http://dx.doi.org/10.3745/KTCCS.2013.2.8.317

An Effective Method for Comparing Control Flow Graphs through Edge Extension

Hyun-il Lim†

ABSTRACT

In this paper, we present an effective method for comparing control flow graphs which represent static structures of binary programs. To compare control flow graphs, we measure similarities by comparing instructions and syntactic information contained in basic blocks. In addition, we also consider similarities of edges, which represent control flows between basic blocks, by edge extension. Based on the comparison results of basic blocks and edges, we match most similar basic blocks in two control flow graphs, and then calculate the similarity between control flow graphs. We evaluate the proposed edge extension method in real world Java programs with respect to structural similarities of their control flow graphs. To compare the performance of the proposed method, we also performed experiments with a previous structural comparison for control flow graphs. From the experimental results, the proposed method is evaluated to have enough distinction ability between control flow graphs which have different structural characteristics. Although the method takes more time than previous method, it is evaluated to be more resilient than previous method in comparing control flow graphs which have similar structural characteristics. Control flow graph can be effectively used in program analysis and understanding, and the proposed method is expected to be applied to various areas, such as code optimization, detection of similar code, and detection of code plagiarism.

Keywords : Control Flow Graph, Graph Structure Comparison, Program Analysis

에지 확장을 통한 제어 흐름 그래프의 효과적인 비교 방법

임 현 일†

요 약

본 논문에서는 바이너리 프로그램의 정적인 구조를 표현하는 제어 흐름 그래프를 비교하는 방법을 제안한다. 제어 흐름 그래프를 비교하기 위해서 기본 블록에 포함된 프로그램의 명령어 및 구문 정보를 비교한 후 기본 블록 사이의 유사한 정도를 측정한다. 또한, 에지 확장을 통해 기본 블록들 간의 제어 흐름을 표현하는 그래프 에지의 유사성을 함께 반영한다. 각 기본 블록 사이의 유사도 결과를 기반으로 기본 블록을 서로 매칭하고, 기본 블록 사이의 매칭 정보를 이용해서 전체 제어 흐름 그래프의 유사도를 측정한다. 본 논문에서 제안한 방법은 자바 프로그램으로부터 추출한 제어 흐름 그래프를 대상으로 제어 흐름 구조의 유사성에 따라 두 가지 기준으로 실험을 수행하였다. 그리고, 성능을 평가하기 위해서 기존의 구조적 비교 방법을 함께 실험하였다. 실험 결과로부터 에지 확장 방법은 서로 다른 프로그램에 대해 충분한 변별력을 가지고 있음을 확인할 수 있다. 프로그램 비교에 좀 더 많은 시간이 소요되지만, 구조가 유사한 프로그램에 대한 매칭 능력에서 기존의 구조적 비교 방법에 비해 우수한 결과를 보였다. 제어 흐름 그래프는 프로그램의 분석에 다양하게 활용될 수 있으며, 제어 흐름 그래프의 비교 방법은 프로그램의 유사성 비교를 통한 코드의 최적화, 유사 코드 검출, 코드의 도용 탐지 등 다양한 분야에서 응용될 수 있을 것이라 기대된다.

키워드 : 제어 흐름 그래프, 그래프 구조 비교, 프로그램 분석

1. 서 론

바이너리 프로그램의 제어 흐름 그래프는 프로그램의 정적인 구조를 효과적으로 기술할 수 있는 자료 구조이다. 그래서, 프로그램의 특성을 이해하고 분석 정보를 활용하기 위해서 정적 분석, 동적 분석 등에서 광범위하게 사용되고 있다. 본 논문에서는 프로그램의 유사성을 판단하기 위해서 제어 흐름 그래프의 에지 확장을 통해 효과적으로 구조를

※ 이 연구결과물은 2011학년도 경남대학교 신진교수 연구비 지원에 의한 것임.
† 정 회 원 : 경남대학교 컴퓨터공학과 조교수
논문접수 : 2012년 12월 18일
수 정 일 : 1차 2013년 4월 12일
심사완료 : 2013년 7월 18일
* Corresponding Author : Hyun-il Lim(hilim@kyungnam.ac.kr)

비교하는 방법을 제안한다. 제어 흐름 그래프의 비교를 위해서 그래프를 구성하는 기본 블록(basic block)들의 특성을 비교하는 방법을 사용하고 있다. 기본 블록에 포함된 프로그램의 명령어나 구문 정보를 비교함으로써 기본 블록 사이의 유사성을 반영할 수 있다. 또한 기본 블록들 간의 제어 흐름을 표현하는 그래프 에지의 유사성을 검출하기 위해서 각 기본 블록은 에지 확장을 통해서 구조적 유사성을 반영하고 있다. 전체 제어 흐름 그래프의 유사도는 각 기본 블록의 유사성 여부에 따라 매칭을 수행하고, 매칭 정보를 전체 프로그램으로 확장함으로써 얻는다.

제어 흐름 그래프는 프로그램의 정적인 구조를 표현하는 가장 중요한 자료 구조 중의 하나이다. 제어 흐름 그래프의 비교는 프로그램의 유사성을 유추할 수 있는 가장 근본적인 방법으로 활용될 수 있으며, 그래프 비교를 통한 코드의 최적화, 유사 코드 검출, 코드의 도용 탐지 등에서 활용되고 있다[1, 2]. 최근에는 다양한 운영 환경에서 급격하게 증가하고 있는 악성 코드의 검출을 위해서 제어 흐름 그래프를 활용하는 방법이 사용되고 있다[3, 4, 5].

본 논문에서 제안하는 에지 확장 방법은 자바로 구현한 프로그램을 대상으로 두 가지 평가 기준에 따라 실험을 수행한다. 성능을 평가하기 위해서 기존의 구조적 비교 방법[12, 13]을 함께 실험하고 성능을 비교한다. 실험 결과를 통해서 본 논문에서 제안한 방법이 유사한 구조를 가진 수정된 프로그램의 비교에서 기존의 방법에 비해 효과적인 결과를 보인다는 것을 확인할 수 있다.

본 논문의 구성은 다음과 같다. 2절에서는 그래프 비교에 대한 기존의 관련 연구에 대해서 요약한다. 3절에서는 본 논문에서 제안하는 제어 흐름 그래프의 비교 방법에 대해서 기술한다. 4절에서는 실험을 통해서 제안한 방법이 제어 흐름 그래프의 비교에 효과적인 결과를 얻을 수 있음을 보여주고, 5절에서는 제안한 방법의 개선 방향 및 향후 연구 계획에 대해서 기술한다. 마지막으로 6절에서 결론을 맺는다.

2. 관련 연구

그래프는 자료의 저장 및 구조를 표현할 수 있는 기초적인 자료 구조로서 유용하게 사용된다. 다양한 형태의 자료들을 표현하기 위해서 그래프 형태가 효과적으로 사용되고 있으며, 이를 비교하는 방법에 대한 연구가 다양하게 진행되고 있다.

일반적인 그래프의 비교 방법은 크게 두 가지 방법으로 분류할 수 있다. 첫 번째는 비교 대상이 되는 그래프의 노드들이 레이블(label)을 가지고 있지 않는 경우이다. 이 경우 각 노드는 어떤 값을 가지고 있지 않는 경우이며, 노드간의 비교는 연결된 에지의 동일성 여부로 판단하게 된다. 두 번째는 그래프의 노드가 레이블을 가지고 있는 경우이다. 노드의 레이블은 노드 간의 동일성 여부를 확인할 수 있는 중요한 정보이기 때문에 첫 번째 방법과는 다른 비교 방법이 필요하다.

일반적인 그래프의 비교 방법으로 두 그래프의 동일성을 비교하기 위해서 최대 공통 부분 그래프(maximum common subgraph) [6]를 이용하는 방법이 있는데, 최대 공통 부분 그래프를 찾는 문제는 NP-Hard 문제로 그 계산의 복잡도가 비효율적이기 때문에 실제 환경에 직접 활용하기 어렵다. 이를 보완하기 위해서 최대 공통 부분 그래프를 활용한 근사 비교 방법 [7, 8]에 관한 연구가 수행되고 있다. 이외에도, 그래프의 편집 거리 (edit distance)에 기반한 방법 [9], 통계학적인 유사도를 측정하는 방법 [10] 등 다양한 비교 방법들이 연구되고 있다.

바이너리 프로그램의 구조를 표현하는 제어 흐름 그래프를 비교하기 위해서는 일반적인 그래프 비교 알고리즘과 다른 접근 방법을 이용한다. 제어 흐름 그래프를 구성하는 노드를 구성하는 명령어 정보 및 에지의 유사성 등을 함께 고려할 필요가 있다. Falke [12]와 Dullien 등 [13]은 부분적인 업데이트가 적용된 바이너리 프로그램의 차이를 비교하는 구조적 비교 방법을 제안하였다. 이 방법은 제어 흐름 그래프의 노드와 에지가 가지는 특성 정보를 비교하고 반복적으로 매칭함으로써 원본 프로그램과 수정된 프로그램의 차이를 분석하는 방법을 제안한다. 이 방법은 부분적으로 수정된 바이너리 프로그램의 비교에서 효과적인 매칭 결과를 보여준다.

3. 에지 확장을 통한 제어 흐름 그래프 비교

3.1 제어 흐름 그래프의 비교 방법

프로그램의 제어 흐름 그래프(control flow graph) [1]는 프로그램의 정적인 형태를 보여주는 자료 구조로서 제어 흐름 그래프 $G=(N,E)$ 의 쌍으로 정의된다. 여기서, N은 제어 흐름 그래프가 가지는 노드의 집합으로 각 노드는 하나의 기본 블록을 표현한다. 이 기본 블록에는 프로그램의 수행 중에 실행되는 명령어나 구문 정보 및 제어 흐름을 변경하는 정보 등을 포함하고 있다. E는 에지(edge)의 집합으로 제어 흐름 그래프에 포함되는 기본 블록 간의 제어 흐름 관계를 나타낸다.

프로그램의 제어 흐름 그래프는 노드와 에지의 집합으로 표현되는 기본 블록과 제어 흐름으로 표현할 수 있고, 그래프의 특수한 형태라고 볼 수 있다. 따라서, 제어 흐름 그래프를 비교하기 위해서 기존의 그래프 비교 알고리즘을 적용할 수 있지만 제어 흐름 그래프에 직접 적용하기에는 한계가 있다.

제어 흐름 그래프의 각 노드는 기본 블록에 해당하는데, 여기에는 유사성을 판단할 수 있는 명령어 및 구문 정보들을 포함하고 있다. 그리고, 이런 정보들은 노드들이 동일성 여부 뿐만 아니라 부분적인 유사성을 유추할 수 있는 정보를 가진다. 반면, 기존의 그래프 알고리즘 [6, 7, 8, 9, 10]에서는 노드에 레이블이 있는 경우에 노드의 동일성 여부를 판단할 수 있지만 부분적인 유사성을 고려하지 않는다.

Falke [12]와 Dullien 등 [13]에서는 제어 흐름 그래프의

차이점을 분석하기 위해서 기본 블록과 에지가 가지는 다양한 특성 시그너처 정보(indegree, outdegree, 함수 호출 정보, 시작 지점에서의 거리 등)을 비교하고 기본 블록이 더 이상 매칭 되지 않는 고정점(fixpoint)를 찾을 때까지 반복적으로 매칭한다. 이 방법은 전체 프로그램의 상당 부분이 원본을 유지하고, 부분적으로 업데이트 패치된 바이너리 프로그램에서 차이점을 찾아내는 방법으로 효과적으로 사용될 수 있지만, 전체적인 프로그램의 수정이 있거나 구조적으로 기본 블록의 시그너처 정보가 변경된 프로그램에 대해서 정확한 매칭을 찾기 어렵다. 따라서, 기본 블록의 매칭을 위해서 특성 시그너처 정보가 동일하게 유지되지 않으면 매칭이 어렵고, 일반적인 제어 흐름 그래프를 비교하는 환경에 활용하기에는 적합하지 않다.

본 논문에서는 바이너리 프로그램의 제어 흐름 그래프사이에 유사도를 검출하기 위해서 우선 기본 블록간의 유사한 정도를 측정한다. 기본 블록 간의 유사성은 에지 확장을 통해 블록 간의 변별력을 크게 가질 수 있다. 전체 제어 흐름 그래프의 유사도는 기본 블록 간의 유사도를 기반으로 최적의 매칭을 찾음으로써 얻을 수 있다. 기본 블록의 매칭은 서로 유사하다고 평가되는 블록들을 순서대로 매칭하게 된다. 본 논문에서 제안한 방법은 다양한 형태 또는 임의로 수정된 제어 흐름 그래프에 대해서 기존의 구조적 비교 방법 [12, 13]보다 광범위하게 적용할 수 있다.

3.2 기본 블록의 비교

기본 블록의 유사성을 비교하기 위해서 기본 블록에 포함되어 있는 명령어 및 구문 정보들을 비교한다. 기본 블록의 비교는 기본 블록이 가지는 구문 정보들을 원소로 하는 집합에 대해서 두 기본 블록에 대한 전체 집합에서 동일한 원소를 나타내는 교집합의 크기가 얼마나 큰지를 평가하는 방법을 적용한다.

두 프로그램의 제어 흐름 그래프 A와 B에 대해서 각각 i 번째와 j 번째 기본 블록을 각각 A_i, B_j라고 할 때, 두 기본 블록의 비교 결과 $m(A_i,B_j)$는 아래의 수식과 같이 구할 수 있다.

$$m(A_i,B_j)=\frac{|Node(A_i)\cap Node(B_j)|}{|Node(A_i)\cup Node(B_j)|}$$

여기서, $Node(A_i)$는 제어 흐름 그래프의 기본 블록 A_i에 포함된 명령어 및 구문의 집합을 의미한다. 그리고 집합 A에 대해서 $|A|$는 집합의 원소의 개수를 의미한다.

기본 블록의 비교에서 제어 흐름 그래프에 포함된 기본 블록들에 대해서 각각 동일한 명령어가 얼마나 포함되어 있는지를 평가하고 유사한 정도를 측정할 수 있으며, 이 측정값은 제어 흐름 그래프의 유사도를 분석하는 시작점이 된다.

그래프는 노드와 에지로 구성되기 때문에 그래프의 유사도를 측정하기 위해서 노드 뿐만 아니라 에지의 유사성 역시 고려하여야 한다. 제어 흐름 그래프를 비교할 때, 기본 블록의 제어 흐름을 나타내는 에지의 유사성을 함께 고려한다면 보다 신뢰성 있는 비교 결과를 얻을 수 있을 것이다. 이를 위해서 제어 흐름 그래프의 기본 블록이 가지는 에지의 비교 결과를 확장함으로써 효과적으로 비교할 수 있다.

3.3 기본 블록의 에지 확장

각 기본 블록은 방향성을 가지는 에지를 가지고 있는데 이 에지는 기본 블록 사이의 제어 흐름을 표현한다. 에지 확장에서는 기본 블록의 비교를 위해서 나가는 제어 흐름을 표현하는 에지(outgoing edge)와 들어오는 제어 흐름을 표현하는 에지(incoming edge)를 확장하는 것을 의미한다.

나가는 에지에 대한 유사성은 이 에지를 통해서 도달하는 기본 블록들이 여러 개 있을 때, 이 기본 블록들에 포함된 명령어 및 구문 정보들의 유사성을 비교한다. 이 비교 결과는 단순히 하나의 에지만을 비교한 결과라기보다는 기본 블록을 중심으로 한 에지들의 유사성을 비교함으로써 기본 블록이 제어 흐름 그래프 상에서 독립된 원소가 아니고 전체 구조에서의 위치를 확인하고 유사한 블록을 신뢰성 있게 매칭하는 데 중요한 정보가 된다. 들어오는 에지에 대한 유사성 또한 동일한 방법으로 얻을 수 있다.

두 제어 흐름 그래프 A와 B에 대해서 각각 i 번째와 j 번째 기본 블록을 각각 A_i, B_j라고 할 때, 기본 블록에서 나가는 에지에 대한 유사성 $m1(A_i,B_j)$와 들어오는 에지에 대한 유사성 $m2(A_i,B_j)$는 아래의 수식을 통해서 구할 수 있다.

$$m1(A_i,B_j)=\frac{|NxtNode(A_i)\cap NxtNode(B_j)|}{|NxtNode(A_i)\cup NxtNode(B_j)|}$$

$$m2(A_i,B_j)=\frac{|BefNode(A_i)\cap BefNode(B_j)|}{|BefNode(A_i)\cup BefNode(B_j)|}$$

여기서, $NxtNode(A_i)$는 제어 흐름 그래프 A의 i 번째 노드에서 나가는 에지에 의해 연결된 기본 블록에 포함된 명령어 및 구문의 집합을 의미하고, $BefNode(A_i)$는 제어 흐름 그래프 A의 i 번째 노드로 들어오는 에지의 시작하는 기본 블록들에 포함된 명령어 및 구문의 집합을 의미한다.

이 수식을 통해 비교한 결과는 하나의 기본 블록에 대해서 그 블록의 실행 전후에 실행되는 기본 블록들에 동일한 구문이 얼마나 많이 포함되어 있는가를 측정한다. 기본 블록의 유사한 정도를 측정하기 위해서 그 연속적으로 실행되는 블록을 비교한 결과를 함께 고려할 수 있도록 에지를 확장한 것이다. 이는 제어 흐름 그래프 상에서 기본 블록 간의 선후 관계를 반영할 수 있고, 제어 흐름 그래프의 비교에 프로그램의 실행 절차를 반영할 수 있다. 따라서, 프로그램이 수정되더라도 절차가 유사하다면 유사성을 확인하는데 중요한 역할을 한다.

에지 확장을 통한 기본 블록 간의 유사성은 세 가지 비교 척도 $m, m1, m2$를 함께 고려하여 구할 수 있다. 두 개의

제어 흐름 그래프 A, B에서 각각 i 번째와 j 번째 기본 블록의 유사성 $S(A_i,B_j)$는 아래의 수식과 같이 구할 수 있다.

$$S(A_i,B_j)=\frac{\alpha\times m(A_i,B_j)+\beta\times m1(A_i,B_j)+\gamma\times m2(A_i,B_j)}{\alpha+\beta+\gamma}$$

이 수식은 기본 블록 자체의 유사성과 기본 블록과 인접한 블록들의 유사성을 적절히 조율함으로써 전체 제어 흐름 그래프 상에서 기본 블록 사이의 유사도를 표현한다. 여기서, α,β,γ는 기본 블록과 에지 확장을 통한 에지의 유사도에 대한 비중을 결정하는 계수이다. 이 값에 의해 기본 블록과 에지에 반영하는 가중치를 조정할 수 있다. 본 논문에서는 기본 블록의 비교와 에지 확장의 비율을 동일하게 평가하고 $\alpha=2,\beta=\gamma=1$로 설정하였다.

기본 블록의 유사성은 두 기본 블록이 가지는 구문 정보와 들어오는 제어 흐름 및 나가는 제어 흐름 모두 동일한 형태를 가질 때 1.0의 유사도를 가지게 되고, 공유하는 구문 정보가 전혀 없고 일치 하지 않을 때 0.0의 유사도를 가지게 된다. 따라서, $S(A_i,B_j)$의 값은 두 기본 블록 간의 유사한 정도에 따라서 0.0에서 1.0 사이의 값을 가진다.

기본 블록 사이의 유사성은 두 개의 제어 흐름 그래프를 구성하는 모든 기본 블록들 사이에 측정되어야 한다. 즉, 제어 흐름 그래프 A와 B가 각각 m개와 n개의 기본 블록을 포함하고 있을 때 기본 블록 간에 총 $m\times n$ 쌍에 대한 유사성 측정값을 얻을 수 있다. 이 측정값은 제어 흐름 그래프의 유사성을 결정하기 위해 기본 블록의 매칭을 위한 기준으로 활용된다.

3.4 기본 블록의 비교 예제

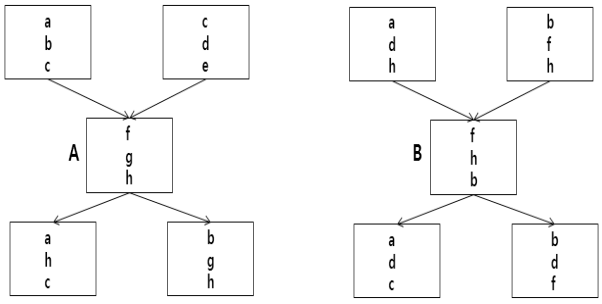

Fig. 1. An example of comparing basic blocks A and B

Fig. 1에서는 두 개의 제어 흐름 그래프에서 각각 기본 블록 A와 B를 보여주고 있으며, A와 B는 연결된 에지를 통해 제어 흐름으로 연결된 기본 블록간의 관계를 보여주고 있다. 각각의 기본 블록 내에 표기된 알파벳 소문자는 기본 블록을 구성하는 프로그램 명령어를 나타낸다고 가정한다. 본 예제에서 기본 블록 A에는 3개의 명령 구문 {f, g, h} 가 포함되어 있고, 기본 블록 B에는 명령 구문 {f, h, b} 가 포함되어 있다. 그래서 두 기본 블록 사이의 유사성 $m(A,B)$는 아래와 같이 구할 수 있다.

$$m(A,B)=\frac{|\{f,g,h\}\cap\{f,h,b\}|}{|\{f,g,h\}\cup\{f,h,b\}|}=\frac{|\{f,h\}|}{|\{f,g,h,b\}|}=0.5$$

제어 흐름 관계를 반영하기 위해서 $m1(A,B)$와 $m2(A,B)$는 아래와 같이 구할 수 있다.

$$m1(A,B)=\frac{|\{a,h,c,b,g\}\cap\{a,d,c,b,f\}|}{|\{a,h,c,b,g\}\cup\{a,d,c,b,f\}|}=\frac{|\{a,c,b\}|}{|\{a,h,c,b,g,d,f\}|}=\frac{3}{7}$$

$$m2(A,B)=\frac{|\{a,b,c,d,e\}\cap\{a,d,h,b,f\}|}{|\{a,b,c,d,e\}\cup\{a,d,h,b,f\}|}=\frac{|\{a,d\}|}{|\{a,b,c,d,e,h,f\}|}=\frac{2}{7}$$

그러므로, 기본 블록 A와 B 사이의 유사도 $S(A,B)$는 다음과 같다.

$$S(A,B)=\frac{2\times m(A,B)+m1(A,B)+m2(A,B)}{4}=\frac{2\times 0.5+\frac{3}{7}+\frac{2}{7}}{4}=0.429$$

따라서 Fig. 1에서 나타난 기본 블록 A와 B의 유사도는 0.429 의 값을 가진다.

3.5 제어 흐름 그래프의 유사도 검출

앞 절에서 제어 흐름 그래프를 구성하는 각 기본 블록들 사이에 유사도를 측정할 수 있는 방법을 제안하였다. 이 측정값들은 전체적으로 두 제어 흐름 그래프 사이에 유사한 기본 블록들의 분포 여부를 효과적으로 판단할 수 있는 근거를 제공한다. 두 제어 흐름 그래프의 기본 블록의 집합을 각각 N_1, N_2 라고 할 때, 두 그래프 사이의 유사성은 아래와 같이 구할 수 있다.

$$Results=\frac{\sum match(N_1,N_2)}{\min(|N_1|,|N_2|)}$$

여기서, $match(N_1,N_2)$는 기본 블록 집합 N_1과 N_2가 각각 $N_1=\{A_1,\cdots,A_m\}, N_2=\{B_1,\cdots,B_n\}$이라고 할 때, 기본 블록 사이의 유사도 $S(A_i,B_j)$ 값이 큰 순서대로 일대일 매칭된 쌍의 유사도 값의 집합을 의미한다. 따라서, $\sum match(N_1,N_2)$는 두 제어 흐름 그래프에서 높은 유사도 순으로 매칭된 기본 블록 쌍의 유사도 총합이 된다.

매칭 과정은 각 기본 블록 간의 유사도 $S(A_i,B_j)$에 대해 크기 순서대로 기본 블록을 일대일 매칭을 반복한다. 크기가 작은 제어 흐름 그래프 (기본 블록의 개수가 작은 그래프)의 기본 블록이 모두 매칭 되면 매칭은 종료되고, 매칭된 쌍의 유사도 합을 이용해서 제어 흐름 그래프 사이의 유사

도를 구할 수 있다. 제어 흐름 그래프의 유사도는 블록 사이의 유사도 총합을 크기가 작은 제어 흐름 그래프의 블록 수로 나눔으로써 정규화할 수 있으며, 전체 유사도의 정도에 따라 0.0 에서 1.0 사이에 값을 가지게 된다. 이 결과 값은 전체 제어 흐름 그래프 사이의 유사한 정도를 나타낸다.

3.6 제어 흐름 그래프의 유사도 검출 예제

두 제어 흐름 그래프의 기본 블록이 각각 $N_1 = \{A, B, C, D, E\}$와 $N_2 = \{W, X, Y, Z\}$ 라고 하면 각 기본 블록 사이의 유사도는 3.2절과 3.3절에서 기술한 과정을 통해 구할 수 있다. 이 유사도 결과가 Table 1과 같다고 할 때 두 제어 흐름 그래프 사이의 유사도는 다음과 같이 구할 수 있다.

Table 1. Matching basic blocks of two control flow graphs by considering similarities

	A	B	C	D	E
W	**0.8**	0.3	0.3	0.4	0.1
X	0.6	0.6	**0.7**	0.5	0.5
Y	0.4	**0.9**	0.5	0.8	0.3
Z	0.5	0.7	0.6	0.3	**0.9**

두 제어 흐름 그래프의 유사도를 검출하기 위해서 먼저 유사도가 높은 기본 블록을 순서대로 매칭한다. Table 1에서 기본 블록 B와 Y, 그리고, 기본 블록 E와 Z가 각각 유사도 0.9로 가장 높다. 그래서 이 두 블록을 순서대로 우선 매칭한다. 그 후 남은 노드들 중에서 가장 유사도가 높은 기본 블록의 쌍은 A와 W이고 유사도는 0.8이다. 마지막으로 기본 블록 C와 X가 매칭이 되고 유사도는 0.7이다. 매칭된 기본 블록의 유사도 집합 $match(N_1, N_2)$은 $\{S(B,Y), S(E,Z), S(A,W), S(C,X)\}$가 된다. 따라서, 제어 흐름 그래프의 비교 결과 전체 유사도는 아래와 같다.

$$Result = \frac{0.9 + 0.9 + 0.8 + 0.7}{\min(5,4)} = \frac{3.3}{4} = 0.825$$

따라서, 두 제어 흐름 그래프 사이의 유사도는 0.825가 되고, 기본 블록 매칭을 통해 제어 흐름 그래프의 유사도를 측정할 수 있다. 에지 확장 방법은 임의의 제어 흐름 그래프의 비교에 적용할 수 있으며, 특히 기본 블록 간의 유사도를 에지 확장을 통해서 보다 정확하게 판단할 수 있는 바이너리 프로그램에서 효과적으로 적용될 수 있다.

4. 실험 및 평가

4.1 실험 환경

본 절에서는 본 논문에서 제안한 방법의 성능을 평가하기 위한 실험 결과를 보여준다. 성능 평가 실험을 위해서 제어 흐름 그래프를 분석하고 비교하는 방법은 C언어와 파이썬[11]으로 구현하였다. 실험 대상 프로그램은 바이너리 프로그램으로서 자바로 구현된 프로그램의 클래스 파일에 포함된 바이트코드에 대한 제어 흐름 그래프를 대상으로 하였다.

에지 확장 방법의 성능을 평가하기 위해서 기존의 제어 흐름 그래프 구조적 비교 방법 [12, 13]을 함께 구현하고 실험 결과를 비교하였다. 기존의 구조적 비교 방법은 제어 흐름 그래프의 구조적 특성을 이용해서 기본 블록을 매칭하는 방법이다. 기본 블록의 매칭을 위해 기본 블록의 indegree, outdegree, 함수 호출 정보, 기본 블록의 재귀적 특성, 시작 지점에서 최단 거리, 종료 지점까지 최단 거리 등의 특성 등을 시그너처 정보로 비교하고 최적의 매칭 블록을 찾으며, 이런 과정은 고정점을 찾을 때까지 반복적으로 수행된다.

Table 2에서 본 실험에서 비교 대상으로 사용한 벤치마크 프로그램의 정보를 보여주고 있다. 대상 프로그램은 자바로 구현된 데이터베이스 프로그램인 Apache Derby, Mckoi SQL Database, SmallSQL Database를 이용하였다. 실험의 신뢰성을 높이기 위해서 대상 프로그램에서 추출한 제어 흐름 그래프 중에서 에지와 기본 블록의 개수가 모두 40개 이상인 197개의 제어 흐름 그래프를 대상으로 하였으며, 에지와 기본 블록의 평균값은 122개와 91개, 최대값은 각각 590개와 506개이다.

Table 2. The information of control flow graphs of benchmark programs

Programs	# of classfiles	Control flow graphs						
		#	# of edges			# of basic blocks		
			Avg.	min	max	Avg.	min	max
Apache Derby	1,436	140	120.0	40	449	79.5	40	352
Mckoi SQL DB	617	40	131.5	41	590	112.6	40	503
SmallSQL DB	164	17	127.6	41	412	135.3	40	506
Sum	2,217	197	122.3	40	590	91.0	40	506

본 논문에서 제안한 방법의 성능을 평가하기 위한 실험은 두 가지 관점에서 수행하였다. 비교를 통한 유사도는 대상의 유사성 정도에 따라 신뢰성 있는 결과를 보여주는 것이 중요하다. 즉, 서로 구조가 유사한 프로그램을 비교한 경우에는 높은 유사도 결과를 줄 수 있어야 하며, 유사하지 않은 프로그램을 비교한 경우에는 상대적으로 낮은 유사도를 보여야 한다. 이 두 가지 기준을 평가하기 위해서 구조가 서로 유사하지 않은 프로그램을 비교한 실험과 서로 유사한 프로그램을 비교한 실험을 각각 수행하였다.

4.2 구조가 서로 다른 제어 흐름 그래프의 비교

서로 다른 구조를 가진 제어 흐름 그래프를 비교하기 위해서 실험 대상이 되는 197개의 제어 흐름 그래프를 각각 서로 비교하였다. 본 논문에서 제안한 비교 방법과 기존의 구조적 비교 방법 [12, 13]을 이용해서 총 19,306 쌍에 대한 비교를 수행하였으면 결과를 분석하였다. 본 실험은 다른 제어 흐름 그래프 사이의 비교 결과에서 얼마나 변별력 있는 유사도 결과를 보여주는지 평가한다. 그래서 서로 다른 제어 흐름 그래프라면 충분히 낮은 유사도 분포를 보이는 것이 효과적이다.

Table 3. Experimental results of comparisons between CFGs of different programs

	Edge extension	Structural comparison
Avg. similarity	11.1%	0.8%
Min. similarity	0.0%	0.0%
Max similarity	100.0%	89.4%
# of comparisons	19,306	19,306
Time (sec)	6,155s	5,598s

Table 3은 본 실험의 결과를 요약해서 보여준다. 에지 확장 비교 방법은 총 비교 결과 나타나는 유사도가 최소 0.0%에서 100.0% 사이에 분포했으며, 평균 11.1%의 유사도를 보였다. 기존의 구조적 비교 방법은 비교 결과 유사도가 0.0%에서 최대 89.4%까지 범위에 분포하였으며 평균 유사도는 0.8% 였다. 본 실험 결과에서 나타나는 실험 결과를 볼 때 기존의 구조적 비교 방법이 에지 확장 방법 보다 낮은 유사도 분포를 보임으로써 서로 다른 프로그램에 대한 변별력이 우수한 것으로 평가된다.

에지 확장에서 높은 유사도를 보인 비교 대상에 대해서 원인을 찾기 위해서 프로그램을 직접 분석하였다. 비교 대상 프로그램의 제어 흐름 그래프에는 Apache Derby에 있는 SQLParserTokenManager.class에서 총 28개의 제어 흐름 그래프와 Mckoi SQL DB에 있는 SQLTokenManager.class에서 총 13개의 제어 흐름 그래프가 포함되어 있었다. 이 프로그램은 개발자가 직접 작성한 코드가 아니고, SQL 구문을 파싱하기 위해서 파서 생성기인 JavaCC를 이용해서 자동 생성된 SQL 파서 프로그램이었다. JavaCC의 코드 생성 패턴의 유사성 때문에 유사한 형태의 제어 흐름 그래프를 포함하는 코드로 생성되었고, 에지 확장을 이용한 방법에서 높은 유사도를 보였던 것이다. 기존의 구조적 방법에서 또한 80% 이상의 높은 기본 블록 매칭을 보이기도 하였다. 따라서 에지 확장을 이용한 방법에서 높은 유사도를 보인 비교 대상은 실제로 유사한 구조를 가진 제어 흐름 그래프로 분석되었고, 전체 평균 유사도 11.1%로써 서로 다른 프로그램을 구분하기에 충분히 낮은 유사도 값을 보이는 것으로 평가된다.

수행 시간은 각각의 비교 방법으로 전체 대상 프로그램을 비교하는데 걸린 시간을 측정하였다. 기존의 구조적 비교 방법은 19,306쌍의 제어 흐름 그래프를 비교하는데 5,598초가 걸렸으며, 에지 확장 방법의 경우 동일한 실험에서 6,155초가 걸렸다. 에지 확장 방법은 기본 블록의 매칭을 위해서 기본 블록에 포함된 명령어들을 모두 비교하기 때문에 기존의 방법에 비해서 시간이 많이 걸렸다. 구조적 비교 방법은 기본 블록의 매칭을 위해서 기본 블록의 특성 정보만으로 빠르게 비교하기 때문에 에지 확장 방법에 비해 속도 향상이 있었지만, 기본 블록 사이에 최적의 매칭에 해당하는 고정점(fixpoint)을 찾기 위해 반복적으로 매칭을 시도하기 때문에 많은 차이는 보이지 않았다.

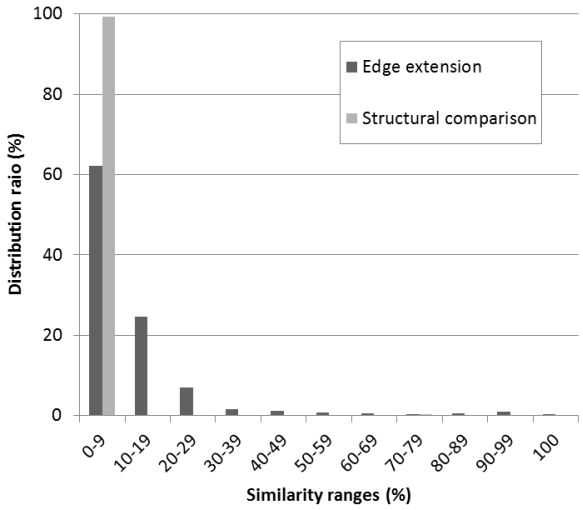

Fig. 2. Similarity distribution ratios for CFGs with different structures

Fig. 2는 실험 결과 나타나는 유사도의 분포 비율을 그래프로 보여주고 있다. 가로축은 비교 결과 나타나는 유사도의 범위를 의미하고, 세로축은 각 유사도 범위에 속하는 결과의 비율을 나타낸다. 낮은 유사도 구간에서 많은 비교 결과가 위치할수록 다른 프로그램에 대한 변별력이 우수하다고 평가할 수 있다. 결과에서 기존의 구조적 비교 방법은 대부분의 비교 결과가 10% 미만의 구간에 위치하고 있으며 변별력이 아주 우수한 것으로 평가된다. 상대적으

로 에지 확장 방법은 높은 유사도 구간에 결과들이 위치하고 있지만, 대부분의 비교 결과는 30% 이하의 낮은 유사도 구간에 분포하고 있으며 서로 다른 프로그램임을 변별할 수 있음을 확인할 수 있다. 또한 70% 이상의 높은 유사도를 보이는 대부분의 제어 흐름 그래프는 JavaCC에 의해서 자동 생성된 프로그램의 제어 흐름 그래프의 비교 결과인 것으로 확인되었다.

서로 다른 프로그램에 대한 변별력에서 기존의 구조적 비교 방법은 기본 블록의 시그너쳐 정보가 다르면 기본 블록의 매칭 대상에서 배제되기 때문에 서로 다른 구조를 가진 경우 효과적인 결과를 보였다. 반면 에지 확장 방법은 서로 시그너쳐 정보가 다르더라도 명령어의 패턴에 일부분 유사성이 있다면 부분적인 매칭을 고려하기 때문에 상대적으로 높은 유사도 분포를 보인 것으로 보인다. 하지만, 유사도 결과는 다른 프로그램을 변별하기에는 충분히 낮은 결과 분포를 보이는 것으로 평가된다.

4.3 구조가 유사한 제어 흐름 그래프의 비교

본 실험은 서로 유사한 구조를 가진 제어 흐름 그래프를 비교할 때 높은 유사도를 결과로 보일 수 있는지를 평가한다. 실험을 위해서 유사한 구조의 프로그램은 원본 프로그램을 변형한 프로그램을 고려하였으며, 난독화 도구인 Smokescreen [14]과 Zelix KlassMaster(ZKM) [15]를 사용하였다. 이 도구는 원본 프로그램에 대해서 실행 결과는 동일하지만 역분석이 어렵고 디컴파일이 안되도록 프로그램으로 변환하는 역할을 한다. 따라서 변환된 프로그램은 원본 프로그램의 기본 골격은 유지하지만 내부적인 구성이 달라진다. 본 실험에서는 Table 2에서 제시한 실험 대상 프로그램에 대해서 두 가지 난독화 도구를 이용해서 변환하고, 원본과 변환된 제어 흐름 그래프를 비교하는 실험을 수행한다.

Table 4. Experimental results of comparisons between CFGs with similar structures(Smokescreen and ZKM)

	Smokescreen		Zelix KlassMaster	
	Edge extension	Structural comp.	Edge extension	Structural comp.
Avg. similarity	92.9%	44.3%	77.1%	30.4%
Min. similarity	49.6%	0.0%	23.3%	0.0%
Max similarity	100.0%	99.8%	100.0%	100.0%
# of comp.	197	197	197	197
Time (sec)	118s	63s	141s	79s

Table 4는 서로 유사한 구조를 가지는 프로그램의 비교 실험 결과를 보여주고 있다. Smokescreen과 ZKM으로 변환한 프로그램에 대해서 각각 비교하였으며, 성능 평가를 위해서 기존의 구조적 비교 방법도 함께 실험하였다. Smokescreen을 이용해 변환한 프로그램의 비교 실험에서 에지 확장 방법의 유사도는 최소 49.6%에서 100% 사이에 분포하고 있으며 평균 유사도는 92.9%로 과반수 이상이 90% 이상의 높은 유사도를 보이고 있음을 확인할 수 있다. 반면에 구조적 비교 방법은 최소 0.0%로 기본 블록의 매칭에 실패한 경우가 많았으며, 평균 44.3%의 유사도를 보이고 있다. ZKM으로 변환한 프로그램에 대한 실험 결과에서 에지 확장은 최소 23.3%의 유사도를 보였으며, 평균 77.1%의 유사도를 보임으로써 Smokescreen을 통한 비교 실험보다 유사도가 다소 낮아진 결과를 보이고 있다. 기존의 구조적 방법도 마찬가지로 유사도의 평균이 30.4%로 Smokescreen에 비해서 낮아졌음을 확인할 수 있다. 본 실험 결과로부터 Smokescreen 이나 ZKM을 이용해 수정된 프로그램에 대해서 에지 확장을 적용한 방법이 기존의 구조적 비교 방법 보다 높은 유사도 값을 보이고 있으며, 구조적 유사성을 탐지하는데 보다 효과적임을 확인할 수 있다.

수행 시간은 각각의 실험에서 총 197쌍의 유사한 구조를 가진 제어 흐름 그래프를 비교하는데 걸린 총 소요 시간을 측정하였다. 기존의 구조적 비교 방법은 Smokescreen과 ZKM을 이용한 실험에서 각각 63초와 79초가 걸렸으며, 에지 확장 방법은 118초와 141초가 걸려서 2배 정도 많은 시간이 걸렸다. 앞 절의 실험에서 10% 정도의 수행 속도 차이가 난 것에 비해 상대적으로 많은 차이가 난 것을 확인하였다. 유사한 제어 흐름 그래프의 경우 기본 블록의 시그너쳐를 통해서 매칭하는 구조적 비교 방법의 경우 매칭이 원활하게 진행되면서 고정점을 찾기 위한 매칭 과정의 반복 횟수가 줄어든다. 이런 이유 때문에 유사한 구조의 제어 흐름 그래프에 대해서는 구조적 비교 방법이 보다 빠른 성능을 보인 것으로 평가된다. 또한 Smokescreen에서 보다 ZKM에서 보다 많은 시간이 걸린 것도 구조적인 차이가 커짐으로 인해서 고정점을 찾기 위한 반복이 증가하기 때문인 것으로 분석된다. 에지 확장 방법은 기본 블록의 매칭을 위해서 기본 블록 내의 명령어들을 함께 비교해야하기 때문에 2배 정도의 시간이 소요되었지만, 실제 환경에서 적용할 수 있는 수준의 효율성을 갖고 있는 것으로 평가된다.

Fig. 3과 Fig. 4는 본 실험 결과로 나온 결과의 유사도 범위별 분포 비율을 보여주고 있다. Smokescreen을 이용한 실험 결과에서 에지 확장 방법은 15% 정도의 결과는 100% 매칭 관계를 확인할 수 있었으며 80% 이상의 결과는 유사도가 90% 이상의 범위에 위치함으로써 매우 높은 유사도 분포를 확인할 수 있었다. 기존의 구조적 방법은 90% 이상 높은 유사도를 보인 결과는 상대적으로 적었으며, 35% 이상의 결과는 10% 이하의 낮은 유사도를 보임으로써 매칭이 효과적으로 이루어지지 않음을 알 수 있다.

ZKM을 이용한 실험 결과에서 에지 확장 방법은

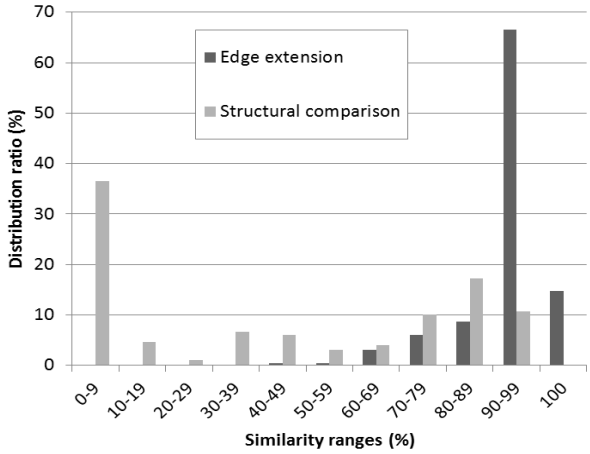

Fig. 3. Similarity distribution ratios for CFGs with similar structures (Smokescreen)

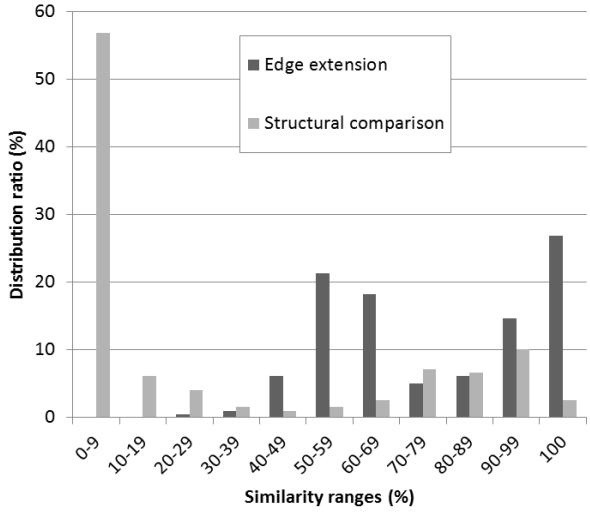

Fig. 4. Similarity distribution ratios for CFGs with similar structures (ZKM)

Smokescreen을 이용한 실험 보다 낮은 유사도 구간에 분포 비율이 높아졌다. 에지 확장 방법의 경우 25% 정도의 결과는 100% 일치하는 결과를 얻을 수 있었지만, 40% 정도의 결과는 50-70% 구간에 분포하고 있다. 그리고, 50% 이상의 상대적으로 높은 유사도 구간에 대부분의 결과들이 분포하고 있다. 기존의 구조적 비교 방법은 과반수에 해당하는 결과가 10% 미만의 유사도를 보였으며, 전체적으로 에지 확장을 통한 방법에 비해서 매칭 능력이 떨어지는 결과를 보였다. ZKM을 이용한 실험에서 유사도가 상대적으로 낮게 나온 것은 ZKM을 통한 변환이 제어 흐름 그래프의 구조적인 형태에 보다 많은 변화를 주기 때문이다.

본 논문에서 제안한 에지 확장 방법이 유사한 프로그램의 비교에서 기존의 구조적 비교 방법 보다 우수한 성능을 보인 것은 프로그램의 내용이 수정되더라도 제어 흐름 그래프의 기본 골격이 유지된다면 선후 관계에 있는 기본 블록의 관계 또한 그대로 유지되는 경향이 강하기 때문인 것으로 분석된다. 유사도가 낮게 나온 결과는 난독화를 통한 변화가 제어 흐름 그래프의 변형 뿐만 아니라 명령어 자체의 치환과 같은 복합적인 수정을 포함하고 있기 때문이다. 기존의 구조적 비교 방법은 기본 블록의 매칭을 위해서 기본 블록 내의 명령어나 구문 정보를 자세히 고려하기 보다는 기본 블록의 시그너처 정보를 이용해서 빠르게 동일한 블록을 매칭하는 것을 목표로 하고 있다. 이런 접근 방법은 수행 성능에 있어서 빠른 비교 및 매칭이 가능하고, 프로그램에서 일부 영역에 한정된 업데이트만 적용된 경우에 서로 다른 부분을 찾아낼 때 효과적으로 적용이 가능하다. 하지만, 프로그램의 전반적인 수정으로 인해 기본 블록의 시그너처 정보의 변화가 많은 경우 매칭을 통한 비교가 효과적이지 못한 것으로 나타났다. 본 실험 결과를 통해서 에지 확장 방법은 임의의 제어 흐름 그래프를 비교할 때 보다 신뢰성 있는 결과를 보여준다는 것을 알 수 있다.

5. 토론 및 향후 연구 계획

본 논문에서는 바이너리 프로그램의 제어 흐름 그래프를 비교하기 위해서 기본 블록의 유사도와 더불어 에지 확장을 통해 제어 흐름 그래프의 구조적인 특성을 비교한다. 각 기본 블록들은 서로간의 유사성 순서에 따라 매칭함으로써 전체 제어 흐름 그래프 사이의 매칭 관계를 확인하고 유사도를 구할 수 있다.

본 논문에서 제안한 방법은 다음과 같은 장점을 가지고 있다. 기존의 바이너리 프로그램의 제어 흐름 그래프를 비교하는 구조적 비교 방법 [12, 13]은 제어 흐름 그래프의 차이를 분석하는데 초점을 맞추고 있다. 따라서 전체적으로 유사하지만 일부분의 수정이나 업데이트 패치가 된 프로그램에서 수정된 부분을 찾고 그 변경된 내용을 빠른 시간에 분석할 수 있다는 장점이 있다. 반면에 임의의 여러 부분에서 변경 된 제어 흐름 그래프를 비교할 때는 기본 블록의 시그처너 정보를 이용해 매칭하는데 어려움이 있었다.

본 비교 방법의 개선을 위해서 다음과 같은 향후 연구 계획을 가지고 있다. 그래프 구조를 비교하기 위해서는 그래프를 구성하는 두 가지 원소인 기본 블록과 에지의 비교 결과를 적절히 조율할 필요가 있다. 본 논문에서는 기본 블록의 유사성을 비교하기 위해서 두 기본 블록 사이의 유사도 $m()$과 에지 확장을 반영하는 유사도 $m1(), m2()$를 함께 반영하고 있다. 이 유사도 값을 평균적으로 반영하기 위해서 가중치 비율을 $\alpha = \beta + \gamma$로 실험을 수행하였다. 프로그램에 따라 하나의 기본 블록에 대해 나가는 에지와 들어오는 에지는 여러 개인 경우가 일반적이며 많은 수의 기본 블록들이 연관되기 때문에 노드와 에지에 대한 가중치 비율을 프로그램의 특성에 맞게 조정할 필요가 있다. 향후 다양한 프로그램에서 실험을 통해 기본 블록과 제어 흐름 사이의 유사도 반영 비율을 조율한다면 비교 결과의 신뢰성을 개선할 수 있을 것이다.

6. 결 론

프로그램의 제어 흐름 그래프는 프로그램의 정적인 구조와 동작 특성을 효과적으로 기술할 수 있기 때문에 정적 분석과 동적 분석 등에서 광범위하게 사용되고 있다. 본 논문에서는 바이너리 프로그램의 구조를 표현하는 제어 흐름 그래프를 비교하는 방법을 제안하였다.

본 논문에서는 제어 흐름 그래프의 비교를 위해서 그래프를 구성하는 기본 블록들이 가지는 내부 정보를 비교하는 방법을 사용하고 있다. 기본 블록에 포함된 프로그램의 명령어나 구문 정보를 비교함으로써 기본 블록 사이의 유사성을 확인할 수 있다. 또한, 기본 블록 사이에 제어 흐름을 표현하는 에지의 정보를 반영하기 위해서 에지 확장 방법을 제안하였다. 에지 확장은 기본 블록을 매칭할 때 들어가는 에지와 나가는 에지로 인접한 기본 블록들의 관계를 함께 고려함으로써 프로그램의 수정에 강인한 제어 구조를 반영할 수 있다. 제어 흐름 그래프 사이의 유사도를 구하기 위해서 기본 블록과 에지 확장을 통해 구해진 유사도에 따라 기본 블록을 매칭하고, 매칭된 결과를 이용해서 전체 프로그램의 유사도를 검출할 수 있다.

본 논문에서 제안한 방법의 성능을 평가하기 위해서 자바 프로그램으로부터 추출한 제어 흐름 그래프를 대상으로 실험을 하였다. 실험은 구조의 유사성에 따라 두 가지 기준으로 실험을 수행하였으며, 비교를 위해서 기존의 구조적 비교 방법을 함께 실험하였다. 기존의 구조적 비교 방법은 서로 다른 프로그램에 대한 변별력은 매우 우수하였지만, 서로 유사한 프로그램에 비교는 충분한 신뢰성을 가지지 못하였다. 에지 확장 방법은 서로 다른 프로그램에 대한 충분한 변별력을 가지고 있었으며, 구조가 유사한 프로그램에 대한 매칭 능력에서 기존의 방법에 비해 효과적인 결과를 보였다. 다만, 수행 성능에 있어서 에지 확장 방법은 기본 블록의 매칭을 위해서 기본 블록 내의 명령어를 모두 고려해야 하기 때문에 기존의 방법에 비해서 상대적으로 많은 시간이 걸리는 경향이 있었다.

제어 흐름 그래프의 비교는 프로그램의 유사성 비교를 통한 코드의 최적화, 유사 코드 검출, 코드의 도용 탐지 뿐만 아니라 최근에는 악성 코드의 검출에도 활용되고 있다. 제어 흐름 그래프의 효율적인 비교는 프로그램의 다양한 분석 방법에 적용할 수 있으며, 향후 여러 응용 사례에서 적용될 수 있을 것이라 기대된다.

참 고 문 헌

[1] Renhard Wilhelm and Dieter Maurer, Compiler Design, Addison-Wesley, 1995.

[2] Ruben Smelik, Arend Rensink, and Harmen Kastenberg, Specification and Construction of Control Flow Semantics, In Proceedings of IEEE Symposium on Visual Languages and Human-Centric Computing, pp.65-72, Sept., 2006.

[3] Danilo Bruschi, Lorenzo Martignoni, and Mattia Monga, Detecting Self-Mutating Malware Using Control-Flow Graph Matching, In Proceedings of Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA 2006), LNCS 4064, pp.129-143, 2006.

[4] Guillaume Bonfante, Matthieu Kaczmarek, and Jean-Yves Marion, Control Flow to Detect Malware, Inter-Regional Workshop on Rigorous System Development and Analysis 2007.

[5] Guillaume Bonfante, Matthieu Kaczmarek, and Jean-Yves Marion, Control Flow Graphs as Malware Signatures, International Workshop on the Theory of Computer Viruses, 2007.

[6] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman, The Design and Analysis of Computer Algorithms, Addison - Wesley, 1974.

[7] John W. Raymondl and Peter Willett, Maximum common subgraph isomorphism algorithms for the matching of chemical structures, Journal of Computer-Aided Molecular Design, 16: pp.521-533, 2002.

[8] Faisal N. Abu-Khzam, Nagiza F. Samatova, Mohamad A. Rizk, and Michael A. Langston, The Maximum Common Subgraph Problem: Faster Solutions via Vertex Cover, In IEEE/ACS International Conference on Computer Systems and Applications (AICCSA 2007), pp.367-373, May, 2007.

[9] Zhiping Zeng, Anthony K. H. Tung, Jianyong Wang, Jianhua Feng, and Lizhu Zhou, Comparing Stars: On Approximating Graph Edit Distance, International Conference on Very Large Data Bases (VLDB 2009), 2009.

[10] Laura Zager, Graph similarity and matching, MS Thesis, Dept of EECS, MIT, 2005.

[11] Python Programming Language, http://www.python.org/

[12] Halvar Flake, Structural Comparison of Executable Objects, In Proceedings of the IEEE Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA), pp.161-173, 2004

[13] Thomas Dullien and Rolf Rolles, Graph-based comparison of executable objects, University of Technology in Florida, 2005.

[14] Smokescreen Java Obfuscator, http://www.leesw.com/smokescreen/

[15] Zelix KlassMaster, http://www.zelix.com/klassmaster/

임 현 일

e-mail : hilim@kyungnam.ac.kr
1995년 KAIST 전산학과(학사)
1997년 KAIST 전산학과(석사)
2009년 KAIST 전산학과(공학박사)
2010년~현 재 경남대학교 컴퓨터공학과 조교수
관심분야 : 소프트웨어 분석, 소프트웨어 보안, 소프트웨어 저작권 보호 및 도용 탐지 등