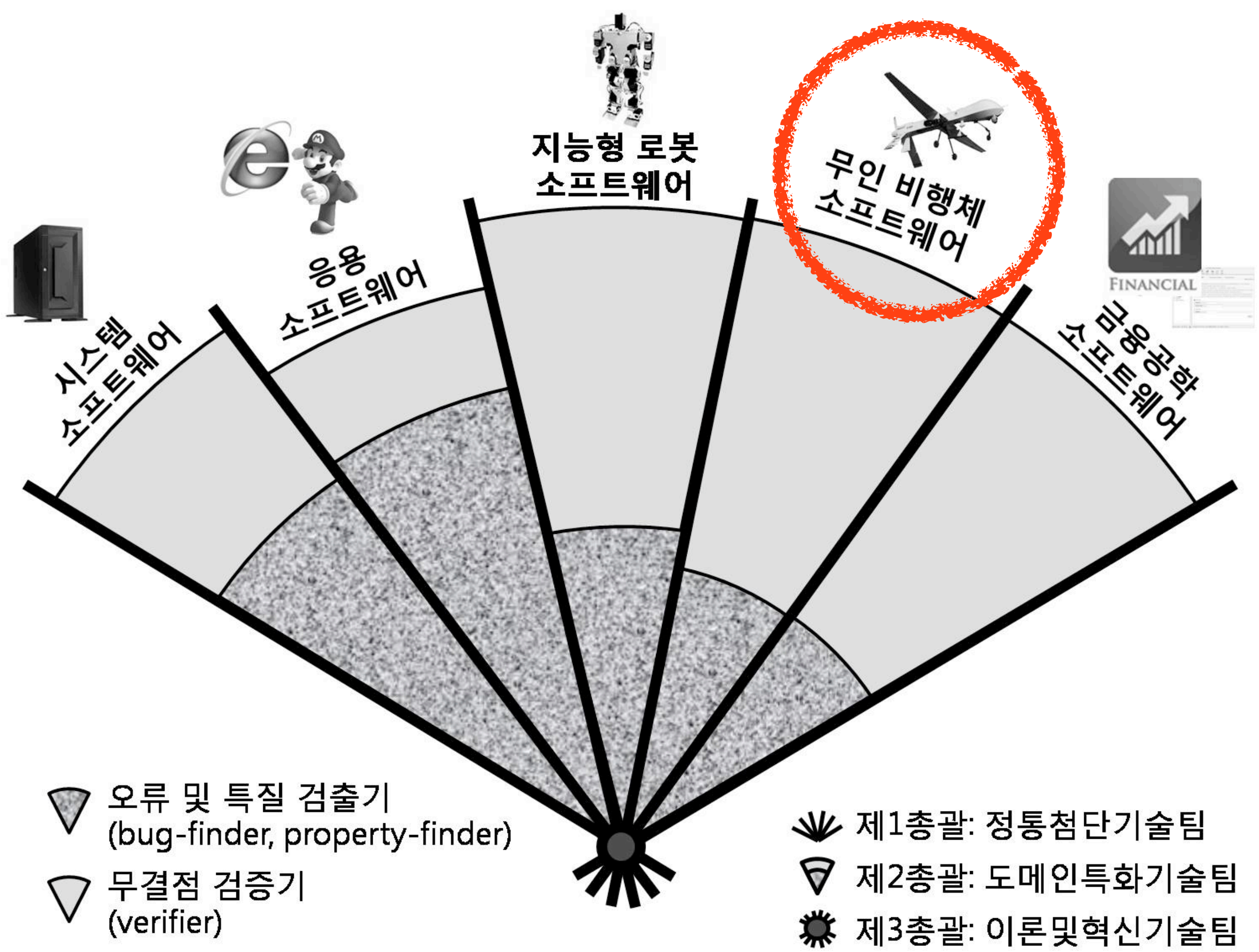
Specialized Static Analyzer for UAV

Suwon Jang Woosuk Lee

2010/01/07
ROSAEC 3rd Workshop

Motivation

- Improve general purpose analyzer difficult
- Developing domain-specific analyzer

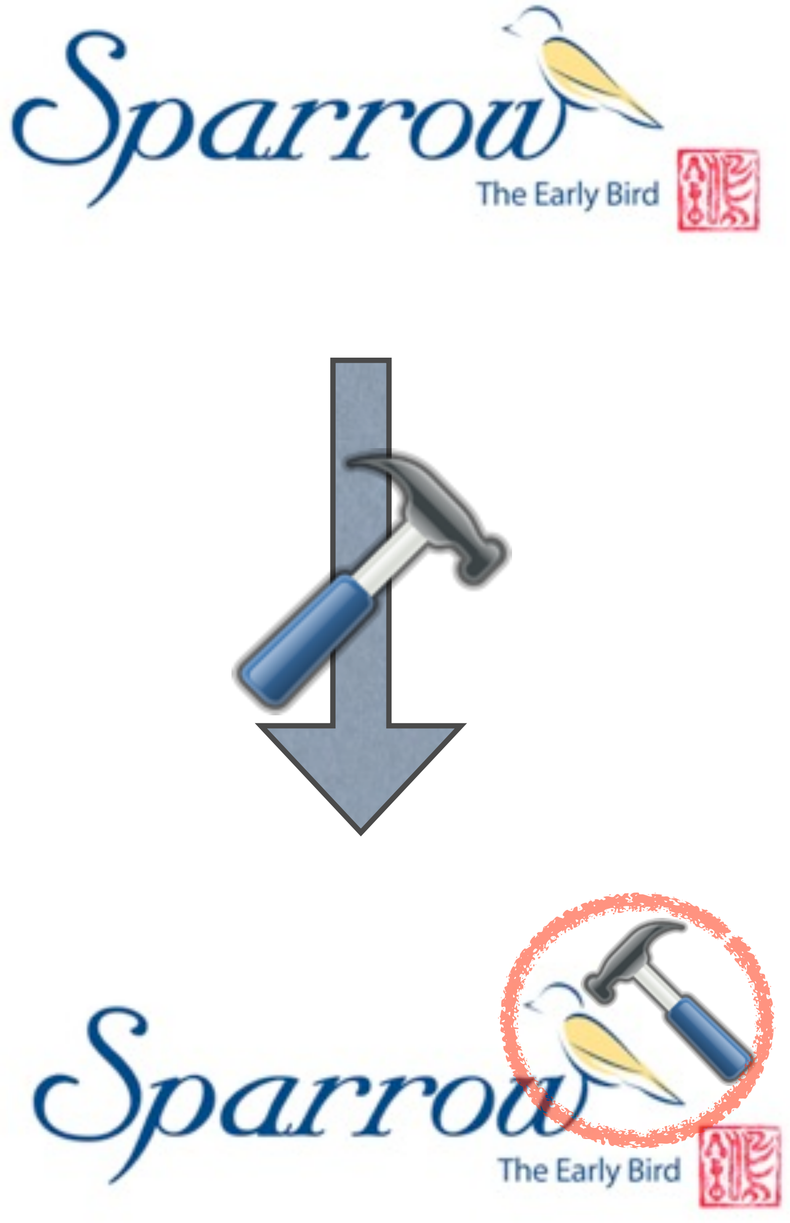


Goal

- Find all buffer overrun bugs
- Without false alarm
- For only one specific target software

Target = FCC program provided by
Dynamics and Control Lab in SNU

Approach



General purpose static analyzer

- Run it!
- Inspect every alarm manually
- Refine it to generate less false alarms

Specialized static analyzer

Current Status

- True alarms : 17
- False alarms : 28 → 10
- 2 types of false alarm are still remaining

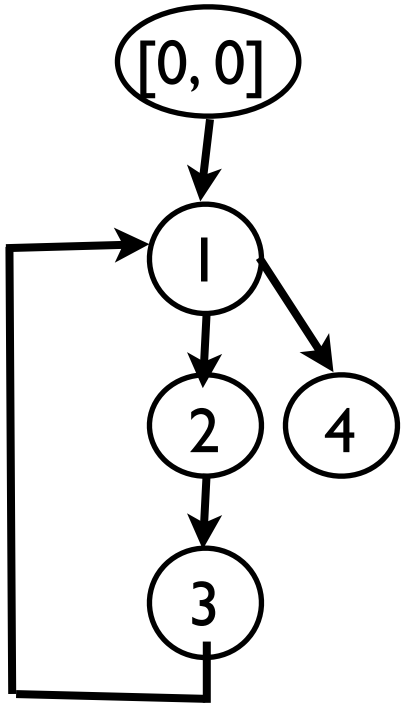
Simple Tuning

- Code is very small (~5500 lines of C)
- 7s. for global analysis
- High cost analysis method can be used
- Delay widening is very effective!

Analysis: Example

- Execute source code in abstract domain
  - e.g. integer interval
- Collect all possible values until fixpoint reached

```
i=0;
/* 1 */
while(i<2) {
  /* 2 */
  i++;
  /* 3 */
}
/* 4 */
```

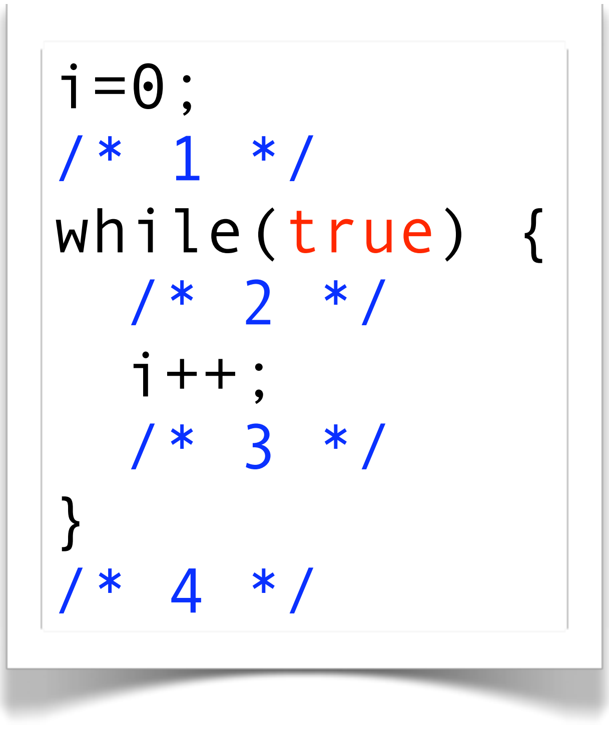


fixpoint!

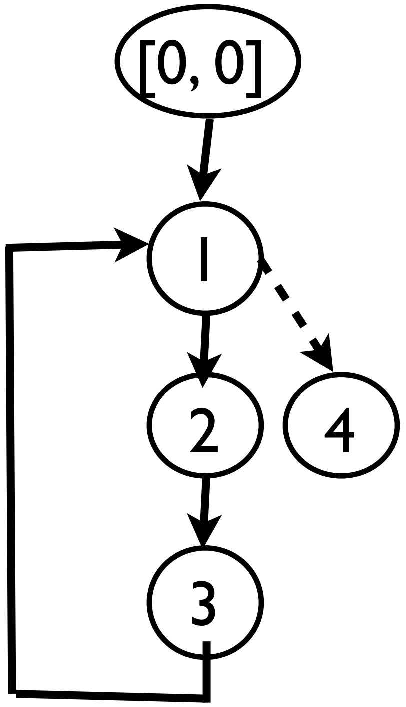
	1	2	3	4
i1	[0,0]	[0,1]	[0,2]	[0,2]
i2	[0,0]	[0,1]	[0,1]	[0,1]
i3	[1,1]	[1,2]	[1,2]	[1,2]
i4	⊥	⊥	[2,2]	[2,2]

Analysis: Example

- Infinite loop? Not terminate



```
i=0;
/* 1 */
while(true) {
   /* 2 */
   i++;
   /* 3 */
}
/* 4 */
```

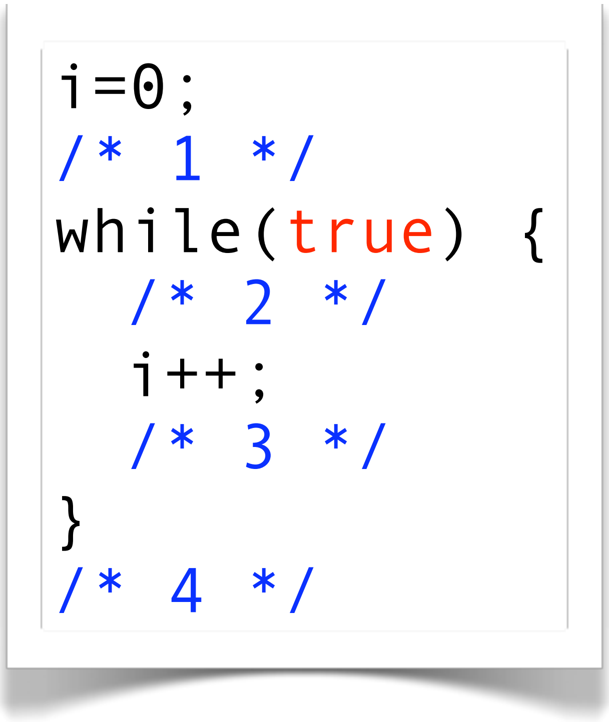


NOT terminate

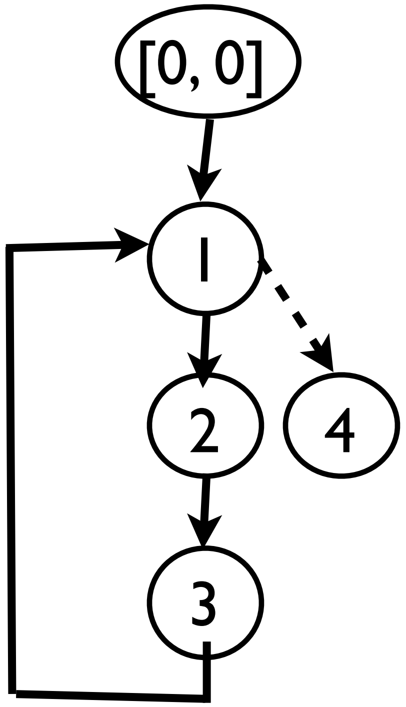
	1	2	3	4	...
i1	[0,0]	[0,1]	[0,2]	[0,3]	...
i2	[0,0]	[0,1]	[0,2]	[0,3]	...
i3	[1,1]	[1,2]	[1,3]	[1,2]	...
i4	⊥	⊥	⊥	⊥	⊥

Analysis: Example

- Infinite loop? Not terminate
- Solution : Widening



```
i=0;
/* 1 */
while(true) {
   /* 2 */
   i++;
   /* 3 */
}
/* 4 */
```



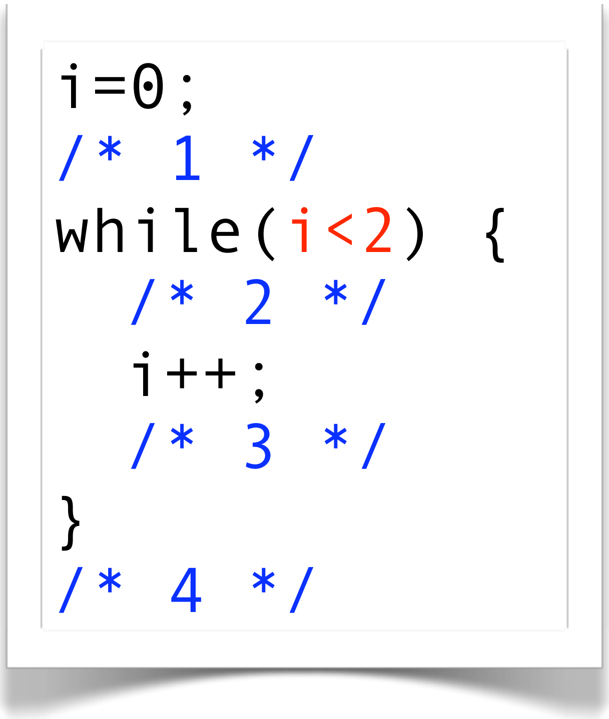
Widening

fixpoint

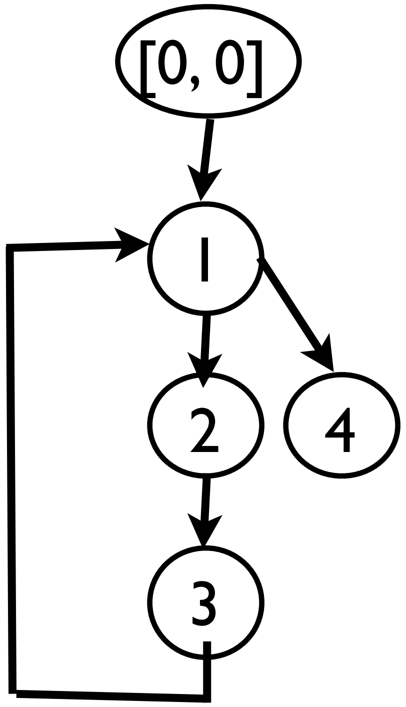
	1	2	3
i1	[0,0]	[0,∞]	[0,∞]
i2	[0,0]	[0,∞]	[0,∞]
i3	[1,1]	[1,∞]	[1,∞]
i4	⊥	⊥	⊥

Analysis: Example

- Widening : precision loss
- Solution : Narrowing



```
i=0;
/* 1 */
while(i<2) {
  /* 2 */
  i++;
  /* 3 */
}
/* 4 */
```



precision loss

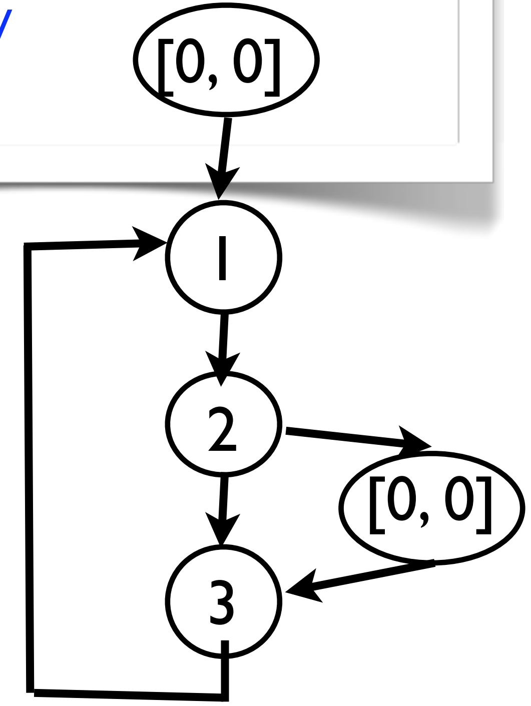
precision recover

	1	2	3	4	5
i1	[0,0]	[0,∞]	[0,∞]	[0,∞]	[0,2]
i2	[0,0]	[0,∞]	[0,∞]	[0,1]	[0,1]
i3	[1,1]	[1,∞]	[1,∞]	[1,2]	[1,2]
i4	⊥	[2,∞]	[2,∞]	[2,∞]	[2,2]

Analysis: Example

- BUT Narrowing MAY not recover precision

```
i=0;
while(1) {
  /* 1 */
  i++;
  /* 2 */
  if (2 == i) i = 0;
  /* 3 */
}
```



widened

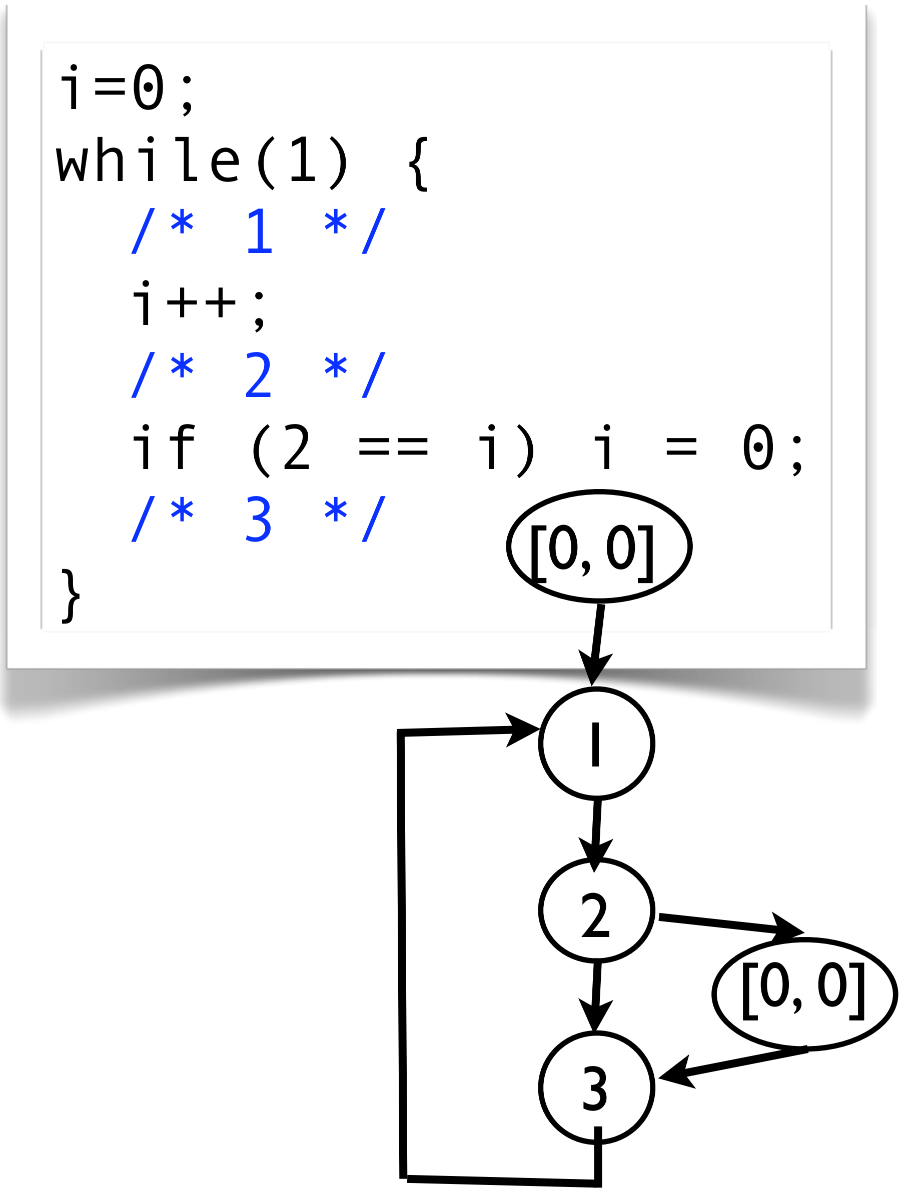
fixpoint

narrowing not working (i ≠2 ∧ [1,∞] = ?)

	1	2	3	4
i1	[0,0]	[0,∞]	[0,∞]	[0,∞]
i2	[1,1]	[1,∞]	[1,∞]	[1,∞]
i3	[1,1]	[1,∞]	[1,∞]	[1,∞]

Analysis: Example

- Delay widening ➔ increase precision



```
i=0;
while(1) {
  /* 1 */
  i++;
  /* 2 */
  if (2 == i) i = 0;
  /* 3 */
}
```

widening delayed

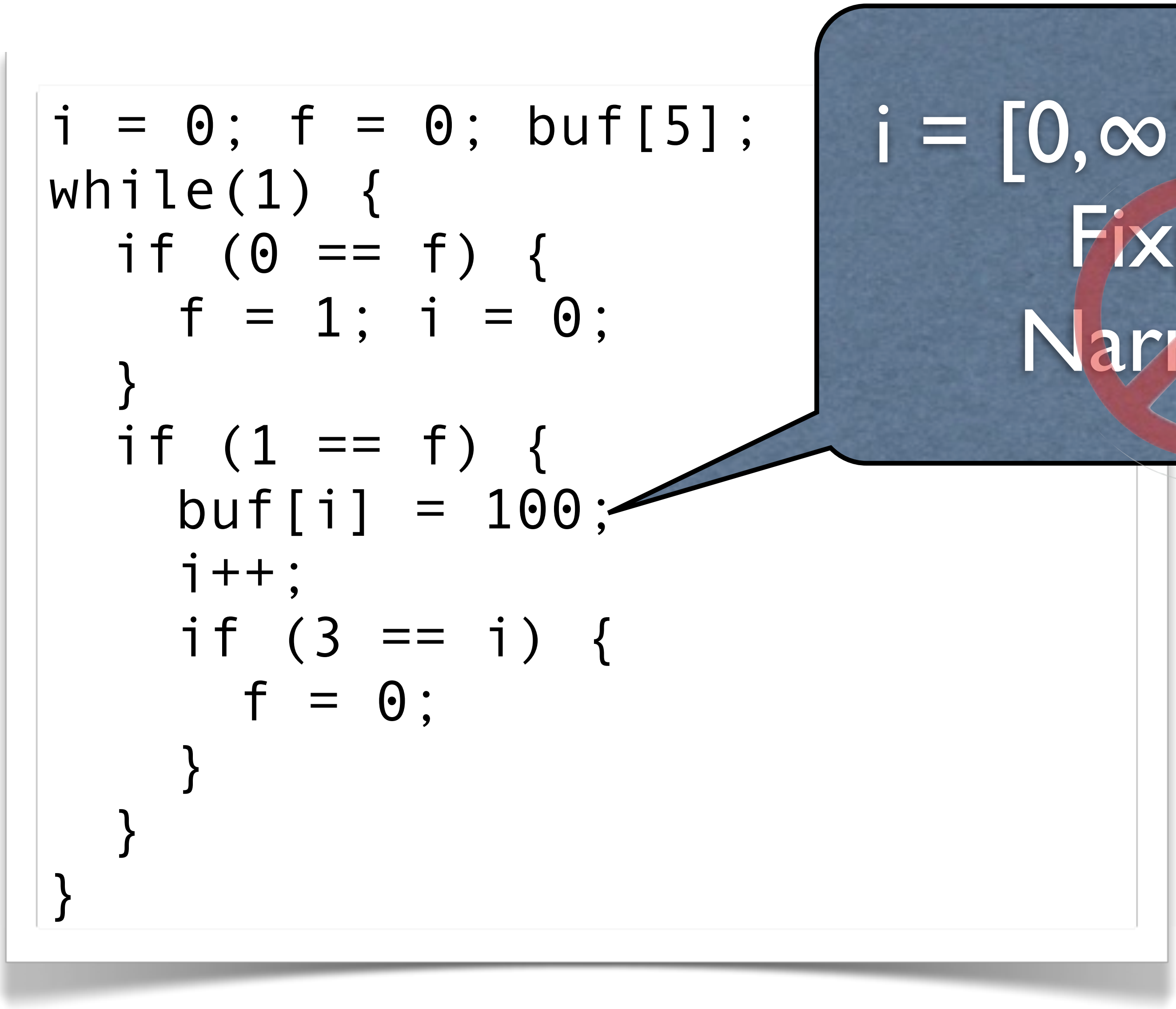
fixpoint

	1	2	3
i1	[0,0]	[0,1]	[0,1]
i2	[1,1]	[1,2]	[1,2]
i3	[1,1]	[0,1]	[0,1]

Delayed Widening

- Sparrow has heuristic delayed widening option
  - Simple loop ➔ find # of delaying automatically (e.g. for (i = 0; i < 100; i++))
- UAV case
  - NOT simple loop
  - BUT can reach to fixpoint w/o widening
- Delay widening 100 times : 28 → 10

False Alarm Case



```
i = 0; f = 0; buf[5];
while(1) {
  if (0 == f) {
    f = 1; i = 0;
  }
  if (1 == f) {
    buf[i] = 100;
    i++;
    if (3 == i) {
      f = 0;
    }
  }
}
```

i = [0,∞] (cf. [0,2])

Fixpoint

Narrowing

Future Work

- Find the reasonable solution for the last case
    - Relational analysis? eg. Octagon domain
    - Memory set domain?
- Find more interesting properties

Thank you