재빠르고 실용적인 C 프로그램 메모리 분석기 개발

이번엔 멀티쓰레드도 생각하여

정승철

한양대학교
프로그램 분석검증 연구실

2010년 8월 28일
4회 소프트웨어무결점연구센터
대명리조트 설악

큰 목표

- 프로그램 실제 개발 과정에 끼어들수 있을 정도로
- 적당히 정확하고
- 빠른 C 메모리 분석기를 개발

현재까지의 분석기

- 리스트, 트리 구조를 고려한 빠른 모양분석기 개발
- 가능한 요약메모리의 집합을 계산
- 여러 셀 뭉치를 하나의 요약셀로 합침

$$x \to \square \to \square \;\Rightarrow\; x \to \bigcirc$$

- 요약셀 하나는 입구 포인터가 하나이고, 출구 포인터가 있거나 없는 여러 셀 뭉치를 나타냄
- 요약셀에 접근하는 명령을 분석할 때는 요약셀을 여러 가능성으로 구체화하고 작업
- 넓히기(widening): (1) 너무 길어지는 구조를 자르고 (2) 여러 실제 셀을 하나의 요약셀로 합침 (3) 비슷한 메모리를 하나로 뭉뚱그려서 경우의 수를 줄임
- 문맥둔감 함수간 분석

분석기 보조 도구

- 오류 위치에서부터 오류 경로를 복구하는 분석기
  - "모양분석기보다 간단한 메모리 도메인 $\times$ 유한한 길이의 경로"를 오류 위치에서 거꾸로 올라가면서 모음

$$\downarrow \qquad \langle x \mapsto \langle next \mapsto \cdot \rangle 5 \cdot p \rangle$$

`5: y = x.next` $\qquad\qquad \uparrow$

$$\downarrow \qquad \langle x \mapsto \cdot, p \rangle$$

- 사용자가 지정한 허위 경보를 제거
  - 허위 경보를 만드는 지점을 역추적 하고, 그 지점에서 허위 경보가 나오지 않도록 메모리 구조를 조정하고 다시 분석
  - *이 구문은 널 포인터를 접근하는게 아닌것 같아 / 널 포인터 가정은 여기 있는 알수 없는 함수호출 때문에 만들어졌네. 그럼 이건 널이 아니라 하자.*

해야 할 일

- 멀티쓰레드 프로그램 분석
- 데이터 문맥민감 함수간 분석
  - 함수 입구의 메모리 별로 함수를 달리 분석
- 함수별 분석결과 보존, 완전한 모듈 분석
- 좀 더 강력하고 편리한 오류 경로 복구, 허위 경보 제거
- 예쁘고 모듈화가 잘 되도록 시스템을 리팩토링

```
Lock l;
Node* h;

thread1 {
  while (?) {
      ...
    acquire(l);
    n = new Node;
    n->next = h;
    h = n;
    release(l);
      ...
  }
}
```

```
main() {
  h = new Node; h->next = NULL;
  execute(thread1); execute(thread2);
}

  thread2 {
    while (?) {
        ...
      acquire(l);
      n = h->next;
      if (n != h) {
        delete h; h = n;
      }
      release(l);
        ...
    }
  }
```

멀티쓰레드 프로그램 분석

- 쓰레드가 엇갈려 수행되는 모든 경우를 고려해야 함
  - 모든 경우를 고려하는 정공법(Yahav01)은 너무 느림
- 우회로
  - 특화된 문제에 집중
  - soundness를 포기
  - 사용자의 도움을 받음
- 어떻게든 가능한 쓰레드 수행가짓수를 줄여 자동으로 분석할 수는 없을까?
  - 락에 의해 보호 받는 영역이라도 제대로 분석해보자

쓰레드별로 모양분석 하기

- A. Gotsman et al., Thread-modular shape analysis, PLDI '07
- 수행 가짓수를 줄일 수 있는 순간: 순차적으로 수행될 수 있는 곳, 즉 락으로 보호받는 영역의 수행
- 자원불변식: 쓰레드가 특정 락을 얻었을 때 자신만이 쓸 수 있는 (다른 쓰레드가 건드릴 수 없는) 메모리 영역에 대한 성질
  - 메모리를 "락 $\mapsto$ 요약된 자원불변식"으로 요약
  - 락을 얻는 지점에서: 그 때까지의 쓰레드 로컬 상태와 그 락의 자원불변식과 함께 "순차적으로, 다른 쓰레드를 생각하지 않으면서" 분석
  - 락을 내놓는 지점에서: 메모리를 쓰레드 지역 메모리와 자원불변식으로 나누어 적용

$$I_0 = h \mapsto \langle\rangle$$

```
thread1 {
 ...
```
s

```
 acquire(l);
```
$s \ * \ h \mapsto \langle\rangle$

```
 ...
```
기존의 순차적인 모양분석기가 분석

$s \ * \ h \mapsto \langle x \rangle \ * \ x \mapsto \langle\rangle$

```
 release(l);
```
결과 메모리를 쓰레드 지역 영역과
자원불변식으로 분리

$$I_1 = \exists x. h \mapsto \langle x \rangle \ * \ x \mapsto \langle\rangle$$

이 불변식을 락 l을 얻는 모든 지점에 반복적으로 적용하여 재분석함.

```
thread2 {
 ...
```
$$s$$
```
 acquire(l);
```
$$s \;*\; h \mapsto \langle x \rangle \;*\; x \mapsto \langle\rangle$$
```
 ...
```
$$s \;*\; h \mapsto \langle x \rangle$$
```
 release(l);
```

새 자원불변식 없음. I_1을 thread1의 락 영역에 계속 적용.
$\rightarrow^*$ 결국 락 l을 얻으면, 그리고 "얻어야만" h에서 시작되는 리스트를 접근한다는 사실을 알아냄.

메모리 나누기

- 락이 풀릴 때의 메모리에서 락으로 보호받는 메모리를 어떻게 잘라낼 것인가?
- 부정확하게 자르는 경우:
  - 쓰레드 지역 메모리를 자원불변식에 포함하면: 락으로 보호받는 영역 밖에서의 접근이 일어날 때 거짓경보 발생
  - 자원불변식의 메모리를 쓰레드 지역 메모리에 포함하면: 다른 쓰레드가 같은 락을 지니고 영역을 접근할 때 거짓경보 발생
- 접근: 락으로 보호받는 포인터 변수를 알아내고, 그 변수에서 접근가능한 힙 덩어리만 꺼냄
  - 락으로 보호받는 포인터 변수를 얼마나 정확히 알아내는지가 중요

목표

- Thread modular analysis 방식으로 C 멀티쓰레드 프로그램을 분석하는 분석기 개발
- 목표 결과: 수만라인의 코드를 수분내에 분석 (reasonable?)
- 덧붙여 가능하다면
  - 오류 경로 찾기
  - 허위 경보 제거
- 다음 워크숍에 번뜩이는 설계와 실험결과를 들고 올 수 있도록 노력할께요