A TOOL THAT DETECTS PLAGIARISM IN PASCAL PROGRAMS

Sam Grier
Department of Astronautics and Computer Science
USAF ACADEMY
USAFA CO 80840

Plagiarism has become a problem in introductory Computer Science courses. Programmed assignments can be copied and transformed with little human effort. A pertinent recommendation has resulted from this realization; an on-line system to detect programs that are "too similar" and hence suspected of plagiarism should be developed [4]. This paper discusses such a system for Pascal programs.

1. INTRODUCTION

As noted in recent literature, plagiarism has become a problem in introductory Computer Science courses [4]. To put it succinctly, students are copying other students' programs.

Detecting this plagriarism is difficult. Not only must graders grade a large volume of programs, but these programs all solve the same problem. Sophisticated plagiarism is not the problem; the sheer volume of code involved is simply overwhelming.

One attempted solution to this problem has been the development of a program at Purdue University by K.J. Ottenstein that quantifies the sameness of Fortran programs [3]. This program utilizes the four basic Software Science parameters suggested by M. Halstead as useful measures of program length [2]. This program utilizes only these parameters, and it counts them in a straightforward manner. The parameters are: (1) the number of unique operators, (2) the number of unique operands, (3) the total number of occurrences of operators, and (4) the total number of occurrences of operands [3]. It seems the first suggestion to use these parameters as measures of similarity or dissimilarity (depending on your viewpoint) came from N. Bulut as a by-product of his study of invariant properties of algorithms [1].

A tool that analyzes Pascal programs to detect those pairs of programs sufficiently similar such that plagiarism is a possibility is not known to exist. Program Accuse attempts to fill this void.

2. DESIGN

Program Accuse attempts to go beyond M. Halstead's four basic Software Science parameters in the belief that additional parameters are available to establish dissimilarity of two or more programs. It uses seven parameters and various counting heuristics that result in the computation of a correlation number that is used to determine the similarity of two programs. Accuse measures 20 parameters. The seven that comprise the correlation number were selected by testing different combinations of them.

An overriding concern of the development of Accuse has been that it be as inexpensive to use as possible. For this reason, the idea of utilizing the front end of a compiler was discarded, and Ottenstein's lead of using a fast counter was followed.

The result is a compromise between speed and comprehensive analysis. Accuse processes over 170 lines per second. However, it will not discover changes made by the sophisticated plagiarist. This is rationalized with the assumption that the student intelligent enough to plagiarize with sophistication has no need to plagiarize. We hope, however, that Accuse is not so simple-minded that it is easy to beat. It is meant to make plagiarism difficult to achieve, and it is meant to do this is such a manner that its repeated use does not compromise its heuristics.

Accuse is a 2800 line program written in Pascal that runs on the E.T.H. Zuerich/ University of Minnesota Compiler. It consists of a modified Pascal scanner that passes tokens to a driver capable of processing compilable input programs. It also contains a host of support routines for the driver.

Accuse presently measures the following 20 parameters:

1. total lines
2. code lines
3. code comment lines
4. multiple statement lines
5. constants and types
6. variables declared (and used)
7. variables declared (and not used)
8. procedures and functions
9. var parameters
10. value parameters
11. procedure variables (includes 9 and 10)
12. for statements
13. repeat statements
14. while statements
15. goto statements
16. unique operators
17. unique operands
18. total operators
19. total operands
20. indenting function

The seven parameters that comprise the correlation number are:

1. unique operators
2. unique operands
3. total operators
4. total operands
5. code lines
6. variables declared (and used)
7. total control statements

One result of Accuse's development has been the failure of an "indenting function" to play a role in the detection of plagiarism. The indenting function is defined as:

```
( (left indentations) mod 1000) *
   1000000 +
((right indentations) mod 1000) *
   1000    +
  (zero indentations) mod 1000
```

If all programs were processed through a "pretty printer," an indenting function might become important. This additional cost is presently considered prohibitive, and it is contrary to the intent of Accuse being inexpensive to use.

The counting heuristics Accuse uses involve "total operators" and "code lines." "Total operators" does not include assignment operators. Additionally, for every assignment operator found, two operands are subtracted from "total operands," and "code lines" is decremented. This should prevent Accuse from being misled by unnecessary initializations and unnecessary assignment statements. "Code lines" ignores blank lines, comment lines, and declarations. It counts only executable lines of code within a program. "Code lines" was found to be an accurate indication of the sameness of two programs.

As Accuse only counts variables, the obvious tactic of changing variable names makes no difference to Accuse. Since Pascal requires declarations, Accuse can keep track of variables declared and subsequently used or not used. Hence excess declarations are an ineffective change to a program. Constants of enumerated types and tag fields in case clauses of record declarations that contain a declaration are considered variables. Since these constants cannot be read or written, their non-use is considered notable.

Accuse is also selective about what it calls operators. Software Science considers a BEGIN END combination as an operator [2]. Because BEGINs and ENDs can be added to Pascal code where not required, Accuse chooses to ignore them. Parentheses and several other operators are ignored by Accuse for essentially the same reason.

3. OUTPUT

Accuse prints four results for the user. The first (Table 1) is a dump of each program's identifier and its values of the 20 parameters measured by Accuse. This dump is sorted on the indenting function (a matter of my preference).

The second result (Table 2) is a

dump of each program's identifier and its respective values of the seven parameters used to compute the correlation number; each parameter list is sorted smallest to largest. In the output, the column headed FOR STMT actually contains the total number of control statements. This is a result of the implementation of summing parameters.

The third result (Table 4) is a frequency distribution graph that indicates the number of pairs of programs with like correlation numbers.

The final result (Table 5) is a list of all pairs of programs with correlation number greater than or equal to 28. Twenty nine is currently identified as the number that indicates the possibility of plagiarism, with 32 the maximum correlation number possible.

4. CORRELATION SCHEME

The scheme that computes the correlation number is only a tentative one. The current scheme was developed and tuned by using a group of 43 programs from an introductory course. Code for three of the programs was written together, but finished individually. The "importance values" for the seven correlation parameters were then adjusted until these three programs were brought into the domain of "those programs suspected of plagiarism."

The current correlation scheme involves computing an increment for each pair of affected programs based on the equation:

increment = "importance value" - (pcounta - pcountb)

where pcounta and pcountb represent parameter counts, and (pcounta - pcountb) is less than or equal to some "window" size, depending on the particular parameter.

The computation of the correlation number may well be subject to improvement by a more elaborate scheme, or by simple changes to the importance values.

A simple, illustrative run of Accuse follows the text of this paper (Tables 1 through 5). This run processed 13 programs, three of which were input twice. Included is a print-out of the triangular matrix (Table 3) that contains correlation values of the pairs of programs. This matrix is not printed in a production model of Accuse.

Below we illustrate the computation of the correlation number for a pair of programs in the run. Before proceeding, it is necessary to note the following "window" sizes and "importance" values for each of the correlation parameters:

1. total operators
   window size = 5
   importance value = 6

2. total operands
   window size = 5
   importance value = 6

3. unique operators
   window size = 3
   importance value = 5

4. unique operands
   window size = 3
   importance value = 5

5. code lines
   window size = 3
   importance value = 5

6. declared variables (and used)
   window size = 2
   importance value = 3

7. control statements
   window size = 1
   importance value = 2

The correlation number for the pair of programs T107 and T102 (see Tables) is computed as follows:

1. T107 - T102 = 8
   Eight is greater than the window size for this parameter, hence these are not "affected" programs.

2. T107 - T102 = 16
   Again, these are not "affected" programs.

3. T107 - T102 = 1
   These programs are now within the window size, and an increment is calculated for this pair of programs.
   increment = 5 - (25 - 24) = 4
   correlation number = 4

4. T102 - T107 = 0
   increment = 5 - (13 - 13) = 5
   correlation number = 9

5. T102 - T107 = 1
   increment = 5 - (64 - 63) = 4
   correlation number = 13

6. T107 - T102 = 0
   increment = 3 - (11 - 11) = 3
   correlation number = 16

7. T102 - T107 = 0
   increment = 2 - (4 - 4) = 2
   correlation number = 18

5. RESULTS

A typical production run of Accuse included 137 input programs consisting of 13,374 lines of code. Accuse processed the code on a CDC machine at a cost of $12.32. It required:

```
FL TO LOAD 110700
FL TO RUN   77100
105237B CM USED
89.956 CP SECS
```

Accuse prints all pairs of programs with correlation number greater than or equal to 28, though 29 is the number that indicates the possibility of plagiarism.

Several points are necessary.

In six runs of Accuse, sabotage occurred in two. There is nothing to prevent a student from removing lines of code from his program. One student shuffled his cards, and another added control characters not found in the character set of the machine. This led to Accuse being used in the context of a larger tool in its last and most successful run. The instructor retrieved the students' programs, compiled them, ran the programs on data the students had never seen, and then sent the source code to a file to be run on Accuse. This is the recommended context for the use of Accuse.

The correlation scheme is admittedly ad hoc. The only thing that can be said in its defense is that it seems to work. The use of Accuse should not be misunderstood. Accuse does not judge plagiarism; it merely indicates its possibility. It is a tool for the user to aid him in its detection; the decision as to plagiarism is left to the user. High correlation numbers may be meaningless; though rare, programs that are completely different may have like values for the seven parameters that are used to compute the correlation number.

6. THE REAL ISSUE?

Finally, as a reviewer noted, Accuse is a tool to discourage dishonesty in students. But, he asks, does anyone care to ask students why they cheat more now, and can we find ways to abort this rising phenomenon? These are pertinent educational issues.

7. ACKNOWLEDGEMENTS

Thanks to Lloyd Fosdick, who conceived this project and let me work on it; special thanks to Malcolm Newey for his insights and encouragement.

8. REFERENCES

1. Bulut, N., "Invariant Properties of Algorithms," PHD Thesis, Purdue University (August 1973) 118-119.

2. Halstead, M.H., "Elements of Software Science," Elsevier North Holland, New York (1977), Chapters 1-4,7.

3. Ottenstein, K.J., "An Algorithmic Approach to the Detection and Prevention of Plagiarism," SIGCSE Bulletin, Dec 76, Vol.8, No.4.

4. Shaw, M.; Jones, A; Knueven, P.; McDermott, J.; Miller, P.; Notkin, D., "Cheating Policy in a Computer Science Department," SIGCSE Bulletin, Jul 80, Vol. 12, No. 2.

***** Accuse was developed as a Masters Thesis under the advisement of Lloyd Fosdick, University of Colorado, Boulder.

	TOTAL LINES	CODE LINES	CODE COMNT LINES	MULT STMT LINES	CONS AND TYPES	DECL VARS	VARS NOT USED	PROCS AND FUNCS	VAR PARAM	VAL PARAM	PROC VARS	FOR STMTS	REP STMTS	WHILE STMTS	GOTO STMTS	UNIQ OPERS	UNIQ OPNDS	TOTAL OPERS	TOTAL OPNDS	I [illegible] ON
T110	161	48	48	0	3	12	0	2	3	2	8	0	0	5	0	25	14	90	68	[illegible]
T113	166	72	39	0	3	12	0	2	3	2	8	0	0	4	0	25	14	156	136	[illegible]
T106	158	58	63	0	3	11	0	2	3	2	8	0	0	4	0	25	13	115	96	[illegible]
T103	158	58	63	0	3	11	0	2	3	2	8	0	0	4	0	25	13	115	96	[illegible]
T112	125	71	43	0	3	10	0	2	3	2	6	0	0	7	0	24	12	100	73	[illegible]
T111	174	66	60	6	3	11	0	2	3	2	8	0	0	4	0	24	14	134	117	[illegible]
T109	189	44	70	0	3	11	0	2	3	2	8	0	0	3	0	24	13	97	81	[illegible]
T101	178	56	64	0	3	11	0	2	3	2	8	0	0	4	0	25	13	117	98	[illegible]
T108	226	79	80	0	3	11	0	2	3	2	8	0	0	4	0	25	13	183	159	[illegible]
T102	200	64	75	0	3	11	0	2	3	2	8	0	0	4	0	24	13	110	90	[illegible]
T105	200	64	75	0	3	11	0	2	3	2	8	0	0	4	0	24	13	110	90	[illegible]
T104	189	63	78	0	3	11	0	2	3	2	8	0	0	4	0	25	13	118	106	[illegible]
T107	189	63	78	0	3	11	0	2	3	2	8	0	0	4	0	25	13	118	106	[illegible]

* VARIABLE(S) WERE DECLARED BUT NOT USED
** PROCEDURE(S) WERE DECLARED BUT NOT USED
*** VARIABLE(S) AND PROCEDURE(S) WERE DECLARED BUT NOT USED

TABLE 1

	TOTAL OPERS		TOTAL OPNDS		UNIQ OPERS		UNIQ OPNDS		CODE LINES		DECL VARS		[illegible] STMTS	
T110	90	T110	68	T102	24	T112	12	T109	44	T112	10	T109	3	
T109	97	T112	73	T105	24	T107	13	T110	48	T102	11	T106	4	
T112	100	T109	81	T111	24	T108	13	T101	56	T107	11	T101	4	
T105	110	T102	90	T112	24	T103	13	T106	58	T105	11	T113	4	
T102	110	T105	90	T109	24	T101	13	T103	58	T108	11	T103	4	
T103	115	T106	96	T104	25	T106	13	T104	63	T111	11	T104	4	
T106	115	T103	96	T106	25	T109	13	T107	63	T101	11	T107	4	
T101	117	T101	98	T113	25	T105	13	T102	64	T109	11	T102	4	
T104	118	T107	106	T107	25	T104	13	T105	64	T106	11	T105	4	
T107	118	T104	106	T106	25	T102	13	T111	66	T104	11	T111	4	
T111	134	T111	117	T110	25	T110	14	T112	71	T103	11	T108	4	
T113	156	T113	136	T101	25	T113	14	T113	72	T113	12	T110	5	
T108	183	T108	159	T103	25	T111	14	T108	79	T111	12	T112	7	

WARNING: HEADING(S) ARE NOT CORRECT. AT LEAST ONE REFLECTS A SUM OF TWO COUNTS.

TABLE 2

	T 1 0 2	T 1 0 3	T 1 0 4	T 1 0 5	T 1 0 6	T 1 0 7	T 1 0 8	T 1 0 9	T 1 1 0	T 1 1 1	T 1 1 2	T 1 1 3
T101	14	26	20	14	26	20	15	13	12	13	10	13
T102		15	18	32	15	18	14	14	11	17	11	12
T103			18	15	32	18	15	13	12	13	10	13
T104				18	18	32	15	13	12	15	10	13
T105					15	18	14	14	11	17	11	12
T106						18	15	13	12	13	10	13
T107							15	13	12	15	10	13
T108								13	12	13	10	13
T109									10	13	14	11
T110										12	9	14
T111											10	13
T112												12

TABLE 3

FREQUENCY DISTRIBUTION GRAPH FOR PAIRS OF PROGRAMS

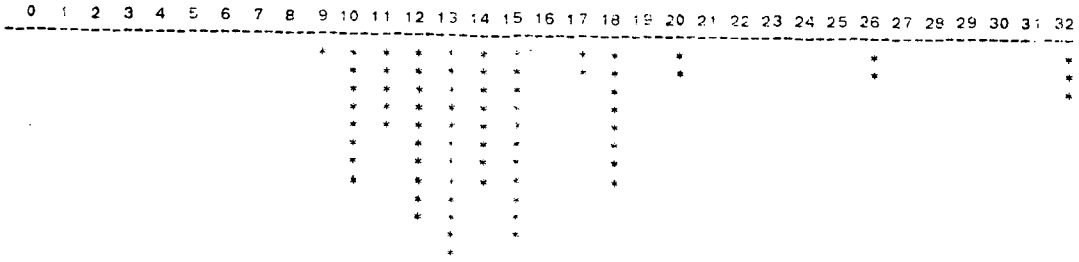

TABLE 4

THE FOLLOWING PAIRS HAVE A CORRELATION OF 32:
T102,T105
T103,T106
T104,T107

TABLE 5