union 멤버의 부적절한 사용 오류 검출 기법

이성원○ 주성용* 조장우
동아대학교 컴퓨터공학과
acezip5@naver.com, SeongYongJ@gmail.com, jwjo@dau.ac.kr

A Method for Detecting Errors Misusing union Members

Sungwon Yi○ Seongyong Joo* Jang-wu Jo
Dept. of Computer Engineering, of Dong-a University

1. 서 론

union 멤버 사용 오류는 union 변수의 값을 읽을 때 union 변수의 멤버 중 값을 정의한 멤버와 다른 멤버를 이용해서 값에 접근하는 것이다. 각 데이터 타입은 값을 표현하는 방식이 서로 상이하기 때문에, union 변수에 저장된 값의 타입과 다른 타입을 가지는 멤버를 이용해서 값을 읽는 경우 예측할 수 없는 문제를 야기할 수 있다.

union 변수에 저장되는 값의 타입은 동적으로 결정되기 때문에, 기존 C나 C++ 컴파일러는 union 변수에 저장된 값을 읽을 때 그 값과 그 값을 읽기 위해서 사용하는 멤버 간의 타입 검사를 수행하지 않는다. 따라서 기존 C/C++ 컴파일러는 union 변수에 저장된 값과 다른 타입의 멤버를 이용해서 값을 읽는 경우 아무런 오류 메시지도 발생하지 않는다. 그러므로 union 멤버 사용 오류는 실행시간 오류를 발생하거나 부정확한 계산을 수행할 수 있다. 이 같은 문제를 해결하기 위해서는 union 멤버의 부적절한 사용 오류를 검출하기 위한 도구가 필요하다.

2. 본 론

<그림 1>은 union 멤버 사용 오류의 예이다. <그림 1>의 11행에서 union 변수 a의 멤버 i에 정수 10을 대입하고, 13행에서 union 변수 a의 멤버 f를 이용해서 union 변수에 저장된 값을 읽고, 그 값을 f1에 대입한다. 이 경우 정수 10과는 전혀 무관한 값이 f1에 저장된다.

```
 1:  union un1
 2:  {
 3:    int i;
 4:    float f;
 5:  };
 6:
 7:  void func()
 8:  {
 9:    un1 a;
10:    float f1;
11:    ...
12:    a.i = 10;
13:    f1 = a.f;
14:    ...
15:  }
```

<그림 1> union 멤버 사용 오류의 예

<그림 1>의 13행과 같은 오류를 검출하기 위해서는 union 멤버를 이용해서 union 변수의 값을 읽을 때 union 변수에 저장된 값과 그 값에 접근하기 위해서 사용하는 멤버 간의 타입 검사가 필요하다. 이를 위해서 타입 한정자를 이용한 union 멤버 사용 오류를 검출하기 위한 기법을 제안한다[1,2]. 제안기법은 union 변수의 값을 읽기 위한 문장에 대해서 union 변수에 저장된 값과 그 값을 읽기 위해서 사용한 멤버 간의 타입을 비교하기 위한 타입 제약식을 생성하고, 만일 타입 제약식의 해가 존재하지 않으면 이를 오류로 보고한다.

<그림 2>는 타입 한정자를 이용해서 union 멤버 사용 오류를 검출하는 예이다. <그림 2> 우측의 제약식은 union 변수가 가질 수 있는 값의 타입을 명세하며, union 멤버를 이용해서 union 변수의 값을 읽는 경우 그 멤

버와 값의 타입간의 타입 검사를 표현한다. 3행의 제약식에서 사용된 u1_q는 타입 한정자 변수이며, union 변수 u1의 타입을 값으로 가진다. 3행에서 union 변수 u1의 타입은 값이 정의 될 때 결정되기 때문에 초기값은 nil로 표현하며, 6행에서 union 변수 u1에 정수값을 대입하기 때문에 u1_q에 int를 대입한다. 그리고 7행에서 u1에 저장된 값을 읽기 위해서 멤버 f를 사용한다. 우측의 제약식은 union 변수에 저장된 값의 타입과 값을 읽기 위해서 사용한 멤버 변수의 타입이 일치해야 함을 표현한 것이다. 함수 typeof은 인수로 전달 받은 식의 타입을 반환하는 함수이다. typeof(u1.f)는 float이고, u1_q는 int이므로 7행은 만족되지 않기 때문에 오류로 보고된다.

소스 코드	제약식
1: union tun { int i; float f; };	
2: ...	
3: tun u1;	u1_q := nil
4: float f1;	
5: ...	
6: u1.i = 10;	u1_q := int
7: f1 = u1.f;	typeof(u1.f) = u1_q
8: ...	

<그림 2> 실제 코드 적용 예

함수의 인수나 결과로 union 타입의 변수를 전달하거나 반환하는 경우 함수의 선언문에 타입 한정자를 기술하고, 이를 기반으로 타입 추론을 이용해서 오류를 검출할 수 있다. 예를 들면 <그림 3>과 같다. <그림 3>에서 Q1과 Q2 그리고 Q3는 타입 한정자이며, 이들은 union 변수가 가져야할 타입을 기술한다. 만일 func의 결과를 식에서 사용하거나, func 내부에서 오류를 검출할 때 이 타입 한정자들을 기반으로 union 변수에 저장된 값의 타입을 결정한다.

```
Q1 union un1 func(Q2 union un2 a1, Q3 union un3 a2);
```

<그림 3> 타입 한정자를 명세한 함수 선언

3. 결 론

본 논문에서는 타입 한정자를 이용하여 union 멤버 사용 오류를 검출하기 위한 기법을 제안하였다. union 멤버 사용 오류는 union 변수에 저장된 값과 상이한 타입의 멤버를 이용해서 그 값을 읽는 연산이다. union 멤버 사용 오류는 기존 컴파일러로 검출되지 않기 때문에 실행시간 오류나 부정확한 계산 결과를 유발할 수 있다.

제안기법은 사용자 정의 함수 중 union 타입의 인수나 반환 값을 가지는 함수에 대해서 타입 한정자 선언을 추가하면 되기 때문에 프로그래머의 큰 노력 없이 멤버 사용 오류를 검사할 수 있다. 또한 타입 한정자는 많은 프로그래밍 언어에서 제공되기 때문에 프로그래머들에게 친숙하다는 장점이 있다.

참고 문헌

[1] Jeffrey Scott Foster, Type Qualifiers: Lightweight Specifications to Improve Software Quality, Ph.D. thesis. University of California, Berkeley. December 2002.

[2] Umesh Shankar, Kunal Talwar, Jeffrey S. Foster, David Wagner, Detecting Format String Vulnerabilities with Type Qualifiers, 10th USENIX Security Symposium, pages 201-218, Washington, D.C., August 2001.

[3] 주성용, 안준선, 조장우, 심볼릭 링크 공격 취약성 검출을 위한 분석 기법, 정보처리학회 논문지 A, Vol 15-A, No. 1, 2008.