Considerations for Future Programming Language Standards Activities

Technical Report # CS76014-E (revised)

by

John A. N. Lee
Computer Science Department
Language Research Laboratory
V.P.I. & S.U.
Blacksburg, Virginia 24061

April 15, 1977

ABSTRACT

This paper reviews the current state of programming language standards activities with respect to the anomalies which exist between the various published and proposed standards for FORTRAN, COBOL, PL/I and BASIC. Proposals are made for the inclusion of formalisms within future standards and the extension of the standards to include additional items such as error conditions and documentation.

KEYWORDS

Programming Languages, Standards, Formalisms, Formal Descriptions, FORTRAN, COBOL, PL/I, BASIC, Vienna Definition Language, VDL.

CR Categories

4.2, 4.29, 4.6, 5.23, 2.4

Introduction

During the past three years, four programming languages have been subjected to public review in preparation to their becoming National Standards. These were: COBOL (X3.23-1974*), PL/I (X3.53-1976), FORTRAN (X3.9-1966, currently being revised as X3J3/76), and BASIC (in process of development, X3J2/76-35). Each in its own way has contributed to a growing controversy as to the usefulness of standards in general and the uselessness of standards for "dead" languages. The updated COBOL standard was criticized for not maintaining the conformance of programs written in the earlier 1968 standard whereas the new FORTRAN proposal has been held up by the lack of "structured" programming elements. PL/I is the first standard which was presented in the form of a totally formal description, though COBOL contained a graphical syntactic specification. This has stirred a controversy related to the intended audience of standards since the lack of a verbal description in the PL/I standard seriously reduced the audience who could respond to the technical content of the proposed standard during the public review period. Finally BASIC added to the overall debate by including in the proposed standard, specifications

----

* X3 numbers refer to the reference code of the American National Standards on Computers and Information Processing, American National Standards Institute, 1430 Broadway, New York, NY, 10018.

on error reporting and recovery, as well as certain documentation requirements.

For the first time also, the FORTRAN and BASIC proposal both include specifications which specify the conformance requirements of programs and implementations. Distilling the conformance requirements from the FORTRAN and BASIC proposals, a conforming program is one which is syntactically correct and which does not violate any of the semantic specifications of the standard. Thus a conforming program does not contain any features which are extensions of the standard language and contains only those elements whose constructs are meaningful within the semantics proscribed in the standard. A conforming implementation is simply then a processor which accepts and correctly processes a conforming program. By this definition a conforming implementation can contain additional features which are extra-lingual and still conform to the standard. Such a requirement is perfectly valid when one considers that it is not the purpose of a standard to inhibit language development. In fact, a "good" standard should be considered to be one in which provision is made to extend the language in a conformable manner. By this means, languages and standards can be evolved naturally and revisions be developed on the basis of tested features.

Reviewing these standards, several questions need to be answered and be considered to determine the program of work of future programing language standards activities:

1. To whom is a standard directed?

2. a) Should formal descriptions of both syntax and semantics be included in all programming language standards and standards which relate to programming languages (such as the numeric representation standard)?
b) Should there be a standard formalism developed for use in all programming language standards?
c) Should a programming language standard include both formal <u>and</u> verbal, explanatory descriptions?

3. Does a programming language standard only impact the language processor and programs or are other features of an implementation included, such as documentation?

4. Should any consideration be given to the concept of standardizing language features and then composing standards of these features (suitably clothed in syntax) and special elements?

The answers must be tempered with the basic tenets of standards development. There are three common criticisms of standards--too soon, too late and who cares? Primarily it must be realized that the purpose of a programing language standards development activity is to produce a standard <u>not</u> to develop a programming language. The materials with which a standardizer works are the <u>existing</u> instances of language constructs, the three primary major tasks being:

1. to select the language features to be included,

2. to choose between various alternative "equivalent" features, and

3. to ensure the consistency of the totality of language features.

The solution to the first of these appears in several different forms; the PL/I and FORTRAN standards choose to define the whole language and to permit future subsetting standards. The COBOL standard presents the language in a complete set of modules which can be combined in a limited number of forms to compose many varieties of an implementation. Initially the proposed BASIC standard covers only the minimal elements of the language and future enhancements standards will build on this core. The choice between several forms of a language feature is difficult and invariably prone to partisan pressures for inclusion in the final product. The solution which permits the inclusion of several alternate forms undermines the efficacy of the resulting standard.

The Audience of a Standard

This question has been the subject of considerable debate with respect to the proposed PL/I standard which was reported out of committee X3J1 without any supporting explanatory verbal documentation. Unfortunately much of the discussion which this produced did not differentiate clearly enough between the effects of the standard and the standard itself. For example, at the NCC-1976, Lois Frampton, chairman of X3J1, stated (paraphrased) that "...the purpose of the PL/I standard is to ensure that implementations conform to a common set of specifications (...and...) that users will be protected by the implementers." From this one is forced to conclude that implementations would not permit the user to develop non-conforming programs. However, the definition of conformance as applied to programs and implementations is not as restrictive as to enforce this expectation. In fact, the PL/I proposed American National Standard does not include any statement with regard to conformance. Obviously the ultimate beneficiary of a standard is not the language processor implementor nor the programmer user. Instead the beneficiary is the customer for whom the program is written. However, we must again distinguish between the standardized language and the standard. That document which is called a "standard" provides a specification for the language processor implementor to follow and is a guarantee to the programmer that what he writes is universally meaningful.

While a standard does not in itself constitute a programming manual, it must provide the basic syntactic and semantic specifications for those manuals. In the event of a failure of a specification in the manual, the standard must provide a detailed description of the operation of each language element and an interpretation of the resulting state of the abstract machine. Thus a standard is a part of the essential library of a programmer. However the primary audience for a standard are the implementer and the procurement agent who must ensure that an implementation conforms to the desired standard.

Formalisms in Programming Language Standards

Experience with the verbal style of programming language standard has revealed that no matter how careful the developers are with their formal style of language, including such terms as to differentiate between "must", "shall" and "is", ambiguities of meaning still arise and require interpretation by the originators. Unfortunately, it is a fundamental principle of law that intention (no matter how well remembered by the authors) cannot be used as the basis for the interpretation of the law. That is, a law must stand on its own feet. Similarly in the domain of standards, the intent of a phrase included in a standard is irrelevant to the interpretation of that phrase at a later time. By coincidence, it may happen that an interpretation conforms to intent; but that is not a requirement. In the cases of both FORTRAN and COBOL, their earliest versions

[1966 and 1964 respectively] required extensive interpretation in the years between the production of the standard and the subsequent revision [ANSI, 1969 and CODASYL, 1968]. On the other hand, little practical experience has been garnered with respect to the need to interpret the formal specification. However, it is expected that since many of the implementation decisions have to be studied in developing a formal specification, that the ambiguities are removed. Experience with developing formal specifications from verbal descriptions has also revealed that illogical constructs are readily identified and can be corrected.

There have been four efforts at developing a formal description of a language related entity within an American National Standard:

1. the syntactic specifications within standard COBOL (X3.23-1974),
2. the syntactic specifications within the standard for the representation of numeric values (X3.42-1975),
3. the syntactic and semantic description of PL/I (X3.53-1976), and
4. the syntactic specification of the Minimal BASIC proposed standard (X3J2/76-35).

Each of these has been met by the industry with varying degrees of concern, the major response being related to the ability of the regular standard user to understand the formalisms.

Syntax *and* Semantics

There are two levels (at least) of formal specification that must be considered. Firstly there is the syntactic specifications for the language. Immediately we find ourselves in a dilemma. Although the Backus Naur Form (BNF) has been the common form of specification of syntax in the literature for at least ten years,

1. there exist several different symbolic forms of BNF,
2. BNF and regular expressions are commonly intermixed, [see both X3.42-1975 and X3.53-1976] and
3. BNF is actually only applicable to context free languages.

Unfortunately, very few of our programming languages are totally context free and thus a decision has to be made as to the next step to be considered for developing a formal (accurate) description of the language. The PL/I solution to this problem is to specify syntactically a super-language, certain elements of which are semantically unacceptable. That is, instances of the language can be developed with respect to the syntactic specifications which are invalid according to the semantic specifications. This approach may be satisfactory on the basis that the specification of the language is to be regarded as a whole and that it is invalid to regard the syntactic specifications as being capable of standing on their own.

There have been several efforts to develop a syntactic specification system which will take into account the context sensitive requirements, [Ledgard 1974, Lee and Dorocak 1973]. However, these have not been totally successful due to their complexity and lack of readability. A recent survey of such efforts [Marcotty, Ledgard and Bochmann 1976] showed* the complexity which is obtained when syntax and semantics are combined into a single system. It is not expected that these academic efforts will be accepted in the foreseeable future.

The PL/I style of semantic specification which is a variation on the Vienna Definition Language [Lucas 1969] (abbreviated to VDL) is actually only semi-formal. That is, there are a number of cases where the complexity of the definitional system itself was so overwhelming that the standard reverts to relying on verbal descriptions and "<u>common sense</u>" in order to specify the particular feature.

The PL/I descriptive techniques suffer from the fact that the language used, while being mathematical in form, is peculiar to this one standard. VDL depends on knowledge of a conceptual machine which operates over a set of functions (usually visualized as trees), quite differently from any known computer. Furthermore, the set of available

*In the opinion of this author

instructions in the conceptual machine is miniscule, relying heavily on parameter passing and recursiveness to accomplish will be able to deal with these abstractions. There are two solutions to this problem;

1. create a new means of specification based on a more realistic abstract machine (the "standard" computer?), or
2. provide a verbal description in parallel with the formal description for use by the programmer.

Standardized Formalisms

An examination of the existing and proposed programming language standards shows a considerable difference in organization besides simply the differences in formalisms used.

There are certain advantages to be attained if the set of standard languages are provided with a common base of definition. Firstly, there is then a common means of comparison between the languages, and secondly there may be developed at a later date a means for formally describing programs based on the formal description of the language in which the program is written. On the basis of a formal description of a program there exists then the possibility of validating the program through a standard verification procedure.

Computer Science is so young that there is not yet any single means of specification which has emerged as the predominantly superior system. Partially this is because most systems [Lucas 1968, Lee 1972, Marcotty, et al 1976, Strachey 1973, McCarthy 1970, and others] have been applied to few languages and there is little overlap in the objective of each specification. Thus it is difficult to judge the comparative effectiveness of the specifications systems. By sheer numbers of specification applications, BNF and VDL win the race for supremacy. However the appropriateness of VDL must be carefully questioned.

Verbal and Formal Descriptions

In the recent case of the proposed PL/I standard, the X3J1 committee felt that the task of producing both a formal description of the language and a verbal, explanatory document was too onerous. The committee (through its chairman as part of the panel discussion at NCC-1976) pointed out that attempts were made during the early portion of their effort to accomplish the task of developing an explanatory standard which was to be supported by the formal documentation. Conversely, no experience has been garnered in the task of developing the explanatory document based on the formal document of PL/I.

Each time a standard is processed through the American National Standards committee X3, a different attitude is taken towards the inclusion of explanatory material and footnotes as part of the standard. In some instances there are explicit statements that the footnotes and appendices are not part of the standard. The removal of material from the body of the standard to appendices is then one means of providing explanatory material in the <u>same</u> document as the formal part of the standard.

The existence of an explanatory document based on a standard which is itself presented in a formal manner, raises the question as to the status of the explanatory material, as indicated above. One possible solution to this dilemma would be to publish the explanatory document not as a standard, but instead as a Technical Report of the standards committee.

CBEMA (Computer and Business Manufacturer's Association) as the secretariat of American National Standards Committee X3, has already undertaken the production of such reports in connection with the vocabulary (originally issued as a standard, X3.12-1970) and documentation procedures. Alternatively, this supplementary document could be developed and produced commercially by a publishing house, and carry some notation that the contents conform to the standard. This does not necessarily imply that the standards development committee itself must place its stamp of approval on the publication, any more than the committee is expected to validate the implementations which are derived from the standard. Since the publication of the original COBOL standard, there has been produced a number of textbooks which purport to explain the COBOL standard [for example, Murach 1975]. Whether these are accurate manifestations of the standard is not known at this time. However, it might be reasonable to assume that publications of this kind should be subject to the same restrictions of the use of the phrase "conforms to the ANSI standard" as are the implementations themselves.

The advantage of developing a technical report as an addendum to the formal standard is that this secondary document could well be the basis for the development of manuals to be used by the vendors. In the very least this explanatory document should include information relating to all the important features of the language which ought to be included in a vendor's manual. While such a hope is implicit here, the next section considers those extra-lingual elements of an implementation which should be included in the standardization process.

The inclusion of both a formal and a verbal description of a language in a standard raises the immediate question of which is the definitive portion? If a comparison is made with standards outside the domain of programming languages, and in particular in the field of media (such as magnetic tapes X3.14-1973 and others, disks X3.46-1974, etc.) then it will be seen that such a combination of formalisms and verbal description does already exist. In these cases the formalism is defined to be the standardizing information, irrespective of any ambiguities in the verbal description. So in the case of programming languages, the verbal description is merely the explanatory material for use by the "standard" programmer and which is subject to interpretation by the formalism when necessary.

Programming Languages as Complete Systems

To this date, the several language standards developed or accepted by the American National Standards Institute have been directed at two elements of the language system: the language processor and the programs in that language. However, these are not the only elements which are included in the language system which is delivered to a user. At the very least the system contains:

1. the language processor (interpreter, compiler, etc.),

2. the set of conforming programs (by implication),

3. the users manual containing:

   a) The description of the language from the points of view of both the syntactic forms which are acceptable, and the semantics of those forms,
   b) the set of directions relating to the operation of the language processor,
   c) the set of error messages which are emitted by the processor and their causes,
   d) the implementation features which are machine dependent, such as the range of numeric representations and the maximum lengths of character strings and
   e) a listing of non-standard or extra-standard features,

4. documentation relating to the installation of the language processor and the maintenance of the system,

5. a statement of conformance with the appropriate standard and

6. a set of test programs [see Hoyt 1976, for example] which either:

   a) validate the conformance of the processor with the standard, or
   b) provide a set of diagnostics related to the operation of the language processor.

To this date, the X3 committee has concerned itself with documentation only from the viewpoint of documenting programs and data elements totally separately from language standards. It is possible and feasible that each standards development committee contain in its program of work the standardization of these essential features of a language processor system. However, there also exists the possibility that a separate, general purpose, standard be produced which specifies that the additional items listed above must also be provided in order for a system to conform to the standard.

Particularly in the case of user manuals, it is essential that some direction be given to the vendor as to what details must be included. For example, in studies of BASIC [Lee et al 1974, and Isaacs 1973] it was found that the majority of users manuals did not include information on the hierarchy of arithmetic operators or the binding time of identifiers in an input statement. Both of these pieces of information are essential to the proper execution of programs and it was only by experiment that the reviewers were able to ascertain the manner of execution of the processors or their resulting programs.

The question of standardizing errors within the standard has been considered only seriously by one standards development committee; X3J2, BASIC. Three elements of error specification need study.

1. the feasibility of identifying errors by the processor or by the host system (in the case of a program),
2. the recovery strategies to be followed after an error, and

3. the error messages.

It must be assumed (though as far as can be determined, has never been explicitly stated) that the current language standards define that anything that is not covered by the terms of the standard are "implementation defined". That is, items which do not conform to the standard can be regarded by the implementer either to be errors or to be language extensions. While it is not the intent of a standard to restrict language development, the standard should include provisions for specifying that certain anomilies must be treated as errors.

These considerations raise two other issues; (a) the problems related to subsetting or supersetting of languages and (b) the questions of distinguishing between those elements which are "errors", those whose semantics are "undecidable", those which will produce "unpredictable" results, those which are "implementation defined" and those which are "undefined". The question of errors as related to various levels of language implementation is most easily resolved when the original language specification is the superset (as in the case of PL/I). In this case all subsets can be assumed to be proper, and thus those elements which are specified (usually by the implementer) not to be in the subset are to be reported as errors. However, where the initial language specifications are in terms of the minimal subset language (such as in the case of BASIC), with all other instances of the language being enhanced versions (supersets), the problem is much more difficult to answer. In this case, it is not known at the time of standardization of the minimal set, what language enhancements might be developed at a later stage. Thus it is important to specify errors in such a manner that language extensions are possible. That is, a program which conforms at the lowest level must also conform at the enhanced level and still develop the same results.

<u>Standardarized Language Features</u>

Ledgard [1971], showed that there exists a set of semantic features which are common to many languages. These include such elements as:

- assignment
- block structure
- functions and procedures
- transfer of control
- parameter passing
- data structures
- input/output

which he used as a pedagogical aid to the understanding of languages in general. Examination of many of our languages show this commonality to be true at some high level but in depth examination of apparently common features reveals only glaring inconsistencies which are the hallmark of particular languages. This is particularly unfortunate since it would be inconceivable at this point in time to go back to revise the existing standards so as to develop a consistency of features. Even more unfortunately, the novice user is unaware of these inconsistencies and expects that (for example) the looping controls in two languages operate in a similar manner.

These latent inconsistencies will preclude the distillation of common features from the existing languages, and may force the consideration of any development of standard language features to such a low level as to only cause further confusion. The concept of a "standard" computer has been raised several times over the years, only to be rejected on the basis of the inability to be totally general (or more correctly, so as not to favor any particular machine architecture), and the restrictiveness of the abstract machine in implementing (easily) certain high level language features. Even over a restricted domain such as the procedural numeric oriented languages the commonality of features would be politically difficult to justify.

If a formal description of a language can be expressed in terms of standardized features without the formalism as used in PL/I, a standard standardizing language might be created which itself can be formally described, but which in itself is understandable by the vast majority of the industry.

Moreover this would give the advantage of expressing a much wider variety of languages (including applications languages) in high level terms. In this latter case, a standard would then be composed merely of the syntactic specifications of the language and a set of transformations into the "standard language". In view of the current trends within the industry to "standardize" programming in terms of a highly restricted set of control structures (the so-called Bohm-Jacopini constructs), it is even more necessary to consider the construction of a standard base language, into which all other language would be transformed for the purpose of standards specification. Languages include not only symbolic elements and arithmetic operators but also a set of intrinsic functions such as SINE, COSINE, etc. To date, no consideration has been given in any programming language standardization effort to this problem. In fact, there exist commercial multilanguage systems containing several different implementations of certain intrinsic functions presumably because even their "in-house" standards did not include such common function specifications.

It might be expected that a more successful standardization effort can be elided. In this case, it is not the language feature itself which is to be standardized, but rather the realization of the mathematical function in terms of a "package" which is supplied by the vendor. No doubt it would be imprudent to define which approximation is to be utilized in implementing the algorithm which represents the function; however, it would be feasible to consider the permissible variance of the chosen approximation from the mathematically "correct" result. The procedures for testing the conformance of a given package to the standard would be subject of extensive research prior to the development of a standard.

In the main, the problem of accuracy of implementations is most important close to discontinuities, often points where even the most sophisticated algorithm has problems due to the finiteness of the host machine. Certain new procedures [Fosdick, 1976] which are being developed for testing programs must be applied to this problem when they are themselves finally shown to be correct, or at least, computable.

Summary and Recommendations

While we must recognize that the state of the art of Computer Science has advanced considerably over the past ten years since the original FORTRAN standard was promulgated, the step towards totally formal standards documents in the field of programming languages is not acceptable. Moreover, if any consistent effort is to be made to introduce formalisms into future standards documents it must be done in a consistent manner. That is, there needs to be developed as soon as possible a standard for standards which will provide the basis for specifying both the syntax and semantics of programming languages in such a manner as to be both unambiguously explicit and acceptable to the computing community. It is not acceptable that there should be considered to be a elite group within the industry to whom standards are addressed. It should be expected that all those concerned with conformance to standards, from the language processor implementer to the journeyman programmer should be provided with a means for accomplishing their appointed tasks. To this end it is essential that programming language standards documents contain both the formal description of the linguistic elements of the language and a verbal, explanatory portion. The simple existence of a formal description system does not ensure that the specification of any language is any more accurate than the equivalent verbal description, for all its faults. To date, only one non-trivial language descriptor [London, 1972] has been validated by a formal means. As attractive as formal descriptions are to the standardizers, the validity of a standard is only as good as the proven validity of its description system. To this end, it is imperative that the standards development groups be provided with a validation system at the earliest possible time, or if that is not possible a means by which programs written in the language can be validated.

There ought to be a more compelling reason for specifying a language in terms of a formalism than simply to provide a more obtuse specification. At the very least, there must be some beneficial side effects. Besides the ability to validate the specification and possibly programs written in the specified language, the formal specification should also be the basis for the development of processor audit routines in order to validate any implementation. To date the benefits of a formally specified language have not been clearly delineated; this task is of paramount importance to our future standards activities.

The adequacy of programming language standards in areas other than the software aspects also needs close attention. Initially it must be recognized that a language processor system, as delivered by a vendor, does not simply consist of a machine readable entity. Rather, the system is composed of a number of distinct items including necessary documentation. These additional items also need to be subject to standardization.

This review has purposely (for the sake of length) not included a closer look at the process of standardization itself. The means by which standards in the field programming languages are developed, and in particular the methods for recording the deliberations of the committees, leaves much to be desired, and should be the subject of a separate study.

During the reviewing process to which this paper was subjected, two reviewers disagreed with several of the contentions made by the author with respect to the domain of programming language standards. For example, one reviewer disagreed that standards were capable of specifying error conditions and vehemently stated (judging from the thickness of the pencil marks) that the statement of recovery procedures was <u>not</u> to be included. Other disagreements included objections to the notion that a programming language <u>system</u> includes the necessary documentation which the vendor supplies. The differentiation between the use of the terms "undecidable", "unpredicable", and "undefined" brought cries of anguish from one reviewer who stated that all such elements should be classified as "implementation defined". This same reviewer also objected to the inclusion of documentation standards and thus would never find out how the implementer had chosen to define the undefined! I agree that we do not want to enter into a process which will overdefine languages to the point where they are stifled; however, there should exist some minimal standards for <u>all</u> elements of a language processor.

One final issue remains; how to make National Standards more acceptable to both vendors and users. As industrious as the standards development committees have been, their labors have been somewhat futile as measured by the industrial application of their standards. While it is realized that standards activities in this country are developed by volunteers, and conformance to known standards is equally voluntary, the ability of those standards to be accepted as examples of the best of current practice must be seriously questioned.

References

ANSI (Committee X3J3), Clarification of Fortran Standards - Initial Progress, Comm. ACM, Vol. 12, No. 5, May 1969.

CODASYL, COBOL Journal of Development, 1968 (and later years).

Fosdick, L.D. and Osterweil, L.J., Data Flow Analysis in Software Reliability, ACM Computing Surveys, Vol. 8, No. 3, September 1976.

Hoyt, P.M., The Navy FORTRAN Validation System, Dept. of the Navy, Washington, D.C., 1976.

Isaacs, G.L., Interdialect Translatability of the BASIC Programming Language, ACT, University of Iowa, Iowa City, 1973.

Ledgard, H.F., Ten Mini-Languages: A Study of Topical Issues in Programming Languages, ACM Computing Surveys, Vol. 3, No. 3, September 1971.

------------, Production Systems: Or can we do better than BNF? Comm. ACM, Vol. 17, No. 2, February 1974.

Lee, J.A.N., Computer Semantics, Van Nostrand Reinhold Pub. Co., New York, 1972.

---------, and Dorocak, J., Conditional Syntactic Specification, Proc. ACM '73, ACM, New York, 1973.

---------, Beckhardt, S.R., and Karshmer, A.I., A Candidate Standard for Fundamental BASIC, National Bureau of Standards, NBS-GCR 73-17, July 1973.

London, R.L., Correctness of a Compiler for a LISP Subset, Proc. ACM Conf. on Proving Assertions about Programs, Las Cruces, NM, SIGPLAN Notices, Vol. 7, No. 1, January 1972.

Lucas, P., and Walk, K., On the Formal Description of PL/I, Annual Review in Automatic Programming, Vol. 6, Part 3, Pergammon Press, Oxford, U.K., 1969.

Marcotty, M., Ledgard, H.F., and Bochmann, G.V., A Sampler of Formal Definitions, ACM Computing Surveys, Vol. 8, No. 2, June 1976.

McCarthy, J., A Basis for a Mathematical Theory of Computation, in Braffort, P., and Hirschberg, D., (Eds), Computer Programming and Formal Systems, North Holland Publishing Company, Amsterdam, 1970.

Murach, M., Standard COBOL, 2nd. Ed., Science Research Associates, Inc., Chicago, 1975.

Strachey, C., The Varieties of Programming Language, Oxford University Computing Laboratory, Programming Research Group, Tech. Memo. PRG-10, March 1973.