슬라이스 복잡도 측정을 위한 VFG의 사용

문유미*, 최완규*, 이성주*

The Use of VFG for Measuring the Slice Complexity

Yu-Mi Mun, Wan-Kyoo Choi, Sung-Joo Lee

요 약

본 논문은 데이터 슬라이스에서의 정보 흐름을 모델링하기 위해서 데이터 값 흐름 그래프(VFG: data Value Flow Graph)라고 하는 새로운 데이터 슬라이스(data slice) 표현을 개발한다. 다음으로, VFG에서의 정보 흐름의 복잡도를 측정하기 위해 기존의 흐름 복잡도를 이용하여 슬라이스 복잡도 척도를 정의한다. 본 연구에서는 각 슬라이스에 대한 슬라이스 복잡도와 전체 슬라이스 복잡도 간의 관계를 보여주고, VFG에서의 극소 수정(atomic modification)과 합성 연산자(concatenation operator)를 통해서 슬라이스 복잡도 척도의 스케일(scale) 인자들을 증명한다.

ABSTRACT

We develop the new data slice representation, called the data value flow graph(VFG), for modeling the information flow on data slices. Then we define a slice complexity measure by using the existing flow complexity measure in order to measure the complexity of the information flow on VFG. We show relation of the slice complexity on a slice with the slice complexity on a procedure. We also demonstrate the measurement scale factors through a set of atomic modifications and a concatenation operator on VFG.

I. 서 론

전체 시스템 비용에서 소프트웨어 비용의 증가에 따라서, 소프트웨어의 심리적 복잡도의 측정에 대한 관심이 증가해왔다. 소프트웨어 개발 단계에서 소프트웨어 측정은 프로그래머에게 중요한 정보를 제공할 수 있고, 이해하기 쉬운 코드의 작성에 도움이 될 수 있다[4].

그러나 측정 분야에서 대부분의 기존의 메트릭들[2, 3]은 프로그램의 표면적인 특징들만을 고려하고 있다고 말할 수 있다. 프로그램의 이해에 지대한 미치는 것은 프로그램의 표면적인 특징들이 아니라 프로그램의 정보 흐름(information flow)의 특징이다[4].

프로그램 이해에 대한 대부분의 어려움은 프로그래머가 프로그램의 이해를 위해서 사용하는 문(statement)들의 묶음(grouping)과 소스 프로그램 리스트와 프로그래밍 환경이 제시하는 문들의 묶음과의 차이 때문이다[4]. Weiser[10, 11]에 의하면 프로그래머들이 프로그램을 이해할 때 순차적 관계 이외의 방법으로

* 조선대학교 컴퓨터공학부 전자계산학과
접수일자: 2000. 8. 10

문(statement)들을 묶는 경향이 있다고 하였다.

일반적으로 묶음(grouping)을 위한 기준은 정보의 흐름인 데이터와 제어의 흐름에 관계가 있다.

이런 정보가 PDG(program dependency graph) [23]와 슬라이스(slice)에 분명하게 표현되므로, 정보 흐름의 복잡도는 PDG와 슬라이스를 이용하여 가장 쉽고 분명하게 측정될 수 있다[4].

프로그램에서 정보 흐름을 정확하고 자세하게 표현하려는 연구들[14]과 주로 프로그램의 응집도를 측정하는 슬라이스에 기반한 대부분의 척도[8, 9, 12, 16, 18]들은 PDG와 슬라이스에 나타나는 정보의 흐름을 측정할 수 없다.

프로그래머가 프로그램을 이해하고 디버깅할 때 슬라이스를 사용한다면[11], 슬라이스의 복잡도의 측정은 호소력을 가지므로[4], 본 연구에서는 프로그램의 정보 흐름의 복잡도를 측정하기 위해서 슬라이스 복잡도를 측정하고자 한다.

슬라이스 복잡도를 측정하기 위해서는 슬라이스 복잡도 척도의 정의, 프로그램의 전체 복잡도 메트릭을 위해서 슬라이스 복잡도를 결합하는 방법들에 대한 정의가 필요하다[4].

슬라이스 복잡도 메트릭이 의미 있는 측정을 제공하기 위해서는 메트릭이 엄밀하게 정의되어야하고, 측정을 위한 속성을 정확히 반영해야하고, 이런 속성들을 포착하는 모델에 근거해야한다[5, 6, 7].

본 연구의 목적은 적합한 슬라이스 복잡도 측정 모델을 정의하며, 슬라이스 복잡도의 정량화를 위해서 정의된 모델을 사용하는 척도를 개발하는 것이다. 또한 측정값들이 슬라이스 복잡도 모델 순서와 일치함을 보이고, 그들의 스케일(scale) 특성들을 결정함으로써 척도를 검증한다.

슬라이스는 프로그램의 특정 위치에서 한 변수의 값에 영향을 미치는 문(statement)들의 묶음에 대한 추상화이다[10, 11, 12].

프로그램의 이해에 슬라이스를 사용한다면[11], 프로그램의 이해는 슬라이스에서의 정보 흐름을 탐색하는 과정이라고 할 수 있다.

슬라이스의 단위로서 데이터 토큰들을 사용하는 데이터 슬라이스(data slices)[8]는 정보 흐름에 따른 변화들이 프로시저의 어느 한 부분에서 변화의 원인이 될 것이다.

그러므로 본 연구에서는 데이터 슬라이스에서의 정보 흐름을 모델링하기 위해서 데이터 값 흐름 그래프(VFG: data Value Flow Graph)라고 하는 데이터 슬라이스 표현 방법을 개발한다. VFG는 데이터 슬라이스에서 정보의 흐름 과정을 분명하게 보여주므로, 관심 있는 변수 값의 변화 과정을 쉽게 탐색할 수 있고, 프로그램의 이해를 증진시킬 수 있다.

슬라이스 복잡도 척도는 모델에서 정량적인 값으로의 사상을 명시한다. 측정값은 복잡도 순서와 일치해야한다. 척도가 복잡도 순서와 일치하는가를 보여주는 방법은 모델과 척도에 대한 코드 수정의 영향을 평가하는 것이다. 본 연구에서는 복잡도 측정값에 적용될 수 있는 스케일 특성들을 보여준다. 척도의 스케일 타입은 매우 중요하고 측정 값 사이에서 의미 있게 수행될 수 있는 산술연산을 결정한다[9, 19, 20].

본 논문은 다음과 같이 구성된다. Ⅱ 장에서는 데이터 슬라이스에 대해여 고찰하고, Ⅲ 장에서는 데이터 슬라이스에서의 데이터 값 흐름을 모델링하는 VFG와 슬라이스 복잡도 척도를 정의한다. Ⅳ장에서는 척도의 스케일 특성을 평가한다. Ⅴ장에서 결론을 제시한다.

Ⅱ. 데이터 슬라이스

슬라이싱은 Weiser[10]에 의해 소개된 프로그램 축소 방법이다. 변수 v에 관한 문장 s에서 프로시저 슬라이스는 s에서 v의 값에 영향을 미칠 수 있는 일련의 모든 문장들과 술어들이다. 슬라이스는 디버깅 도구와 프로그램 이해를 위한 보조 수단으로 제안되었다. 슬라이스는 특정한 변수의 값에 영향을 미치는 모든 노드들을 얻기 위해 PDG(Program Dependency graph)에서 후향(backward)으로 탐색하여 얻어진다.

Weiser[12]는 슬라이싱(slicing)에 근거한 메트릭들을 제안했으며, Lognworth는 응집도 척도로 이들의 사용을 처음으로 연구하였다[15].Linda[16]는 메트릭 슬라이스(metric slice)에 근거한 응집도 척도를 제안했다. 메트릭 슬라이스는 use와 used by 관계에 의하여 계산된다. Bieman[8,9]은 데이터 토큰을 사용하기 위해서 메트릭 슬라이스를 수정한 데이터 슬라이스(data slices)에 근거한 응집도 척도를 제안했다.

데이터 슬라이스는 각 출력에 대해서 계산된다. 데이터 토큰 v를 위한 데이터 슬라이스를 v의 전향

(forward)과 후향(backward) 슬라이스를 포함하는 문장 내의 일련의 모든 데이터 토큰들로 간주한다. 후향 슬라이스는 프로시저의 종결부분으로부터 계산되고, 전향 슬라이스는 후향 슬라이스 선두로부터 계산된다.

예를 들어서, 그림1의 프로시저SumAndProduct에서 출력 SumN과 ProdN에 대한 데이터 슬라이스는 다음과 같은 일련의 데이터 토큰들이다.

S(SumN)={N1, SumN1, I1, SumN2, 01, I2, 12, N2, SumN3, SumN4, I3}
S(ProdN)={N1, ProdN1, I1, ProdN2, I1, I2, 12, N2, ProdN3, ProdN4, I4}

```
procedure SumAndProduct(
    N:integer; var SumN, ProdN:integer)
var I:integer;
begin
    SumN:=0;
    ProdN:=1;
    for I:=1 to N do begin
                SumN:=SumN+I;
                ProdN:=ProdN+I;
    end
end
```

그림 1. SumAndProdN 프로시저[Jame 94]

여기서 vi는 프로시저에서 변수 v에 대한 i번째 토큰을 나타낸다.

슬라이스 기반 척도들[8, 9, 12, 16, 18]은 측정 값을 계산하기 위해서 슬라이스 추상화(slice abtraction)와 슬라이스 프로파일(slice profile)을 사용한다. 슬라이스 추상화와 슬라이스 프로파일은 응집도의 중요한 속성들을 가시화 시킬 수 있는 메카니즘을 제공하지만, 슬라이스 프로파일에 근거하여 응집도를 분석할 때, 하나의 토큰이 데이터 슬라이스 내에 있다는 것을 아는 것은 중요하지만, 토큰의 이름은 중요하지 않기 때문에[9], 슬라이스 추상화와 슬라이스 프로파일의 사용은 슬라이스에서의 정보 흐름을 나타낼 수 없다. 따라서 슬라이스에서 정보 흐름을 표현할 수 있는 새로운 데이터 슬라이스 표현이 필요하므로 본 연구에서는 데이터 슬라이스에서의 정보흐름을 표현할 수 있는 데이터 값 흐름 그래프(data Value Flow Graph)를 제안한다

Ⅲ. Value Flow Graph와 슬라이스 복잡도 척도

1) data Value Flow Graph(VFG)

슬라이스 복잡도의 조사를 위하여 본 연구에서는 데이터 슬라이스 내에 정보 흐름을 모델링하기 위해서 VFG(data Value Flow Graph)를 다음과 같이 정의한다.

VFG(S)로 표현되는 슬라이스 S에 대한 VFG는 방향성 그래프, VFG(S)=(N, E)이다. 여기서, N은 슬라이스 S에 포함된 데이터 토큰들의 집합이고, E는 노드들 간의 데이터 값 흐름을 나타내는 간선들의 집합이다.

Ni, Nj∈N이라고 할 때, Ni에서 Nj으로의 정보흐름 간선(information flow edge)은 Ni → ifNj로 표현되고, Ni의 값이 Nj의 값에 영향을 미치면 Nj는 Ni를 사용한다라고 한다. Ni를 사용하는 연산의 결과가 Nj에 배정되면 Nj는 Ni를 직접 사용한다라고 한다. Nj가 Ni를 직접사용하지 않으면 Nj는 Ni를 간접 사용한다라고 한다. Nj가 Ni를 사용하면 Ni에서 Nj로 정보가 흐른다라고 한다.

vi, vj∈N인 vi와 vj가 데이터 슬라이스 S에서의 하나의 변수 v에 대한 데이터 토큰들이라고 할 때, 프로그램의 제어흐름 관계에서 vi가 vj보다 앞서고 vj가 vi를 사용하지 않으면 vi에서 vj로 정보가 흐른다라고 한다.

Ni, Nj, Nk∈N일 때, 다음 조건들을 만족하면 VFG는 Ni에서 Nj로의 정보흐름 간선(즉, Ni→ifNj)을 포함한다.

① Ni에서 Nj로 정보가 흐른다.
② Ni에서 Nk로 그리고 Nk에서 Nj로 정보가 흐르는 노드 Nk가 존재하지 않는다.

그림 1의 출력 SumN과 ProdN의 데이터 슬라이스에 대한 VFG는 그림 2와 3과 같다.

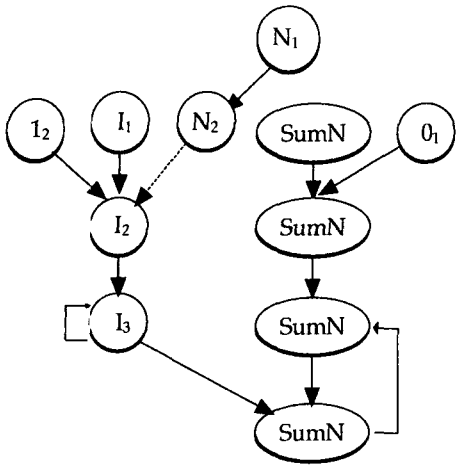

그림 2. SumN에 대한 VFG

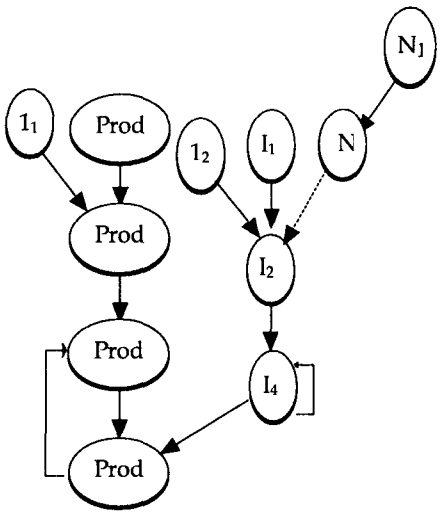

그림3. ProdN에 대한 VFG

프로시저 P에 대한 VGF는 프로시저 P에서의 각 데이터 슬라이스에 대한 결합(concatenation)으로 정의된다.

정의 1: NSi와 ESi가 VFG(Si)의 노들과 간선들의 집합일 때, 프로시저 P에 대한 VFG는 다음과 같이 정의된다.

$$VFG(P) = VFG(S1) \circ VFG(S2) \circ \cdots \circ VFG(Sn) \quad \cdots\cdots (1)$$

$$= (NS1 \cup NS2 \cup \cdots \cup NSn\ ,\ ES1 \cup ES2 \cup \cdots \cup ESn)$$

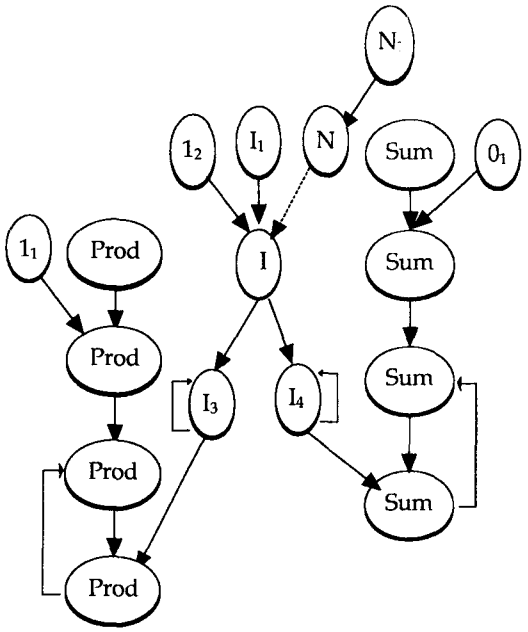

그림 4. SumAndProduct에 대한 VFG

그림 2와 3을 결합한 프로시저SumAndProduct에 대한VFG(SumAndProduct)=VFG(SumN) ∘ VGF(Prod N)는 그림 4와 같다.

슬라이스의 토대로서 데이터 토큰들의 사용은 정보 흐름에 따른 변화들이 프로시저의 어느 한 슬라이스에서의 변화의 원인이 될 것이고[9], 슬라이스에서의 변화들은 VFG에 반영되고, 또한 슬라이스 복잡도에 영향을 미치게된다. 이러한 변화들에는 데이터 토큰의 추가와 삭제, 주어진 정황에서 데이터 값 흐름의 변화 등이 있는데, 이러한 변화들은 최소한 하나의 VFG에서의 변화를 유발한다.

2) 슬라이스 복잡도 척도

프로그램의 이해와 디버깅에 슬라이스를 사용한다면[11], 프로그램의 이해는 슬라이스에서의 정보 흐름을 탐색하는 과정이라고 할 수 있다. 즉, VFG(S)의 종결 노드에서 출발하여 종결노드에 영향을 미치는 노드들을 후향(backward)으로 탐색하는 과정이라고 할 수 있다. VFG(S)에서 종결노드에 이르는 경로가 많을수록, 즉 VFG(S)를 후향으로 탐색하는 과정에서 분기 노드들이 많을수록 출력에 영향을 미치는 과정을 추적하기 어려우므로, 슬라이스 S에 대한 복잡도는 증가한

다. 그러므로 VFG(S)에서의 슬라이스 복잡도는 간선들의 개수와 노드들의 개수에 관계하여 흐름 복잡도를 측정하는 기존의 복잡도 척도를 사용하여 측정될 수 있다. VFG(S)에서의 슬라이스 복잡도는 McCabe가 제안한 복잡도 척도인 식 (2)[1]를 이용하여 정의된다.

$$SC(VFS(S)) = |E| - |N| + 1 \quad \cdots\cdots (2)$$

SC(VFS(S)) : VFS(S)에 대한 슬라이스 복잡도
|E| : VFG(S)에서의 간선들의 개수
|N| : VFG(S)에서의 노드들의 개수

프로시저 P에 대한 슬라이스 복잡도는 VFG(P)에서 식(2)와 동일하게 식 (3)과 같이 정의된다.

$$SC(VFG(P)) = |E| - |N| + 1 \quad \cdots\cdots (3)$$

그림 1의 프로시저에 대한 슬라이스 복잡도는 다음과 같다.

SC(SunN) = 12 - 11 + 1 = 2
SC(ProdN) = 12 - 11 + 1 = 2
SC(SumAndProd) = 20 - 17 + 1 = 4

합성되는 각 슬라이스들간에 실질적인 상호작용이 존재하고19], 슬라이스 결합을 통한 슬라이스 재순서화(reordering)가 반드시 더욱 구조적으로 복잡하지 않으므로[6], 프로시저 P에 대한 슬라이스 복잡도 측정 SC(VFG(P))와 프로시저 P에서의 슬라이스 Si에 대한 복잡도 측정 VFG(Si)간에는 다음 관계가 성립한다.

정리 1 : VFG(P) = VFG(S1)∘VFG(S2)∘…∘VFG(Sn)이면, 다음 식이 성립한다.

$$SC(VFG(P)) \le SC(VFG(S1)) + SC(VFG(S2)) + \cdots + SC(VFG(Sn)) \quad \cdots\cdots (4)$$

증명 : VFG(P)=VFG(S1)∘VFG(S2)에 대해서, NS1, NS2가 VFG(S1), VFG(S2)의 노드들의 집합이고, ES1, ES2가 간선들의 집합이라 가정한다.

SC(VFG(S1)∘VFG(S2)) =
|ES1∪ES2| - |NS1∪NS2| + 1
|NS1∪NS2| = |NS1| + |NS2| - |NS1∩| NS2|
|ES1∪ES2| = |ES1| + |ES2| - |ES1∩ES2|.

그러므로,
SC(VFG(S1)∘VFG(S2)) =
(|ES1| + |ES2| - |ES1∩ES2|) -
(|NS1| + |NS2| - |NS1∩NS2|) + 1
SC(VFG(S1))+SC(VFG(S2)) =
|ES1| - |NS1| + 1 + |ES2|-|NS2|+1

이때,
NS1∩NS2의 노드들로 구성된 스패닝 그래프에서 |NS1∩NS2| - |ES1∩ES2| ≤ 1이 성립하므로
SC(VFG(S1)∘VFG(S2)) ≤ SC(VFG(S1)) + SC(VFG(S2)) 이다.

Ⅳ. Validation

Fenton[21]은 검증(validation)을 척도가 주장된 속성의 적당한 숫자적 특징임을 보증하는 과정이라고 정의했다. 측정되는 속성이 확실히 이해되지 않을 때, 이런 종류의 검증은 매우 어렵다. 본 연구에서 제안한 VFG의 상대적 등급의 결정을 위하여, VFG가 슬라이스 복잡도 척도 값과 일치하는가를 확인하기 위해서, 인간의 직관에 의존할 필요가 있다. Zuse는 소프트웨어 척도들이 그들의 스케일 특성들에 의해서 검증될 수 있다는 것을 보여준다[19, 20].

본 연구에서는 스케일 특성에 의하여 슬라이스 복잡도 척도를 분석 검증하기 위해서 Zuse[19]와 Weyuker[22]의 방법을 토대로 한다. 먼저 척도가 인간의 직관과 일치하는 ordinal 스케일에 있음을 보여주고, 다음으로 척도가 ratio 스케일의 요구조건을 만족하는 가를 결정한다.

1) 척도와 ordinal 스케일

슬라이스 복잡도 척도가 ordinal 스케일로서 기술되기 위해서는 경험 관계(empirical relation) 또는 뷰포트(viewport)라 불리는 슬라이스 복잡도 척도에 대한 직관은 세 가지 공리를 만족해야한다: 반사성, 이행성,

완전성[19]. 이것들은 약순서(weak order)의 필요충분 조건들이다. Zuse[19, 20]의 정의에 따라서, 프로시저 P에서 P1, P2∈P 일 때, 이진 관계로 슬라이스 복잡도 뷰포트를 다음과 같이 정의한다.

P1 • >P2 P1이 P2보다 더욱 복잡하다.
P1 • =P2 P1과 P2의 복잡도는 같다.
P1 • ≥P2 P1 • >P2 혹은 P1 • =P2이다.

슬라이스 복잡도 뷰포트의 일반적인 정의를 내리는 것은 가능하지 않기 때문에 원소 뷰포인트(elementary viewpoint)라 불리는 위의 관계의 부분집합을 사용할 수 있다[9]. 원소 뷰포인트는 VFG에서의 극소수정(atomic modification)[19]들의 유한 집합에 의하여 정의된다. 슬라이스 복잡도 척도가 ordinal 스케일에 있다는 것을 보여주기 위해서, 그것이 극소 수정과 일치함을 보여줄 필요가 있다[9]. 그래서 본 연구에서는 VFG에서의 극소 수정들의 직관적으로 분명한 효과에 의하여 슬라이스의 복잡도 순서를 평가한다.

극소 수정은 VFG에서 간선들이나 노드들의 첨가, 삭제, 이동과 같은 임의의 수정에 의해 VFG(S)의 VFG(S')로의 변환이므로[19], 극소 수정은 슬라이스 복잡도를 고려하지 않고 VFG에 적용될 수 있다.

슬라이스 복잡도 척도는 극소 수정들에 대해서 특별한 반응을 보이는데, 이것을 슬라이스 복잡도 척도의 부분특성(partial properties)[19]이라 한다.

슬라이스 복잡도 척도의 특성은 다음과 같은 극소 수정 AM1, AM2, AM3에 대한 부분 특성들에 의해 완전하게 서술될 수 있다[19].

AM1 = VFG내의 임의의 위치에 하나의 노드와 하나의 간선의 추가.
AM2 = VFG의 한 위치로부터 다른 위치로 간선의 이동.
AM3 = VFG 내의 임의의 위치에 간선의 추가.
즉,
P1 • =M1P2 ⇒ SC(P1) = SC(P2)
P1 • =M2P2 ⇒ SC(P1) = SC(P2)
P1 • >M3P2 ⇒ SC(P1) > SC(P2)

그러므로, 극소수정 AM1, AM2, AM3에 의해 서술되는 부분특성들에 의하여 프로시저 P에 대한 슬라이스 복잡도를 ordinal 스케일로 사용할 수 있다.

2) 척도와 ratio 스케일

척도가 ratio 스케일에 있는가를 보여주기 위해서, 관계형 시스템에 프로그램 합성 연산자 "⊕"를 도입하여 확장된 관계형 시스템 (S, • ≥, ⊕)를 가진다. Zuse[19, 20]에 따르면, 척도가 실수 값 함수 m이고, ordinal 스케일 상에 있고, 다음 공리를 만족하면, 척도는 ratio 스케일 상에 있다.

P1 • >P2 ⇔ m(P1) ≥ m(P2)
m(P1⊕P2) ⇔ m(P1) + m(P2) ..... (5)

첫번째 공리는 m이 측정된 속성에 의해 부과된 프로시저의 직관적 순서와 일치할 것을 요구한다. 즉 m이 ordinal 스케일에 있어야 한다. 두번째 공리는 m이 가법적(additive)임을 요구한다. 슬라이스 복잡도 척도가 이 공리들과 일치하는 지를 검증하기 위하여 이진 연산자 ⊕를 다음과 같이 정의한다.

정의 2: 두 개의 프로시저 P1과 P2에서의 VFG(P1)과 VFG(P2)에 대해서, 이진연산자 ⊕는 VFG(P1)의 임의의 노드로부터 VFG(P2)의 임의의 노드로 간선을 추가한다. 이진 연산자 ⊕는 두 프로시저간의 호출과 반환 관계에 의한 데이터 값의 흐름을 나타낸다.

정리 2: 이진 연산자 ⊕에 대해서 다음이 성립한다.
SC(VFG(P1)⊕VGP(P2)) ≥
SC(VFG(P1))+SC(VGP(P2)) ..... (6)

증명: |Ef|가 VGF(P1)과 VFG(P2)간의 데이터 값 흐름을 나타내는 간선의 개수라 가정한다.

데이터 슬라이스에 근거한 VGF(P1)과 VFG(P2)간에는 반드시 하나 이상의 호출 및 반환 관계가 존재하므로 |Ef|≥1이다.

NP1, NP2가 VFG(P1), VFG(P2)의 정점들의 집합이고, EP1, EP2가 간선들의 집합이라 할 때,

SC(VFG(P1)⊕VFG(P2)) =
|EP1|+|EP2|-|NP1|-|NP2|+1+|Ef|
SC(VFG(P1))+SC(VFG(P2)) =

$$|EP1|-|NP1|+1+|EP2|-|NP2|+1$$

이때, $|Ef| \geq 1$ 이므로,

$$SC(VFG(P1) \oplus VGP(P2)) \geq SC(VFG(P1))+SC(VGP(P2))$$

여기에서 정의되는 이진 연산자 ⊕에 대해서 슬라이스 복잡도 척도가 Zuse가 요구하는 가법성을 만족하지는 않지만, 식(7)과 같은 weyuker[22]의 8번째 공리를 만족한다.

$$\mu(P)+\mu(Q) \leq \mu(P;Q) \quad \cdots\cdots (7)$$

Weyuker의 8번째 공리는 ratio 스케일에 대해서 의미가 있다. 즉, 슬라이스 복잡도 척도가 ratio 스케일을 나타내면 이 공리는 의미를 갖는다[19].

그러므로, 이진 연산 ⊕에 대한 Weyuker의 8번째 공리에 근거하고, ordinal 스케일의 부분특성을 수용하면, 슬라이스 복잡도 척도를 ratio 스케일로 사용할 수 있다.

V. 결 론

본 연구에서는 측정 이론으로부터의 원칙들을 이용하여 슬라이스 복잡도 척도를 유도했다. 먼지, 데이터 슬라이스에서의 정보 흐름을 모델링하기 위해 데이터 값 흐름 그래프(VFG: Data Value Flow Graph)라고 하는 데이터 슬라이스 표현을 개발하였다. 다음으로, VFG에서의 정보 흐름의 복잡도를 측정하기 위해 기존의 흐름 복잡도를 이용하여 슬라이스 복잡도 척도를 정의하였다.

또한, 각 슬라이스에 대한 슬라이스 복잡도 측정값과 전체 프로시저에 대한 슬라이스 복잡도 측정값의 관계를 보였다. 간단한 극소 수정들의 집합에 의해 부과되는 순서가 슬라이스 복잡도 척도에 관한 우리의 직관과 일치하는 정도로 슬라이스 복잡도 척도가 ordinal 스케일의 요구조건을 만족함을 보여준다. 또한 슬라이스 복잡도 척도가 Zuse가 요구하는 가법성(additive)을 만족하지 않지만 ratio 스케일에 대해서 의미 있는 Weyuker의 8번째 공리를 만족함을 보여주었다. 사용자가 슬라이스 복잡도 척도의 가법성을 전제하면 측정값을 분석할 때 ratio 스케일이 아니라 ordinal 스케일 연산을 사용할 수 있다. 그러나 사용자가 Weyuker의 8번째 공리에 동의하면 측정값을 분석할 때 ordinal 스케일뿐만 아니라 ratio 스케일 연산도 사용할 수 있다.

현재까지의 작업은 단일 프로시저에서의 슬라이스 복잡도 측정에 집중하였지만, 프로시저간 슬라이스(interprocedure slice)뿐만 아니라 객체 지향 환경에서의 데이터 슬라이스에 근거한 슬라이스 복잡도 측정에 대한 연구가 필요하다.

참고문헌

[1] T. J. McCabe, "A complexity measure", IEEE Transaction Software Engineering SE-2, pp.308-320, 1976.

[2] N. F. Schneidewind, H. M. Hoffman, "An experiment in software error data collection and analysis", IEEE Transaction Software Engineering SE-5, 3, pp.276-286, May, 1979.

[3] M. H. Halstead, "Element of software science", Elsevier North Holland, New York, 1977.

[4] Karl J. Ottensteion, Linda M. Ottensteion, "The program dependence graph in a software development environment", Proceeding of the ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environment, ACM SIGPLAN Notices, vol.19, no.5, pp.177-184, May, 1984.

[5] Rachel Harrison, Steve J. Counsell, "An Evaluation of MOOD set of object-oriented software metrics", IEEE Transaction Software Engineering, vol.24, no.6, pp.491-496, June, 1989.

[6] B.A. Kitchenham, N. Fenton, S. Lawrence, "Towards a framework for software measurement validation", IEEE Transaction Software Engineering, vol.21, no.12, pp.929-944, December, 1995.

[7] A.L.Baker, J.M.Bieman, N.E.Fenton, A.C.Melton, R.W. Whitty, "A philosophy for software measurement", Journal of Systems and Software, vol.12, no.3 pp. 277-281, July, 1990.

[8] J.M.Bieman, B.K.Kang, "Measuring design-level cohesion", IEEE Transaction Software Engineering, vol.24, no.2, pp.111-124, 1998.

[9] J.M.Bieman, L.M.Otto, "Measuring functional cohesion", IEEE Transaction Software Engineering, vol.20, no.2, pp.111-124, 1994.

[10] M. Weiser, "Program slicing", In Prodeedings of the 5th International conference on Software Engineering, pp.439-449, 1981.

[11] M. Weiser, "Programmers use slices when debuggung", Communication of the ACM, vol.25, no.7, pp.446-452, 1982.

[12] M. Weiser, "Program slicing", IEEE Transaction Software Engineering, vol.10, no.4, pp.352-357, 1984.

[13] Sallie Henry, Dennis Kafura, "Software structure metrics based on information flow", IEEE Transaction Software Engineering SE-7, 5, pp.510-518, Sept. 1981

[14] H.E. Dunsmore, J.D.Gannon, "Data referencing: an empirical investigation", IEEE Computer, vol.12, no.12, pp.50-59, Dec., 1979.

[15] H. D. longworth, "Slice based program metrics", Master's thesis, Michigan Technological university, 1985.

[16] Linda M. Ott, Jeffrey J. Thuss, "Slice based metrics for estimating cohesion", Proc. IEEE-CS International Software Metrics Symposium, Baltimore, pp.71-81, May, 1993.

[17] Linda M. Ott, James M. Bieman, B.K. Kang, Bindu Mehra, "Developing measures of class cohesion for object-oriented software", 7th Annual Oregon Workshop on Software Metrics, June, 1995.

[18] James M. Bieman, B.K. Kang, "Cohesion and reuse in an object-oriented system", Proceeding ACM Symposium on Software Reusability (SSR'95), pp.259-262, April, 1995.

[19] Horst Zuse, "Software Complexity-Measures and Methods", pp.25-37, Walter de Gruyter, New York, 1991.

[20] Horst Zuse, "Support of validation of software measures by measurement theory", Invited Presentation at the 15th International Conference on Software Engineering(ICSE-15) and the First IEE-CS International Software Metrics Symposium, Baltimore, May, 1993.

[21] N.Fenton, "Software Metrics-A Rigorous Approach", Chapman and Hall, Kondon, 1991.

[22] E.Weyuker, "Evaluating software complexity measure", IEEE Transaction Software Engineering, vol.14, no.9, pp.1357-1356, Sept. 1988.

[23] Jeanne Ferrante, Karl J. Ottenstein, Joe D. Warren, "The program dependence graph and its use in optimization", ACM Transaction on Programming Language and Systems, vol.9, no.3, pp.319-349, July, 1987

문 유 미(Yu-mi Mun)
1983년 조선대학교 전자계산학과 (이학사)
1987년 조선대학교 산업공학과(이학 석사)
1998년~현재 조선대학교 컴퓨터공학부 전자계산학과 박사과정(software engineering lab)
조선대학교 공과대학 컴퓨터공학부 출강
송원대학교 금융정보학과 겸임교수
※관심분야 : 소프트웨어 공학, 소프트웨어 메트릭스, 프로그램복잡도 러프 및 퍼지이론.

최완규(Wan Kyoo Choi)
1988년 서울대학교 종교학과 졸업 (학사)
1992년~1993년 (주)공성통신 전산실
1993년~1995년 한양시스템 전산실
1997년 조선대학교 전자계산학과 (이학석사)
2000년 조선대학교 전자계산학과(이학박사)
2000년~현재 광주대학교 컴퓨터전자통신공학부 전임강사
※관심분야 : 소프트웨어 공학, 프로그래밍 언어, 객체지향 시스템, 리프집합

이 성 주(Sang-Hyun Bae)
1970년 한남대학교 물리학과 (학사)
1992년 광운대학교 전자계산학과 (이학석사)
1998년 대구 가톨릭대학교 (이학박사)
1988년~1990년 조선대학교 전자계산소 소장
1995년~1997년 조선대학교 정보과학대학장
1981년~현재 조선대학교 컴퓨터공학부 교수
※관심분야 : 소프트웨어 공학, 프로그래밍 언어, 객체 지향 시스템, 리프집합