Variable Reference Graph 의 설계 및 구현

이헌기, 이문수, 신규상
한국전자통신연구원 컴퓨터.소프트웨어기술연구소
e-mail : {lee, mslee, gsshin}@etri.re.kr

A Design and Implementation of Variable Reference Graph

Heon-Ki Lee, Mun-Su Lee, Gyu-Sang Shin
ETRI-Computer & Software Technology Laboratory

요 약

Variable Reference Graph 는 C 언어로 작성된 프로그램으로부터 상호 절차적인 자료 흐름 분석 정보를 수평적 방향 그래프(directed graph)로 자동 생성해주는 역공학(reverse engineering) 도구들 중 하나이다. 본 논문에서는 판독성 있는 구조적 정보를 제공하기위한 그래픽 표현의 전략을 바탕으로 JAVA 로 구현된 그래픽 사용자 인터페이스(graphic user interface) 및 그래프 레이아웃 알고리즘(graph layout algorithm)을 기술한다. 이 알고리즘은 4 단계로 구성되어 있다: 정보 모형, 레벨 알고리즘, 순서 알고리즘, 위치 알고리즘. 각 단계별에서 수행되는 주요 알고리즘을 살펴 본다. 특히, 이 알고리즘들은 사이클(cycle) 및 비사이클(acyclic) 방향 그래프, 그리고 트리(tree)를 수평적 계층 구조를 생성하는데 사용될 수 있다. 본 논문에서 구현된 Variable Reference Graph 는 소프트웨어 재공학 도구를 개발하는 RESORT(RESearch on object-oriented SOftware Reengineering Technology) 과제에서 개발되었다.

1. 서론

소프트웨어를 유지 보수하거나 점증적으로 발전시켜 나갈 때 가장 중요한 과제는 컴퓨터 프로그램을 이해하기 위한 과정 또는 활동인 프로그램의 이해이다 [1]. 프로그램 이해는 유지 보수 작업의 기본과 핵심이 되며, 시간과 비용 측면에서도 가장 큰 비중을 차지하는 작업 중 하나이다. 그러나, 대부분 유지 보수자들의 어려운 점들은 수정이 가해져서 익숙치 않는 코드를 유지 보수해야 하거나, 그리고 관련된 문서들이 구 버전이거나, 부적절하거나, 일치성 없거나, 또는 부재하다는 것이다.

이런 경우, 유지 보수자는 프로그램을 이해하기 위해서 단지 코드에만 의존할 수 밖에 없다. 유지 보수자가 소스 코드의 정보들을 파악하기 위해 어떻게 체계적인 방법을 발견하는가가 핵심적인 요소이다 [3].

이 문제를 해결하는 방법 중 하나는 좀더 쉽게 이해할 수 있는 방법으로 프로그램의 의미와 구조를 소스 코드로부터 추출하여 그래픽하게 표현하는 도구의 개발이다. 역공학 도구들 중 하나인 Variable Reference Graph 는 자료(전역변수 및 구조체)의 정의와 사용에 관한 상호 절차적인 자료 흐름 분석을 지원하는 도구이다. 즉, C 프로그램에서 함수들이 어떻게 전역 변수 또는 구조체들을 정의/사용하는지를 식별함과 동시에 정의/사용하는 함수들 간의 호출 관계를 수평적 계층 방향 그래프로 자동 생성하고, 그리고 자료의 참조 빈도수를 나타내는 각 함수의 참조 복잡도 정보들을 시각적으로 제공하는 도구이다.

일반적으로, 소프트웨어 도구들에서 그래프 표현은 판독성 있는 구조적 정보를 제공하여야 한다. 시각화에 도움을 줄 수 있는 그래프의 표현 및 레이아웃은 시각화의 실제 내용보다 더 중요하게 구축되어야 할 것이다.

그래프를 자동 레이아웃하는 여러 가지의 알고리즘들이 개발되었지만 [4, 5, 6], 본 논문에서는 Sugiyama 알고리즘을 확장하여 JAVA 로 Variable Reference Graph 를 구현하였다. Variable Reference Graph 는 RESORT 시스템 중에서 역공학 도구인 RESORT/R 의 한 부분으로써 구현 되었으며, 나머지 RESORT/R 의 도구 특징은 다음과 같다 [7, 8, 9].

- Call Graph : 함수들 간의 호출 관계 표현
- Control Flow Graph : 함수 내부의 제어흐름 표현
- Structure Chart : 모듈들 간의 호출 관계 및 자료/제어 커플(couple) 표현
- Source Code Browser : 하이퍼텍스트(hypertext)를 이용하여 소스코드를 하이라이팅(highlighting) 표현

본 논문에서 제 2 장은 그래픽 사용자 인터페이스의 구성 요소 및 뷰의 종류를 설명하고, 제 3 장은 4 단계로 구성된 그래프 레이아웃 알고리즘들을 기술하며, 이를 바탕으로 구현된 사례를 살펴보고, 마지막으로 제 4 장에서 결론을 맺는다.

2. 그래픽 사용자 인터페이스 환경

일반 사용자에게 보다 편리한 사용자 인터페이스 환경을 제공하기 위해 Variable Reference Graph 의 화면은 <그림 4>와 같이 그래프(graph)와 트리(tree) 프레임(frame)으로 구성된다.

첫째, 그래프 프레임은 전역 변수와 함수들 간의 정의/사용 관계를 수평적 계층 방향 그래프의 Variable Reference Graph 로 보여준다. 이 프레임에서 한 노드(함수)를 선택하면, 그의 모든 후손 노드들을 하이라이팅하여 시각화 해준다. 그래프 프레임의 주요 표현 정보는 다음과 같다.

- 뷰 형태
  - Variable Reference Graph

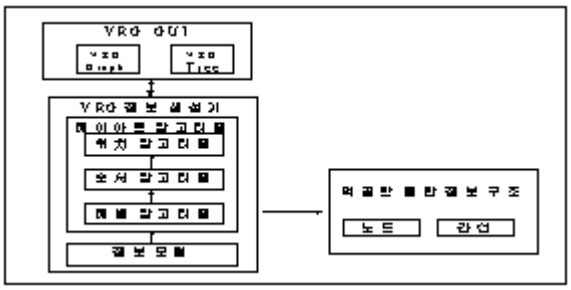
- 노드의 종류
  - 자료형 종류
    . 전역 변수
    . 구조체
  - 함수의 종류
    . 전역 변수(구조체)를 정의하는 함수
    . 전역 변수(구조체)를 사용하는 함수
    . 전역 변수(구조체)를 정의/사용하는 함수

- 구조적 측정 종류
  - 전역 변수(구조체) 참조 복잡도(reference complexity)

둘째, 트리 프레임은 전역 변수(구조체)와 모든 함수들 간의 참조 관계를 수평적 트리 형태인 Variable Reference Tree 로 보여준다. 이 프레임에서 한 노드(전역 변수/구조체)를 선택하면, 그와 관련된 함수들을 그래픽 프레임 상에서 Variable Reference Graph 를 생성한다. 트리 프레임의 주요 표현 정보는 다음과 같다.

- 뷰 형태
  - Variable Reference Tree

- 노드의 종류
  - 자료형 종류
    . 전역 변수
    . 구조체
  - 함수의 종류
    . 사용자 정의 함수



<그림 1> Variable Reference Graph 의 구조도

3. 수평적 방향 그래프 레이아웃 알고리즘

Variable Reference Graph 의 수평적 방향 그래프 레이아웃 알고리즘은 Sugiyama 알고리즘을 [4] 확장하여 JAVA 로 구현되었고, <그림 1>과 같이 크게 2 가지로 구성되어 있다. (1) 정보 모형은 필요한 입력 그래프(input graph)인 Variable Reference Graph 의 노드와 간선 정보들을 인접 리스트(adjacency list) 구조로 저장한다. (2) 레이아웃 알고리즘은 노드의 레벨 부여 및 간선의 종류 식별, 노드의 순서화, 노드와 간선의 실제 좌표 부여 등의 절차에 의해 사이클(cycle) 및 비사이클(acyclic) 방향 그래프의 수평적 계층 레이아웃을 생성한다. 7 만 라인(line) 이하의 중 규모인 C 언어로 된 소프트웨어인 (1) Calculator, (2) CrazyWebBoard, (3) ncFTP (4) xFig 등의 프로그램들을 대상으로 역공학 변환하는 과정에서 나타나는 프로그램의 구성 요소의 사례결과는 <표 1>과 같다.

<표 1> RESORT 시스템 적용 결과

	File 수	C 코드 라인	사용자 함수	라이브러리 함수	전역 변수
Calculator	1	762	28	10	13
Crazy Web Board	19	3,639	56	59	6
ncFTP	67	20,043	377	144	180
xFig	160	65,451	1,487	271	1,537

3.1 정보 모형

정보 모형은 C 언어로 작성된 프로그램을 분석하여 RESORT/R 구성 요소인 매개 언어(Internal Program Representation: IPR)로 정보를 구축하고 [8, 9], 필요한 역공학 정보를 추출하여 역공학 통합 저장 구조에 저장한 후, Variable Reference Graph 의 생성에 필요한

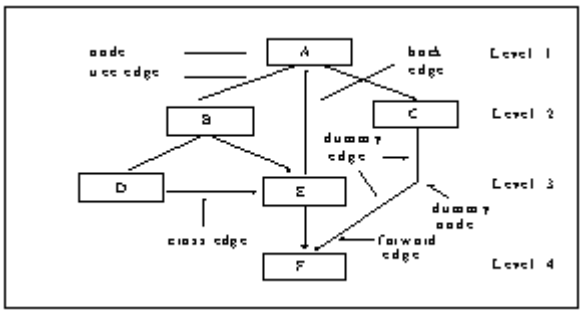
입력 그래프의 노드와 간선 정보들을 아래와 같은 인접 리스트(adjacency list) 구조의 Class VRGModel 에 저장한다.

```
Public Vector VRGModel {
        private Vector functionList;
        private Vector functionEdgeList;
        private Vector globalVariableList;
        private Vector relationshipList;
}
```

3.2 레벨 알고리즘

레벨의 주요 알고리즘은 아래와 같이 구성되고, Sugiyama 알고리즘을 확장한 것이다. 이 단계는 모든 노드들을 계층적으로 배치하는 것이다.

```
Public void Leveling_VRG() {
    List_DFS();
}
```



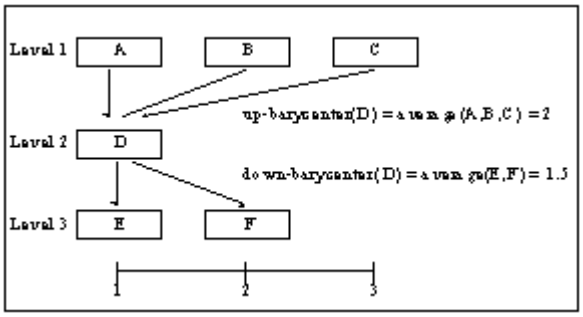
<그림 2> 레이아웃 알고리즘의 용어

레이아웃의 첫 단계인 레벨 알고리즘은 깊이 우선 탐색(Depth-First Search, DFS)을 확장하여 노드들의 레벨 및 간선의 종류를 식별한다.

확장된 DFS 알고리즘은 <그림 2>의 예제와 같이 노드들의 레벨을 정하고, 간선들의 종류를 식별하고, 그리고, 모든 연결 리스트를 방문하여 하나 또는 다수의 레이아웃이 생성될 수 있도록 설계되었다.

인접 리스트의 저장 순서에 따라, 확장된 DFS 알고리즘은 시작 노드를 방문하여 레벨을 부여하고, 인접 노드를 방문할 때, 2 가지 처리 과정이 존재한다. (1) 인접 노드가 있다면, 레벨이 되어 있는지를 식별한다. 레벨이 없다면, 그 노드들 간의 간선은 트리 간선(tree edge)으로 분류하고, 인접 노드의 레벨은 "시작 노드 레벨 + 1"이 된다. (2)그와 반대이면, 그 노드들 간의 간선은 비트리 간선(non-tree edge)으로 분류하고, 인접 노드의 레벨은 수정되지 않는다. 이 처리 과정은 연결된 노드들이 모두 방문 될 때까지 반복 수행한다.

확장된 DFS 알고리즘은 간선을 2 가지 유형으로 분류하여 사이클을 처리하는데 유용하다. (1) 트리 간선은 노드들 간의 레벨 순서를 결정한다. (2) 주어진 레벨 순서에 따라 비트리 간선은 3 가지 유형으로 분류된다. 역 방향 간선(back edge)은 한 노드를 한 조상에 연결한다. 앞 방향 간선(forward edge)은 한 노드를 2 레벨 이상의 후손에 연결한다. 크로스 간선(cross edge)은 한 노드를 그의 조상도 후손도 아닌 노드에 연결한다. 그러한 크로스 간선은 트리 구조가 아니기 때문에, 이 간선은 트리 및 앞 방향 간선으로 수정하여 제거되어야 한다. 예를 들어, <그림 3>에서 크로스 간선인 edge(D,E)는 트리 간선인 edge(D,E)로 수정되고, 동시에 트리 간선인 edge (B,E)는 앞 방향 간선인 edge(B,E)로 수정되어, 결국 크로스 간선 edge(D, E)는 제거된다.



<그림 3> 위 및 아래 무게 중심의 예

일반적으로, 비사이클 및 트리를 방향 그래프로 레이아웃하는 많은 알고리즘들이 있지만, 사이클을 방향 그래프로 레이아웃 하기 위해 휴리스틱(heuristic) 접근 방법들이 제시되고 있다.

Sugiyama 알고리즘은 사이클에 포함된 모든 노드들을 하나의 노드인 프락시(proxy)로 처리하기 때문에 사이클을 레이아웃하기가 불가능하다 [4]. 또한, Sugiyama 알고리즘의 문제점을 해결하기 위해, 간단한 휴리스틱으로 Sugiyama 알고리즘을 확장한 Rowe et al. 알고리즘은 이행적 폐쇄(transitive closure)를 수행하는 동안 사이클을 찾아, 레이아웃 알고리즘을 수행 이전에 사이클을 형성하고 있는 간선을 임시적으로 수정함으로써, 그 사이클을 제거한다 [2]. 그러나, 그래프를 화면에 출력할 때, 수정된 간선은 원래의 방향(역 방향)으로 그려진다 [2].

따라서, Sugiyama 알고리즘은 사이클을 프락시로, Rowe et al. 알고리즘은 사이클을 내부적으로 비사이클로 처리하여 레이아웃을 생성한다. 그러나, 이들 알고리즘에서 제시된 사이클 처리 방법은 유용하게 사용될 수는 있지만, 근본적으로 사이클을 사이클로 수용하는 레이아웃 알고리즘을 제시하지 못했다.

본 알고리즘은 3 종류의 비트리 간선 개념을 도입하여 사이클을 제거하지 않고 레이아웃할 수 있도록 설계 및 구현되었다. 비트리 간선 중에서 사이클을 형성하는 간선은 역 방향 간선뿐이다.

이 레벨 알고리즘에 의해 수행된 레이아웃 결과는 노드들을 각 레벨에 배치되었지만, 각 레벨에 있는 노드들은 아직 간선의 교차점을 최소화 하기 위한 노드의 순서화가 이루지지 않은 상태이다.

3.3 순서 알고리즘

순서의 주요 알고리즘은 아래와 같이 3 단계로 구성되고, Sugiyama 알고리즘을 기반으로 이루어졌다. 이 단계는 판독성 및 추적성을 제공하기 위한 휴리스틱한 과정으로 간선들 간의 교차점을 줄여, 각 레벨상에서 모든 노드들의 순서를 결정하는 것이다.

```
Public void Ordering_VRG() {
        Estimate_Up_Barycenter();
        Estimate_Down_Barycenter();
        Estimate_Up_Barycenter();
}
```

레벨화된 각 노드들의 수평적 위치를 측정하는 무게 중심(barycenter)은 각 레벨상에서 모조 노드를 포함한 노드들의 순서를 결정하는데 사용된다. 본 알고리즘에서는 2 가지 유형의 무게 중심이 활용된다.

첫째, 위-무게 중심(up-barycenter)은 [4] 그래프의 탑(top) 레벨에서 시작하여 아래 방향으로 측정하며, 노드가 재배치되어지는 레벨은 2,…,n 레벨이다. <그림 3>의 예제와 같이, Level 1 에 있는 A, B, C 의 노드들을 서수의 오름 차순으로 가중치를 부여한 후, 인접한 모든 조상들의 평균 위치로 Level 2 에 있는 D 노드의 위-무게 중심을 측정한다. 각 레벨에서 위-무게 중심이 수행된 후, 모든 노드들은 측정된 가중치에 의해 오름 차순으로 정렬 한다. 정렬 한다는 것은 상위 레벨에서 교차하고 있는 간선의 수를 최소화 하는 것이다. 이 과정은 각 레벨에서 반복 수행한다. 그리고, 아래-무게 중심을 수행 후, 다시 한번 더 수행된다.

둘째, 아래-무게 중심(down-barycenter)은 [4] 위-무게 중심과 동일한 방법이지만, 단지 그래프의 끝(bottom) 레벨에서 시작하여 위 방향으로 측정한다. 노드가 재배치되어지는 레벨은 n-1,…,1 레벨이다. 이 과정은 각 레벨에서 반복 수행한다.

이 순서 알고리즘에 의해 수행된 레이아웃 결과는 각 레벨에 있는 노드들을 위 및 아래 무게 중심에 의해 간선의 교차점의 수는 최소로 줄일 수 있었고, 그 결과 각 레벨에 있는 노드들의 위치가 결정되었다. 그러나, 각 레벨에 있는 노드들은 아직 실제 X, Y 좌표의 값이 계산되지 않은 상태이다.

3.4 위치 알고리즘

위치의 주요 알고리즘은 아래와 같이 X 좌표 생성을 포함한 3 단계로 구성되고, Sugiyama 알고리즘을 기반으로 이루어졌으며, 그래프의 미적인 휴리스틱을 추가 하였다. 이 단계는 레벨들 간의 간격 및 노드들 간의 간격의 옵셑(offset)을 위한 휴리스틱한 과정으로 수평적 계층 구조를 생성하기 위한 실제 노드와 간선의 X, Y 의 좌표를 생성한다.

```
public void Position_VRG(){
{
        Position_X();
        setFunctionXYCoordinates();
        setFunctionCallRelationshipXYCoordinates();
}

Public void Position_X() {
            Calculate_Upper_Barycenter();
            Calculate_Lower_Barycenter();
}
```

레이아웃의 마지막 단계인 위치 알고리즘은 이미 레벨 알고리즘의 결과로 얻어진 Y 좌표 값과 정렬된 레벨을 가지고 각 노드의 X 좌표의 위치를 계산한 후, Variable Reference Graph 캔버스(canvas)에 정의된 레벨 및 노드들의 간격에 의해 수평적 계층 구조의 실제 노드 및 간선 X, Y 좌표 값을 생성한다.

첫째, 본 알고리즘의 무게 중심은 순서 알고리즘에서 사용된 무게 중심 방법들과 동일한 방법으로 결정된다 [4]. 본 알고리즘에서는 2 가지 유형의 무게 중심이 활용된다.

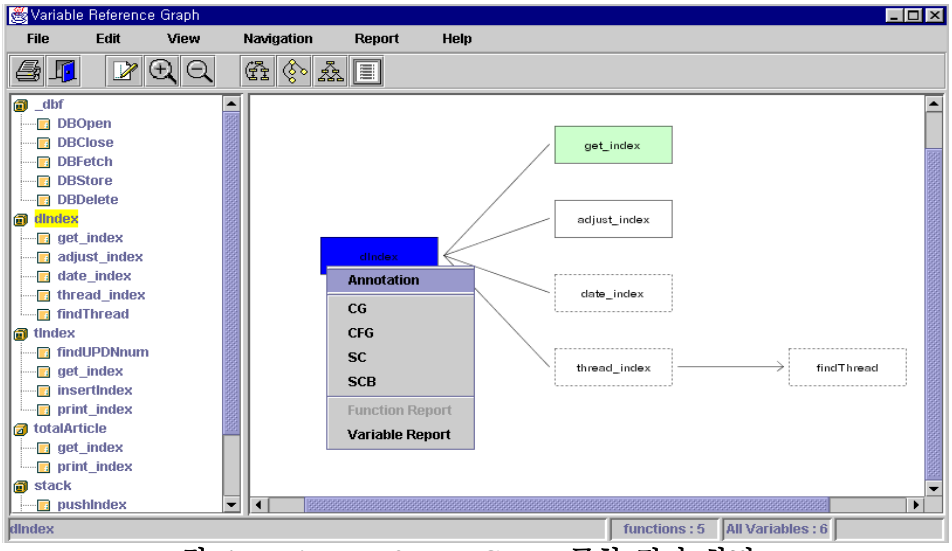
(1) 위쪽-무게 중심(upper-barycenter)은 [4] 각 레벨의 노드들을 무게 중심의 가중치를 측정하여 재배치한다. 그러나, 순서 알고리즘에 의해 결정된 노드들의 순서는 변하지 않는다. 이 과정은 각 레벨에서 반복 수행한다.

(2) 아래쪽-무게 중심(lower-barycenter)은 [4] 위쪽-무게 중심과 동일한 방법이지만, 단지 진행 방향만 다르다. 이 과정은 각 레벨에서 반복 수행한다.

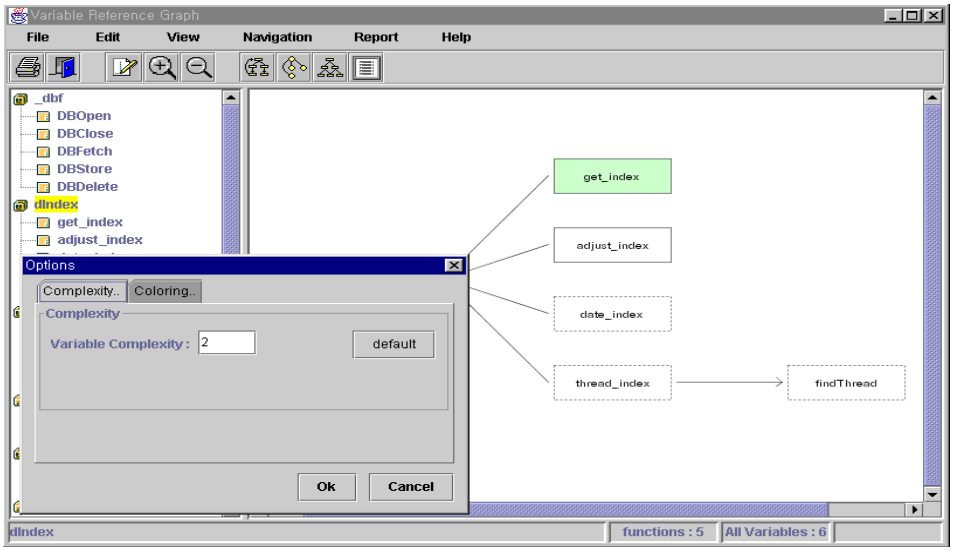
둘째, Variable Reference Graph 캔버스(canvas)에 출력하기 위해서, 각 노드들과 간선들에 대한 수평적 계층 구조의 X, Y 좌표 위치를 결정한다. Variable Reference Graph 에서 레벨들 간의 간격과 노드들 간의 간격은 고정된 값으로 설계 되었다.

(1) 그래프의 탑 레벨에서 시작하여 끝 레벨까지 아래 방향으로 진행하면서 각 노드들의 수평적 계층 위치를 구한다. 노드들 간의 간격에 따라, 노드의 X 좌표와 폭의 수평적 계층 위치가 결정되고, 레벨들 간의 간격에 의해 노드의 Y 좌표와 높이의 수평적 계층 위치가 결정된다. 이 과정은 각 레벨에서 반복 수행한다.

(2) 그래프의 탑 레벨에서 시작하여 끝 레벨까지 아래 방향으로 진행하면서 각 간선들의 수평적 계층 위치를 구한다. 먼저, 시작 X 좌표는 노드의 X 좌표와 폭의 위치에 의해 수평적 계층 위치가 결정되고, 그리고 시작 Y 좌표는 노드의 Y 좌표와 높이의 위치



<그림 4> Variable Reference Graph 구현 결과 화면



<그림 5> Variable Reference Graph 참조 복잡도 화면

에 의해서 수평적 계층 위치가 결정된다. 마지막으로, 끝 X 좌표는 노드의 X 좌표와 폭의 위치에 의해 수평적 계층 위치가 결정되고, 그리고 끝 Y 좌표는 노드의 Y 좌표와 높이의 위치에 의해서 수평적 계층 위치가 결정된다. 이 과정은 각 레벨에서 반복 수행한다.

<그림 4>는 예제 C 소스 코드인 "CrazyWebBoard"를 수평적 방향 그래프 레이아웃 알고리즘 결과로 구현된 Variable Reference Graph 이다. <그림 5>는 참조 복잡도의 화면으로써 사용자가 그 복잡도의 기준치를 새로이 정의할 수 있는 화면이다. 본 시스템에서는 "default" 값으로 각 함수의 전역 변수(구조체) 참조도가 2 이상이면, 다양한 외부 자료를 참조하는 복잡한 함수로 판정하여 칼라로 표현된다.

4. 결론

Variable Reference Graph 는 C 소스 코드로부터 정적인 자료 흐름 분석 정보를 수평적 방향 그래프로 자동 레이아웃하기 위해 개발된 역공학 도구이다. 이 도구는 함수들간의 자료 흐름 분석 정보를 수평적 트리 및 그래프 구조로 사용자에게 제공할 뿐만 아니라, 각 함수에 대한 참조 복잡도 등의 측정 정보를 제공함으로써 유지 보수 되어야 할 프로그램을 이해하고 분석하는데 유용하게 사용될 수 있다.

본 논문에서는 Sugiyama 알고리즘을 확장하여 JAVA 로 구현된 그래프 레이아웃 알고리즘을 기술하였다. 이 알고리즘은 4 단계로 구성되어 있다: 정보 모형, 레벨 알고리즘, 순서 알고리즘, 위치 알고리즘. 순서와 위치 알고리즘은 휴리스틱 처리 과정이다.

특히, 레벨 알고리즘에서는 DFS 를 확장하여 역 방향 간선인 사이클을 제거 하지 않고 레이아웃할 수 있도록 설계되어 있다.

마지막으로, 좀 더 연구되어야 할 부분은 다음과 같다.

- 판독성 및 추적성을 강화하기 위한 레이아웃
- 에디터 브라우저(browser) – 재사용, 정보의 복구
- 다양한 레이아웃 – circular graph, symmetric graph
- 객체지향 언어인 JAVA, C++ 등에 대한 Variable Reference Graph

참고문헌

[1] D. J. Robon, K. H. Bennett, B. J. Cornelius, and M. Munro, Approaches to Program Comprehension, *Journal of Systems and Software,* Vol. 14, Feb., 1991, pp. 79-84.

[2] L. A. Rowe, M. Davis, E. Messinger, and C. Meyer, A Browser for Directed Graphs, *Software-Practice and Experience,* Vol. 17(1), Jan., 1987, pp.61-76

[3] A. von Mayrhauser and A.M. Vans, Program Comprehension During Software Maintenance an Evolution, *IEEE Computer,* August 1995, Vol.25, No.8, pp44-55

[4] K. Sugiyama, S. Tagawa, and M. Toda, Methods for Visual Understanding of Hierarchical System Structures, *IEEE Trans. On Systems, Man, and Cybernetics,* Vol.SMC-11, No. 2, Feb., 1981, pp. 109-125

[5] E. R. Gansner, S. C. North, and K. P. Vo, Dag – a program that draws directed graphs, *Software-Practice and Experience,* Vol. 18(11), Nov., 1988, pp.1047-1062

[6] J. N. Warfield, Crossing Theory and Hierarchy Mapping, *IEEE Trans. On Systems, Man, and Cybernetics,* Vol.SMC-7, No. 7, July, 1977, pp. 505-523

[7] 이헌기, 신규상, 영공학에서 재문서화를 지원하는 도구, *정보처리학회 추계 학술발표논문집,* 1998, 제 5 권 제 2 호, pp. 433-436

[8] 이헌기, 장진호, 역공학 소프트웨어 지원도구 – RESORT/R, *한국정보과학회, 소프트웨어공학회지,* 1999, 3, 제 12 권 제 1 호, pp. 22-32

[9] 한국전자통신연구원, *객체지향 소프트웨어 재공학 기술 개발,* 정보통신부, 1999