제26회 한국정보처리학회 추계학술발표대회 논문집 제13권 제2호 (2006. 11)

표절검사를 위한 프로그램 추적기법

지정훈*, 우균, 조환규
부산대학교 컴퓨터공학과
e-mail:jhji@pusan.ac.kr

The Tracing Method of Program for Plagiarism Detection

Jung-Hoon Ji*, Gyun Woo, Hwan-Gyu Cho
Dept of Computer Engineering, Pusan National University

요 약

표절을 검사하는 방법으로는 문서 내의 특정 정보들을 추출하여 비교하는 지문법(fingerprint)과 파스트리(parse tree)와 같이 프로그램의 특정한 구조를 이용하여 문서의 구조적 유사성을 검사하는 구조적(structure metrics) 검사방법들이 있다. 본 논문에서는 표절검사를 위한 프로그램 추적 기법을 제안한다. 프로그램 추적 기법은 프로그램을 구문단계에서 정적으로 수행을 하여 그 수행되는 함수들의 순서에 따라 주요 키워드를 추출하여 새롭게 정렬하는 방법이다. 실험결과 사용하지 않는 코드 삽입, 함수 위치 변경 및 합성 등과 같은 표절 스펙트럼에서 정의한 표절 방법에 대하여 효과적으로 검출할 수 있었다.

1. 서론

프로그램 표절은 다른 사람이 작성한 프로그램을 소스코드에 대한 이해 없이 복사 또는 간단한 수정을 통하여 단시간에 동일한 동작을 수행하는 프로그램을 만드는 것이다. 프로그램 표절은 대학의 프로그래밍 언어 관련 교과목에서 빈번하게 발생하며 표절 여부를 가려내기 위해서는 많은 시간과 비용이 든다. 그러므로 자동화된 표절 검사 시스템이 요구된다[1]. 자동화 된 표절 검사 시스템으로는 YAP[2], MOSS[3], JPLAG[4] 등이 있다.

Faidhi와 Robinson은 프로그램 표절 유형을 6단계의 표절 스펙트럼으로 정의하였다[5]. 프로그램 표절 방법은 변수나 함수의 이름 변경과 같은 간단한 방법에서부터 함수의 합성이나 제어구조 변경과 프로그램의 구조를 변경하는 방법 등이 있다.

프로그램 표절 검사 방법은 지문법(fingerprints)과 구조적 특징을 고려한 표절 검사 방법이 있다. 지문법은 사용된 단어의 빈도 수, 단어, 문장, 문단의 평균길이, 부호 사용 횟수 등의 정보를 이용하여 표절을 검사한다. 지문법은 일반 문서의 표절 검사에는 효율적이지만 구조적인 특성과 흐름을 가지고 있는 프로그램의 표절 검사에는 한계가 있다.

프로그램 표절 검사에는 주로 구조적 표절 검사 방법이 사용된다. 구조적 표절 검사 방법은 크게 두 부분으로 나눌 수 있다. 첫 단계는 표절 검사를 위한 전처리 단계로 소스코드를 중간표현으로 변환한다. 중간 표현으로는 일련의 문자 스트링이나 파스트리(parse tree), 토큰 리스트 등이 사용된다. 두 번째 단계는 유사도 비교 알고리즘을 중간표현에 적용시켜 두 프로그램 사이의 표절여부를 판단한다.

정확한 표절 검사를 위해서는 소스코드로부터 표절을 가려내는데 필요한 정보를 추출하여 중간표현을 만드는 방법과 두 중간표현들 사이에 유사도를 산출하는 유사도 비교 알고리즘의 정확성이 요구된다.

본 논문은 프로그램 표절 검사의 정확성을 높이기 위한 방법으로 표절 검사의 전단부(front-end)에 해당하는 구조적 소스코드 분석 방법에 대하여 제시한다. 본 논문의 구조는 다음과 같다. 2절에서는 관련연구로 프로그램의 주요 표절 방법 및 검사 방법에 대하여 알아보고, 3절에서는 본 논문에서 제시하는 표절검사를 위한 구조적 소스코드 분석 방법에 대하여 알아본다. 그리고 4절에서는 실험을 통하여 본 논문에서 제시하는 방법의 효율성을 알아본다. 마지막으로 5절에서는 본 논문의 결론과 향후 연구에 대한 방향을 제시한다.

2. 프로그램 표절 및 검사 방법

2.1 프로그램 표절 방법

Faidhi와 Robinson은 프로그램 표절 유형을 6단계의 프로그램 표절 스펙트럼으로 정의하였다[5]. 표 1은 표절 스펙트럼이다.

<표 1> 프로그램 표절 스펙트럼

단계	표절방법
1	주석삽입
2	식별자 이름 수정
3	변수나 함수의 위치변경
4	함수의 합성(procedure combination)
5	동일한 의미의 문장(statement)으로 수정
6	프로그램의 전체적인 제어구조 변경

위의 표절 스펙트럼 중 4단계 이상의 표절방법들은 소스코드의 구조적 특성이 변하기 때문에 표절을 밝혀내기가 어렵다.

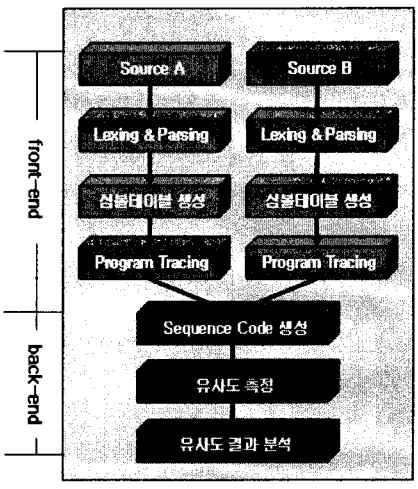
2.2 프로그램 표절 검사 방법

프로그램 표절 검사의 대표적인 방법은 두 가지가 있다. 일반 문서의 표절 검사 방법에 사용되는 지문법과 문서의 구조적 특징을 고려한 구조적 표절 검사 방법이 있다. 일반 문서의 경우 단락의 삭제나 재배치, 일부 문장 편집, 문서에서 일부 단어를 의미가 동일한 단어로 교체와 같은 방법으로 문서를 표절한다. 지문법은 이러한 일반 문서의 표절 검사에 주로 사용되는 방법이다. 지문법은 사용된 단어의 빈도 수, 단어, 문장, 문단의 평균길이, 부호 사용 횟수 등의 정보를 이용하여 두 문서간의 표절을 검사한다. 지문법은 일반 문서의 표절 검사에는 효율적이지만 구조적인 특성과 흐름을 가지고 있는 프로그램의 표절 검사에는 한계가 있다.

구조적 검사 방법은 프로그램의 구조적 변화를 이용하는 표절 방법까지도 검출할 수 있다. 구조적 검사 방법에는 프로그램 소스 전체를 하나의 문자 스트링으로 보고 특별한 작업 없이 그대로 이용하는 방법과 전체 프로그램을 프로그램 parsing tree로 재배열하여 그렇게 배열된 트리를 순회법(traversal)을 이용하여 어떤 선형의 순서를 만들어 내는 방법 등이 있다. 본 논문에서는 소스코드의 구조적 분석 방법으로 프로그램의 구문단계에서 정적으로 수행을 하여 그 수행되는 함수들의 순서에 따라 주요 키워드를 추출하여 새롭게 정렬하는 방법을 사용한다.

3. 프로그램 추적(tracing)을 이용한 표절검사

구조적 표절 검사는 크게 전단부(front-end)와 후단부(back-end)의 두 단계로 나뉜다. 전단부에서는 프로그램의 소스코드를 분석하여 토큰을 나누고, 파스 트리 구성과 심볼 테이블을 생성한다. 생성된 파스 트리와 심볼 테이블을 이용하여 프로그램을 추적(tracing) 하면서 프로그램의 구조적 특징을 포함하는 시퀀스 코드(sequence code)를 생성한다. 후단부에서는 유사도 비교 알고리즘을 사용하여 두 프로그램 사이의 표절여부를 가려낸다. 그림 1은 프로그램 표절도 검사의 수행 과정을 나타낸 것이다.



(그림 1) 표절 검사 수행 과정

3.1 프로그램 추적

표절검사 전단부는 소스코드를 처음부터 끝까지 모두 읽어 들여 파싱과 심볼 테이블을 생성한 다음 프로그램의 시작부터 끝까지 추적하면서 유사도 측정에 이용되는 시퀀스 코드(sequence code)를 생성한다. 시퀀스 코드는 후단부 작업의 유사도 측정 알고리즘의 입력 데이터로 사용된다. 프로그램은 main() 함수에서 시작하여 main() 함수의 마지막까지 실행하면 끝난다. 일반 문서는 문서의 처음부터 끝까지 순차적으로 추적을 하면서 정보를 추출하면 문서의 특성을 파악할 수 있지만 프로그램은 순차적인 특성보다는 함수, 제어 구문과 같이 흐름적인 특성을 포함하고 있다. 그리고 프로그램을 추적하면서 유사도 측정 데이터를 수집하면 2절에서 설명한 프로그램 소스코드의 주요 표절 기법들 중 주석문 삽입, 실제로 사용되지 않는 코드 삽입, 정의된 함수의 위치 변경, 두 개의 함수 코드를 하나로 합치는 표절 등 구조적 특성을 변경하는 표절 기법에 대해서 효율적으로 검출할 수 있다.

프로그램을 추적할 때 주의해야 할 사항은 함수가 순환적(recursive)으로 호출될 때 무한 반복에 빠지지 않는 것이다. 무한 반복을 방지하려면 직접순환(direct recursion)외에도 간접순환(indirect recursion)까지 고려해야 한다. 본 논문에서는 이런 경우를 해결하기 위하여 함수 호출 스택(function call stack)을 이용한다. 함수의 호출이 있으면, 함수명과 파라미터 정보를 스택에 삽입하고 함수에서 반환(return)하면 스택에서 제거한다. 함수 호출 시에 스택을 검사하여 동일한 함수가 스택에 있으면 호출하지 않고 체크만 한다.

프로그램 추적 방법을 이용하면 프로그램의 실행에 관련된 코드 정보들만 유사도 측정 데이터에 포함시킬 수 있으므로 표절 검사를 효율적으로 수행할 수 있다.

3.2 유사도 측정

두 프로그램의 유사도 측정 단계에서는 적응적 지역정렬 알고리즘[6]을 사용하여 유사도 점수를 계산한다. 적응적 지역정렬에서는 빈도가 높은 키워드의 일치에 대해서는 낮은 점수를 부여하고, 빈도가 낮은 키워드의 일치에 대해서는 높은 점수를 부여한다.

유사도 행렬을 이용하여 유사구간과 유사도 점수를 구했으면 다음으로 유사도를 산출한다. 유사도는 유사도 점수를 0~100%로 정규화한 것이다. 유사도 점수는 다음 함수로 정의된다.

$$SIM_{abs}(A,B)=\sum_{(a,b)\in align(A,B)} M^{P}(a,b)$$

*align*은 두 프로그램 A와 B를 정렬하여 유사구간을 산출하는 함수이다.

정규화된 유사도를 산출하는 방식은 두 가지가 있다. 일반적으로 사용되는 방식은 두 프로그램의 유사도 점수 두 배를 각 프로그램에 대하여 자신과 정렬시킨 후 유사도 점수의 합으로 나누는 방식이다. 유사도 $SIM_{sum}(A,B)$ 함수는 다음과 같이 정의된다.

$$SIM_{sum}(A,B)=\frac{2SIM_{abs}(A,B)}{SIM_{abs}(A,A)+SIM_{abs}(B,B)}$$

유사도 점수를 정규화 하는 다른 방법으로서 $SIM_{abs}(A,A)$와 $SIM_{abs}(B,B)$중 작은 값으로 나누는 방식이 있다. 작은 점수를 기준으로 정규화한 유사도를 본 논문에서는 $SIM_{\min}(A,B)$로 정의한다.

$$SIM_{\min}(A,B)=\frac{SIM_{abs}(A,B)}{\min\{SIM_{abs}(A,A),SIM_{abs}(B,B)\}}$$

4. 실험 및 평가

4.1 실험 데이터

실험 데이터로는 고급프로그래밍 교과목 과제에 대하여 학생들이 제출한 프로그램을 사용하였다. 표 2는 실험데이터를 정리한 것이다. 프로그램 그룹 1은 연관배열(associative array)을 다룬 문제이고, 그룹 2는 정수(integer)보다 더 큰 범위의 정수를 제공하는 클래스를 만드는 문제이다. 마지막으로 그룹 3은 템플릿을 이용한 리스트(list)를 다루는 문제이다.

<표 2> 고급컴퓨터프로그래밍 데이터 셋

순번	프로그램 그룹	파일 개수	순서쌍 수	소스코드 라인 수			
				최대	최소	평균	표준편차
1	assoc	48	1126	809	156	306.17	126.74
2	hugeint	33	528	2309	352	858.79	364.10
3	genlist	41	820	1413	308	924.32	246.51

4.2 유사도 분포

실험 데이터에 대하여 프로그램 추적 방법과 순차적 방법을 적용하여 유사도 분포를 조사하였다. 유사도 산출 방법은 $SIM_{sum}(A,B)$을 적용하였다. 표3은 유사도 분포 실험결과이다.

<표 3> 유사도 분포 실험 결과

유사도	assoc		hugeint		genlist	
	순차적	추적	순차적	추적	순차적	추적
0	754	717	456	411	703	687
5	153	108	64	86	82	91
10	84	124	3	20	24	35
15	44	79	1	7	2	3
20	23	48	2	2	1	1
25	27	30	2	0	1	0
30	16	8	0	0	1	0
35	12	5	0	1	1	0
40	2	1	0	0	0	0
45	1	0	0	0	2	0
50	0	0	0	0	1	0
55	0	0	0	0	0	1
60	0	0	0	0	1	0
65	1	0	0	0	0	0
70	2	0	0	0	0	0
75	2	1	0	0	0	0
80	0	1	0	0	1	0
85	0	0	0	0	0	0
90	1	1	0	0	0	1
95	1	1	0	0	0	0
100	1	2	0	1	0	1
총계	1126	1126	528	528	820	820

실험결과 대부분의 코드들 사이의 유사도는 0 ~ 20% 사이에 분포하였다. 그리고 각 문제에 대하여 표절로 의심되는 코드들을 발견할 수 있었다.

4.3 추적기법을 이용한 표절 검사의 정확성

표절검사에서 가장 중요한 것은 정확하게 표절 여부를 판단하는 것이다. 실제 표절에 대하여 표절 검사의 정확성은 판별의 양성·음성으로 나타낼 수 있다. 표 4는 표절 검사의 정확성을 분류한 것이다.

<표 4> 표절 검사의 정확성 분류

판별여부 / 표절여부	유사하지 않음	유사
표절	잘못된 음성판정 (false negative)	올바른 양성판정 (true positive)
표절하지 않음	올바른 음성판정 (true negative)	잘못된 양성판정 (false positive)

각 그룹에서 유사도가 80% 이상인 코드들에 대하여 표절여부를 조사하였다. 프로그램 추적 방법을 이용한 표절검사에서는 올바른 양성판정을 하였으나 순차적 방법에서는 잘못된 음성판정을 내렸다.

5. 결론

본 논문은 표절검사를 위한 소스코드 구조분석 기법으로 프로그램 추적 기법을 제안하였다. 프로그램 추적은 프로그램을 구문단계에서 정적으로 수행하여 그 수행되는 함수들의 순서에 따라 주요 키워드를 추출한다. 이 방법은 사용하지 않는 코드 삽입, 함수 위치 변경 및 합성 등과 같은 표절 스펙트럼에서 정의한 표절 방법에 대하여 효과적으로 표절을 검출할 수 있었다.

참고문헌

[1] B. Cheang, A. Kurnia, A. Lim and W. Oon. On automated grading of programming assignments in an academic institution. *Computers and Education*, 41:121-131, 2003.

[2] Michael J. Wise. Detection of similarities in student programs: YAP'ing may be preferable to Plague'ing. *ACM SIGSCE Bulletin(Proc of 23rd SIGCSE Technical Symp)*, 24(1):268-271, March 1992.

[3] Alex Aiken. MOSS(Measure Of Software Similarity) plagiarism detection system. http://www.cs.berkeley.edu/~moss/(as of April 2000) and personal communication, 1998. University of Berkeley, CA.

[4] Lutz Prechelt, Guido Malphol, and Michael Philipsen. Finding plagiarisms among a set of programs with JPlag. *Journal of Universal Computer Science*, 8(11):1016-1038, 2002.

[5] J. A. W. Faidhi and S. K. Robinson, An empirical approach for detecting program similarity and plagiarism within a university programming environment, *Comput. Educ.* vol. 11. pp. 11-19, 1987.

[6] 지정훈, 우균, 조환규 "제한된 프로그램 소스 집합에서 표절 탐색을 위한 적응적 알고리즘" *한국정보과학회*, 제33회 추계학술발표회 발표예정, 2006.