요약해석을 이용한 버퍼오버런 분석에서 루프의 분석결과를 정교화 하는 방법[1)]

오학주○ 이광근
서울대학교 컴퓨터공학부 프로그래밍 연구실
pronto@ropas.snu.ac.kr, kwnag@ropas.snu.ac.kr

Buffer-overrun Abstract Interpretation Refinement for Loops

Hakjoo Oh○ Kwangkeun Yi
Programming Research Laboratory, Seoul Natonal University

1. 문제정의

구간(interval)을 도메인으로 사용하는 버퍼오버런 분석기는 루프에서 많은 허위경보를 발생시킨다. 이러한 허위경보들은 대부분 루프의 시작지점에서 적용되는 넓히기(widening)[3] 연산 때문에 발생한다. 넓히기 연산은 루프의 분석이 종료함을 보장하기 위해서 꼭 필요한 연산이지만 루프내에서 변화하는 변수들의 값을 그 변수가 루프내에서 가질수 있는 모든 값을 포함하는 값으로 어림잡는다. 예를 들어 다음 C 프로그램을 보자.

```
char p[64];
char d[64];
char* pd = &p[0];
for (j = 0; j < 8; j++)
{
        d[0] = (pd[0]+4)>>3;
        d[1] = (pd([1]+4)>>3;
        ...
        d[7] = (pd[7]+4)>>3;
        d += 8;
        pd += 8;
}
```

구간 도메인을 사용하는 버퍼오버런 분석기(이하 분석기)는 위의 루프내의 모든 버퍼 접근에 대하여 경보를 발생한다. 분석기는 루프의 조건식으로부터 변수 j가 루프를 돌때마다 1씩 증가하고 최대 8번까지 변화한다는 것은 알수 있지만 변수 j와 d, j와 pd 사이의 관계(j가 1증가할때마다 d와 pd는 8씩 증가한다는 사실)는 유추할 수 없기 때문이다. 그래서 분석기는 포인터 변수 d와 pd가 루프 내에서 8번 변화한다는 사실을 알 수 없고, 무한번 변화할수 있다고 근사시킨다. 때문에 루프내에서 d와 pd가 가리키는 주소에 접근하는 모든 식들에 대하여 버퍼오버런 가능성을 보고하게 된다. 이로써 위의 예와 같이 분석기는 루프의 실행 횟수(iteration number)를 쉽게 파악할 수 있는 루프에서 조차도 많은 수의 허위경보가 발생한다. 우리의 목표는 이러한 루프에서 발생하는 허위경보를 안전하게 제거하는 것이다.

2. 해결방법

정적 프로그램 분석기는 모든 종류의 프로그램에 대해서 효과적일 수 없다. 한 예로 구간도메인을 이용하는 퍼오버런 분석기는 많은 상용프로그램에 대하여 유용함이 밝혀졌지만[6] 루프내에서 버퍼에 접근하는 방식이 많은 프로그램에 대해서는 많은 수의 허위경보를 발생시킨다. 일반적으로 이러한 정확도

1) 이 연구는 교육인적자원부 두뇌한국21사업의 지원을 받았음을 밝힙니다.

문제를 해결하는 방법은 더 자세한 방법으로 분석하는 것일 것이다. 하지만 더 자세한 분석방법은 분석 비용을 증가시켜서 분석기의 확장성을 떨어뜨린다. 또한 일관되게 자세한 분석을 모든 프로그램의 분석에 적용하는 것은 자세한 분석이 필요없는 특정 프로그램에 대해서도 동일하게 적용되어 분석비용을 증가시키므로 오히려 비효율적일 수 있다.

우리의 목표는 분석기의 전체적인 성능을 크게 떨어뜨리지 않으면서 실제 프로그램에서 자주 나타나는 유형의 허위경보들을 효과적으로 제거하는 것이다. 우리가 제안하는 방법은 요약해석 분석결과를 정교화(abstract interpretation refinement)하는 방법이다. 이는 기존의 가벼운 분석방법으로 프로그램을 분석하여 얻은 분석결과에서 더 자세한 분석이 필요한 부분만을 선택한 후 그 부분을 더 자세하게 다시 분석하는 방법이다. 요약해석 분석결과를 정교화 시키는 방법은 두 가지로 나눌 수 있다. 1) 정확도 향상이 필요한 프로그램 부분을 기존의 것보다 정확한 요약 실행 함수(abstract semantics function)으로 재분석 하는 방법과, 2) 정확도 향상이 필요한 프로그램 부분 p를 p'으로 변형하여 결과적으로 기존의 요약 실행 함수가 더 자세한 분석을 할 수 있도록 하는 것이다. 우리는 첫번째 방법을 이용하여 정확도 향상이 필요한 루프를 재분석 하였다.

요약해석 정교화를 실제 사용 프로그램 분석에 적용하려면 최대한 효율성을 높여야 한다. 이를 위해서는 다음 세 가지를 고려해야 한다. 첫째는 정교화를 통하여 정확도를 높이려는 대상이 명확해야 한다. 특정 정교화 방법이 모든 종류의 허위경보를 제거하는데에 적합한 것은 아니기 때문이다. 우리는 허위경보를 일으키는 루프들 중 바운드 루프를 대상으로 잡았다. 둘째, 정교화를 위해 선택된 프로그램 부분들 중 재분석을 적용했을 때 실제로 정확도가 올라갈 것인지를 미리 예측해야 한다. 우리의 실험결과 정확도가 더 올라갈 여지가 도저히 없는데 다시 분석하는 것은 정교화의 성능을 많이 떨어뜨렸다. 이 부분에 대한 예측은 기존의 분석결과를 보고 어느정도 파악할 수 있다. 셋째, 대상에 적합한 효율적인 재분석 방법을 적용해야 한다. 보통, 정교화를 적용할 대상이 명확히 정해지면 효율적인 재분석 설계가 쉬워진다.

우리는 루프 관련 허위경보 제거에 요약해석 정교화 방법을 응용하였다. 일단 프로그램은 기존의 구간 분석을 이용하여 분석된다. 그 후에 분석된 결과를 보고 더 자세한 분석이 필요한 부분만을 판단하고 더 자세한 분석을 적용했을시 정확도가 올라갈 부분만을 다시 추려낸 후 재분석을 진행한다.

3. 참고문헌

1. Bruno Blanchet, Patrick Cousot, Radia Cousot, Jerome Feret, Laurent Mauborgne, Antoine Mine, David Monniaux, and Xavier Rival. A static analyzer for large safety-critical software. In PLDI '03:Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation, pages 196-207, New York, NY, USA, 2003. ACM Press.

2. K. Cooper, T.Harvey, and K.Kennedy. A simple, fast dominance algorithm, 2001.

3. Patrick Cousot and Radhia Cousot. Abstract Interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In Prodeedings of ACM Symposium on Principles of Programming Languages, pages 238-252, January 1977.

4. Yungbum Jung, Jaehwang Kim, Jaeho Shin, and Kwnagkeun Yi. Taming false alarms from a domain-unaware C analyzer by a Bayesian statistical post analysis. In SAS 2005: 12th Annual International Static Analysis Symposium, volume 3672 of Lecture Notes in Computer Science, pages 203-217. Springer 2005.

5. B.W.Kernighan and D.M.Ritchie. The C Programming Language. Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 1978.

6. Jaeho Shin, Jaehwang Kim, Hakjoo Oh, Yikwon Hwang, and Kwangkeun Yi. Airac5: Array index range analyzer for c. http://ropas.snu.ac.kr/2005/airac5

7. Mine,A. A new numerical abstract domain based on difference-bound matrices. In PADO II, vol. 2053 of LNCS, Springer-Verlag, pp. 155-172.