Babbage died in London on 18 October 1871. His youngest son, Henry, who had spent most of his life in various military and civil appointments in India, did what he could to carry on his father's work, and published a collection of papers relating to it. The eldest son, Herschel, migrated in 1851 to South Australia, where he became a prominent member of the colony.

Bibliography

1864. Babbage, C. *Passages from the Life of a Philosopher.* London, 1864; facsimile edition, London, 1968; reprint (ed. M. Campbell-Kelly) London: Pickering, 1994.
1889. Babbage, H. P. (ed.) *Babbage's Calculating Engines.* London. This book has now been reprinted as vol. 2 of the Babbage Institute reprint series, I. Tomash (ed.), Cambridge, MA: MIT Press, 1984.
1971. Wilkes, M. V. "Babbage as a Computer Pioneer," Report of the Babbage Memorial Meeting, British Computer Society. Reprinted in *Historica Mathematica*, **4**, 415 (1977).
1982. Hyman, A. *Charles Babbage.* Princeton, NJ: Princeton University Press.
1989. Campbell-Kelly, M. (ed.) *The Works of Charles Babbage.* 11 volumes. London: Pickering and Chatto.
1990. Wilkes, M. V. "Herschel, Peacock, Babbage, and the Development of the Cambridge Curriculum," *Notes and Records of the Royal Society*, **44**, 205.

Maurice V. Wilkes

BACKUS–NAUR FORM (BNF)

For articles on related subjects *see* ALGOL; GRAMMARS; METALANGUAGE; PROCEDURE-ORIENTED LANGUAGES: SURVEY; PROGRAMMING LINGUISTICS; SYNTAX, SEMANTICS, AND PRAGMATICS; and VIENNA DEFINITION LANGUAGE.

Backus–Naur Form, named after John W. Backus of the US and Peter Naur of Denmark, and usually written BNF, is the best-known example of a *metalanguage* (*q.v.*), i.e. one that syntactically describes a programming language. Using BNF it is possible to specify which sequences of symbols constitute a syntactically valid program in a given language. (The question of *semantics*—i.e. what such valid strings of symbols mean—must be specified separately.) A discussion of the basic concepts of BNF follows.

A *metalinguistic variable* (or *metavariable*), also called a *syntactic unit*, is one whose values are strings of symbols chosen from among the symbols permitted in the given language. In BNF, metalinguistic variables are enclosed in brackets, ⟨ ⟩, for clarity and to distinguish them from symbols in the language itself, which are called terminal symbols or just *terminals*. The symbol `::=` is used to indicate metalinguistic equivalence; a vertical bar (|) is used to indicate that a choice is to be made among the items so indicated; and concatenation (linking together in a series) is indicated simply by juxtaposing the elements to be concatenated. In the form that this notation was introduced in the *Algol Report*, a *metalinguistic variable* name should be a word or phrase that describes a language element (e.g. unsigned integer), and that is also used in presenting the semantics of that element. Our examples follow this practice.

For an example, here is how the definition of an Algol (*q.v.*) integer is built up. First, we have a definition of what a digit is, according to the usual meaning:

```
⟨digit⟩ ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

Next we have a statement that an unsigned integer consists either of a single digit or an unsigned integer followed by another digit:

```
⟨unsigned integer⟩ ::= ⟨digit⟩
                     | ⟨unsigned integer⟩⟨digit⟩
```

This definition may be applied *recursively* to build up unsigned integers of any length whatever. Since there must be a limit on the number of digits in any actual computer implementation, this would have to be stated separately in conjunction with each particular implementation or, as in some extensions to BNF, by an addition to the definition of *unsigned integer* (e.g. placing [10] above `::=` could indicate a limit of 10 digits). Finally, the definition of an integer is completed by noting that it may be preceded by a plus sign, a minus sign, or neither:

```
⟨integer⟩ ::= ⟨unsigned integer⟩
            | + ⟨unsigned integer⟩
            | - ⟨unsigned integer⟩
```

For a second example, suppose that the metalinguistic variables ⟨unsigned number⟩, ⟨variable⟩, ⟨function designator⟩, and ⟨Boolean expression⟩ have all been defined earlier, with their usual meanings, and that the vertical arrow stands for exponentiation. Here, then, is the complete sequence of definitions that culminates with the definition of an Algol ⟨arithmetic expression⟩:

```
⟨adding operator⟩ ::= + | -
⟨multiplying operator⟩ ::= × | / | ÷
⟨primary⟩ ::= ⟨unsigned number⟩
    | ⟨variable⟩
    | ⟨function designator⟩
    | (⟨arithmetic expression⟩)
⟨factor⟩ ::= ⟨primary⟩
    | ⟨factor⟩↑⟨primary⟩
⟨term⟩ ::= ⟨factor⟩
    | ⟨term⟩⟨multiplying operator⟩⟨factor⟩
⟨simple arithmetic expression⟩ ::=
      ⟨term⟩
    | ⟨adding operator⟩⟨term⟩
    | ⟨simple arithmetic expression⟩
        ⟨adding operator⟩⟨term⟩
⟨if clause⟩ ::= if
      ⟨boolean expression⟩ then
⟨arithmetic expression⟩ ::=
      ⟨simple arithmetic expression⟩
    | ⟨if clause⟩
      ⟨simple arithmetic expression⟩
        else ⟨arithmetic expression⟩
```

It is no error that the third definition contains ⟨arithmetic expression⟩, enclosed in parentheses, even though it is ⟨arithmetic expression⟩ that we are trying to define. This is another example of a recursive definition, and simply says in this case that one choice for a ⟨primary⟩ is just any ⟨arithmetic expression⟩ enclosed in parentheses.

The words **if**, **then**, and **else**, since they are not enclosed in metalinguistic brackets, stand for themselves; they are, like the character set, basic elements of the Algol language that are not further defined.

An extended version of BNF (EBNF) is used in the *Pascal User Manual and Report* (Jensen *et al.*, 1985) and in the definition of Modula-2 (Wirth, 1985). An EBNF specification of the syntax of a programming language consists of a collection of rules (*productions*), collectively called a *grammar*, that describe the formation of sentences in the language. Each production consists of a nonterminal symbol and an EBNF expression separated by an equal sign and terminated by a period. The nonterminal symbol is a *metaidentifier* (a syntactic constant denoted by an English word), and the EBNF expression is its definition. An EBNF expression is composed of zero or more terminal symbols, nonterminals, and metasymbols, summarized in Table 1.

The superficial difference between BNF and EBNF is that, in the former, nonterminals need to be delimited (by angle brackets) and terminals are allowed to stand for themselves, whereas in EBNF terminals must be delimited (by quotation marks) and nonterminals are allowed to stand for themselves. The more profound difference is that the bracket and brace notation of EBNF allows a simpler presentation of definitions that must be expressed recursively in BNF. Consider, for example, these contrasting definitions of a Pascal ⟨identifier⟩:

```
BNF: ⟨identifier⟩ ::= ⟨letter⟩
                    | ⟨identifier⟩ ⟨letter⟩
                    | ⟨identifier⟩ ⟨digit⟩
```

Table 1. Metasymbols in EBNF

Metasymbol	Meaning
=	is defined to be
|	alternatively
.	end of production
[X]	0 or 1 instance of X
{X}	0 or more instances of X
(X|Y)	a grouping: either X or Y
"XYZ"	the terminal symbol XYZ
MetaIdentifier	the nonterminal symbol MetaIdentifier

Meaning: an identifier is either a single letter or else something that is already a valid identifier followed by either a letter or a digit.

```
EBNF: Identifier = Letter
                   {Letter | Digit}.
```

Meaning: an identifier is a single letter followed by any number of letters or digits, possibly none.

The information conveyed through an EBNF production can be displayed pictorially through the use of *syntax diagrams* (*railroad diagrams*), a technique popularized by the Pascal report of Jensen *et al.* (1985). The syntax diagram for a Pascal identifier is shown in Fig. 1a.

The term "railroad diagram" stems from comparison of the directed line segments to tracks along which a train can proceed on any physically realizable path. Since the train cannot avoid passing through ⟨letter⟩, an identifier consists of at least a single letter. Thereafter, the train may pass straight through to the right, indicating that a single letter is a sufficiently complete definition of an ⟨identifier⟩, or else it may traverse either the top or bottom circuits any number of times in arbitrary order (before finally using the main track to end the definition).

A more complex syntax diagram, one that defines a ⟨signed real⟩ number in Pascal, is shown in Fig. 1b.

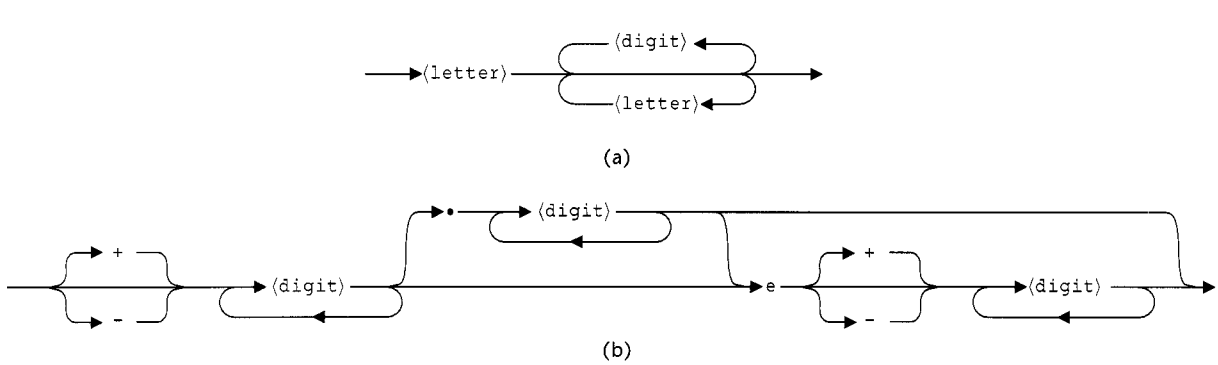

Figure 1. (a) ⟨identifier⟩ (b) ⟨signed real⟩.

The diagram can be used to show that such character strings as `5.7`, `-19.0`, `+3.9`, `-7.36e-3`, and `123e4` are valid signed real numbers but that `-7.`, `.2`, and `123.e4` are not.

BNF (or EBNF or syntax diagrams) can be used to describe any context-free language (*see* GRAMMARS), and hence, since most programming languages are context-free, most programming languages. There are competitive notations, however, such as the Vienna Definition Language (*q.v.*), used to describe PL/I, and the notation typically used to describe Cobol, which is closer to EBNF than to BNF.

Bibliography

1985. Jensen, K., Wirth, N., Mickel, A. B., and Miner, J. F. *Pascal User Manual and Report, 3rd Ed.: ISO Pascal Standard.* New York: Springer-Verlag.

1985. Wirth, N. *Programming in Modula-2, Third Corrected Edition.* New York: Springer-Verlag.

Daniel D. McCracken and Edwin D. Reilly

BANDWIDTH

For articles on related subjects *see* COMMUNICATIONS AND COMPUTERS; and DATA COMMUNICATIONS.

The *bandwidth* of an analog communication network is a measure of the range of frequencies it can transmit at or near maximum power levels. As an example, consider a normal telephone system, which still has analog parts in its communication network (line to the customer) normally designed to carry voice traffic in the frequency range 300–3,400 Hz. The equipment in the telephone exchange collects incoming data from the sound spectrum and arranges to attenuate the signals sharply outside that part of the spectrum. Within the preserved range, however, there is still attenuation, since the power of signals passing through the telephone transmission system is reduced. A typical measurement of attenuation on the US telephone network is shown in Fig. 1, which indicates that somewhere below 300 Hz and above 3–4 KHz, the attenuation rises very rapidly. The range of frequencies in which the power level stays at above one-half its peak value (the so-called 3 dB (decibel) points) is the *nominal bandwidth* of the circuit. This is typically 3 KHz in a switched telephone line.

The term *bandwidth* is now also applied—somewhat erroneously—to a digital communication network. In this context the term means the speed at which digital data can be transmitted. Thus, in digital communications, bandwidth has become a synonym for what had been known as the information *transfer rate.* For example, the bandwidth or capacity of an Ethernet network may be 100 (or more) Mb/sec. The digital

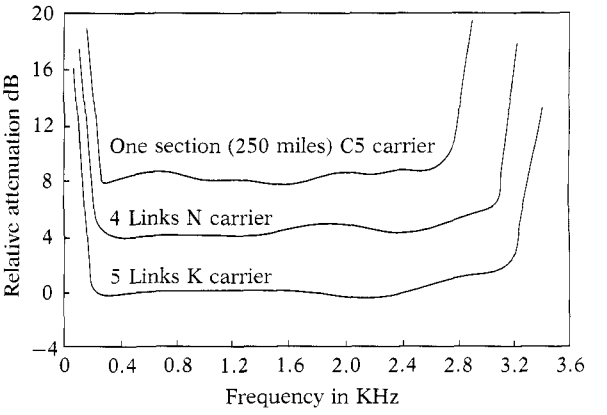

Figure 1. Attenuation for frequency division multiplexing (FDM) systems (reproduced from *Communication-Networks for Computers* by D. W. Davies and D. L. A. Barber, New York: Wiley, 1973, Fig. 2.16).

bandwidth of a channel, or *capacity*, depends directly upon the analog or "electrical" bandwidth of the communication medium and the channel's noise levels. The channel capacity is governed by the Hartley–Shannon law of information theory (*q.v.*).

Bibliography

1999. Bateman, A. *Digital Communications: Design for the Real World.* Reading, MA: Addison-Wesley.

Vicky J. Hardman and Peter T. Kirstein

BAR CODE

See UNIVERSAL PRODUCT CODE.

BASIC

For articles on related subjects, *see* PROGRAMMING LANGUAGES; PROCEDURE-ORIENTED LANGUAGES; STRUCTURED PROGRAMMING; and TIME-SHARING.

The Birth of Basic

Basic (Beginner's All-purpose Symbolic Instruction Code) was invented at Dartmouth College in 1964 to allow all students, especially those with no interest in science, to learn about the computer, i.e. to write simple programs. The students used Basic on a time-sharing system, which allowed them to reach the computer using terminals in their dorms. (The reader should realize that almost all computer work in those days required punched cards to be carried to the computer center. The only terminals were electric typewriters, such as the Teletype, which printed at 10 characters per second. There were no video terminals or personal computers.)