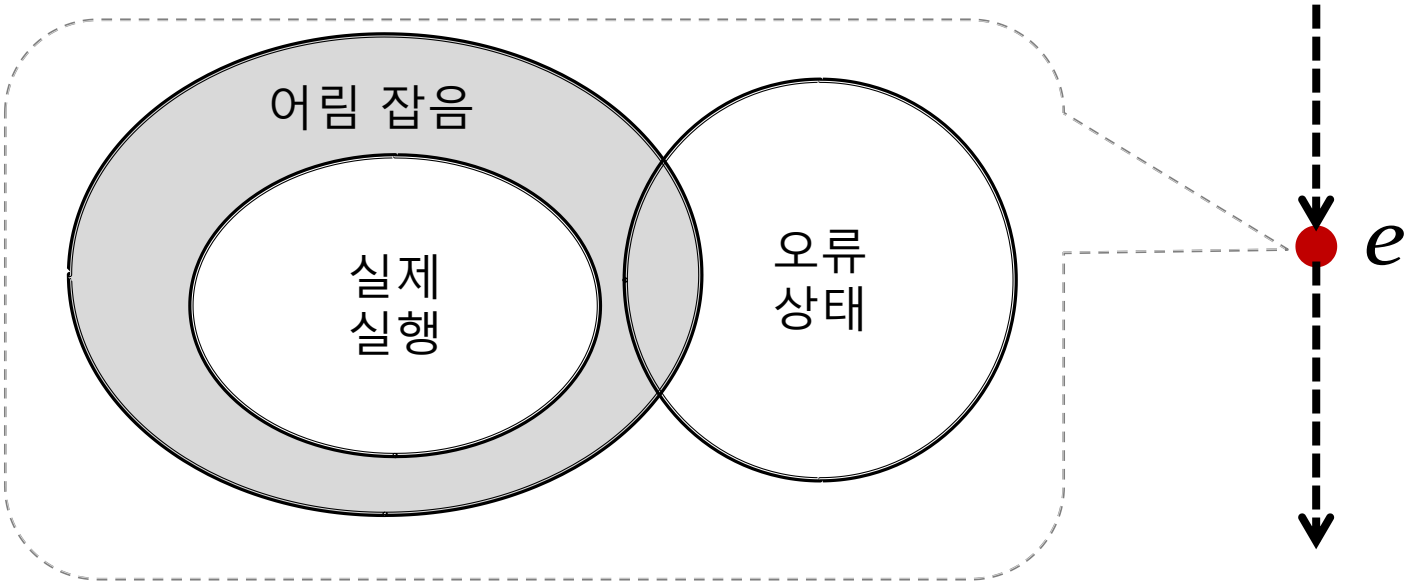
정적 분석 경보 사이의 종속관계를 이용한 대표 경보 찾기

이우석, 김유일
프로그래밍 연구실
서울대학교

거짓 경보란?



• 정적 분석 : 프로그램이 실행되면서 실제 도달 가능한 기계 상태를 프로그램을 실행하지 않고 엄밀하게 모음.

• 어림 잡지 않고 정확히 모으려면 분석이 끝나지 않을 수 있음.

해결 방안

- 중요한 경보를 먼저 보여주는 방법
  - Kremenek, T., “Using statistical analysis to counter the impact of static analysis approximations.”
- 더 정확한 분석을 적용하는 방법
  - Y. Kim et al., “Filtering false alarms of bufferoverflow analysis using smt solvers.”
- **경보 간 관계를 이용하여 대표 경보만 보여주는 방법**

경보들 사이의 관계

- 우리 방법으로 자동으로 찾은 경보들 사이의 관계들 예제

```
invmergerules[8];
…
int lookup_mergearcs (char *rule)
{
  for (int i = 1; invmergerules[i]; i++)
    if (strcmp(rule,…))
       return (i);
  return (0);
}
```

```
int apply_rule (char *rule, struct sketch *sketch)
{
…
else if ((code = lookup_mergearcs (rule)))
     res = rule_mergearcs (sketch, code, rcount);
```

code = [0,7]

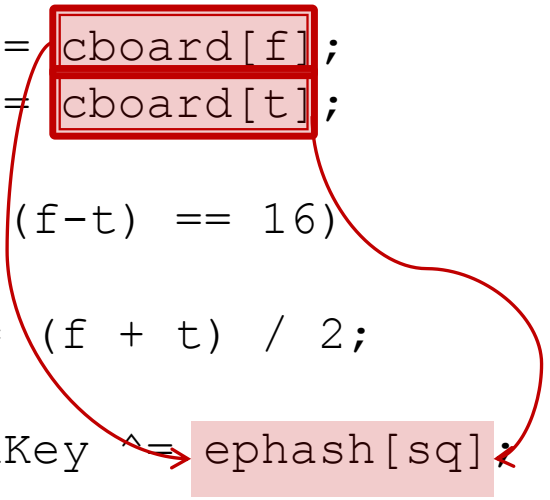
```
int rule_mergearcs
(struct sketch *s, int rule, int rcount)
{
…
 if (debug)
    printf ("%s count %d\n",invmergerules[rule], rcount);
```

rule = [0,7]

```
cboard[64];
epash[64];

void MakeMove (int side, int *move)
{
f = …;
t = …;
fpiece = cboard[f];
tpiece = cboard[t];
…
if (abs(f-t) == 16)
{
    sq = (f + t) / 2;

    HashKey ^= ephash[sq];
}
```

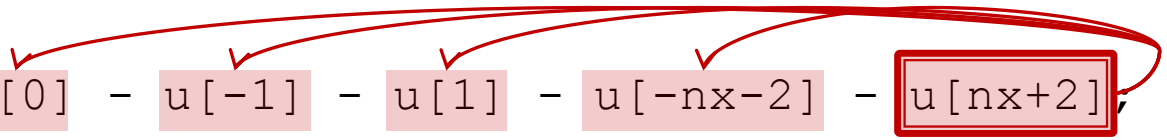


```
upad [21632];

void residual (double *u, *ac, *ax, *ay, *q;)
{
  int nx = 50;

  u  = &upad[nx+2];

  for  (…)
  {
     r[0] = u[0] - u[-1] - u[1] - u[-nx-2] - u[nx+2];
     u++;
      …
  }
}
```



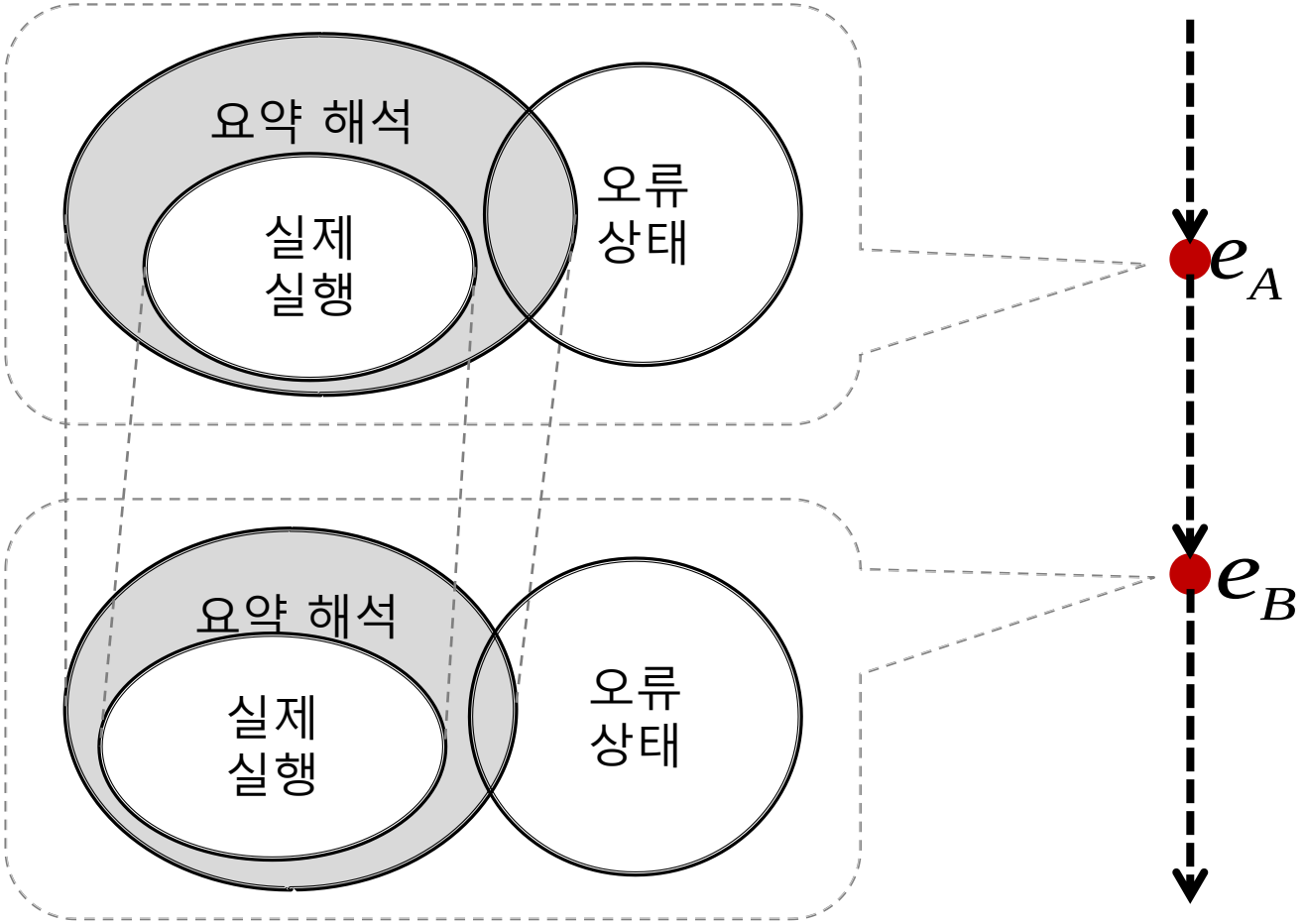
```
static int is_unterminate_line(const char *line)
{
    int i, len;
    len = strlen(line);
    if (line[len - 1] != '\n')
        return 0;
    for (i = len - 2; i >= 0; i--) {

        if (line[i] == '\\') {
            return 1;
        }
    …
}
```

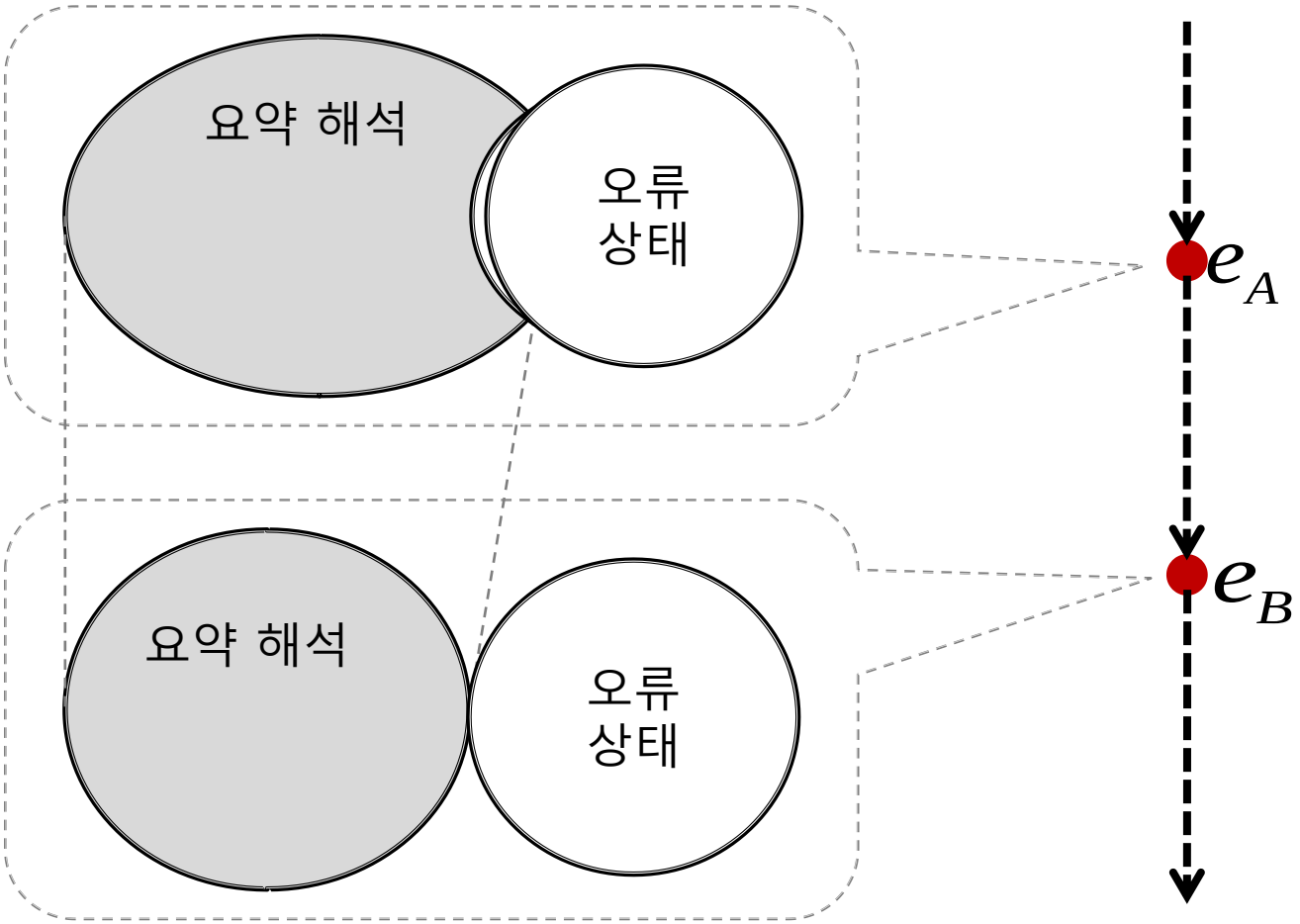
방법 및 이론

- 경보 메시지간의 종속 관계를 안전하게 계산하는 틀

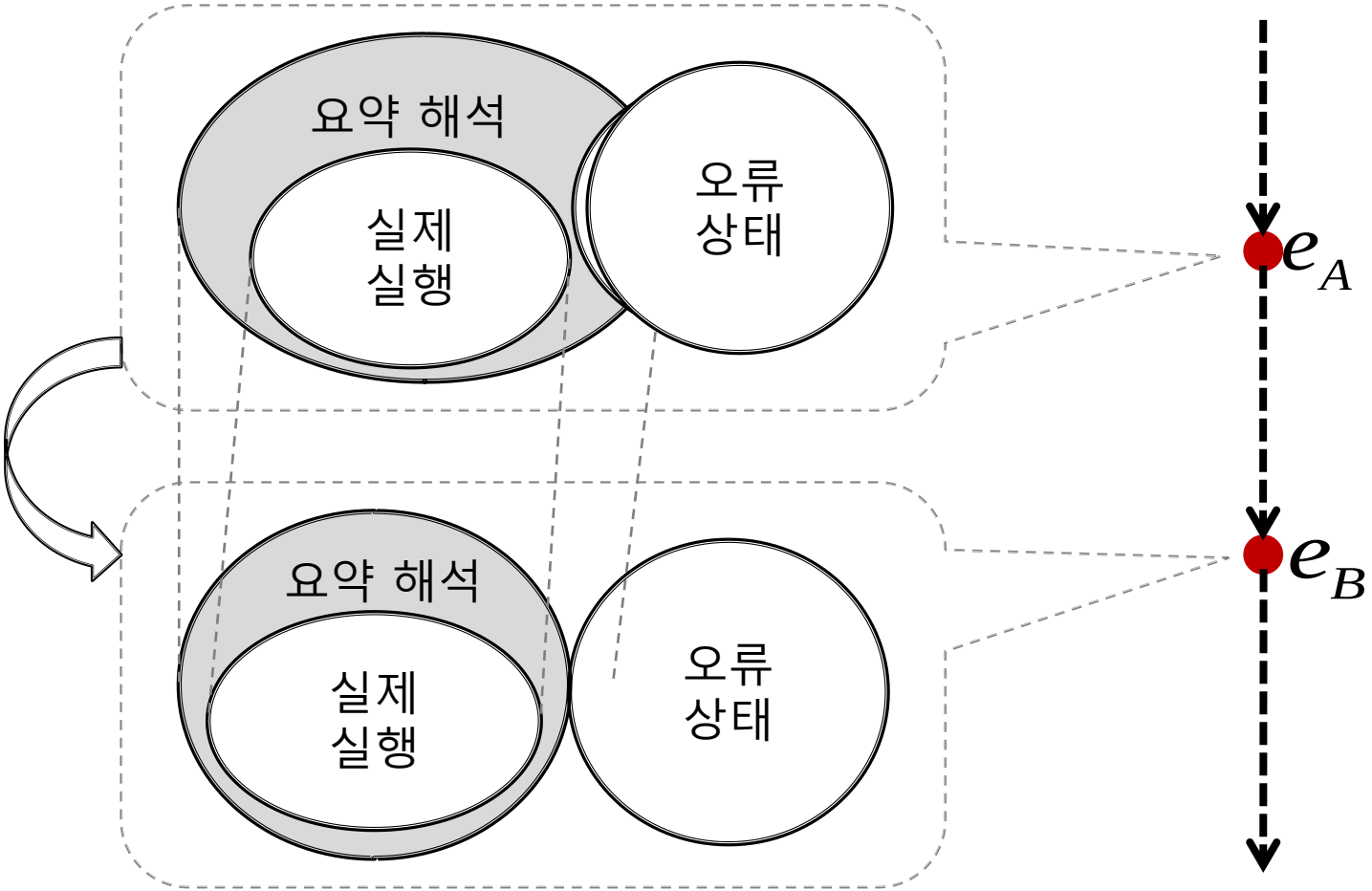
거짓 경보



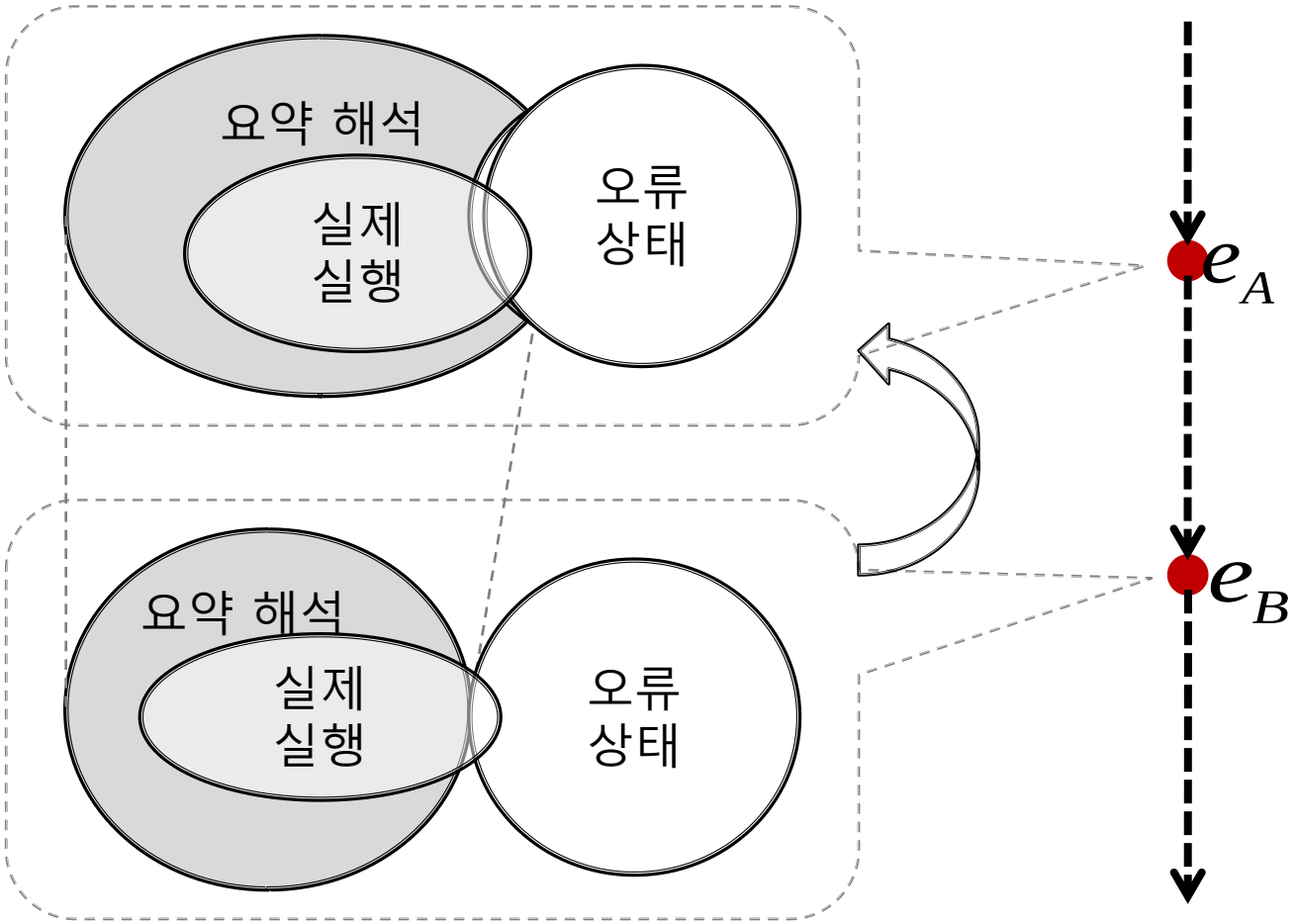
거짓 경보



거짓 경보간의 종속 관계



오류간의 종속 관계

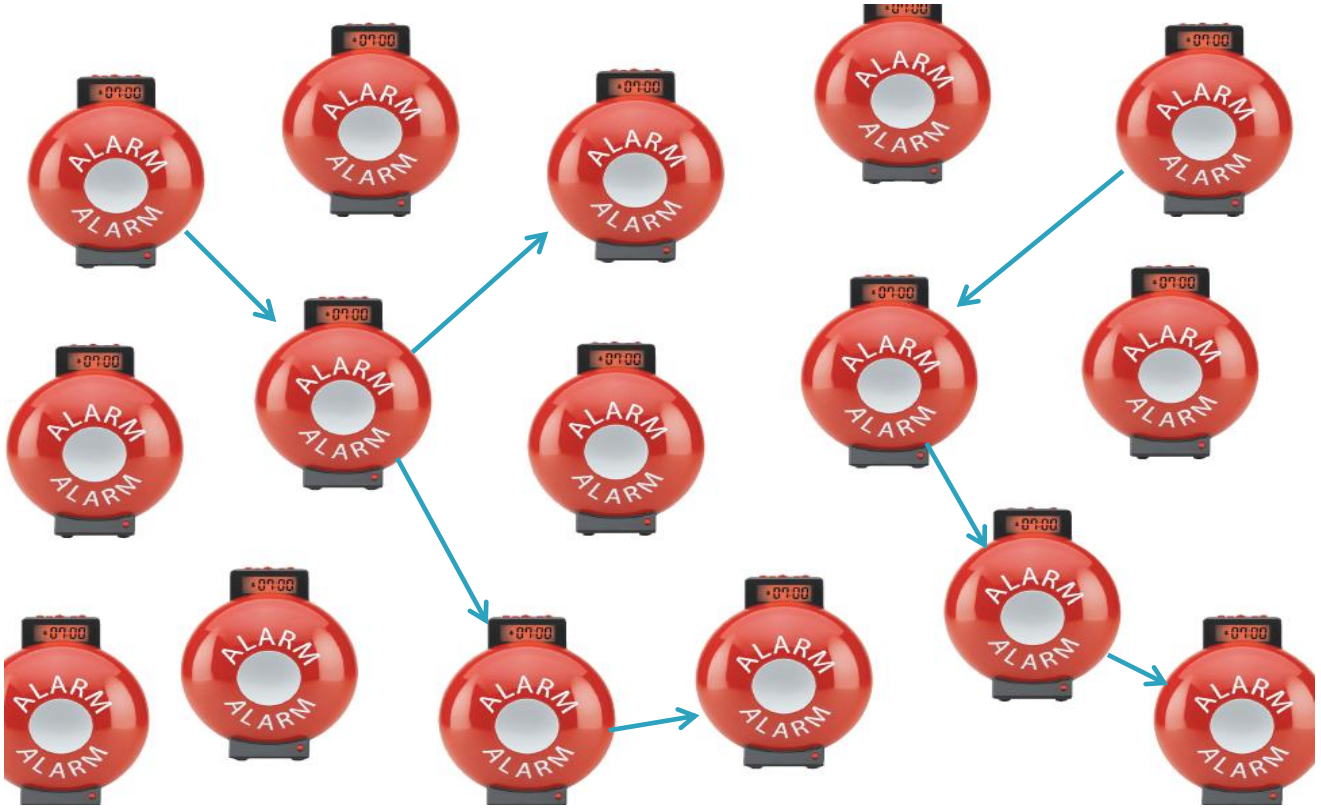


경보간의 종속 관계

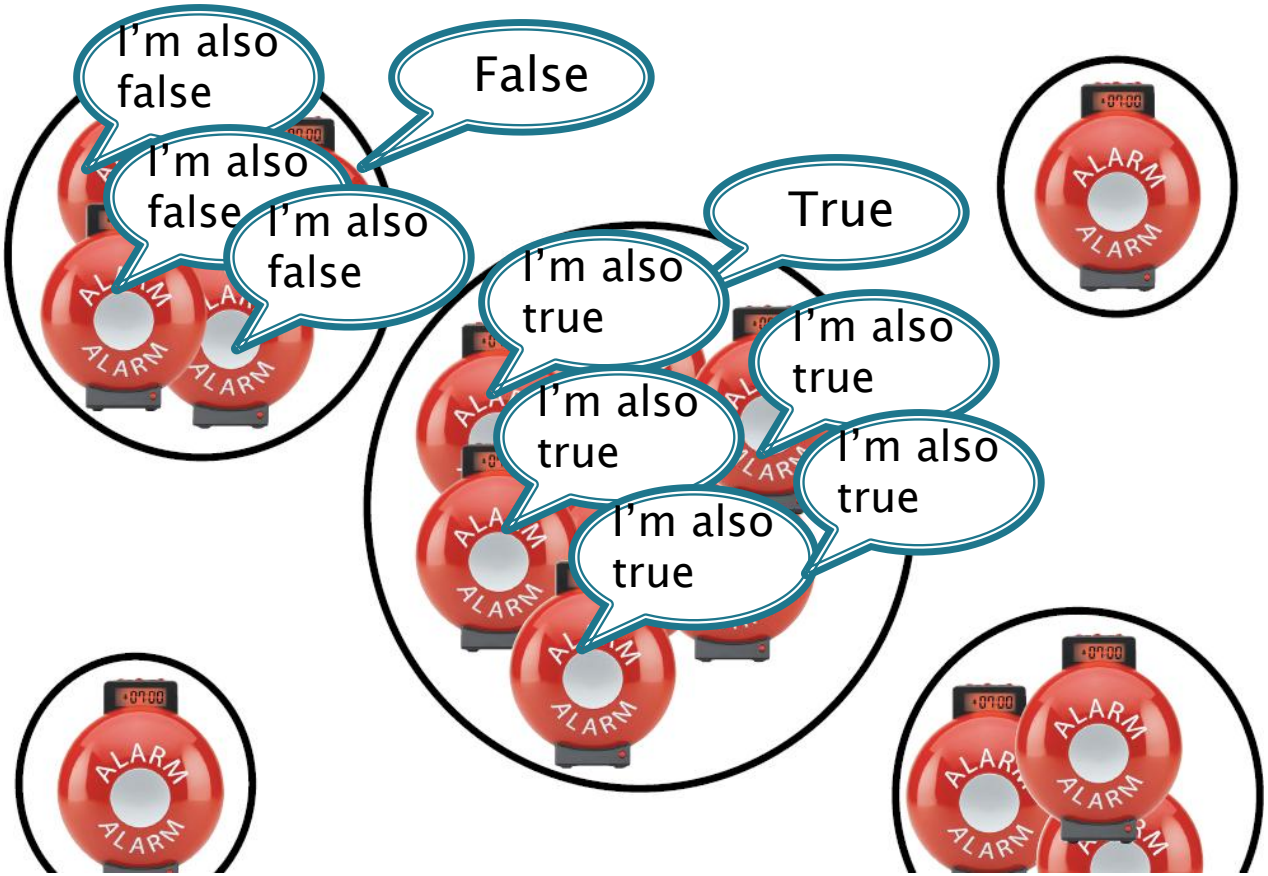
- 가정: 앞의 조건을 만족하는 경우에
- 정리 1: 거짓 경보간의 종속 관계
  - 경보 1이 가짜면, 경보 2도 가짜다.
- 정리 2: 오류간의 종속 관계
  - 경보 2가 진짜면, 경보 1도 진짜다.
- 두 정리는 모든 요약 해석에 일반적으로 적용 가능

종속 관계를 이용한 대표경보 찾기

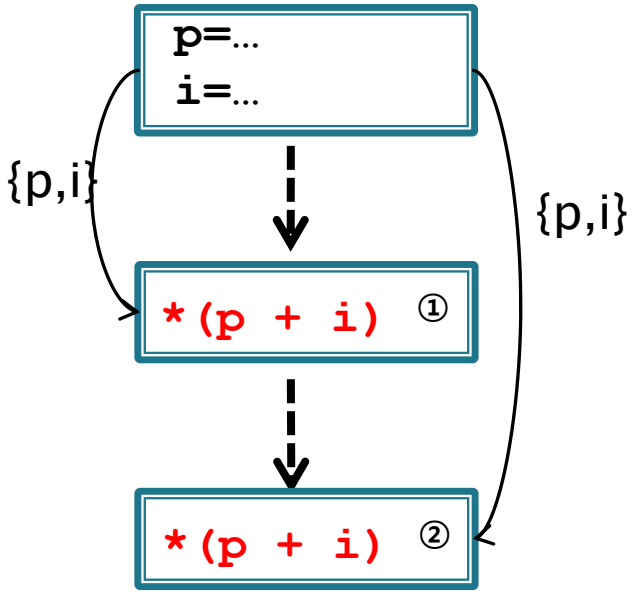
종속 관계를 이용한 대표경보 찾기

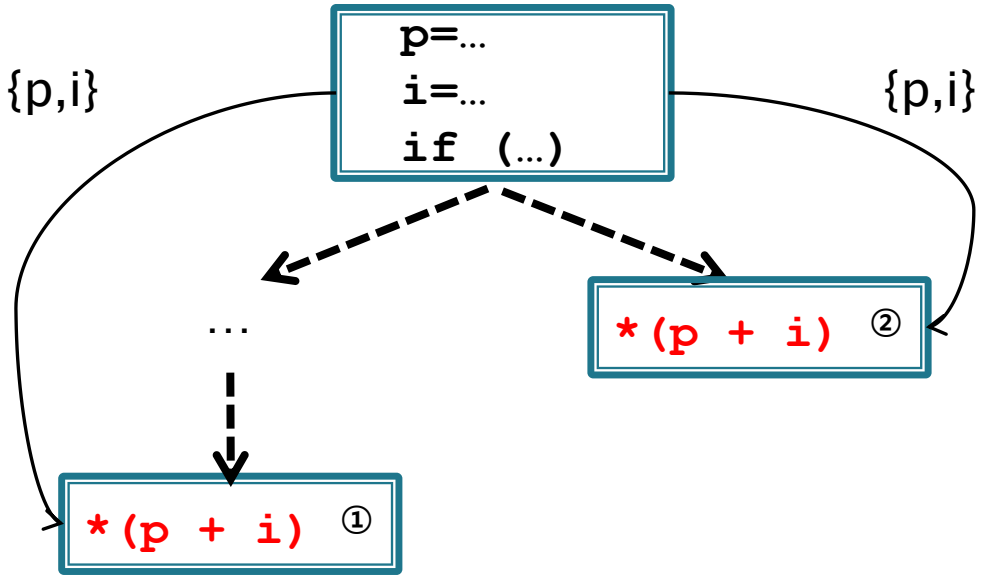


종속 관계를 이용한 대표경보 찾기



변수 정의 지점을 이용한 경보 동일 군집





- 겉보기에 똑같은 두 경보에 대해서,
  - 그들을 구성하는 변수들의 정의 지점이 같은지 검사. ➔ 같은 것들은 동일한 경보로 묶음

반박 가정을 이용한 관계 유추

```
void main()
{
    int small[5];
    int big[10];
    int i;
    i = complex_function();

    /* i = [0, +∞] */
    small[i]② = …;
    REFINE(0 <= i <= 4, alarm 2)
    big[i]① = …;
    REFINE(0 <= i <= 9, alarm 1)
}
```

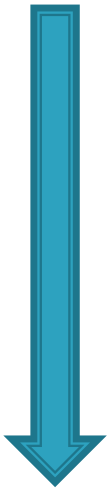
- 각 경보가 거짓이라는 가정 및 그에 따른 요약 상태 정보를 경보 지점들 바로 뒤에 삽입.

반박 가정을 이용한 관계 유추

```
void main()
{
    int small[5];
    int big[10];
    int i;
    i = complex_function();

    /* i = [0, +∞] */
    small[i]② = …;
    REFINE(0 <= i <= 4, alarm 2)
    big[i]① = …;
    REFINE(0 <= i <= 9, alarm 1)
}
```

- 분석 진행



반박 가정을 이용한 관계 유추

```
void main()
{
    int small[5];
    int big[10];
    int i;
    i = complex_function();

    /* i = [0, +∞] */
    small[i]② = …;
    REFINE(0 <= i <= 4, alarm 2)
    big[i]① = …;
    REFINE(0 <= i <= 9, alarm 1)
}
```

- 다른 경보의 반박가정에 의해 어떤 알람이 사라짐

→ 사라진 알람이 거짓이 가정된 경보에 종속

실험 결과

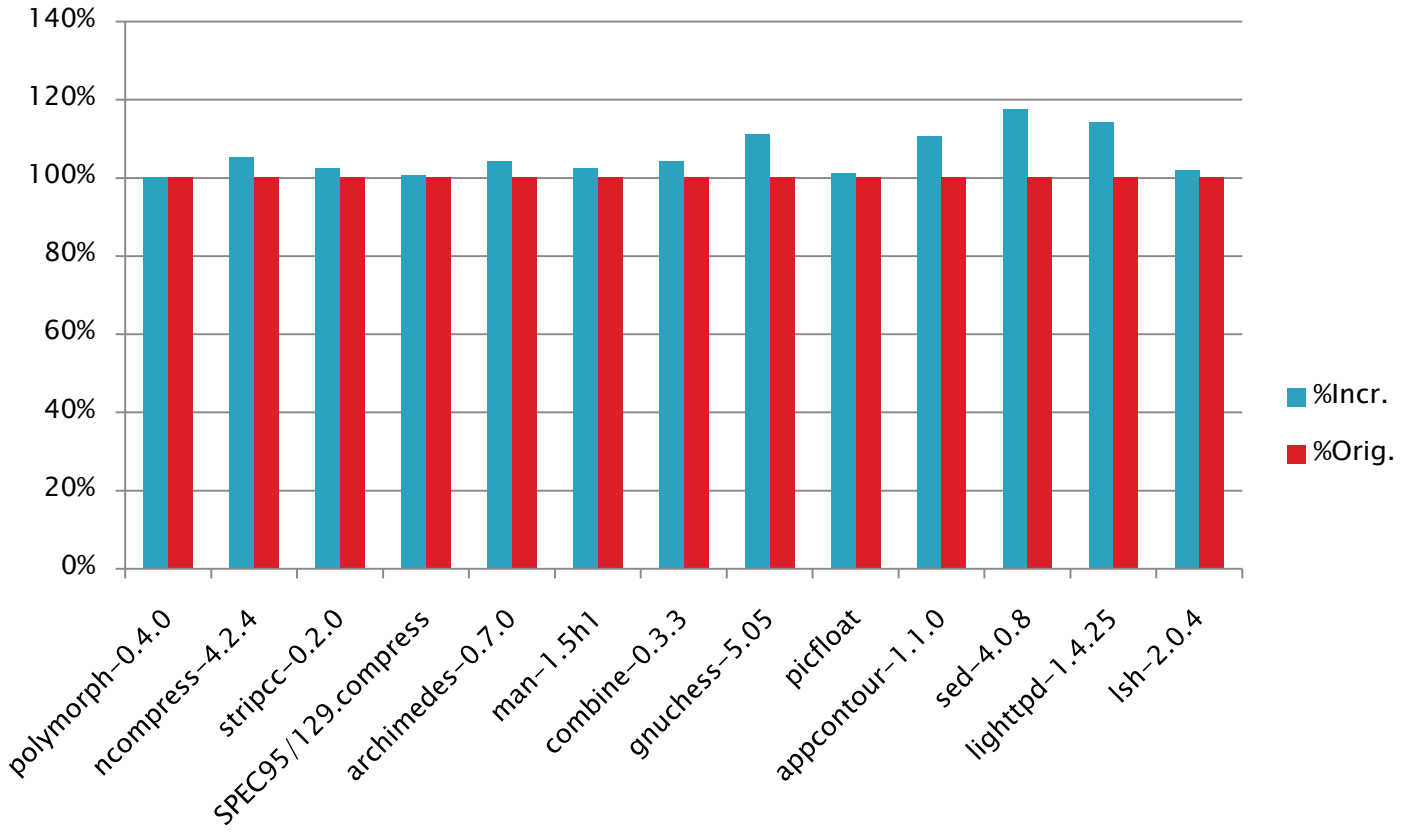
- 이러한 관계들을 이용하여 경보 수를 얼마나 줄였나?
  (Airac 버퍼 오버런 경보 메시지 출력을 개선)

실험 결과

프로그램	라인 수	본래 경보 수	최종 경보 수	감소 비율(%)
polymorph-0.4.0	1,357	25	13	48%
ncompress-4.2.4	2,195	66	37	44%
stripcc-0.2.0	2,555	247	155	37%
SPEC95/129.compress	5,585	57	25	56%
man-1.5h1	7,232	276	176	36%
archimedes-0.7.0	7,569	711	215	70%
combine-0.3.3	11,472	733	297	59%
gnuchess-5.05	11,629	972	311	68%
picfloat	11,959	41	21	49%
appcontour-1.1.0	14,930	533	365	32%
sed-4.0.8	25,903	843	562	33%
lighttpd-1.4.25	56,518	1481	652	56%
lsh-2.0.4	110,898	616	317	49%

평균 52% 감소

추가적인 분석 시간



평균 3% 증가

기존 기술과 결합

- “보다 정확한 후 분석을 통한 허위 경보 제거”기술과 결합

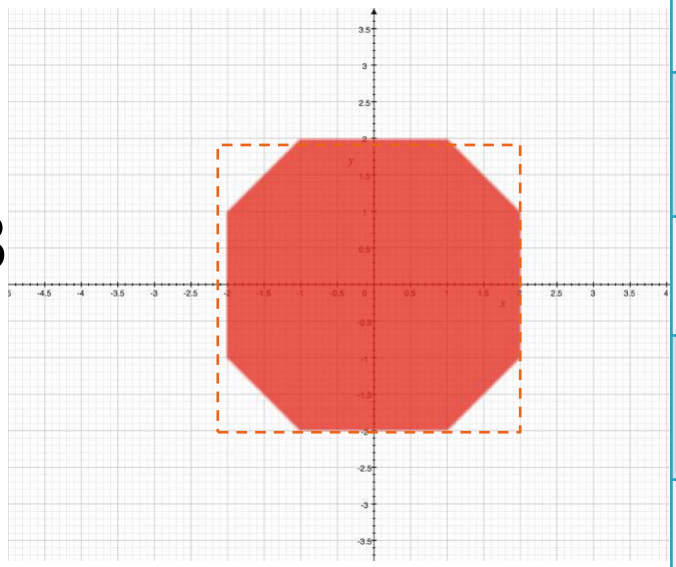
옥타곤 후 분석(post-analysis)

- 무인비행체 소프트웨어에 특화된 분석기에 사용된 방법
- 옥타곤 분석 : 각 변수가 갖는 값의 범위만 찾는 인터벌 분석과 달리 변수들 사이의 관계를 분석
- 인터벌 분석 후 경보가 난 지점에 선택적으로 더 정확한 옥타곤 분석을 적용

옥타곤 도메인

$$\pm X \pm Y \le c \quad (X, Y \in Variable,\ c \in \mathbb{Z}, \mathbb{W} \text{ or } \square)$$

- $-2 \le x \le 2$
- $-2 \le y \le 2$
- $-3 \le x+y \le 3$
- $-3 \le x-y \le 3$



	+x	−x	+y	−y
+x	0	4	3	3
−x	4	0	3	3
+y	3	3	0	4
−y	3	3	4	0

옥타곤 후 분석(post-analysis)

프로그램	라인 수	본래 경보 수	옥타곤 분석 후	허위 경보 제거 비율
BIND-1	931	46	42	9%
BIND-2	1,095	42	40	5%
BIND-3	380	2	2	0%
BIND-4	645	21	11	48%
FTP-1	503	2	2	0%
FTP-2	744	43	43	0%
FTP-3	689	27	11	59%
TOTAL	4987	183	151	17%

* Zitser가 제시한 버퍼오버런 벤치마크(bind, wu-ftpd 에서 추출)

옥타곤 후 분석 + 대표 경보 구하기

프로그램	라인 수	본래 경보 수	대표 경보 구하기 후	옥타곤 분석 후	허위 경보 제거 비율
BIND-1	931	46	38	34	26%
BIND-2	1,095	42	40	38	10%
BIND-3	380	2	2	2	0%
BIND-4	645	21	8	5	76%
FTP-1	503	2	2	2	0%
FTP-2	744	43	33	33	23%
FTP-3	689	27	11	11	59%
TOTAL	4987	183	134	125	**32%**

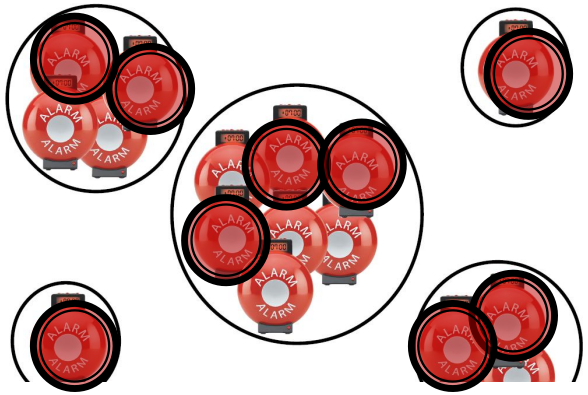
- 옥타곤 후 분석에 우리 방법 접목 후
  - 옥타곤 후 분석 시간 20% 감소
    (42.61초→ 33.93초)

앞으로의 연구 계획

- 관계 분석을 이용한 종속 관계 유추

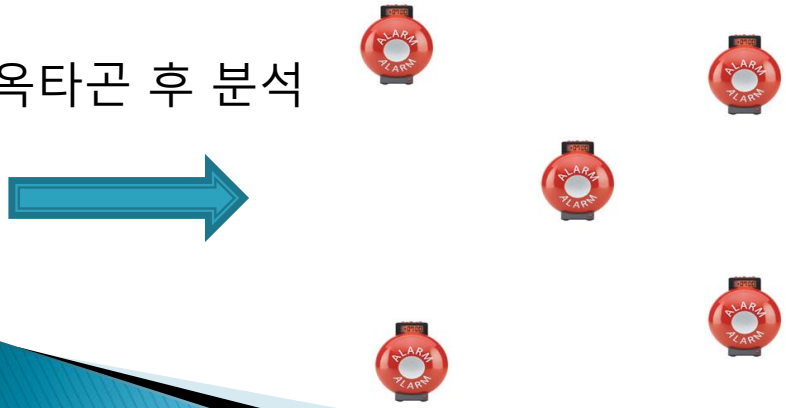
계층적으로 경보 줄이기

인터벌 그룹핑

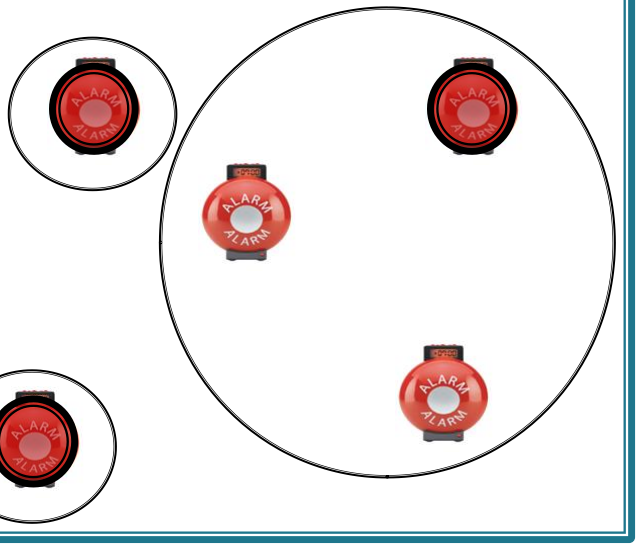


TODO

옥타곤 후 분석



옥타곤 그룹핑



옥타곤 분석을 이용한 관계찾기

```
int Mvboard[64]
t = complex_function(...)
...
g->mvboard = Mvboard[t];
if (t & 0x04)     /* King side */
 {
    rookf = t + 1;
    rookt = t - 1;
 }
 else          /* Queen side */
 {
    rookf = t - 2;
    rookt = t + 1;
 }
((*a) |= BitPosArray[rookf])

((*a) &= NotBitPosArray[rookt])
```

BitPosArray.size = NotBitPosArray.size

$t - 2 \le rookf \le t + 1$

$t - 1 \le rookt \le t + 1$

$0 \le t - 2 \le rookf \le t + 1 \le size - 1$

$\Rightarrow 1 \le t - 1 \le rookf \le t + 1 \le size - 1$

(from gnuchess-5.05)

결론

- 경보 간 종속 관계를 안전하게 계산하는 틀
  - 보고되는 정적 분석 경보의 수를 줄임.
  - 결과를 믿을 수 있음 (통계적 방법과 차이)
  - 추가 비용이 낮은 편
    - 3% 추가시간으로 52% 경보 감소
  - 기존의 기술과 잘 어울림
    - 옥타곤 후 분석의 추가 시간을 낮추고, 함께 작용하여 허위 경보 제거 효과를 높임