안전한 모듈 지원 함수 흐름 분석*

이욱세 이광근†
한국과학기술원 전산학과

Deriving a Safe Modular Version of Control-Flow Analysis

Oukseh Lee and Kwangkeun Yi
Dept. of Computer Science, KAIST

요 약

예외상황 분석기를 SML/NJ에 장착하면서 모듈 시스템에 효율적으로 장착할 수 있는 분석이 필요하였다. 문제를 축소하여 다루기 쉬운 함수 흐름 분석(control-flow analysis)에 대하여, 간단한 모듈 지원 분석을 설계해 보았다. 설계된 모듈 지원 분석이 옳은 결과값을 준다는 것을 증명하기 위하여, 모듈 지원 분석의 작동 과정을 설명할 수 있는 함수 흐름 분석 모델을 끌어 내고, 이로부터 안전함을 증명하였다. 설계된 모듈 지원 분석보다 향상된 다른 모듈 지원 분석을 설계하고, 같은 모델을 사용하여 안전함을 증명함으로써, 이러한 증명 과정이 효과적임을 보였다.

1 배경

예외상황 분석기[5]를 SML/NJ[2]에 장착하는 과정에서, 모듈 시스템에 효율적으로 장착하기 위한 연구가 필요하였다. 효율적인 장착이란

- 어떤 모듈이 컴파일되는 시간과 분석되는 시간이 같고,
- 불필요한 정보를 저장하지 않고 동일한 계산을 반복하지 않으며,
- 분석 결과값의 정확도가 떨어지지 않도록

하는 것을 말한다. 그러나, 예외 상황 분석은 쉽게 효율적으로 장착할 수 없다. 비슷한 분석으로 함수 흐름 분석(control-flow analysis; 이하 CFA)[4]에 대해 살펴 보자. 다음과 같은 프로그램에서,

```
structure A: sig
    val f: (('a -> 'a) -> 'b) -> 'b
  end = struct
    val g = fn x=>x
    val f = fn z=>z g
end
```

모듈 밖에서 보이는 변수는 `f`이므로, `f`의 분석 결과값, 즉, `f`가 `fn z=>z g` 값을 갖는다는 정보만 저장한다면, 다음과 같은 이어지는 프로그램에서,

```
structure B = struct
  val h = A.f (fn w=>w)
end
```

`h`가 `fn x=>x` 값을 가질 수 있다는 것을 분석해 낼 수 없다. 이러한 문제는 함수 값이 함수 인자로 전달될 수 있는 (high-order) 언어에서 CFA와 관련있는 분석이면 발생하는 문제이다.

보다 일반적인 문제로, 모듈을 지원않는 분석이 주어졌을 때 모듈을 지원하는 분석을 설계하는 문제를 해결하고자 한다. 모듈 지원 분석을 끌어내고, 끌어낸 분석이 옳은 결과값을 준다는 것을 증명하는 과정을, 자동적으로 만들어 낼 수 있는 프레임웍을 제공하거나, 경험적인 방법론을 제시하고자 한다.

첫번째 시도로 ML의 CFA에 대하여 모듈 지원 분석을 설계해 보았다. CFA는 프로그램 내의 변수 또는 계산식이 어떤 함수값이 될 수 있는지를 분석하는 것으로, 예외상황 분석에 비하여 비교적 단순하므로 첫번째 시도의 소재로 삼았다.

본 논문에서는, 0-CFA[4]로부터 간단한 모듈 지원 분석을 끌어내고, 그 안전함을 증명하는 과정을 설명한다. 0-CFA의 안전함으로부터 모듈 지원 분석의 안전함을 증명할 수 없다. 그러므로, 다른 0-CFA보다 자세한 결과값을 주는 CFA을 제시함으로써, 그로부터 간단한 모듈 지원 분석의 안전함을 증명한다. 또, 향상된 모듈 지원 CFA를 제시하고, 같은 방법으로 안전함을 증명함으로써, 어떠한 모듈 지원 CFA를 대상으로 하더라도 제시된 방법으로 편리하게 안전함을 증명할 수 있음을 보인다.

2 언어와 0-CFA

대상 언어는 그림 1에 있다. 변수, 함수, 함수적용문, 선언문이 있고, 각 계산식은 위치를 나타내기 위한 위치표시(label)를 위첨자로 붙인다. 프로그램 내의 모든 변수는 함수 인자 변수(x)와 선언 변수(f)로 나뉘되, 모두 구분된다고 가정한다. σ는 시그네춰(signature)로 모듈 밖에서 사용될 수 있는 선언 변수의 집합이다. 모듈 없는 언어는 *core* 이고, 모듈 있는 언어는 *prog* 이다.

소재가 되는 0-CFA는 그림 1과 같다[1]. 모듈 없는 언어에 대하여 정의되었다. $n \to m$은 n의 분석 결과가 m의 분석 결과를 포함한다는 것을 의미한다. $\lambda(x,l)$은 x를 받아서 l의 식을 계산하는 함수 값을 의미한다. 더 이상 조건이 생성되지 않을 때까지 조건을 만들어서, 조건을 만족하는 가장 작은 집합을 만들면 분석 결과가 된다.

3 $\mathcal{T}$: 간단한 모듈 지원 0-CFA

모듈을 지원하기 위해서는 프로그램을 보는 규칙과 들여다

*본 연구는 과학기술부 창의적연구진흥사업 추진으로 얻어진 결과임.
†e-mail: {cookcu; kwang}@cs.kaist.ac.kr

$$
\begin{array}{rcl}
v & ::= & x \mid f \\
e & ::= & v \mid \lambda x.e^l \mid e^l\ e^l \\
\mathit{core} & ::= & (f = e^l)^* \\
\sigma & ::= & \{f^*\} \\
\mathit{prog} & ::= & (\texttt{struct}\ S\ \sigma\ \mathit{core})^*
\end{array}
$$

$$n ::= v \mid l \mid \lambda(x,l)$$

$$\frac{f = e^l \in C}{f \to l}\quad(\mathrm{DEC}_0)$$

$$\frac{v^l \in C}{l \to v}\quad(\mathrm{VAR}_0)$$

$$\frac{(\lambda x.e^{l_0})^l \in C}{l \to \lambda(x,l_0)}\quad(\mathrm{LAM}_0)$$

$$\frac{(e_1^{l_1}\ e_2^{l_2})^l \in C \quad l_1 \to \lambda(x,l_0)}{l \to l_0 \quad x \to l_2}\quad(\mathrm{APP}_0)$$

$$\frac{n_1 \to n_2 \quad n_2 \to n_3}{n_1 \to n_3}\quad(\mathrm{TR}_0)$$

그림 1: 대상 언어와 0-CFA.

보지 않는 규칙을 나누어야 한다. 모듈 A, B가 있을 때, B를 분석할 때는 A 프로그램은 볼 수가 없다. 그러므로, 0-CFA 규칙에서 $e^l \in C$와 같이 프로그램을 보는 것을 나누어 주어야 한다. 즉, 프로그램에서 조건을 끌어내는 과정과 푸는 과정으로 나누어야 한다.

모듈을 지원하기 위해서는 분석 결과를 저장할 수 있는 중간 언어가 있어야 한다. 0-CFA 에서는 분석 결과가 조건들의 집합이 나올 수 있다. 이 조건들 자체가 중간 언어로 사용될 수 있으며, 덧붙여 프로그램을 보는 규칙에서 나온 정보를 저장하는 $@(l;l,l) \to \Box$ 조건이면 충분하다.

모듈을 효율적으로 지원하기 위해서는 모듈에서 분석된 결과를 필요한만큼만 걸러내는 규칙이 필요하다. 필요한 정보라는 것은 나중에 이 모듈에서 선언된 변수를 사용하는 모듈을 분석할 때, 옳은 결과를 낼 수 있을 만큼의 정보를 말한다. 간단한 모듈 지원 분석은 모듈 밖으로 나가는 노드들을 모아 (σ), 이 노드들이 연관된 조건만을 걸러 낸다 ($\mathrm{OUT}_\mathcal{T}$). 시그네춰에 있는 변수는 나갈 수 있고 ($\mathrm{SIG}_\mathcal{T}$), 나가는 노드의 내부식(sub-expression)은 나갈 수 있고 ($\mathrm{SUB}_\mathcal{T}$), 나가는 노드의 분석값에 영향을 주는 것은 나갈 수 있다 ($\mathrm{EXPN}_\mathcal{T}$). $\mathcal{R}_\mathcal{T}$에서 $n \rightsquigarrow m$이 걸러진 조건을 의미한다.

이처럼 프로그램에서 조건 끌어내기($\mathcal{D}$), 풀기($\mathcal{S}$), 걸러내기($\mathcal{R}$)의 세과정이 있으면 효율적으로 CFA를 장착할 수 있다. 모듈 A:σ_0, B가 있을 때, A로부터 조건을 끌어낸 후, 풀고 나서, σ_0로 걸러진 조건들과, B로부터 끌어내진 조건들을 합쳐서 풀면, 즉, $\mathcal{S}(\mathcal{R}(\mathcal{SD}(A),\sigma) \cup \mathcal{D}(B))$ 과정으로 풀 수 있다. B의 시그네춰가 σ_1이고, 모듈 C가 또 있으면, 위의 분석 결과를 E라 할 때, $\mathcal{S}(\mathcal{R}(E,\sigma_1) \cup \mathcal{D}(C))$ 로 분석할 수 있다. B를 수정한 경우, A의 풀어 걸러낸 것까지는 같으므로 생략하고, 그 이후를 반복한다. 실제 컴파일러도 모듈간 의존도에 의해 수정된 이후의 모듈들을 컴파일하므로 효율적으로 분석할 수 있다.

4 $\mathcal{T}$의 안전함

$\mathcal{T}$의 안전함을 0-CFA로부터 증명할 수 있으면 좋겠지만 불가능하다. 0-CFA로부터 증명하려면, 0-CFA에서 나오는 모든 결과를 $\mathcal{T}$에서도 얻을 수 있어야 한다. 그러나, 다음과 같

$$n ::= v \mid l \mid \Box \mid \lambda(x,l) \mid @(l;l,l)$$

조건을 끌어내기($\mathcal{D}$) :

$$\frac{f = e^l \in S}{f \to l}\quad(\mathrm{DEC})$$

$$\frac{v^l \in S}{l \to v}\quad(\mathrm{VAR})$$

$$\frac{(e_1^{l_1}\ e_2^{l_2})^l \in S}{@(l;l_1,l_2) \to \Box}\quad(\mathrm{APD})$$

$$\frac{(\lambda x.e^{l_0})^l \in S}{l \to \lambda(x,l_0)}\quad(\mathrm{LAM})$$

풀기($\mathcal{S}$) :

$$\frac{@(l;l_1,l_2) \to \Box \quad l_1 \to \lambda(x,l_0)}{l \to l_0 \quad x \to l_2}\quad(\mathrm{APP})$$

$$\frac{n_1 \to n_2 \quad n_2 \to n_3}{n_1 \to n_3}\quad(\mathrm{TR})$$

걸러내기($\mathcal{R}$) :

$$\frac{x \in \sigma_0}{x \in \sigma}\quad(\mathrm{SIG}_\mathcal{T})$$

$$\frac{l \in \sigma \quad l' \in l}{l' \in \sigma}\quad(\mathrm{SUB}_\mathcal{T})$$

$$\frac{n \in \sigma \quad n \to m}{m \in \sigma}\quad(\mathrm{EXPN}_\mathcal{T})$$

$$\frac{n,m \in \sigma \quad n \to m}{n \rightsquigarrow n'}\quad(\mathrm{OUT}_\mathcal{T})$$

그림 2: 간단한 모듈 지원 0-CFA ($\mathcal{T}$).

은 반례가 존재한다.

```
struct A {f}
  f = λx.x
  g = (f λy.y) λz.z
end
```

```
struct B {i}
  i = A.f λw.w
end
```

0-CFA에서는 `f λy.y` $\to$ `x` $\to$ `λw.w`의 결과가 나와, 결국 `w` $\to$ `λz.z`를 얻는다. 하지만, $\mathcal{T}$에서는 `g`가 시그네춰에 없고 `f`와의 연관성이 없기 때문에, 앞의 결과가 보존되지 않아, 모듈 B를 분석할 때, `w` $\to$ `λz.z`를 얻을 수 없다.

의미구조로부터 $\mathcal{T}$가 안전하다는 것을 증명할 수 있겠지만 복잡할 수 있으므로, 다음 조건을 만족하는 모델을 통해 증명하도록 하겠다.

- 0-CFA로부터 쉽게 끌어낼 수 있다.
- 안전함이 명백하다.
- 모듈 지원 분석보다 적은 결과를 준다.
- 모듈 지원 분석의 안전함을 증명하는데 편리하다.

k-CFA[4]가 유력한 후보이나, k가 유한한 경우 위 조건을 만족하는 모델이 될 수 없다. k-CFA는 각 계산식 또는 변수의 동적 호출 역사(dynamic call sequence)에 따라 구별하여 분석하는 CFA를 말한다. k는 동적 호출 역사를 k 크기까지만 기록한다는 의미이고, k보다 역사가 길어지면 최근 것만을 기록한다. 0-CFA보다 작은 결과를 주지만 역시 $\mathcal{T}$보다 큰 결과를 줄 수 있다. 다음은 1-CFA 에 대한 반례이나, k-CFA의 반례도 유사하게 만들 수 있다.

```
struct A {f}
  f = λx.(λy.y x)
  g = (f λz.z) λw.w
end
```

```
struct B {i}
  i = A.f λu.u
end
```

$$n ::= l_M \mid v_M \mid \lambda(x, e^l)$$
$$c ::= e^l{}_M \mid n \to n$$

$$\frac{x^l{}_M}{l_M \xrightarrow{1} x_M}\ (\mathrm{VAR}_1)$$
$$\frac{f^l{}_M \quad f \in N}{l_M \xrightarrow{1} f_N}\ (\mathrm{IVAR}_1)$$
$$\frac{f = e^l \in M}{f_M \xrightarrow{1} l_M \quad e^l{}_M}\ (\mathrm{DEC}_1)$$
$$\frac{(\lambda x.e^{l_0})^l{}_M}{l_M \xrightarrow{1} \lambda(x, e^{l_0})}\ (\mathrm{LAM}_1)$$
$$\frac{(e_1^{l_1}\ e_2^{l_2})^l{}_M}{e_1^{l_1}{}_M \quad e_2^{l_2}{}_M}\ (\mathrm{APD}_1)$$
$$\frac{(e_1^{l_1}\ e_2^{l_2})^l{}_M \quad l_{1M} \xrightarrow{1} \lambda(x, e^{l_0})}{x_M \xrightarrow{1} l_{2M} \quad l_M \xrightarrow{1} l_{0M} \quad e^{l_0}{}_M}\ (\mathrm{APP}_1)$$
$$\frac{n_1 \xrightarrow{1} n_2 \quad n_2 \xrightarrow{1} n_3}{n_1 \xrightarrow{1} n_3}\ (\mathrm{TR}_1)$$

그림 3: 1M-CFA

x는 g와 i에서 호출한 것으로 구별되지만, y는 "λy.y x"에서 호출한 것으로 동일하게 본다.

$\mathcal{T}$의 안전함을 증명할 수 있는 모델로 1M-CFA를 제안한다. k-CFA의 변형으로, 동적 호출 역사를 기록하되, 하나만 기록하고, 위치 단위가 아닌 모듈 단위로 기록하고, 역사가 하나보다 길어지는 경우, 최근 것이 아닌 가장 오래된 것을 기록한다. 1M-CFA는 0-CFA로부터 쉽게 이끌어 진다. 왜냐하면, k-CFA와 비슷한 원리로 만들어 졌기 때문이다. 1M-CFA는 안전한 분석임이 명백하다. 왜냐하면, 무한 k-CFA를 요약(abstraction)한 것이기 때문이다[3].

$\mathcal{T}$의 안전함을 1M-CFA로부터 쉽게 증명할 수 있다. 여기서는 두 모듈, 즉, $A{:}\sigma$와 B가 있어, B가 A의 변수를 사용한 경우만 고려한다. $n \xrightarrow{A;B} m$은 $n \to m$이 $\mathcal{T}$에 의해 B를 분석한 결과에 있음을 의미한다.

정리 1 ($\mathcal{T}$의 안전함) *다음은 사실이다.*

- e^l_B*이면,* $l \in B \cup \sigma$*이다.*
- $n_B \xrightarrow{1} m_\kappa$*이면,* $n \xrightarrow{A;B} m$*이다.*
- $n_B \xrightarrow{1} \lambda(x, e^{l_0})$*이면,* $n \xrightarrow{A;B} \lambda(x, l_0)$*이다.*

세번째 문장을 보면 모듈 B 내의 n이 1M-CFA에서 어떤 함수값을 가지면, $\mathcal{T}$의 결과에도 n이 그 함수값을 가진다는, 안전함을 말하고 있다.

5 $\mathcal{X}$: 향상된 모듈 지원 0-CFA

$\mathcal{T}$의 걸러내기를 교체하여 향상된 모듈 지원 0-CFA를 설계할 수 있다. $\mathcal{T}$의 걸러내기는 필요한 것보다 많은 조건들이 나가게 되는데, 나가는 노드들을 정확히 분류하여 향상된 걸러내기를 만들 수 있다. (TR) 규칙을 고려하지 않은 경우, 모듈 밖으로부터 직접 들어 올 수 있는 노드("$\Rightarrow n$")와 직접 나갈 수 있는 노드("$n \Rightarrow$")들을 관찰할 수 있는데, 그 노드들을 정확히 모아서 그림 4와 같은 걸러내기 규칙을 만들 수 있다.

$\mathcal{T}$의 안전함을 증명할 때와 마찬가지로, $\mathcal{X}$의 안전함도 1M-CFA로부터 쉽게 증명될 수 있다. 마찬가지로 모듈 두 개에 대해서만 증명하였다. $n \xrightarrow{A} m$는 $n \to m \in \mathcal{SD}(A)$를 의미하고, $n \xrightarrow{A?} m$은 $n \xrightarrow{A} m$ 또는 $n = m$을 의미한다. $n \xrightarrow{A;B} m$은 $n \to m$이 $\mathcal{X}$에 의해 B를 분석한 결과에 있음을 의미한다.

$$\frac{f \in \sigma_0}{\Rightarrow f}\ (\mathrm{SIG}_{\mathcal{X}})$$
$$\frac{\Rightarrow n \to \lambda(x, l)}{n \rightsquigarrow \lambda(x, l) \quad x \Rightarrow \quad \Rightarrow l}\ (\mathrm{LAM}_{\mathcal{X}})$$
$$\frac{@(l; l_1, l_2) \to \Box \quad l_1 \Rightarrow}{@(l; l_1, l_2) \rightsquigarrow \Box \quad l \Rightarrow \quad \Rightarrow l_2}\ (\mathrm{APP}_{\mathcal{X}})$$
$$\frac{@(l; l_1, l_2) \to \Box \quad l_1 \to n \Rightarrow}{@(l; l_1, l_2) \rightsquigarrow \Box \quad l_1 \rightsquigarrow n \quad l \Rightarrow \quad \Rightarrow l_2}\ (\mathrm{APP}'_{\mathcal{X}})$$
$$\frac{\Rightarrow n \to m \Rightarrow}{n \rightsquigarrow m}\ (\mathrm{TRA}_{\mathcal{X}})$$

그림 4: 향상된 모듈 지원 0-CFA의 걸러내기 ($\mathcal{R}_{\mathcal{X}}$).

정리 2 ($\mathcal{X}$의 안전함) *다음은 사실이다.*

- $n_B \xrightarrow{1} \lambda(x, e^{l_0})$*이면,* $n \xrightarrow{A?} m \xrightarrow{A;B} \lambda(x, l_0)$*이고* $m \in B$ *또는* $m \Rightarrow$*인* m*이 존재하거나,* $n \xrightarrow{A} \lambda(x, l_0)$*이다.*
- $n_B \xrightarrow{1} m_\kappa$*이면,* $n \xrightarrow{A?} n' \xrightarrow{A;B} m' \xrightarrow{A?} m$*이고* $n' \in B$ *또는* $n' \Rightarrow$, $m' \in B$ *또는* $\Rightarrow m'$*을 만족하는* n', m'*이 존재하거나,* $n \xrightarrow{A?} m$*이다.*

첫번째 문장을 보면, 모듈 B의 n이 어떤 함수값을 가지면, $n \xrightarrow{A} \cdots$가 불가능하므로, 즉, $n \xrightarrow{A;B} \lambda(x, l_0)$가 존재함을 뜻한다. 그러므로, $\mathcal{X}$가 1M-CFA보다 큰 결과를 준다는 의미이다.

6 결론 및 향후계획

본 논문에서는 0-CFA의 모듈 지원 분석 $\mathcal{T}$와 $\mathcal{X}$를 제시하고, 그들의 안전함을 1M-CFA를 통하여 증명할 수 있음을 보였다. 모듈 지원 분석은 분석을 두 과정으로 나누고, 걸러내는 규칙을 설계하여 쉽게 만들 수 있었다. 그리고, 설계된 분석의 안전함은 다른 모델을 통하여 간접적으로 증명할 수 있었다.

0-CFA가 아닌 다른 분석에 대해서도 모듈 지원 분석을 설계하고 그 안전함을 증명하고자 한다. 그러한 과정들을 통하여 모듈을 지원 않는 분석에 대하여 모듈 지원 분석을 쉽게 설계하고 안전함을 증명하는 일반적인 방법을 제시하는 연구가 필요하다.

참고 자료

[1] Nevin Heintze and David McAllester. Linear-time subtransitive control flow analysis. In *Proceedings of the SIGPLAN Conference on Programming Language Design and Implementation*, pages 261–272, June 1997.

[2] David B. MacQueen and Andrew W. Appel. Standard ML of New Jersey. Technical memorandum, Lucent Bell Laboratory, 1993.

[3] Flemming Nielson and Hanne Riis Nielson. Infinitary control flow analysis: a collecting semantics for closure analysis. In *ACM Symposium on Principles of Programming Languages*, 1997.

[4] Olin Shivers. Control flow analysis in scheme. In *Proceedings of the SIGPLAN Conference on Programming Language Design and Implementation*, June 1988.

[5] Kwangkeun Yi and Sukyoung Ryu. Towards a cost-effective estimation of uncaught exceptions in SML programs. In *Lecture Notes in Computer Science*, volume 1302, pages 98–113. Springer-Verlag, proceedings of the 4th international static analysis symposium edition, 1997.