INTERNATIONAL STANDARD

ISO/IEC 14977

First edition
1996-12-15

Information technology — Syntactic metalanguage — Extended BNF

Technologies de l'information — Métalangage syntaxique — BNF étendu

Reference number
ISO/IEC 14977:1996(E)

Contents

Page

Foreword iv

Introduction v

1 Scope 1

2 Normative references 1

3 Definitions 1

4 The form of each syntactic element of Extended BNF 1
4.1 General 2
4.2 Syntax 2
4.3 Syntax-rule 2
4.4 Definitions-list 2
4.5 Single-definition 2
4.6 Syntactic-term 2
4.7 Syntactic exception 2
4.8 Syntactic-factor 2
4.9 Integer 2
4.10 Syntactic-primary 2
4.11 Optional-sequence 3
4.12 Repeated sequence 3
4.13 Grouped sequence 3
4.14 Meta-identifier 3
4.15 Meta-identifier-character 3
4.16 Terminal-string 3
4.17 First-terminal-character 3
4.18 Second-terminal-character 3
4.19 Special-sequence 3
4.20 Special-sequence-character 3
4.21 Empty-sequence 3
4.22 Further examples 3

ISO/IEC Copyright Office • Case Postale 56 • CH-1211 Genève 20 • Switzerland
Printed in Switzerland

5 The symbols represented by each syntactic element 3
5.1 General 3
5.2 Terminal-string 4
5.3 Meta-identifier 4
5.4 Grouped-sequence 4
5.5 Optional-sequence 4
5.6 Repeated-sequence 4
5.7 Syntactic-factor 4
5.8 Syntactic-term 4
5.9 Single-definition 5
5.10 Definitions-list 5
5.11 Special-sequence 5
5.12 Empty-sequence 5

6 Layout and Comments 5
6.1 General 5
6.2 Terminal-character 5
6.3 Gap-free-symbol 6
6.4 Gap-separator 6
6.5 Commentless-symbol 6
6.6 Comment-symbol 6
6.7 Bracketed-textual-comment 6

7 The representation of each terminal-character in Extended BNF . . 6
7.1 General 6
7.2 Letters and digits 6
7.3 Other terminal characters 6
7.4 Alternative representations 6
7.5 Other-character 7
7.6 Gap-separator 7
7.7 Terminal-characters represented by a pair of characters 8
7.8 Invalid character sequences 8

8 Examples 8
8.1 The syntax of Extended BNF 8
8.2 Extended BNF used to define itself informally 10
8.3 Extended BNF defined informally 10

Annexes

A Two-level grammars 11

B Bibliography 12

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

In the field of information technology, ISO and IEC have established a joint technical committee ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75% of the national bodies casting a vote.

International Standard ISO/IEC 14977 was prepared by BSI (as BS 6154) and was adopted, under a special "fast-track procedure", by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, in parallel with its approval by national bodies of ISO and IEC.

Annexes A and B of this International Standard are for information only.

Introduction

A syntactic metalanguage is an important tool of computer science. The concepts are well known, but many slightly different notations are in use. As a result syntactic metalanguages are still not widely used and understood, and the advantages of rigorous notations are unappreciated by many people.

Extended BNF brings some order to the formal definition of a syntax and will be useful not just for the definition of programming languages, but for many other formal definitions.

Since the definition of the programming language Algol 60 (Naur, 1960) the custom has been to define the syntax of a programming language formally. Algol 60 was defined with a notation now known as BNF or Backus-Naur Form. This notation has proved a suitable basis for subsequent languages but has frequently been extended or slightly altered. The many different notations are confusing and have prevented the advantages of formal unambiguous definitions from being widely appreciated. The syntactic metalanguage *Extended BNF* described in this standard is based on Backus-Naur Form and includes the most widely adopted extensions.

Syntactic metalanguages

A syntactic metalanguage is a notation for defining the syntax of a language by use of a number of rules. Each rule names part of the language (called a non-terminal symbol of the language) and then defines its possible forms. A terminal symbol of the language is an atom that cannot be split into smaller components of the language. A syntactic metalanguage is useful whenever a clear formal description and definition is required, e.g. the format for references in papers submitted to a journal, or the instructions for performing a complicated task.

A formal syntax definition has three distinct uses:

a) it names the various syntactic parts (i.e. non-terminal symbols) of the language;

b) it shows which sequences of symbols are valid sentences of the language;

c) it shows the syntactic structure of any sentence of the language.

The need for a standard syntactic metalanguage

Without a standard syntactic metalanguage every programming language definition starts by specifying the metalanguage used to define its syntax. This causes various problems:

Many different notations — It is unusual for two different programming languages to use the same metalanguage. Thus human readers are handicapped by having to learn a new metalanguage before they can study a new language.

Concepts not widely understood — The lack of a standard notation hinders the use of rigorous unambiguous definitions.

Imperfect notations — Because a metalanguage needs to be defined for every programming language, almost inevitably, the metalanguage contains defects. For example errors occurred in the drafting of RTL/2 (BS5904) and CORAL 66 (BS5905) because the metalanguages could not be typed easily.

Special purpose notations — A metalanguage defined for a particular programming language is often simplified by taking advantage of special features in the language to be defined. However, the metalanguage is then unsuitable for other programming languages.

Few general syntax processors — The multiplicity of syntactic metalanguages has limited the availability of computer programs to analyse and process syntaxes, e.g. to list a syntax neatly, to make an index of the symbols used in the syntax, to produce a syntax-checker for programs written in the language.

In practice experienced readers have little difficulty in picking up and learning a new notation, but even so the differences obscure mutual understanding and hinder communication. A standard metalanguage enables more people to crystallize vague ideas into an unambiguous definition. It is also useful because other people needing to provide formal definitions no longer need to reinvent similar concepts.

The objectives to be satisfied

It is desirable that a standard syntactic metalanguage should be:

a) concise, so that languages can be defined briefly and thus be more easily understood;

b) precise, so that the rules are unambiguous;

c) formal, so that the rules can be parsed, or otherwise processed, by a computer when required;

d) natural, so that the notation and format are relatively simple to learn and understand, even for those who are not themselves language designers; (The meaning of a symbol should not be surprising. It should also be possible to define the syntax of a language in a way that helps to indicate the meaning of the constructions.)

e) general, so that the notation is suitable for many purposes including the description of many different languages;

f) simple in its character set and with a notation that avoids, as far as is practicable, using characters that are not generally available on standard keyboards (both typewriters and computer terminals) so that the rules can be typed and can be processed by computer programs;

g) self describing, so that the notation is able to describe itself;

h) linear, so that the syntax can be expressed as a single stream of characters. (This simplifies printing a syntax. Computer processing of a syntax is also simpler.)

Some common syntactic metalanguages

Unfortunately none of the existing syntactic metalanguages was suitable for adoption as the standard, for example:

a) COBOL (ISO 1989:1985) lists alternatives vertically and uses brackets spreading over many lines. This is inconvenient for computer processing and cannot be prepared on typewriters.

b) Backus-Naur Form (used in ALGOL 60) has problems if the metasymbols < > | ::= occur in the language being defined. Some common forms of construction (e.g. comments) cannot be expressed naturally, other constructions (e.g. repetition) are long-winded.

c) The obsolete FORTRAN 77 (ISO 1539:1980) had 'railroad tracks'. These are easy to understand but difficult to prepare and to process on a computer or typewriter. The current version, FORTRAN 90 (ISO/IEC 1539:1991), no longer uses this notation.

Most other languages use a variant of one of these metalanguages. Most of them cannot be candidates for standardization because they use characters not in the language being defined as metasymbols of the metalanguage. This simplifies the metalanguage but prevents it from being used generally.

POSIX (ISO/IEC 9945-2:1993) includes two complementary facilities which both assume an ISO/IEC 646:1991 character set is applicable: LEX permits the definition and lexical analysis of regular expressions, but is inadequate for the description of an arbitrary context-free grammar, and YACC (Yet Another Compiler Compiler) is a parser generator for an LALR(1) grammar.

The standard metalanguage Extended BNF

Extended BNF, the metalanguage defined in this International Standard, is based on a suggestion by Niklaus Wirth (Wirth, 1977) that is based on Backus-Naur Form and that contains the most common extensions, i.e.:

a) Terminal symbols of the language are quoted so that any character, including one used in *Extended BNF*, can be defined as a terminal symbol of the language being defined.

b) [and] indicate optional symbols.

c) { and } indicate repetition.

d) Each rule has an explicit final character so that there is never any ambiguity about where a rule ends.

e) Brackets group items together. It is an obvious convenience to use (and) in their ordinary mathematical sense.

The main differences in *Extended BNF* are further features that experience has shown are often required when providing a formal definition:

a) *Defining an explicit number of items.* Fortran contains a rule that a label field contains exactly five characters; an identifier in PL/I or COBOL has up to 32 characters: rules such as these can be expressed only with difficulty in Backus-Naur Form. In practice, such definitions are often left incomplete and the rules qualified informally in English.

b) *Defining something by specifying the few exceptional cases.* An Algol **end**-comment ends at the first **end**, **else** or semicolon. A rule like this cannot be expressed concisely or clearly in Backus-Naur Form and is also usually specified informally in English.

c) *Including comments.* Programming languages and other structures with a complicated syntax need many rules to define them. The syntax will be clearer if explanations and cross-references can be provided; accordingly *Extended BNF* contains a comment facility so that ordinary text can be added to a syntax for the benefit of a human reader without affecting the formal meaning of the syntax.

d) *Meta-identifier.* A meta-identifier (the name of a non-terminal symbol in the language) need not be a single word or enclosed in brackets because there is an explicit concatenate symbol. This also ensures that the layout of a syntax (except in a terminal symbol) does not affect the language being defined.

e) *Extensions.* A user may wish to extend *Extended BNF*. A special-sequence is provided for this purpose, the format and meaning of which are not defined in the standard except to ensure that the start and end of an extension can always be seen easily. Various possible extensions are outlined in the following paragraphs.

Limitations and extensions

The main limitation of *Extended BNF* is that the language being defined needs to be linear, i.e. the symbols in a sentence of the language can be placed in an ordered sequence. For example knitting patterns and recipes in cooking are linear languages, but electric circuit diagrams are not.

A further limitation is that *Extended BNF* is inadequate for defining more complex forms of grammars. Such facilities were not provided because it was thought the main need was to define a notation sufficient for the simpler and commoner requirements.

Instead *Extended BNF* has been designed so that various extensions can be made in a natural way. There are two simple ways of extending the standard metalanguage. Firstly, the special-sequence concept provides a basic framework for any extension, the format between the special-sequence-characters being almost completely arbitrary. This method would be suitable for an action grammar, i.e. one specifying actions that are to take place as a sentence is parsed. Secondly, a meta-identifier can never be followed immediately by a left parenthesis in the standard metalanguage; thus another method of extending the metalanguage is to define the syntax and meaning of a meta-identifier followed by a sequence of parameters enclosed in parentheses. This would be reasonable in an attribute grammar where the rules ensure consistency between different parts of a sentence in the language being defined.

More complicated extensions are also possible. Annex A suggests how *Extended BNF* might be extended to define a two-level grammar.

Information technology — Syntactic metalanguage — Extended BNF

1 Scope

This International Standard defines a notation, *Extended BNF*, for specifying the syntax of a linear sequence of symbols. It defines both the logical structure of the notation and its graphical representation.

Extended BNF has applications in the definition of programming and other languages, as well as in other formal definitions, for example the commands to an operating system, or the precise format of data and results.

Examples of *Extended BNF* are given in clause 8.

NOTE — Like many other notations, *Extended BNF* can still be misused; thus it does not prevent someone from trying to define an unparsable or ambiguous language.

2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this International Standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this International Standard are encouraged to investigate the possibility of applying the most recent editions of the standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO 2382-15 : 1985, *Data processing — Vocabulary — Part 15: Programming languages.*

ISO/IEC 646 : 1991, *Information technology — ISO 7-bit coded character set for information interchange.*

ISO/IEC 6429 : 1992, *Information technology — Control functions for coded character sets.*

BS 6154 : 1981, *Method of defining — Syntactic metalanguage.*