論文
4-10-13

프로그램 과제물의 유사도 평가 시스템 설계 및 구현
A Design and Implementation of the Similarity Evaluation System for Program Assignments

김영철(Young-Chul Kim)* / 조용윤(Yong-Yoon Cho)** /
박호병(Ho-Byung Park)***

요약

본 논문은 대학 시절에 많이 행해지는 프로그램 복제와 부분 변형에 대한 대처 방안으로 작성되었다. 대학에 만연하고 있는 복제에 대해서 강력히 대처함으로써 학생들의 학습 의욕을 고취하고 보다 깊은 프로그램의 세계로 이끌 수 있을 것이다. 본 논문은 학생들이 제출한 프로그램에 대하여 에러 유무를 가려내고, AST를 이용하여 제출된 프로그램들의 유사 정도를 평가하는 시스템이다. 또한 유사 정로를 눈으로 확인할 수 있도록 AST를 시각적으로 도식화함으로써 정밀 분석이 가능하도록 하였다. 또한 본 시스템은 가상 대학, 주문형 강의, 전자 도서관 등에서 다양한 형태로 활용될 수 있으며, 컴퓨터 교육 분야에 많은 영향을 줄 것으로 기대된다.

Abstract

This paper has been studied on the prevent of "program reproduce" and "partial modification" in computer programming. The system is made for the improvement for the prevention of prevailing "program reproduce" and "program partial modification", this research will be helpful to the student who is interested in depth study programmer. The research using AST will do the work such as finding errors of program and will evaluate how much each program is similar to each other. The research using AST will show you, on the screen, by using pictures how much each programs are assimilated. This system supports all of technique available in various forms on the areas of cyber education at present, cyber university, lecture on demand, electronic library, etc. It is expected that the new system can be contributed to opening the new chapter of computer education.

*정회원 숭실대학교 대학원 박사과정
**정회원 숭실대학교 대학원 박사과정
***정회원 숭실대학교 대학원 박사과정

논문접수 : 2003. 10. 1.
심사완료 : 2003. 10. 9.

1. 서론

학생들이 제출한 프로그램 과제물의 복제 여부를 판단하는 일은 결코 쉬운 일이 아니다. 특히, 인터넷이 활성화되면서 학생들이 많은 참고문헌을 토대로 프로그램 과제물을 제출하기 때문에 제출한 과제물이 유사한 경우도 상당히 많다. 이러한 사유로 제출된 과제물을 정확히 비교, 평가하기란 상당히 어렵다. 또한 하나의 과제물에 대한 평가를 여러 명의 강사가 검사한다면 강사들끼리의 주관적인 과제 평가 방법에 의하여 하나의 과제물에 대한 평가가 각기 다를 수 있게 된다[1,2].

하나의 과제물에 대해 복제 여부를 검사하기 위해서는 모든 다른 과제물에 대한 비교 평가를 해야만 한다. 프로그램의 양과 학생 수가 많아지면 비교평가는 더욱 많은 시간과 노력이 필요하게 된다. 또한 표절이나 특정 패턴의 과제물을 알아내기 위해서는 세밀한 작업을 요하게 된다[3,4]. 복제에 의한 과제물 수행은 요즘 문제가 되고 있는 불법 복제에 의한 소프트웨어 사용의 싹이 될 수 있으며, 이는 소프트웨어 분야 산업의 붕괴를 가져올 수 있다는 점을 주지하고 이를 사전에 방지할 수 있는 길을 모색해야할 것이며, 교육적인 측면에서도 지양되어야 할 문제로 인식되고 있다[5]. 따라서 프로그래밍 언어의 교육단계에서부터 불법 복제의 관행을 막을 수 있는 철저한 평가만이 프로그램 개발자는 물론 사용자로서 불법 복제의 심각성을 인식하여 이러한 불행한 결과를 방비할 수 있는 유일한 방법일 것이다. 본 연구에서는 이러한 프로그램 도용의 방지를 위한 복제 검색 도구를 제공하여 프로그래밍 학습을 시작하는 과정에서부터 이러한 관행이 뿌리내리지 못하도록 하는데 기여하고자 한다.

J. R. Edlun[6]은 자신의 연구 논문에서 프로그램 복제에 관하여 정의하였다. 다음은 Edlun이 정의한 문서와 프로그램 복제에 관한 인용문이다.

> 프로그램 복제란 타인의 프로그램 소스코드를 복사해서 간단한 편집과정을 거치거나 이미 만들어진 원저자의 참조 사항을 달지 않는 것을 말한다.

이처럼 복제란 타인의 글이나 프로그램 소스코드를 출처를 밝히지 않고 무단으로 도용하거나 원저자의 참조사항을 달지 않는 것을 말한다. 프로그램의 복제는 갈수록 더 다양해지고 있다. 심지어 몇몇의 학생들은 과제물을 수행하는 시간보다 프로그램 소스 코드를 편집하는 과정에 더 시간을 투자하고 있다는 기사가 나와[조선일보 2003. 2] 눈길을 끌고 있다. 이처럼 여러 가지 현실적/환경적 요인으로 볼 때 프로그램의 소스 코드의 복제 여부를 판단은 아주 중요한 요소가 되었다. 특히, 두 프로그램의 복제 여부를 자동으로 판단하는 일은 그 어느 때 보다도 절실해졌다.

본 논문은 다음과 같이 구성되었다. 제2장에서는 관련연구에 대해서 기술하였으며, 제3장에서는 프로그램 유사도 평가 시스템에 대해서 기술하였다. 또한 제4장에서는 실험 및 평가에 대해서 언급했으며, 마지막으로 제5장에서는 결론을 기술하였다.

2. 관련 연구

복제와 관련된 연구는 크게 2가지 형태로 과거 20-30년 전부터 진행되어 왔다. 초기의 복제와 관련된 연구는 문서(Document)의 복제 여부에 관한 연구가 주를 이루어 왔으며, 최근에는 이 기술을 이용하여 프로그램 소스 코드(Program Source Code) 복제에 관한 연구가 진행되고 있다. 또한 본 장에서는 두 프로그램 사이의 복제 관련 유사도와 관련된 연구를 다룬다.

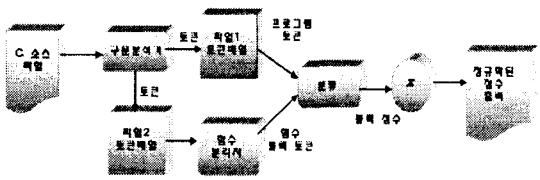
2.1 문서(Document)의 복제

일반적으로 문서 복제를 검사할 때, 구조적인 특징보다 통계적인 특징을 추출하여 이를 토대로 복제 검사를 하는 기법이 지문법(fingerprint)[7]이다. 지문법은 두 개의 문서에서 사용된 단어들의 유사성을 살피거나 사용된 단어의 빈도수 등을 비교하는 방법이다. 지문법의 특징은 문서의 통계학적인 방법을 사용하기 때문에 문서 길이에 영향을 받지 않는다는 특징이 있다. 지문법을 이용하는 시스템은 대표적으로 Plagiarism[8], IntergirGuard[9] 등이 있다.

2.2 프로그램 소스 코드의 복제

프로그램 복제 연구와 관련된 연구는 문서의 복제와 관련된 연구로 이어져 왔다. 초기의 가장 단순한 소스 코드의 복제 검사 방법은 단순 스트링 비교 방법을 사용하였다. 먼저 소스 코드에서 설명문을 제거하고 여백(White Space) 등 스타일을 변경한 후, 유닉스 명령어인 diff, grep 등을 사용하여 두 파일을 비교하는 기법이다. 또한 두 프로그램의 공통적인 단어의 개수를 얻어내어 유사성을 비교하였다. 초기에는 문서의 복제 확인 방법으로 많이 사용되는 지문법을 사용하다가 프로그램 구조의 특성을 이용해서 검사하는 방법론이 대두되었다.

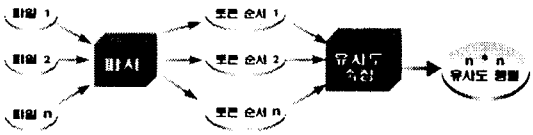
초기의 프로그램 복제 검사 방법은 검사하고자 하는 두 프로그램을 단순히 텍스트 형태로 비교하는 방법을 이용하였다. 이렇게 단순히 비교하는 검사 기법은 각각의 프로그램에 대해서 다양한 소프트웨어 특징을 계산하고, n-차원으로 매핑한다. 이러한 방법을 기반으로 만들어진 가장 초기의 시스템은 두 프로그램 사이에 유사성 정도를 검사하기 위한 Halstead 매트릭스[10] 즉, H=(μ1, μ2, N1, N2)를 이용한 시스템이다. 여기서, μ1은 유일한 오퍼레이터 수를 나타내며, μ2는 유일한 오퍼랜드 수, N1은 오퍼레이터의 출현 수, N2는 오퍼랜드의 출현 수를 나타낸다. Halstead가 제안한 매트릭스를 이용해 만든 최초의 복제 검색 시스템은 Ottenstein[11]이 개발한 시스템이다. 이 시스템은 Fortran 프로그램을 비교하기 위하여 만들어졌다. 이 시스템에서 두 프로그램이 Halstead 매트릭스인 μ1, μ2, N1, N2가 같으면, 유사한 프로그램이라 판결하고 세밀한 조사를 한다. 가장 최근에 개발된 시스템들은 토큰 스트링 등 프로그램 특성을 이용하여 평가하는 시스템이다. 이전 시스템과는 달리 토큰 스트링을 이용함으로써 프로그램 스타일이나 설명문, 들여쓰기 등의 프로그램 구문과 상관없는 요소에 민감하지 않다는 특징을 가지고 있다. 프로그램의 토큰을 이용하여 복제 검사를 하는 시스템은 대표적으로 YAP3[12], MOSS[13], Jplag[14] 등이 있으며, 복제 검사에 관련된 연구는 David Gitchell이 개발한 Sim(software Similarity tester)[15]과 Xin Chen 등이 개발한 SID[16]가 있다. Sim은 C++와 Tcl/Tk로 구현되었으며, 서로 다른 두개의 C 프로그램을 비교하며, 소스 코드에 있는 토큰들이 다른 곳에서 참조되는 횟수를 계산해서 히스토그램을 만든다. 그리고 이 히스토그램을 비교해서 복제 여부를 확인한다. 그림 1은 sim의 모델을 보여준다.



[그림 1] sim 시스템 모델
[figure 1] sim system model

SID(Software Integrity Diagnosis system)는 Kolmogorov 복잡도를 기반으로 하고 있다. SID은 검사하고자 하는 두 프로그램의 "공유정보"를 이용하여 유사도를 측정한다는 점이다. 다음 그림 2는 SID의 블록 다이어그램을

보여준다. SID 역시 sim과 마찬가지로 0~1 사이의 유사도 값을 나타낸다.



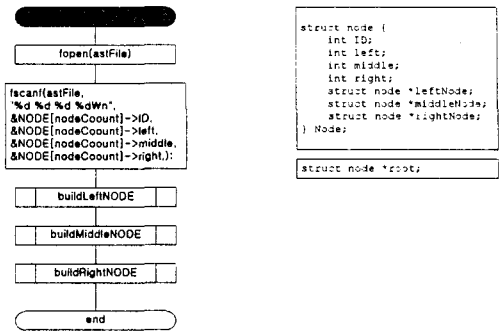
[그림 2] SID 시스템 모델
[figure 2] SID system model

3. 프로그램 유사도 평가 시스템

학생들이 제출한 프로그램은 JLex[17]와 JavaCUP[18]에 의하여 토큰화와 문법 검사를 수행하게 되고 이 과정에서 AST가 생성된다. 생성된 AST는 유사도 평가 시스템에 의하여 복제 검사를 하게 되며 유사 정도에 따라 강한, 중간, 약한 유사로 나뉘어 지게 된다. 약한 유사는 통과되어 처리되지만 중간 이상의 유사도는 복제했다고 판단되므로 개별적으로 다시 검사하거나 AST 트리를 정밀 검사하게 된다. 만약 문법 에러가 있는 프로그램이라면 AST가 생성되지 못하므로 프로그램 오류를 검사할 수 있다.

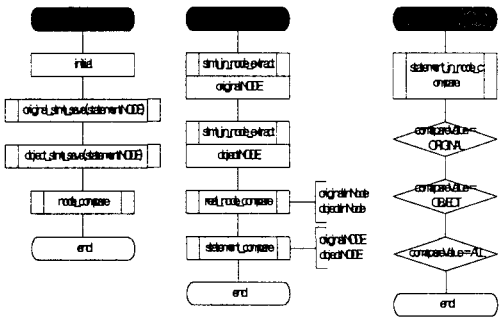
3.1 AST의 설계 및 비교

다음 그림 3은 AST를 생성하기 위한 구조와 AST 노드의 자료구조를 보여준다. 그림에서처럼 buildAST 함수는 노드를 트리 형태로 만들어 주는 함수로, 인자로 파일의 이름을 받아 파일 안에 있는 AST 자료들을 모아서 트리를 형성한다. 그림에서 보여주듯이 트리는 ID 정수 값, 왼쪽, 중간, 오른쪽을 나타내는 정수 값, 왼쪽, 중간, 오른쪽을 가리키는 노드 포인터들로 구성되어 있다.



[그림 3] AST 노드의 자료구조
[figure 3] data structure of AST node

이처럼 생성된 AST는 다음 그림 4에서처럼 ast_compare 함수에 의해 비교된다. ast_compare 함수는 인자로 받은 원본 노드들과 대상 노드들을 비교하는 함수 이다. find_statement()라는 함수를 이용해 statement 들을 저장하고 나중에 statement 만을 비교하거나 statement 안의 내용을 비교하기 위해 사용한다.



[그림 4] AST 노드 평가 개략도
[figure 4] AST node evaluation framework

3.2 유사도 평가 알고리즘

본 논문에서 이용된 유사도 평가 알고리즘은 Ira D.B axter[3]에서 제시된 알고리즘을 이용

하여 유사도를 평가한다. Ira D.B axter의 연구에서는 중복된 코드를 "clone"이라 칭하였으며, 중복된 코드를 통하여 유사도를 평가하는 방법을 제시하였다. 이 연구에서 제시한 유사도 평가 수식은 다음과 같다.

$$\text{유사도(Similarity)} = 2 * \frac{S}{2*S + L + R}$$

여기서, S는 공유 노드의 수를 나타내며, L은 원본 소스에 있는 트리 중 대상 소스에 없는 노드의 수를 나타낸다. 또한 R은 다른 소스에 있는 트리 중 원본 소스와 다른 노드의 수를 나타낸다. Ira D.B axter가 제시한 중복 코드 발견 시스템의 기본 알고리즘은 다음과 같다.

```
1. Clones = ∅;
2. for each subtree i;
      if mass(i) >= MassThreshold then
hash i to bucket;
3. for each subtree i, j in same bucket
  if      compareTree(i,      j)      >
similarityThreshold          { for each
subtree s of i
      if isMember(Clones, s)
         RemoveClonePair(Clones, s);
         for each subtree s of j
         if isMember(Clones, s)
               RemoveClonePair(Clones,
s);
               AddClonePair(Clones, i, j);
         end for
      end for
   }
end for
```

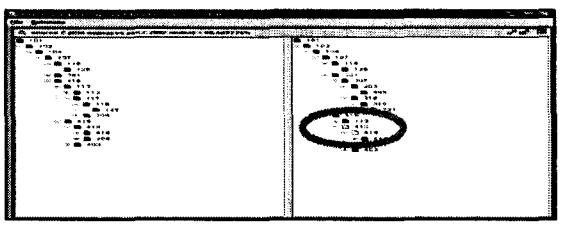
알고리즘에서 두 트리가 같으면, "clone"으로 취급하여 유사도에 이용된다. 본 논문에서는 clone에 해당되는 노드를 찾아내어 유사도를 측정한다. 또한 AST를 시각화하여 "clone"에 해당되는 코드 조각을 찾아낸다. 자세한 실험은 제 4장 실험 환경 및 구현에서 설명한다.

4. 실험 및 평가

본 시스템에서는 프로그램 유사도 평가를 수행하기 위하여 1989년에 제정된 ANSI C 언어로 작성된 프로그램을 대상으로 하며, 다양한 시스템에서 활용하기 위하여 Java 언어로 구현하였다. 또한 C 언어의 어휘 분석 및 구문 분석을 수행하기 위하여 컴파일러 보조도구인 JLex[17]와 JavaCUP[18]을 이용하였다. 또한 본 논문의 실험을 위해서 부록 A와 같은 3개의 프로그램이 있다고 가정한다.

4.1 실험

다음 그림 5는 부록 A에 있는 source.C와 sim.C 프로그램을 비교한 평가 결과를 보여준다. 그림에서 노드에 붉은(■)표시가 되어 것은 두 트리에서 공통된 노드에 해당한다. 즉, 두 프로그램에서 AST의 노드가 일치하였을 때를 나타내는 부분으로 이전 장에서 언급한 "clones"에 해당된다. 따라서 그림에서처럼 두 프로그램은 거의 대부분이 일치된다고 볼 수 있으며, 부분적으로 다른 노드가 나타남을 알 수 있다. 이것은 부분적으로 코드 조각(code fragment)이 다르다는 것을 나타낸다.



[그림 5] 프로그램 source.C와 sim.C의 유사도 평가 결과

[figure 5] similarity evaluation result of program source.C and sim.C

위의 그림에서 보여준 것처럼 두 프로그램의 유사도 평가는 다음과 같이 이루어졌다..

$$\text{유사도(Similarity)} = 2 * \frac{S}{2*S + L + R}$$

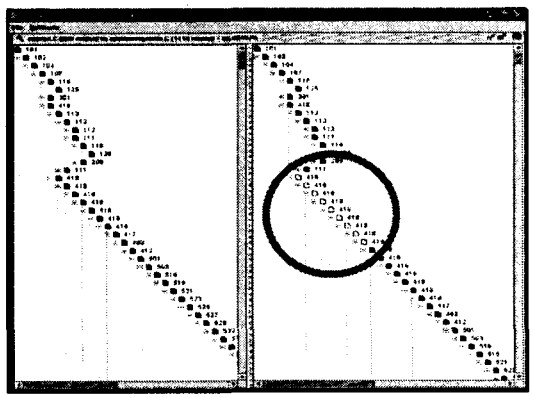
$$= 2 * \frac{828}{2*828 + 0 + 53} = 0.96563774$$

이 유사도의 의미는 복제 검사 대상(sim.C) 프로그램이 원본 프로그램(source.C)과 거의 같음을 의미한다.

4.2 평가

AST를 이용하여 유사도를 측정하는 방법은 다음과 같이 다양한 장점을 가지고 있다. 첫째, AST는 방대한 프로그램에서 작은 프로그램까지 유사도 검사에 적용될 수 있다. 둘째, 복제를 한 당사자가 약간의 프로그램을 수정했을 경우에도 유사도 측정이 가능하다. 셋째, 컴퓨터가 정확히 판별하기 어려운 프로그램도 검사자에 의해서 유사도 측정이 가능하다. 넷째, AST는 프로그램에 오류가 있는 경우 트리를 생성할 수 없으므로 에러를 쉽게 판별해 낼 수 있다. 다섯째, AST는 함수 각각에 대해서도 유사도를 검사할 수 있다. 여섯째, 복제를 한 당사자가 프로그램의 변수나, 함수 이름, 변수에 할당된 값 등을 바꾸었을 경우 유사도를 정확히 판별해 낼 수 있다. 일곱째, while이나 if, switch문 등의 안에 복제자가 간단한 변수나, 수치를 추가했을 경우 AST에 대한 전체적인 트리는 영향을 받지 않으므로 유사도를 검사할 수 있다. 여덟째, 함수나 순환문, if then, switch 등의 순서를 바꾸었을 경우, 즉 문장의 순서를 바꾸었을 경우에도 쉽게 유사도를 검사할 수 있다.

이러한 장점과는 달리 본 시스템은 단점도 가지고 있다. 즉, 프로그램 복제 대상 프로그램에 여분 코드(dummy code)가 많이 있다면 본 시스템의 유사도는 현저하게 떨어질 수도 있다. 예를 들면, 부록 A의 source.C 파일과 adddummycode.C를 비교해보자. 먼저 두 프로그램의 비교 후의 AST 형태는 다음 그림 6과 같다.



[그림 6] 프로그램 source.C와 adddummycode.C의 유사도 평가 결과

그림 6에서 두 프로그램의 비교 결과 유사도 값이 적게 나온 이유는 여분 코드(dummy code)가 많이 들어갔기 때문이다. 이 두 프로그램의 유사도 평가는 다음과 같이 이루어진다.

$$\text{유사도(Similarity)} = 2 * \frac{S}{2*S + L + R}$$

$$= 2 * \frac{835}{2*835 + 0 + 279} = 0.8568497$$

위의 실험에서 두 프로그램은 똑같은 결과를 가지고 있다. 하지만 실험에서 나타나듯이 복제 대상 프로그램(adddummycode.C)에 쓸모없는 여분 코드를 삽입함으로써 본 시스템은 중간 유사(유사도 0.7 ~ 0.9) 값을 나타내는 85.68497%의 유사도를 나타냈다. 이것은 본 시스템이 여분의 코드를 삽입하였을 경우에는 취약점을 드러내고 있음을 의미한다. 이러한 단점을 보완하기 위해서는 여분 코드 검사기

(dummy checker) 및 코드 최적화기(code optimizer)가 필요하다. 즉, 본 논문에서 제시한 프로그램 유형 복제 검사를 수행하기에 앞서 여분 코드 검사기 및 코드 최적화기를 수행시키면 여분의 코드나 불필요한 코드를 여과(filtering)해주면 정상적으로 수행할 수 있다. 이러한 여분 코드 검사기나 코드 최적화기는 또 다른 하나의 큰 연구 과제이므로 본 논문에서는 취급하지 않으며, 향후 연구과제로 남긴다.

5. 결론

본 논문에서는 AST를 이용하여 서로 다른 두 프로그램의 유사도를 검사하는 프로그램 유사도 평가시스템을 제시하고 구현하였다. AST를 이용한 프로그램 유사도 검사는 프로그램의 구조적인 방법으로 유사도를 검사할 수 있을 뿐 아니라 어휘 분석과 구문 분석을 수행하여 구문 오류를 검사할 수도 있었다. 또한 여러 형태의 복제 유형에 대해서 효율적으로 복제 검사를 할 수 있다는 것을 보여주었다.

또한 본 논문에서는 프로그램 유사도 평가 시스템을 구현하기 위하여 [3]에서 제시한 알고리즘을 이용하였으며, AST를 시각화하여 보여주었다. 평가 부분에서는 AST를 이용하여 프로그램 유형 복제 검사를 함으로써 얻는 장점과 단점에 대해서 언급했다. 특히 본 시스템은 많은 장점을 가지고 있으나 취약성도 가지고 있음을 논의하였다. 즉, 본 시스템은 불필요한 변수나 문장 등을 추가한 코드(dummy code)에 문제점을 드러내고 있다. 따라서 이 문제를 해결하기 위해서는 소프트웨어 공학 분야에서 연구되고 있는 코드 최적화기나 여분 검사기(dummy checker)를 수행함으로써 해결할 수 있다는 것을 논의하였다.

본 연구의 향후 과제로는 이전에 언급한 여분 검사기 외에도 수행 속도에 관한 연구가 필요하다. 본 시스템을 실험한 결과 약 1000 라인 이상인 프로그램에 대해서는 속도가 현저하게 떨어진다는 것을 알 수 있었다. 따라서 알고리즘의 속도 향상 측면에서 연구가 이루어져야 할 것이다. 이외에도 XML 문서의 유사도 측정, 코드 생성 및 최적화를 이용한 유사도 검사, 웹 기반의 프로그램 복제 검사 시스템 등에 활용할 수 있을 것으로 기대된다.

참고 문헌

[1] J. K. Harris, "Plagiarism in Computer Science Courses", In proc. Ethics in Computer Age, pp. 133-135, 1994.

[2] M. Joy & M. Luck, "Plagiarism in Programming Assignments", IEEE Transaction in Education, 42(2), pp. 129-133. 1999.

[3] I. D. Baxter, A. Yahin, L. Moura, M. Sant'Anna & L. Bier, "Clone Detection using Abstract Syntax Trees", In proc. of the international Conference on Software Maintenance, Bethesda, Maryland, pp. 368-378, Nov., 1998.

[4] J. Carter, "Collaboration or Plagiarism: What Happens when Students Work Together?", In proc. ITiCSE, pp. 52-55, Cracow, Poland, 1999.

[5] R. Irving, "Plagiarism Detection: Experiences and Issues", Presented at JISC Fifth Information Strategies Conference: Focus on Access and Security, London. 2000.

[6] J. R. Edlun, "What is "Plagiarism" and why do people do it?", available at http://www.calstatela.edu/centers/write_cn/plagiarism.htm, University Writing Centre Director, California State University, LA, 1998.

[7] J. H. Jonson, "Identifying Redundancy in Source Code using Fingerprints". In

proc. of CASCON 93, pp. 171-183, 1993.
[8]Plagiarism Site available at http://www.plagiarism.org.
[9]Integriguard Site available at http://www.integriguard.com.
[10] M. Howard & Halstead, Elements of Software Science, Elsevier, 1977.
[11] K. J. Ottenstein, "an Algorithmic Approach to the Detection and Prevention of Plagiarism". ACM SIGSCE Bulletin, 8(4), pp. 30-41, 1976.
[12] M. 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), pp. 268-271, Mar., 1992.
[13] A. Aiken, "MOSS(Measure Of Software Similarity) Plagiarism detection system", available at http://www.cs.berkeley.edu/~moss/, University of Berkeley, CA, Apr., 2000.
[14] L. Prechelt, G. Malpohl & M. Philppsen, "JPlag: Finding Plagiarism Among a Set of Programs", available at http://wwwipd.ira.uka.de/EIR/ D-76128 Karlsruhe, Germany, Technical Report 2000-1, Mar., 2000.
[15] D. Gitchell & N. Tran, "Sim: A Utility For Detecting Similarity in Computer Programs", available at ftp://ftp.cs.vu.nl/pub/dick/similarity_tester/, In proc. of 30th SCGCSE Technical Symp., pp. 266-270, New Orleans, USA, 1998.
[16] X. Chen, M. Li, B. Mckinnon & A. Seker, "A Theory of Uncheatable Program Plagiarism Detection and Its Practical Implementation", University of California, SantaBarbara, May., 2002.
[17] J. Lin & JLex Tutorial, available at http://bmrc.berkeley.edu/courseware/cs164/spring98/ proj/jlex/ tutorial.html.
[18] S. E. Hudson, "CUP Parser Generator for Java", available at http://www.cs.princeton.edy/~appel/modern/java/CUP/.

부록 A. 실험에 이용된 프로그램 데이터

1) source.C

```
#include <stdio.h>
void main(void)
{
  long result;
  int n1, n2, m1, m2, divs[100], lnum, i, flag;
  int index=0;
  printf("Input 2 numbers for calculating GCM...\n");
  scanf("%d %d", &n1, &n2);
  m1= n1; m2 = n2;
  while(1) {
    lnum = (m1>=m2?m1:m2);
    flag = 0;
    for(i=2; i<=lnum; i++) {
      if(m1%i==0 && m2%i==0) {
        m1/=i; m2/=i;  divs[index++]=i;
        flag=1; break;
      }
    }
    if(flag==0) break;
  }
  result = m1 * m2;
  for(i=0; i<index; i++) result *= divs[i];
  printf("LCM of %d and %d is %ld.\n", n1, n2, result);
}
```

2) sim.C

```
#include <stdio.h>
void main(void) {  /* 스타일 변화 */
  long result;
  int index=0; /* 문장 위치 바꿈 */
  int n1, n2, m1, m2, divs[100], lnum, i, flag;
  printf("Input 2 numbers for calculating
GCM...\n");
  scanf("%d %d", &n1, &n2); m1= n1; m2 =
n2;
  for(;;) {/* 제어 구조 변환 while ==> for */
    flag = 0;
    lnum = (m1>=m2?m1:m2); /* 문장 바꿈
*/
    for(i=2; i<=lnum; i++) {/* 스타일바꾸기
*/
      if(m1%i==0 && m2%i==0) {
        m1/=i; m2/=i; divs[index++]=i;
        flag=1; break;
      }
    }
    if(flag==0) break;
  }
  result = m2 * m1;  /* 오퍼랜드 바꾸기 */
  result = m2 * m1;  /* 중복 코드 추가 */
  result = m2 * m1;  /* 중복 코드 추가 */
  for(i=0; i<index; i++) result *= divs[i];
  printf("LCM of %d and %d is %ld.\n", n1,
n2, result);
}
```

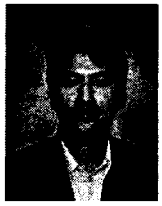
3) adddummycode.C

```
#include <stdio.h>
void main(void) {
  long result;
  int n1, n2, m1, m2, divs[100], lnum, i, flag;
  int index=0;
  printf("Input 2 numbers for calculating
GCM...\n");
  scanf("%d %d", &n1, &n2);
  m1= n1; m2 = n2;
  while(1) {
    lnum = (m1>=m2?m1:m2);
    flag = 0;
    for(i=2; i<=lnum; i++) {
      if(m1%i==0 && m2%i==0) {
        m1/=i; m2/=i;
        divs[index++]=i;
        flag=1;
        break;
      }
    }
    if(flag==0) break;
  }
  result = m1 * m2;
  result = m1 * m2;/* insert to dummycode
*/
  result = m1 * m2;/* insert to dummycode
*/
  result = m1 * m2;/* insert to dummycode
*/
  result = m1 * m2;/* insert to dummycode
*/
  result = m1 * m2;/* insert to dummycode
*/
  result = m1 * m2;/* insert to dummycode
*/
  for(i=0; i<index; i++) result *= divs[i];
  printf("LCM of %d and %d is %ld.\n", n1,
n2, result);
  m1 = m2;  /* insert to useless code */
  m1 = m2;  /* insert to useless code */
  m1 = m2;  /* insert to useless code */
}
```

김영철

1990년 한남대학교 전자계산학과 졸업(학사)
1996년 숭실대학교 대학원 전자계산학과 졸업(석사)
2002~현재 숭실대학교 박사과정, (주)뉴스텍 시스템즈 이사, 명지전문대학 겸임조교수
관심분야 : (웹)프로그래밍 언어, 컴파일러, 컴퓨터 통신, XML, 망관리

조 용 윤



1995년 시립 인천대학교 전자계산과 졸업(학사)
1998년 숭실대학교 대학원 전자계산학과 졸업(석사)
1999~현재 숭실대학교 박사과정
관심분야 : 프로그래밍 언어, 컴파일러, XML, HCI

박 호 병

1999년 숭실대학교 전자계산학과 졸업(학사)
2002년 숭실대학교 컴퓨터학과 졸업(석사)
2002~현재 숭실대학교 박사과정
관심분야 : 프로그래밍 언어, XML, 구문지향편집기