김일성종합대학학보
(자연과학)
주체105(2016)년 제62권 제5호

JOURNAL OF **KIM IL SUNG** UNIVERSITY
(NATURAL SCIENCE)
Vol. 62 No. 5 JUCHE105(2016).

프로그람작성능력평가를 위한 문제류형과 자동채점의 한가지 방법

정만흥, 김예화

콤퓨터에 의한 자동시험채점의 연구는 많이 진행되였으나 프로그람작성문제의 자동채점법은 잘 알려져있지 않다. 현재 프로그람작성의 자동평가에서 리용되는 기술은 대다수 번역체계에 의존한다. 이 방법은 정확한 번역의 전제하에서 평가하려는 프로그람을 실행시키고 여러번의 입력자료에 따라 출력되는 결과가 예상결과에 맞는가 맞지 않는가를 검사하고 그 정도를 판정하는것에 의해 점수를 채점하는 방법이다.

프로그람이 반드시 실행되여야 그 정확성을 평가할수 있는 시험채점방법의 결함을 극복하기 위한 방법으로서 Levenshtein편집거리법[1, 2, 4]과 정규식작성법[3] 등을 들수 있다. 그러나 사람마다 프로그람작성방법이 서로 다르고 프로그람에서 리용되는 변수, 명령문, 함수 등의 리용이 다양한것으로 하여 선행연구[4]에서 제기한 방법은 많은 목표프로그람들을 작성하여야 한다. 한편 선행연구[3]에서 제기한 정규식작성법은 많은 목표프로그람을 작성해야 하는 부족점은 극복하였지만 번역기에 의존하는 정규문법에 따르는 개별적인 프로그람들에 대한 정규식을 작성하여야 하는데 정규식의 작성 역시 쉽지 않다.

론문에서는 프로그람작성능력이자 프로그람오유수정능력이라는 사실에 기초하여 프로그람작성능력평가를 위한 5가지 문제류형을 제기하고 본문류사도계산에 의해 프로그람작성능력에 대한 자동채점방법을 론의한다.

1. 프로그람작성능력자동평가를 위한 문제류형

프로그람의 오유에는 문법오유, 실행오유 그리고 의미적오유가 있다는것을 고려하여 다음과 같은 5가지 류형의 능력평가프로그람들을 제시하고 수정하도록 한다.

① 문법오유수정문제

프로그람작성에서 가장 많이 범하게 되는 오유는 문법오유이다. 프로그람은 문법이 정확할 때에만 실행될수 있으며 프로그람코드에 오유가 있으면 프로그람해석기는 오유통보를 현시한다. 프로그람문법은 프로그람의 구조와 그 규칙에 관계된다.

② 실행오유수정문제

두번째 형태의 오유가 실행오유이다. 프로그람이 실행되기까지 오유가 나타나지 않는다고 하여 실행오유라고 부른다. 때때로 례외적인 현상이 나타난다고 하여 실행오유를 또한 례외적오유라고도 한다.

실례로 실행오유는 타당하지 않은 연산 등을 진행할 때 나타날수 있으며 례외적오유는 기억기관리 등과 같은 규칙을 바로 지키지 못하여 생길수 있다.

실행오유의 한가지 실례코드를 보자.

```
import math
signal_power = 9
noise_power = 10
ratio = signal_power / noise_power
decibels = 10 * math.log10(ratio)
print decibels
```

이 코드는 Python언어를 리용하여 음성 및 화상처리에서 자주 보게 되는 신호대잡음비

$$SNR_{db} = 10\ \log_{10}(P_{signal}/P_{noise})$$

를 계산하는 프로그람코드이다.

프로그람코드는 문법적으로 오유가 없지만 Python 2에서 실행시키면 다음과 같은 오유통보가 현시된다.

```
Traceback (most recent call last):
File "snr.py", line 5, in ?
decibels = 10 * math.log10(ratio)
OverflowError: math range error
```

오유통보는 5번째 행을 가리킨다. 그러나 그 행에는 아무러한 오유도 존재하지 않는다. 실제적인 오유를 찾기 위해 ratio의 값을 출력해보면 출력값은 0이다. 결국 오유는 4번째 행에 있다. 그것은 실례코드의 4번째 행에서 옹근수나누기를 진행하였기때문이다. 오유를 수정하자면 코드의 4행에서의 나누기를 류동소수점나누기로 수정해야 한다. 이와 같이 실행오유를 능숙하게 찾아 수정하자면 해당 프로그람에 대한 문법규칙을 잘 알아야 할뿐아니라 해당 문법의 의미를 잘 아는것이 중요하다. 이것은 프로그람작성능력을 높이는데서 중요한 역할을 한다.

일반적으로 오유통보는 오유를 가지고있는 코드행을 지적하는것이 아니라 자주 오유가 없는 코드행을 지적하게 된다.

③ 의미적오유수정문제

세번째 형태의 오유는 의미적오유이다. 만일 프로그람에 의미적오유가 있다면 콤퓨터는 어떠한 오유통보도 현시하지 않고 프로그람을 성과적으로 실행시킨다. 그러나 실행결과는 옳은 결과가 아니다. 이때의 프로그람은 우리가 작성하려는 프로그람이 아니라는것을 의미한다. 즉 프로그람의 의미가 달라진것이다. 프로그람의 실행결과를 보고 현재 프로그람이 무엇을 수행하고있는가를 료해하고 프로그람을 수정하여야 하는것만큼 의미적오유를 발견하는것은 일반적으로 능란한 수완을 요구한다.

Python프로그람에서 큰 옹근수를 입력할 때 1,000,000와 같이 보통 세자리수를 단위로 반점을 리용하여 입력한다고 하자. 이때 Python해석기는 1,000,000을 반점으로 분리된 옹근수들의 렬로 리해한다. 즉

```
>>> 1,000,000
(1, 0, 0).
```

이 코드는 오유통보없이 실행되지만 옳은것은 아니다. 이외에 의미적오유는 연산자 +를 연산자 –로, 연산자 *를 연산자 /로 잘못 입력할 때에도 발생하게 된다. 그러나 이런

경우는 일반적으로 문제풀이의 알고리듬을 잘못 리해하여 생기는 의미적오유이다.

④ 프로그람변경문제

어떤 문제를 푸는 완성된 프로그람을 제시하고 이 프로그람과 류사한 다른 문제를 푸는 프로그람으로 변경시키는 문제이다.

실례로 입력된 수렬에 대한 짝수값원소들의 합을 구하는 프로그람이 제시될 때 그 프로그람을 홀수값원소들의 합을 구하는 프로그람으로 변경시키는 문제를 생각할수 있다. 우리는 이와 같은 류형의 문제를 프로그람변경문제(알고리듬파악문제)로 정의한다.

⑤ 코드보충문제

이 류형의 문제는 현재 많이 리용되고있는 시험평가체계들에서 제시하고있는 문장채우기문제와 같다. 실례로 100이하의 정의 옹근수가운데서 5와 7로 동시에 나누어지는 수들의 개수를 구하는 프로그람에 대하여 아래와 같은 완성된 프로그람을 보자.

```
int k; count=0;
    for( k=1; k<=100; k++ )
  { [……]
        count=++ }
    print( "개수=%d", count );
```

이때 프로그람의 빈 자리 [……]에 코드 if(k%5==0 && k%7==0)을 써놓으면 정답으로 되는데 이것을 코드보충문제라고 한다. 선행한 시험평가체계에서는 단순히 학생이 입력한 코드가 대답자료기지에 들어있는 정확한 코드와 일치하는가에 따라 점수평가를 진행하고있다.

5가지 류형의 문제중에서 프로그람변경문제와 코드보충문제는 프로그람작성능력을 가지고있다 할지라도 문제를 풀기 위한 알고리듬을 잘 모르면 완성된 프로그람을 작성할수 없다는 리유로부터 출발하여 제기한 문제류형이다.

2. 프로그람류사도평가방법

프로그람을 인공언어로 작성된 본문으로 리해할수 있다. 그러므로 본문류사도평가기술을 리용하여 프로그람들사이의 류사도를 평가할수 있으며 류사도값에 따라 프로그람작성능력의 자동평가를 진행할수 있다.

본문류사도는 2개이상의 본문들사이의 정합정도를 표시하는 척도파라메터이다. 정합성정도가 크다는것은 두 본문사이의 류사도가 높다는것을 의미하며 반대로 정합성정도가 작다는것은 두 본문사이의 류사도가 낮다는것을 의미한다.

본문류사도의 계산은 본문정보에 대한 문헌검색, 본문무리짓기 그리고 본문분류 등의 처리들에서 가장 많이 응용되고있다.

전통적인 본문류사도비교방법으로는 주로 벡토르공간모형(VSM)에 기초한 방법을 들수 있는데 벡토르공간모형을 리용하는 경우 본문에 대한 품사처리와 통계처리를 진행하여야 한다. 따라서 이 방법은 산법이 비교적 복잡한 부족점을 가질뿐아니라 본문의 아래우문맥의 의미관계를 고려하지 못하는 결함을 가지고있다.

특히 프로그람본문은 일반본문과 달리 특수한 문법과 구조화의 특징을 가지고있으므로 벡토르공간모형을 리용하여 프로그람들사이의 류사도를 평가하는 문제는 매우 어려운 문제이다.

그러므로 우리는 프로그람본문류사도를 평가하기 위해 편집거리에 의한 프로그람작성문제의 자동채점방법을 제기한다.

Levenshtein거리는 문자렬류사도비교에 자주 리용되는 거리이다. 이 거리는 입력된 문자렬을 비교하려는 문자렬로 변환하는데 최소 얼마만한 문자를 추가 및 삭제 그리고 수정해야 하는가에 따르는 회수에 의해 계산된다.

문자의 추가, 수정 및 삭제의 개념은 문서편집과정에 쓰이는 개념이므로 Levenshtein거리를 Levenshtein편집거리라고 부르게 된다.

Levenshtein편집거리에 의한 두 문자렬사이의 류사도평가는 벡토르공간모형에 기초한 방법에 비해 산법이 상대적으로 간단하며 일정한 정도의 문법구조와 순서관계를 고려한다.

두 문자렬의 순서를 바꾸는 경우 일반적으로 벡토르공간모형에 기초한 방법에 의해 계산된 류사도값은 변하지 않지만 Levenshtein편집거리에 의해 계산된 류사도값은 변하게 된다. 이 사실은 Levenshtein편집거리에 의한 방법이 벡토르공간모형에 기초한 방법에 비해 프로그람의 류사도평가의 요구에 보다 더 부합된다는것을 의미한다.

Levenshtein편집거리의 또 다른 하나의 우점은 정합오차가 있는 단어에 대하여 정합오차의 정도에 따라 오차가 없는 단어와의 류사도를 얻을수 있다는것이다. 이것은 프로그람에 대한 평가를 보다 정확하게 평가할수 있게 한다.

여기로부터 론문에서는 문법오유, 실행오유 및 의미오유를 포함하는 프로그람을 비롯하여 다양한 류형의 능력평가프로그람들을 학생들에게 제시하고 수정하도록 하며 수정한 프로그람들과 정답(원천)프로그람사이의 Levenshtein편집거리에 의해 프로그람코드의 류사도를 평가하는 방법으로 프로그람작성능력평가를 위한 자동평가방법을 론의한다.

프로그람작성능력을 평가하기 위한 절차는 다음과 같다.

① 프로그람작성능력을 평가받으려는 대상자(학생)에게 능력평가프로그람 p_t를 제시한다.

② 프로그람작성능력을 평가받으려는 대상자(학생)는 프로그람 p_t에 포함되여있는 오유를 능력껏 수정하여 수정프로그람 p_c를 얻는다.

③ 수정프로그람 p_c와 대응하는 원천프로그람 p_s의 류사도를 계산하여 해당 문제에 대한 점수평가를 진행한다.

점수평가는 다음과 같이 한다.

우선 능력평가프로그람과 원천프로그람사이, 수정프로그람과 원천프로그람사이의 Levenshtein편집거리를 각각 계산한다. 즉

$$L_{ts}=L(p_t,\ p_s),\ L_{cs}=L(p_c,\ p_s).$$

다음 편집거리를 리용하여 수정프로그람 p_c와 원천프로그람 p_s사이의 류사도를 다음의 식에 의해 계산한다.

$$\mathrm{sim}D(p_c,\ p_s)=1-\frac{L_{cs}}{L_{ts}+L_{cs}} \tag{1}$$

이때 L_{cs}가 L_{ts}와 같은 경우는 한 문자도 수정하지 못한 경우이며 L_{cs}가 L_{ts} 보다 큰 경우는 오유가 없는 코드부분도 수정한 경우이다. 따라서 L_{cs}가 L_{ts}보다 크거나 같을 때에는 $\mathrm{sim}D(p_c,\ p_s)=0$으로 한다. 분명히 프로그람에 포함되여있는 오유를 모두 정확히 수정하였을 때에는 L_{cs}= 0이므로 $\mathrm{sim}D(p_c,\ p_s)=1$로 된다.

끝으로 5점채점법에 따라 점수평가를 하는 경우 류사도에 5를 곱한것을 점수로 한다. 즉 FM을 평가점수라고 할 때

$$\mathrm{FM}=5\times\mathrm{sim}D(p_c,\ p_s) \tag{2}$$

와 같다.

한편 Levenshtein편집거리는 다음과 같이 계산한다.

계산하려는 두 문자렬

$$S=s_1s_2\cdots s_n,\ T=\ t_1t_2\cdots t_m$$

이 주어졌다고 하자. 여기서 s_j와 t_j들은 각각 문자렬 S와 T를 구성하고있는 문자들이다.

이때 두 문자렬 S와 T에 대하여 다음과 같은 $(n+1)\times(m+1)$형 정합관계행렬 D를 만든다.

$$D=\begin{pmatrix} d_{00} & d_{01} & \cdots & d_{0m} \\ d_{10} & d_{11} & \cdots & d_{1m} \\ \vdots & \vdots & \ddots & \vdots \\ d_{n0} & d_{n1} & \cdots & d_{nm} \end{pmatrix}$$

여기서

$$d_{ij}=\begin{cases} i\ , & j=0 \\ j, & i=0 \\ \min\{d_{(i-1)j}+1,\ d_{i(j-1)}+1,\ d_{(i-1)(j-1)}+\alpha(i,\ j)\}, & i,\ \ j>0 \end{cases}$$

이며

$$\alpha(i,\ j)=\begin{cases} 0 & s_i=t_j \\ 1 & s_i\neq t_j \end{cases} \qquad (i=1,\ 2,\ \cdots,\ n, j=1,\ 2,\ \cdots,\ m)$$

이다.

이러한 행렬 D의 오른쪽 아래원소 d_{nm}이 바로 문자렬 S와 T의 Levenshtein편집거리이다.

3. 실 험 분 석

이미 교원들에 의해 최우등, 우등, 보통, 락제생으로 평가된 학생들을 대상으로 제안한 학생들의 프로그람작성능력평가체계의 정확성을 대비분석하였다. 대비분석에 리용된 방법은 선행한 시험체계[1]이다.

선행한 시험체계[1]는 프로그람작성능력평가를 위해 프로그람의 여러 개소에 오유코드 또는 빈칸코드를 주고 학생들이 수정 또는 보충한 코드를 미리 구축된 정답코드와 비교하고 정확히 비교된 코드의 개수를 리용하여 점수를 평가하는 방법이다. 대비분석결과는 표와 같다.

표. 교원채점결과

번호	방법	최우등(5)	우등(4)	보통(3)	락제(2)
1	교원채점	9	23	6	2
2	선행체계	11	26	3	0
3	론문방법	8	22	7	3

표에서 최우등생 9명, 우등생 23명, 보통생 6명, 락제생 2명이라는 결과는 5명의 교원들의 채점성적에 대한 평균점수결과이다.

표에서 보는것처럼 론문에서 제안한 방법이 선행한 시험체계에서보다 교원의 채점결과와 류사하다는것을 보여준다.

맺 는 말

프로그람작성능력의 평가를 위한 5가지 프로그람문제류형과 Levenshtein편집거리를 리용한 프로그람류사도계산식을 제기하고 학생들의 프로그람작성 및 분석능력을 자동평가하기 위한 방법을 론의하였으며 그 효과성을 검증하였다.

참 고 문 헌

[1] 리충건 등; 정보기술통보, **2**, 12, 주체103(2014).
[2] 吉胜军; 电脑知识与技术, **5**, 9, 2177, 2009.
[3] 余石泉 等; 计算机机技与发展, **17**, 7, 244, 2007.
[4] 周汉平; 计算机应用与软件, **28**, 5, 209, 2011.

주체105(2016)년 1월 5일 원고접수

A Method on the Problem Types for Estimating Programming Capacity and an Automatic Marking

Jong Man Hung, Kim Ye Hwa

We proposed five types of problems for estimating programming capacity and considered a method of automatic marking based on Levenshtein edit-distance.

Key words: automatic marking, e-examination