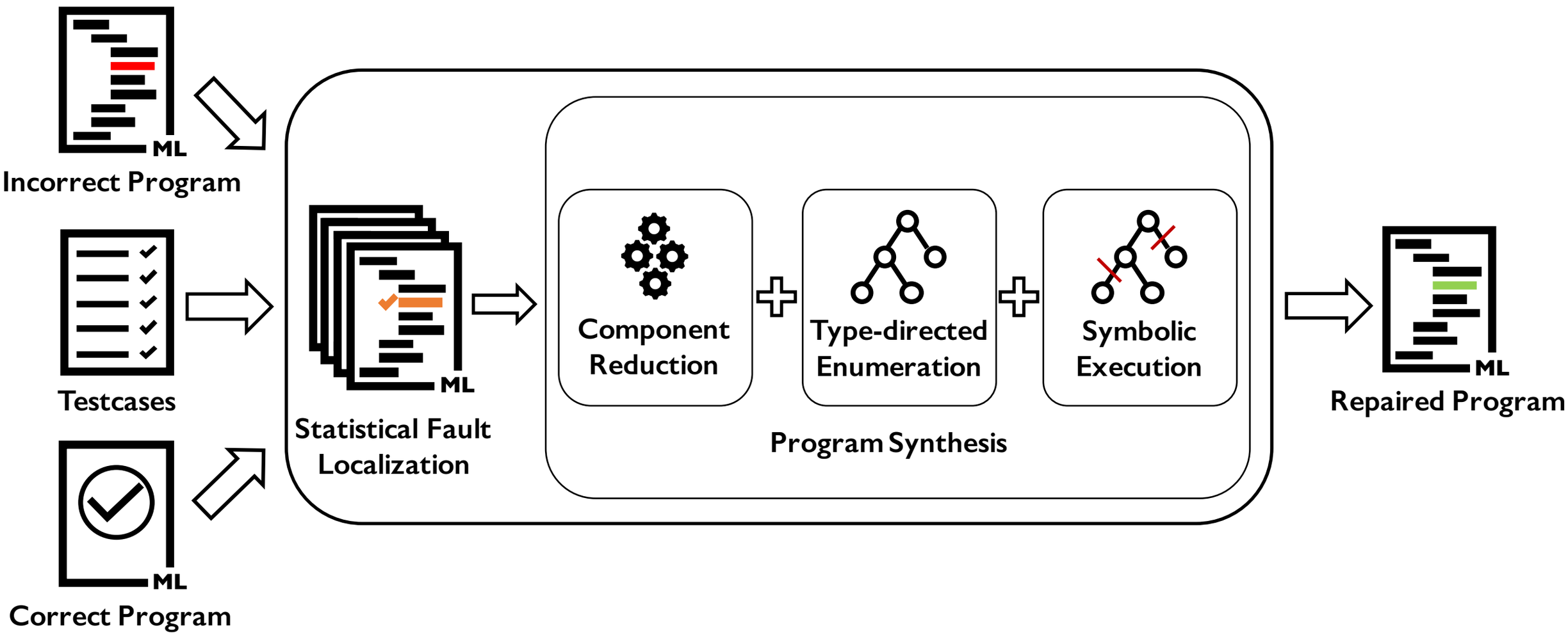
모범답안

```
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)
```

((Sum lst)::tl)

OOPSLA'18

The FixML System



SAS'17

Program Synthesis

- Can we automate the process of writing computer programs?

SAS'17

Program Synthesis

- Can we automate the process of writing computer programs?

$$\text{reverse}(12) = 21,\ \text{reverse}(123) = 321$$

```
reverse(n){
  r := 0;
  while (        ){
    [          ]
  };
  return r;
}
```

2.5s

```
reverse(n){
  r := 0;
  while (n > 0){
    x := n % 10;
    r := r * 10;
    r := r + x;
    n := n / 10;
  };
  return r;
}
```

Performance

- Better than humans for introductory programming tasks

Domain	No	Description	Vars		Ints	Exs	Time (sec)		
			IVars	AVars			Base	Base+Opt	Ours
Integer	1	Given n, return $n!$.	2	0	2	4	0.0	0.0	0.0
	2	Given n, return $n!!$ (i.e., double factorial).	3	0	3	4	0.0	0.0	0.0
	3	Given n, return $\sum_{i=1}^{n} i$.	3	0	2	4	0.1	0.0	0.0
	4	Given n, return $\sum_{i=1}^{n} i^2$.	4	0	2	3	122.4	18.1	0.3
	5	Given n, return $\prod_{i=1}^{n} i^2$.	4	0	2	3	102.9	13.6	0.2
	6	Given a and n, return a^n.	4	0	2	4	0.7	0.1	0.1
	7	Given n and m, return $\sum_{i=n}^{m} i$.	3	0	2	3	0.2	0.0	0.0
	8	Given n and m, return $\prod_{i=n}^{m} i$.	3	0	2	3	0.2	0.0	0.1
	9	Count the number of digit for an integer.	3	0	3	3	0.0	0.0	0.0
	10	Sum the digits of an integer.	3	0	3	4	5.2	2.2	1.3
	11	Calculate product of digits of an intger.	3	0	3	3	0.7	2.3	0.3
	12	Count the number of binary digit of an integer.	2	0	3	3	0.0	0.0	0.0
	13	Find the nth Fibonacci number.	3	0	3	4	98.7	13.9	2.6
	14	Given n, return $\sum_{i=1}^{n}(\sum_{m=1}^{i} m))$.	3	0	2	4	$\perp$	324.9	37.6
	15	Given n, return $\prod_{i=1}^{n}(\prod_{m=1}^{i} m))$.	3	0	2	4	$\perp$	316.6	86.9
	16	Reverse a given integer.	3	0	3	3	$\perp$	367.3	2.5
Array	17	Find the sum of all elements of an array.	3	1	2	2	8.1	3.6	0.9
	18	Find the product of all elements of an array.	3	1	2	2	7.6	3.9	0.9
	19	Sum two arrays of same length into one array.	3	2	2	2	44.6	29.9	0.2
	20	Multiply two arrays of same length into one array.	3	2	2	2	47.4	26.4	0.3
	21	Cube each element of an array.	3	1	1	2	1283.3	716.1	13.0
	22	Manipulate each element into 4th power.	3	1	1	2	1265.8	715.5	13.0
	23	Find a maximum element.	3	1	2	2	0.9	0.7	0.4
	24	Find a minimum element.	3	1	2	2	0.8	0.3	0.1
	25	Add 1 to each element.	2	1	1	3	0.3	0.0	0.0
	26	Find the sum of square of each element.	3	1	2	2	2700.0	186.2	11.5
	27	Find the multiplication of square of each element.	3	1	1	2	1709.8	1040.3	12.6
	28	Sum the products of matching elements of two arrays.	3	2	1	3	20.5	38.7	1.5
	29	Sum the absolute values of each element.	2	1	1	2	45.0	50.5	12.1
	30	Count the number of each element.	3	1	3	2	238.9	1094.1	0.2
	Average						$>$ 616.8	165.5	6.6

IJCAI'18

Synthesizing Pattern Programs

```
★★★★★
★★  ★★
★  ★  ★
★★  ★★
★★★★★
```

```
★★★★★★★
★★     ★★
★  ★  ★  ★
★   ★   ★
★  ★  ★  ★
★★     ★★
★★★★★★★
```

for i in N do:
  for j in N do:
    if ($i = 1 \,||\, i = N \,||\, j = 1 \,||\, j = i \,||$
      $j = N - i + 1 \,||\, j = N$): print ★
    else: print ␣
  print ↵

```
★     ★     ★
 ★ ★ ★ ★
  ★     ★
```

```
★       ★       ★
 ★     ★ ★     ★
  ★   ★   ★   ★
   ★       ★
```

for i in N do:
  for j in $4 * N - i - 2$ do:
    if ($j = 2 * N - i \,||\, j = 2 * N + i - 2 \,||$
      $j = 4 * N - i - 2 \,||\, j = i$): print ★
    else: print ␣
  print ↵

고려대학교 소프트웨어 분석 연구실

- Research areas: programming languages, software engineering, software security
  - program analysis and testing
  - program synthesis and repair
- Publication: top-venues in PL, SE, Security, and AI:
  - PLDI('12,'14), OOPSLA('15,'17,'17,'18,'18), TOPLAS('14,'16,'17,'18,'19), ICSE('17,'18), FSE'18, ASE'18, S&P'17, IJCAI('17,'18), etc
- We are recruiting graduate students and undergrad research interns!

http://prl.korea.ac.kr