제어 흐름 그래프의 명령어 유사성에 기반한 자바 버스마크*

박희완, 임현일, 최석우, 한태숙
한국과학기술원 전자전산학과 전산학전공
e-mail:{hwpark, hilim, swchoi}@pllab.kaist.ac.kr, han@cs.kaist.ac.kr

A Java Birthmark Based on Similarity Between Instructions of Control Flow Graph

Heewan Park, Hyun-il Lim, Seokwoo Choi, and Taisook Han
Division of Computer Science, Dept. of EECS, KAIST

요 약

소프트웨어 버스마크는 프로그램을 식별하는데 사용될 수 있는 프로그램의 고유한 특징을 말한다. 본 논문에서는 자바 메소드의 제어 흐름 그래프 유사도에 기반한 자바 버스마크를 제안한다. 제어 흐름 그래프 유사도는 노드의 유사도와 에지의 유사도로 나누어 계산하였다. 노드의 유사도는 인접 노드의 유사도를 함께 고려했으며, 에지 유사도는 이미 매칭된 노드들 사이의 거리를 측정하는 방법을 사용했다. 본 논문에서 제안한 버스마크를 평가하기 위해서 서로 다른 프로그램을 구별할 수 있는 신뢰도와 프로그램 최적화나 난독화에 견딜 수 있는 강인도에 대한 실험을 하였다. 실험 결과로부터 본 논문에서 제안하는 버스마크가 기존의 정적 버스마크보다 신뢰도가 높으면서도 난독화나 컴파일러 변경에 강인하다는 것을 확인하였다.

1. 서론

최근 많은 프로그램들이 오픈 소스로 개발되고 있다[1]. 오픈 소스를 활용하면 프로그램 개발 시간과 비용을 줄일 수 있다는 장점이 있다. 그러나 오픈 소스도 지정된 라이선스로 배포되기 때문에 섣불리 오픈 소스를 사용하여 상용 프로그램을 만들 경우에는 라이선스 위반을 야기할 수 있다[2]. 특히 GPL(General Public License)로 배포된 오픈 소스는 누구나 마음대로 사용할 수는 있지만 이로부터 개발된 프로그램도 GPL에 의거하여 공개하는 것을 원칙으로 한다. 따라서 GPL 오픈 소스를 사용했으면서도 프로그램의 소스를 공개하지 않고 판매한다면 라이선스 위반으로 처벌을 받을 수 있다.

만일 라이선스 위반이 의심되는 프로그램의 소스 코드를 얻을 수 있다면 소스 표절 검사 도구[3]를 사용하여 GPL 오픈 소스와의 유사도를 측정하고 이로부터 라이선스 위반 여부를 판단할 수 있다. 그러나 프로그램은 대부분 컴파일된 클래스 파일이나 바이너리 형태로 배포되기 때문에 소스 코드를 얻을 수 없는 경우가 일반적이다.

소프트웨어 버스마크(Software Birthmark)[4,5,6]는 이런 상황에서 프로그램이 가지고 있는 고유한 특징을 자바 클래스나 바이너리로부터 직접 추출하여 유사도를 비교하는 방법이다. 본 논문에서는 자바 메소드의 제어 흐름 그래프 유사도를 기반으로 하는 버스마크를 제안하고 효용성을 평가한다.

* 이 논문은 2009년도 정부(교육과학기술부)의 재원으로 한국과학재단의 지원을 받아 수행된 연구임 (No. R01-2008-000-11856-0)

2. 관련 연구

최초의 버스마크 기법은 자바 클래스를 대상으로 하여 Tamada에 의해서 제안되었다[4]. 이 방법은 클래스 파일을 분석하여 필드 변수의 상수값, 메소드 호출 순서, 클래스 상속 구조, 사용된 클래스 정보를 추출하여 클래스 파일의 유사도를 계산한다. Tamada 버스마크는 자바 클래스의 구조적인 특징 위주의 버스마크이기 때문에 서로 다른 알고리즘을 구별하는 신뢰도가 낮다는 단점이 있다.

k-gram 버스마크[5]는 자바 바이트코드 명령어의 연속된 k개의 시퀀스 집합을 버스마크로 사용한다. 이 방법은 클래스에 포함된 모든 바이트코드 명령어에 대해서 유사성을 비교를 하기 때문에 서로 다른 프로그램을 구별하는 능력은 매우 뛰어나지만 프로그램의 제어 흐름을 분석하지 않기 때문에 분기문이나 반복문을 변경하는 것만으로도 버스마크가 손상될 수 있다는 문제가 있다.

트레이스 버스마크[6]는 자바 클래스로부터 제어 흐름을 분석하여 실행 가능한 모든 바이트코드 트레이스를 추출하고, 트레이스 집합의 유사도를 서열 정렬 알고리즘을 이용해서 비교하는 방법이다. 이 방법은 제어 흐름 분석을 거쳐서 추출된 트레이스를 비교하기 때문에 난독화와 같은 프로그램 변환에 강하다는 장점이 있다. 그러나 분

기문의 개수에 대해서 기하급수적으로 트레이스의 개수가 증가하기 때문에 큰 프로그램에는 적용하기 어렵다.

3. 제어 흐름 그래프 유사도 기반 버스마크

3.1 소프트웨어 버스마크

Myles는 copy 관계를 이용하여 소프트웨어 버스마크를 정의했다. 다음은 버스마크에 대한 Myles의 정의[5]이다.

정의 1 (버스마크)
프로그램 p와 q에 대한 함수 f가 다음 조건을 만족할 때, f(p)를 프로그램 p의 버스마크라고 한다.
조건 1. f(p)는 부가적 정보 없이 p 자신에서부터 얻는다.
조건 2. 프로그램 p와 q가 서로 copy 관계에 있다면 f(p) = f(q) 이다.

다음 두 가지 속성은 Tamada와 Myles가 제시한 버스마크가 만족시켜야 하는 평가 기준을 나타낸다.

속성 1 (신뢰도 Credibility)
같은 기능을 하는 두 프로그램 p와 q에 대해서, p와 q가 독립적으로 개발되었을 때, f(p)≠ f(q)을 만족해야 한다.
속성 2 (강인도 Resilience)
프로그램 p'이 프로그램 p로부터 변환되었다고 할 때, f(p) = f(p')을 만족해야 한다.

3.2 제어 흐름 그래프 유사도 기반 자바 버스마크

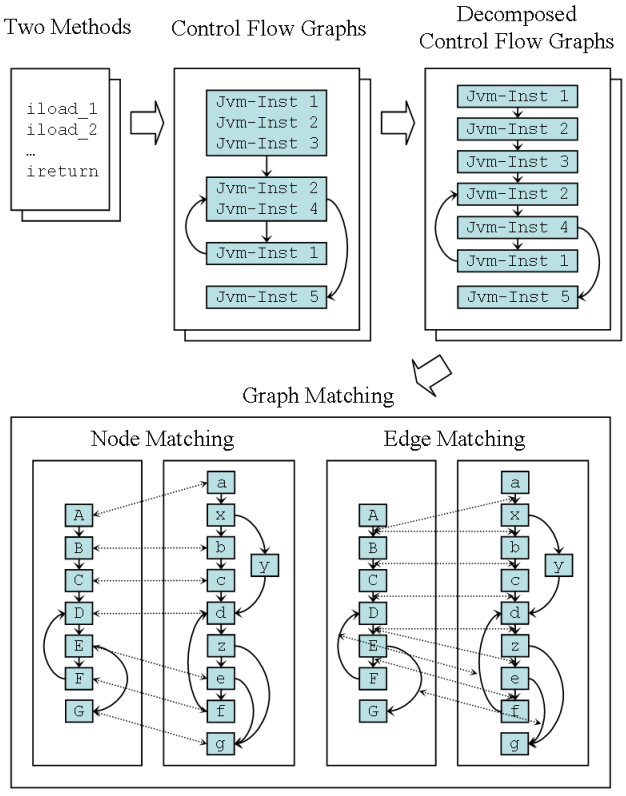

그림 1. 그래프 유사도 기반 버스마크 시스템

제어 흐름 그래프 유사도 기반 버스마크 시스템의 전체 구조는 그림 1과 같다. 버스마크 시스템은 먼저 두개의 자바 메소드를 입력으로 받는다. 두 메소드로부터 각각 기본 블록 단위의 제어 흐름 그래프를 생성하고, 모든 노드가 1개의 바이트코드 명령어만 가지도록 기본 블록을 분리시킨 제어 흐름 그래프를 생성한다. 그 이유는 노드와 노드를 비교할 때 유사도 계산을 간단하게 하기 위해서이다. 그래프 유사도 비교는 노드 유사도 비교와 에지 유사도 비교로 구분하여 수행하고, 두 가지 유사도 결과를 이용해서 전체 메소드 유사도를 산출해낸다.

지금까지 잘 알려진 그래프 비교 알고리즘[7]에는 그래프 동형(Isomorphism), 편집 거리(Edit distance) 계산, 최대 공통 부분 그래프(Maximum common subgraph) 생성, 통계학적 유사도(Statistical Similarity) 비교 방법 등이 있다.

프로그램의 실행 시퀀스를 내포하고 있는 제어 흐름 그래프를 버스마크의 용도로 사용하기 위해서는 너무 엄밀한 비교를 해서는 안 된다. 왜냐하면 다양한 난독화 기법에 의해서 노드나 에지가 그래프에 추가될 수 있기 때문이다. 또한 너무 함축된 통계 정보를 이용해서도 안 된다. 서로 다른 알고리즘을 구별할 정도의 신뢰도가 보장되어야 하기 때문이다. 즉, 버스마크의 신뢰도와 강인도를 모두 충족시키는 새로운 그래프 비교 방법이 필요하다. 이 논문에서는 제어 흐름 그래프의 노드 매칭과 에지 매칭을 통한 새로운 그래프 유사도 비교 방법을 제안한다.

4. 제어 흐름 그래프 매칭 알고리즘

4.1 노드 매칭(Node Matching)

노드 매칭을 위해서 본 논문에서는 제어 흐름 그래프를 구성하는 노드를 다음과 같이 두 가지로 구분하여 2단계 노드 매칭을 수행한다.

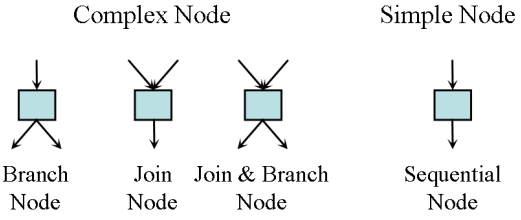

그림 2. 제어 흐름 그래프의 노드 분류

노드는 그림 2와 같이 복합(Complex) 노드와 단순(Simple) 노드로 구분하였다. 복합 노드는 나가는 에지가 두개 이상 포함된 분기(Branch) 노드이거나, 들어오는 에지가 두 개 이상 포함된 합류(Join) 노드, 또는 이 두가지가 혼합된 노드를 의미한다. 단순 노드는 들어오는 에지와 나가는 에지가 각각 1개 이하인 순차(Sequential) 노드를 의미한다.

복합노드는 제어 흐름 그래프 유사도 비교에서 중요한 매칭 포인트가 된다. 따라서 단순 노드보다 우선하여 1차 매칭 대상이 된다. 복합 노드의 매칭이 끝나면 복합 노드 사이에 위치한 단순 노드들을 대상으로 2차 매칭을 시도한다. 예를 들어 그림 1의 노드 매칭에서 노드 D와 E는 복합노드이다. 따라서 먼저 1차 매칭을 하고, 노드 A,B,C,F,G는 단순 노드이므로 2차 매칭을 한다. 이와같이 2단계 노드 매칭 기법을 사용하면 노드 사이의 매칭 대상을 줄일 수 있기 때문에 빠른 매칭이 가능하고, 서로 연관된 노드들 사이의 매칭을 유도하여 연관성이 없는 무분별한 매칭을 배제시키는 효과가 있다.

노드 매칭은 노드 간 유사도 점수가 높은 순서대로 매칭한다. 노드 간 유사도 점수는 0과 1사이의 값으로 정해지는데 비교 대상인 노드가 서로 같다고 해서 1점을 주지는 않는다. 같은 노드가 여러개 존재할 수 있기 때문이다. 예를 들어서 iload와 같이 자주 사용되는 바이트코드 명령어는 매칭 대상이 너무 많아질 수 있다. 따라서 노드 매칭을 할때 주변 노드들의 매칭 점수를 함께 고려하는 방법을 사용하였다. 두 노드 X와 Y의 유사도 점수를 구하는 계산식은 다음과 같다.

$$NodeSim(X,Y) = \frac{P_2(X,Y)+2\cdot P_1(X,Y)+3\cdot C(X,Y)+2\cdot N_1(X,Y)+N_2(X,Y)}{9}$$

where,

$$P_2(X,Y)=\frac{|\,prev(prev(X))\cap prev(prev(Y))\,|}{|\,prev(prev(X))\,|}$$

$$P_1(X,Y)=\frac{|\,prev(X)\cap prev(Y)\,|}{|\,prev(X)\,|}$$

$$C(X,Y)=\begin{cases}1 & if\ \ X=Y\\ 0 & if\ \ X\neq Y\end{cases}$$

$$N_1(X,Y)=\frac{|\,next(X)\cap next(Y)\,|}{|\,next(X)\,|}$$

$$N_2(M,N)=\frac{|\,next(next(X))\cap next(next(Y))\,|}{|\,next(next(X))\,|}$$

$$prev(X)=\{x\,|\,x \text{ is a set of previous nodes of } X\}$$

$$next(X)=\{x\,|\,x \text{ is a set of next nodes of } X\}$$

여기서 P_1과 N_1은 비교 대상 노드인 X, Y와 인접하여 거리가 1인 노드의 집합이다. 그리고, P_2와 N_2는 비교 대상 노드와 거리가 2인 노드의 집합이다. 즉, 비교 대상 노드와 인접한 주변 노드들의 유사도를 함께 고려하여 최종 유사도를 구했으며, 유사도에 대한 가중치는 거리가 2인 노드, 거리가 1인 노드, 비교 대상 노드에 대해서 각각 1:2:3으로 정하였다. 만일 원본 메소드에 포함된 노드가 비교 대상 메소드의 노드와 매칭되지 않고 남은 경우는 감점의 의미로 -1점을 주었다.

4.2 에지 매칭(Edge Matching)

에지 매칭의 기본 개념은 원본 메소드에 포함된 에지가 비교 대상 메소드에도 존재하는지를 체크하는 것이다. 즉, 원본 그래프에서 존재하는 에지가 대상 그래프에도 존재할 경우에는 1점을 얻는다. 그러나 난독화에 의해서 에지 중간에 다른 노드가 끼어드는 경우가 발생할 수 있기 때문에 이것을 고려하여야 한다. 본 논문에서는 매칭된 노드 사이의 최단거리의 역수를 에지 매칭 점수로 계산하였다. 원본 메소드에 포함된 두 노드 A, B 사이의 에지를 (A,B)라고 하고, 비교 대상 메소드에서의 에지를 (a,b)이라고 했을 때 에지 유사도는 다음과 같이 구한다.

$$EdgeSim((A,B),(a,b)) = \frac{1}{ShortestPa\ th(a,b)}$$

만일 노드 a와 b사이에 에지가 존재한다면 에지 유사도는 1이 되지만, a와 b사이에 난독화에 의해서 다른 노드가 1개, 또는 2개가 삽입되었을 경우에는 에지 유사도 점수가 각각 1/2, 1/3로 감소한다. 예를 들어 그림 1의 에지 매칭에서 (A,B)는 (a,b)와 매칭이 되었는데 노드 x가 중간에 삽입되었기 때문에 (a,b) 노드 사이의 거리가 2로 증가하였고, 그 결과 EdgeSim((A,B),(a,b))= 1/2 이 된다.

만일 원본 메소드에 존재하는 에지가 비교 대상 메소드에 존재하지 않는 경우에는 감점으로서 -1점을 주었다.

4.3 메소드 유사도 계산

노드 매칭과 에지 매칭이 끝나면 최종 메소드 유사도를 계산할 수 있다. 메소드 유사도는 메소드에 포함된 모든 노드의 유사도의 합계와 모든 에지의 유사도의 합계를 이용하여 구한다. 두가지 유사도가 모두 고려되어야 하기 때문에 본 논문에서는 다음과 같이 노드 유사도 합과 에지 유사도 합의 평균으로 정의하였다.

$$MethodSim\ (M,N) = \frac{\sum NodeSim\ + \sum EdgeSim}{2}$$

5. 실험 및 평가

제어 흐름 그래프 유사도 기반 자바 버스마크의 효용성을 검증하기 위해서 대표적인 정적 버스마크 기법인 k-gram 버스마크, 트레이스 버스마크와 비교 실험을 하였다. 버스마크는 같은 일을 수행하지만 서로 독립적으로 구현된 프로그램을 구별할 수 있는 신뢰도가 중요하기 때문에 정수 배열을 정렬하는 6개의 자바 메소드를 대상으로 신뢰도를 측정하였다. 그리고 강인도를 측정하기 위해서 난독화 도구인 Smokescreen[8]을 사용하였고, 컴파일러가 바뀌었을 때의 강인도를 측정하기 위해서 Jikes 컴파일러[9]를 이용해서 실험하였다.

<표 1> 3-gram 버스마크의 신뢰도 실험

	Bubble	Insertion	Merge	Quick	Selection	Shell
Bubble	-	0.330	0.132	0.187	0.220	0.165
Insertion	-	-	0.188	0.293	0.323	0.241
Merge	-	-	-	0.017	0.115	0.037
Quick	-	-	-	-	0.374	0.228
Selection	-	-	-	-	-	0.115
Shell	-	-	-	-	-	-

* 평균 : 0.198

<표 2> 트레이스 버스마크의 신뢰도 실험

	Bubble	Insertion	Merge	Quick	Selection	Shell
Bubble	-	0.273	0.139	0.222	0.278	0.222
Insertion	-	-	0.212	0.121	0.273	0.061
Merge	-	-	-	0.246	0.205	0.067
Quick	-	-	-	-	0.256	0.111
Selection	-	-	-	-	-	0.077
Shell	-	-	-	-	-	-

* 평균 : 0.184

<표 3> 그래프 유사도 기반 버스마크의 신뢰도 실험

	Bubble	Insertion	Merge	Quick	Selection	Shell
Bubble	-	0.000	0.213	0.194	0.000	0.322
Insertion	-	-	0.198	0.203	0.064	0.285
Merge	-	-	-	0.000	0.000	0.034
Quick	-	-	-	-	0.298	0.011
Selection	-	-	-	-	-	0.000
Shell	-	-	-	-	-	-

* 평균 : 0.121

표 1, 2, 3은 각각 3-gram 버스마크, 트레이스 버스마크, 그래프 유사도 기반 버스마크의 신뢰도 실험 결과이다. 신뢰도 실험에서는 서로 다른 정렬 알고리즘의 유사도를 비교한 결과이기 때문에 결과 값이 낮을수록 신뢰도가 높은 것을 의미한다. 측정 결과 평균값은 3-gram 버스마크와 트레이스 버스마크가 각각 0.198와 0184로서 비슷했으며 그래프 유사도 기반 버스마크는 0.121로서 기존 버스마크보다 7.7%, 6.3% 신뢰도가 높았다.

<표 4> 3-gram 버스마크, 트레이스 버스마크, 그래프 유사도 기반 버스마크의 강인도 실험

	Smokescreen			Jikes		
	3-gram	Trace	Graph	3-gram	Trace	Graph
Bubble	0.944	0.967	0.985	0.611	1.000	1.000
Insertion	0.939	0.970	0.982	0.606	1.000	1.000
Merge	0.763	0.916	0.933	0.829	1.000	1.000
Quick	0.842	0.905	0.953	0.875	1.000	1.000
Selection	0.718	0.809	0.868	0.718	1.000	1.000
Shell	0.911	0.974	0.990	0.600	1.000	1.000
평균	0.853	0.924	0.952	0.707	1.000	1.000

표 4는 강인도 실험 결과이다. 강인도는 난독화나 프로그램 변환에도 변하지 않는 성질을 의미하기 때문에 유사도가 높을 수록 높은 강인도가 높은 것을 의미한다. Smokescreen 난독화 도구를 사용한 실험에서는 그래프 유사도 기반 버스마크가 기존의 버스마크 보다 9.9%, 2.8% 높은 강인도를 보였고, Jikes를 이용한 실험에서는 트레이스 버스마크와 동일하게 100%의 유사도를 보이며 3-gram 버스마크에 비해서 29.3% 높은 강인도를 보였다.

Jikes 컴파일러는 분기문 블록을 javac 컴파일러와 다르게 컴파일 한다. 예를 들면, true와 flase 블록의 위치가 바뀌기도 하는데 k-gram 버스마크는 연속된 명령어의 유사도를 측정하기 때문에 명령어 블록이 바뀌는 프로그램 변환에 취약하다. 트레이스 기반 버스마크는 분기문에 대해서 양쪽 트레이스를 모두 생성하여 비교하기 때문에 이러한 문제가 발생하지 않았고, 본 논문에서 제안한 그래프 유사도 기반 버스마크도 그래프 노드의 좌, 우 위치가 바뀔 뿐 노드 자체는 바뀌지 않았기 때문에 높은 강인도를 얻었다.

6. 결론 및 향후 연구 과제

본 논문은 프로그램의 제어 흐름 그래프 유사도 기반 버스마크를 제안하였다. 그래프 유사도를 노드 유사도와 에지 유사도로 구분하였는데, 노드 유사도는 비교 대상 노드와 인접한 주변 노드들의 유사도를 함께 고려하는 방법을 사용했으며, 에지 유사도는 이미 매칭된 노드들 사이의 거리를 측정하는 방법을 사용했다. 본 논문에서 제안한 버스마크를 평가하기 신뢰도와 강인도 실험을 하였다. 실험 결과로부터 본 논문에서 제안하는 버스마크가 기존의 정적 버스마크보다 신뢰도가 높으면서도 난독화나 컴파일러 변경에 강인하다는 것을 확인하였다.

향후 연구 과제는 다음과 같다. 노드 매칭에 있어서 자료 의존성 분석(Data Dependency Analysis)을 통해서 노드 매칭의 정확도를 높이는 방법을 고려하고 있으며 유사도 비교 대상을 애플리케이션 단위로 확장하는 것을 고려하고 있다.

참고 문헌

[1] “The Open Source Definition," http://www.opensource.org/docs/osd.
[2] "Open Source Licenses," http://www.opensource.org/licenses/.
[3] L. Prechelft, G. Malpohl, and M. Philippsen, "Finding plagiarisms among a set of programs with jPlag", Journal of Universal Computer Science, vol. 8, no. 11, pp. 1016-1038, 2002.
[4] Tamada, H., Nakamura, M., Monden, A., Matsumoto, K. Java birthmark Detecting the software theft. IEICE Transactions on Information and Systems, E88-D, 9 (Sept. 2005), 2148-2158.
[5] Ginger Myles and Christian Collberg. k-gram Based Software Birthmarks. In Proceeding of the 2005 ACM Symposium on Applied Computing, pp. 314-318. Santa Fe, New Mexico, USA, 2005.
[6] 박희완, 임현일, 최석우, 한태숙. 정적 트레이스 기반의 자바 버스마킹 기법. 한국정보처리학회 2008년 춘계학술대회 논문집 pp. 281-284. 2008년 5월.
[7] Graph similarity and matching, Laura Zager, MS Thesis, EECS, MIT, 2005
[8] "Smokescreen" http://www.leesw.com/smokescreen/.
[9] "Jikes Java Compiler" http://jikes.sourceforge.net/.