Evaluation of compiler diagnostics

L. Presser and J. Benson

*Department of Electrical Engineering and Computer Science, University of California, Santa Barbara, California, 93106, USA**

This brief paper presents a technique for the evaluation (specification) of translator diagnostics. The approach is based on the definition of a weighted error range and synthetic modules that exercise the language constructs of interest. As an illustration three different FORTRAN translators, available on the IBM System 360 computers, are evaluated. These are the WATFIV, G, and H translators. The results obtained substantiate the fact that WATFIV is superior to the other two systems as far as diagnostic power is concerned.

(Received February 1973)

Today's computer management is faced with increasing flexibility in the configuration of computer systems. Particularly so in view of the present trend towards separate pricing of hardware and software facilities. Specifically, anyone selecting a translator (e.g. compiler, interpreter) for a popular programming language such as FORTRAN, is offered a number of alternative packages. In choosing a translator many different characteristics have to be considered in detail (Lucas and Presser, 1973). One of the most important characteristics is the diagnostic information provided and the degree of error recovery effected by the translator. Generally, this has been, at best, evaluated (specified) in a vague manner. The purpose of this short paper is to present a simple and practical approach to the evaluation of translator diagnostics.

Method

The approach (Presser, 1968) consists of defining a weighted error range, such as that shown in **Table 1,** and a synthetic module that exercises the language constructs of interest. A synthetic module (Lucas and Presser, 1973) is a program written to model the characteristics of the anticipated job stream. Once defined, the synthetic module, which in this case contains errors, is run through each of the translators under consideration. Based on the results obtained and the weighted error range defined, each translator is assigned a total score; the higher the score the better the performance. The overall quality of the diagnostic messages is also taken into consideration by adjustments to the total scores.

Finally, it is necessary to distinguish between translation and execution time diagnostics and error recovery.

Example

To illustrate the technique we present a relative comparison of the results obtained when the diagnostic power of the WATFIV[1], FORTRAN G[2] and FORTRAN H[3] compilers are exercised. These three compilers are implemented on IBM's System 360; the first one is available from the University of Waterloo and the other two from IBM. The optional DEBUG facility available with FORTRAN G is not considered in the discussion that follows.

Translation time diagnostics

In the recent article (1971) Knuth reported statistics that indicate that the most popular FORTRAN language constructs are: ASSIGNMENT, IF, GO TO, and DO. Therefore, based on Knuth's work and on our own experience, it is reasonable to restrict our analysis to these four statements. Hence, four synthetic error submodules were coded. Each submodule was utilised to evaluate the response of the translators to common errors in each of the four FORTRAN statement types mentioned above. The set of 'common' errors present in a submodule consisted of the union of the errors associated with the statement in question, that each of the three translators is capable of detecting (Cress, Dirksen and Graham, 1970; IBM, 1970), plus a few others. The type of errors present in the four modules are listed in **Tables 2, 3, 4 and 5.** The modules were run on the three compilers and their performance evaluated using the rating scale detailed in **Table 1**. The overall quality of the diagnostic messages was also taken into consideration. The results are presented in **Table 6.** The percentages shown were calculated by scoring each error, totalling the scores, adjusting for overall features and then normalising.

We can observe that in each case the diagnostic performance of WATFIV is superior to that of the G and H compilers. This substantiates what is generally well known (Siegel, 1971). However, it is of interest to note that FORTRAN H does almost as good a job as WATFIV on the IF statement, while FORTRAN G does a very poor analysis on a DO statement. Also, it is of some value to compare the G and H scores.

Execution time diagnostics

To exercise the diagnostic power of these systems during execution time another synthetic error module was prepared. The selection of the errors present in this synthetic module was influenced by the type of execution errors observed in a sample of programs from an undergraduate programming class. The module tested the three systems for their reactions to the following execution time errors:

(*a*) Overflow
(*b*) Integer input too large
(*c*) Incorrect input type
(*d*) Computed GO TO out of range
(*e*) Array reference out of range

Table 1 Weighted error range

+5	Detects and corrects specific error
+4	Detects specific error and specific location
+3	Detects general error and specific location or specific error and general location
+2	Detects general error and general location
+1	Detects general error
0	No error detected
−1	Misleading or redundant error information
−2	Wrong (error) information

*This work was supported in part by the National Science Foundation, Grant GJ-31949.

[1]Version 1, level 2, Aug. 1970.
[2]Level 18.
[3]September 1969 release.

Table 2 ASSIGNMENT statement errors tested

Variable name too long
Integer constant too large
Undimensioned array reference
Incorrect number of subscripts
Invalid delimiter
Non-numeric character in numeric constant
Exponent too large
Complex constant not composed of reals
Operator terminates expression
Comma in real constant
Invalid variable name
Non-subscripted array item
Complex constant with different length reals
Complex exponent
Complex base with non-integer exponent
Logical variable base
Missing operand
Extra parenthesis
Too few parentheses
Missing operator after parenthesis
Logical operator with period missing
Invalid character in columns 1-5
Real constant greater than 16 digits
Two decimal points in constant
Constant greater than 7 digits with exponent
Number on left of equal sign
Multiple assignment
NOT used as a binary operator
Relational operator with logical operand
Relational operator with complex operand
Mixed mode (logical with arithmetic)
Logical subscript
Zero subscript
Negative subscript
Illegal sequence of operators

Table 3 IF statement errors tested

Complex expression in arithmetic IF
Undefined label in arithmetic IF
Unlabelled statement following arithmetic IF
Invalid statement following logical IF
Duplicate statement labels
Illegal statement label in arithmetic IF
Logical IF following a logical IF
Equal sign in logical IF
Statement label greater than 99999 in arithmetic IF
Arithmetic expression in logical IF
Format statement label in arithmetic IF

(*f*) Incorrect argument for SINE
(*g*) Incorrect argument for ALOG
(*h*) Incorrect argument for SQRT

The scores obtained when this module was executed, and the weighted error range displayed in Table 1 utilised, were:

WATFIV	G	H(HO)
70%	25%	25%

We observe anew that the performance of WATFIV is superior to that of the G and H systems!

In general, when the WATFIV system encounters an error it stops execution. The G and H systems, on the other hand, perform some standard corrective action and continue with execution. We consider this latter course of action preferable.

Table 4 GO TO statement errors tested

Self transfer
Transfer to a FORMAT
Assigned GO TO index assigned by an arithmetic statement
Assigned GO TO index used in an arithmetic statement
Index of computed GO TO undefined
Non-integer GO TO index
Missing comma in assigned GO TO
Illegal statement label
Non-existent statement label
Missing parenthesis
Invalid delimiter
Non-integer computed GO TO variable
Non-integer variable in assigned GO TO
Invalid assigned variable in ASSIGN statement
Invalid delimiter in ASSIGN statement
Invalid delimiter in GO TO statement

Table 5 DO statement errors tested

DO statement is the object of a DO
Illegal transfer into DO range
Object of DO precedes DO
Improperly nested DO's
Parameter redefined within the loop
Non-integer DO parameter
Equal initial and final values
Do parameter redefined in an input list
Initial value negative
Invalid delimiter
Subscripted DO variable
Subscripted test value
Subscripted increment value

Table 6 Translation time diagnostic performance

	WATFIV	G	H(HO)*
ASSIGNMENT	66%	55%	47%
IF	73%	58%	69%
GO TO	72%	51%	42%
DO	65%	12%	45%

*The FORTRAN H compiler allows one of three levels of optimisation to be specified: H0, H1, H2; the lowest level is H0.

Thus, the G and H scores were adjusted slightly to give credit for their strategy.

Summary

We have presented here a simple and practical method for the evaluation (specification) of the diagnostic power of translator systems, which is a problem of interest. The approach is based on the definition of a weighted error range and synthetic modules that exercise the language constructs of interest. The results depend directly on the weights assigned to the various levels in the error range and on the types of errors included in the synthetic modules. These decisions should be made in the context of the specific translators to be evaluated as well as their intended use. To illustrate the method three popular FORTRAN translators available on the IBM System 360 were evaluated. The results indicate that, indeed, WATFIV offers very good error analysis facilities.

Finally, it should be noted that it is possible to employ a similar strategy in the evaluation of the diagnostic power of other sections of an operating system.

References

CRESS, P., DIRKSEN, P., and GRAHAM, J. W. (1970). *FORTRAN IV with WATFOR and WATFIV*, Prentice-Hall, Englewood Cliffs, New Jersey, 1970.

IBM System/360 Operating System FORTRAN IV (G and H) Programmer's Guide, Third Edition, International Business Machines Corp., June 1970.
KNUTH, D. (1971). An Empirical Study of FORTRAN Programs, *Software Practice and Experience*, Vol. 1.
LUCAS, H. C. Jr., and PRESSER, L. (1973). A Method of Software Evaluation: The Case of Programming Language Translators. *The Computer Journal*, Vol. 16, No. 3.
PRESSER, L. (1968). *The Structure, Specification and Evaluation of Translators and Translator Writing Systems*, Ph.D. dissertation, Report 68-51, Department of Engineering, University of California, Los Angeles, October 1968.
SIEGEL, S. (1971). WATFOR ... Speedy Fortran Debugger, *DATAMATION*, November 15, 1971.

Book reviews

Standard FORTRAN Programming Manual, by R. Bornat, 1972; 152 pages + indexes + appendices. (*NCC*, £4·75 hard cover, £3·50 paperback)

This manual is intended to show FORTRAN programmers how to write programs in Standard FORTRAN and 'to bridge the gap between the few who understand Standard FORTRAN and the many who do not'. The first edition was reviewed in references 1, 2 and 3.

To prepare the second edition the publishers have simply applied, not totally successfully, the amendments in the three-page errata list to the first edition and have added as appendices a reproduction of the American National Standard for FORTRAN (X3.9-1966) and the first set of clarifications to the Standard. These additions are welcome and it is perhaps ungracious to mention that the copy of the Standard is one without line numbers, making the clarifications harder to identify, and that the second set of clarifications (reference 4) has inexplicably been omitted.

Apart from the errata no more than a handful of minor changes have been made to the text which is therefore subject to the same praise and criticism as before. The manual is unique in the literature: the author Richard Bornat, has probed thoroughly and makes many nice points which will have escaped all but the most devoted readers of the Standard. Nevertheless there are still a few factual errors and a number of dubious explanations and fact and opinion are not separated as clearly as one would wish. Regarded as a personal treatise, and with this caveat, the book is highly recommended. Appearing now with the silver and black ménage à trois symbol it still, in this reviewer's opinion, falls short of the standards that ought to be required of one of the NCC's Computers and the Professional series.

References

1. *Computer Bulletin* (1971) Vol. 15, p. 47.
2. *Computer Bulletin* (1971) Vol. 15, p. 245.
3. *Computing Reviews* (1972) Vol. 13, p. 79.
4. American National Standards Institute Sub-committee X3J3 (1971), Clarification of Fortran Standards—Second Report. *CACM*, Vol. 14, pp. 628-642.

D. T. MUXWORTHY (Edinburgh)

Digital Interface Design, by D. Zissos and F. G. Duncan, 1973; 174 pages. (*Oxford University Press*, £4·00)

The idea behind this book is a good one: to treat interfacing to a computer in a generalised systematic way. To some extent the authors achieve this but not as successfully as they might have done if the book had not also been based on a lecture course. The truly valuable part of the text lies in the introductory part of each chapter. It is the amplification and detailed analysis that follows which is of dubious value.

The first chapter, for instance, starts well by defining the nature of the interface and its functional components. The chapter continues with an account of Logic Design by Zissos' method which covers this wide and complex subject too briefly to be of use for more than lecture notes. It is in this chapter, too, that the authors chose to write binary numbers with the m.s.d. on the right. Zissos successfully makes a virtue of disregarding conventions in his recent excellent book on Logic Design but here it is apt to confuse.

The second chapter on the 'Components of a Digital Computer' starts well with the generalised introduction. The rest of the chapter consists of a quite long description of the internal logic of a kind of 2½D core store. Possibly this is a useful exercise in logic design but it has little relevance to a book on interfacing.

There are chapters on Machine Code Programming, and what should be an important part of the book, Programmed and Autonomous Data Transfers. These regrettably conform to the first two chapters—good beginnings but little else. In aid of generalisation it is perhaps a good idea to invent a machine code and a hypothetical small computer, but it seems hardly necessary.

In summary, about a fifth of the book is good work appropriate to the title. The remainder is largely a series of contrived exercises in logic design, suited to the tutorial work of a compressed course, but hardly to an authoritative work on an important subject.

B. S. WALKER (Reading)