Augmented Backus–Naur form

In computer science, **augmented Backus–Naur form** (**ABNF**) is a metalanguage based on Backus–Naur form (BNF), but consisting of its own syntax and derivation rules. The motive principle for ABNF is to describe a formal system of a language to be used as a bidirectional communications protocol. It is defined by *Internet Standard 68* (https://tools.ietf.org/html/std68) ("STD 68", type case sic), which as of December 2010 is RFC 5234 (https://datatracker.ietf.org/doc/html/rfc5234), and it often serves as the definition language for IETF communication protocols.[1][2]

RFC 5234 (https://datatracker.ietf.org/doc/html/rfc5234) supersedes RFC 4234 (https://datatracker.ietf.org/doc/html/rfc4234), 2234 (https://datatracker.ietf.org/doc/html/rfc2234) and 733 (https://datatracker.ietf.org/doc/html/rfc733).[3] RFC 7405 (https://datatracker.ietf.org/doc/html/rfc7405) updates it, adding a syntax for specifying case-sensitive string literals.

Overview

An ABNF specification is a set of derivation rules, written as

```
rule = definition ; comment CR LF
```

where rule is a case-insensitive nonterminal, the definition consists of sequences of symbols that define the rule, a comment for documentation, and ending with a carriage return and line feed.

Rule names are case-insensitive: `<rulename>`, `<Rulename>`, `<RULENAME>`, and `<rUlENamE>` all refer to the same rule. Rule names consist of a letter followed by letters, numbers, and hyphens.

Angle brackets (`<`, `>`) are not required around rule names (as they are in BNF). However, they may be used to delimit a rule name when used in prose to discern a rule name.

Terminal values

Terminals are specified by one or more numeric characters.

Numeric characters may be specified as the percent sign `%`, followed by the base (`b` = binary, `d` = decimal, and `x` = hexadecimal), followed by the value, or concatenation of values (indicated by `.`). For example, a carriage return is specified by `%d13` in decimal or `%x0D` in hexadecimal. A carriage return followed by a line feed may be specified with concatenation as `%d13.10`.

Literal text is specified through the use of a string enclosed in quotation marks (`"`). These strings are case-insensitive, and the character set used is (US-)ASCII. Therefore, the string `"abc"` will match “abc”, “Abc”, “aBc”, “abC”, “ABc”, “AbC”, “aBC”, and “ABC”. RFC 7405 added a syntax for case-sensitive

strings: `%s"aBc"` will only match "aBc". Prior to that, a case-sensitive string could only be specified by listing the individual characters: to match “aBc”, the definition would be `%d97.66.99`. A string can also be explicitly specified as case-insensitive with a `%i` prefix.

Operators

White space

White space is used to separate elements of a definition; for space to be recognized as a delimiter, it must be explicitly included. The explicit reference for a single whitespace character is `WSP` (linear white space), and `LWSP` is for zero or more whitespace characters with newlines permitted. The `LWSP` definition in RFC5234 is controversial[4] because at least one whitespace character is needed to form a delimiter between two fields.

Definitions are left-aligned. When multiple lines are required (for readability), continuation lines are indented by whitespace.

Comment

```
; comment
```

A semicolon (`;`) starts a comment that continues to the end of the line.

Concatenation

```
Rule1 Rule2
```

A rule may be defined by listing a sequence of rule names.

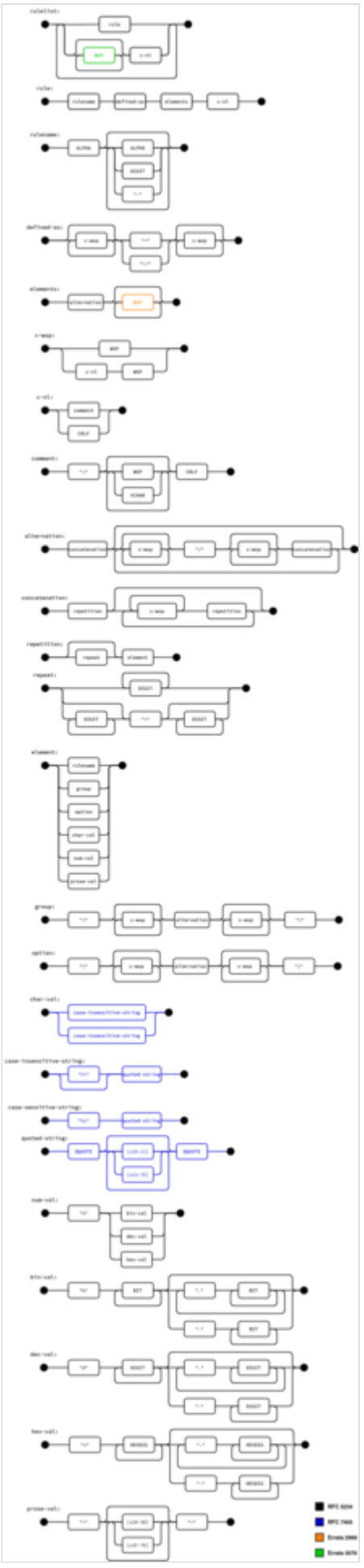
To match the string “aba”, the following rules could be used:

- `fu = %x61 ; a`
- `bar = %x62 ; b`
- `mumble = fu bar fu`

Alternative

```
Rule1 / Rule2
```

A rule may be defined by a list of alternative rules separated by a solidus (`/`).



ABNF syntax diagram of ABNF rules

To accept the rule *fu* or the rule *bar*, the following rule could be constructed:

- `fubar = fu / bar`

Incremental alternatives

`Rule1 =/ Rule2`

Additional alternatives may be added to a rule through the use of `=/` between the rule name and the definition.

The rule

- `ruleset = alt1 / alt2`
- `ruleset =/ alt3`
- `ruleset =/ alt4 / alt5`

is therefore equivalent to

- `ruleset = alt1 / alt2 / alt3 / alt4 / alt5`

Value range

`%c##-##`

A range of numeric values may be specified through the use of a hyphen (`-`).

The rule

- `OCTAL = %x30-37`

is equivalent to

- `OCTAL = "0" / "1" / "2" / "3" / "4" / "5" / "6" / "7"`

Sequence group

`(Rule1 Rule2)`

Elements may be placed in parentheses to group rules in a definition.

To match "a b d" or "a c d", the following rule could be constructed:

- `group = a (b / c) d`

To match “a b” or “c d”, the following rules could be constructed:

- `group = a b / c d`

- `group = (a b) / (c d)`

Variable repetition

`n*nRule`

To indicate repetition of an element, the form `<a>*<b>element` is used. The optional `<a>` gives the minimal number of elements to be included (with the default of 0). The optional `<b>` gives the maximal number of elements to be included (with the default of infinity).

Use `*element` for zero or more elements, `*1element` for zero or one element, `1*element` for one or more elements, and `2*3element` for two or three elements, cf. regular expressions `e*`, `e?`, `e+` and `e{2,3}`.

Specific repetition

`nRule`

To indicate an explicit number of elements, the form `<a>element` is used and is equivalent to `<a>* <a>element`.

Use `2DIGIT` to get two numeric digits, and `3DIGIT` to get three numeric digits. (`DIGIT` is defined below under "Core rules". Also see *zip-code* in the example below.)

Optional sequence

`[Rule]`

To indicate an optional element, the following constructions are equivalent:

- `[fubar snafu]`
- `*1(fubar snafu)`
- `0*1(fubar snafu)`

Operator precedence

The following operators have the given precedence from tightest binding to loosest binding:

1. Strings, names formation
2. Comment
3. Value range
4. Repetition
5. Grouping, optional
6. Concatenation
7. Alternative

Use of the alternative operator with concatenation may be confusing, and it is recommended that grouping be used to make explicit concatenation groups.

Core rules

The core rules are defined in the ABNF standard.

Rule	Formal definition	Meaning
ALPHA	%x41–5A / %x61–7A	Upper- and lower-case ASCII letters (A–Z, a–z)
DIGIT	%x30–39	Decimal digits (0–9)
HEXDIG	DIGIT / "A" / "B" / "C" / "D" / "E" / "F"	Hexadecimal digits (0–9, A–F, a-f)
DQUOTE	%x22	Double quote
SP	%x20	Space
HTAB	%x09	Horizontal tab
WSP	SP / HTAB	Space and horizontal tab
LWSP	*(WSP / CRLF WSP)	Linear white space (past newline)
VCHAR	%x21–7E	Visible (printing) characters
CHAR	%x01–7F	Any ASCII character, excluding NUL
OCTET	%x00–FF	8 bits of data
CTL	%x00–1F / %x7F	Controls
CR	%x0D	Carriage return
LF	%x0A	Linefeed
CRLF	CR LF	Internet-standard newline
BIT	"0" / "1"	Binary digit

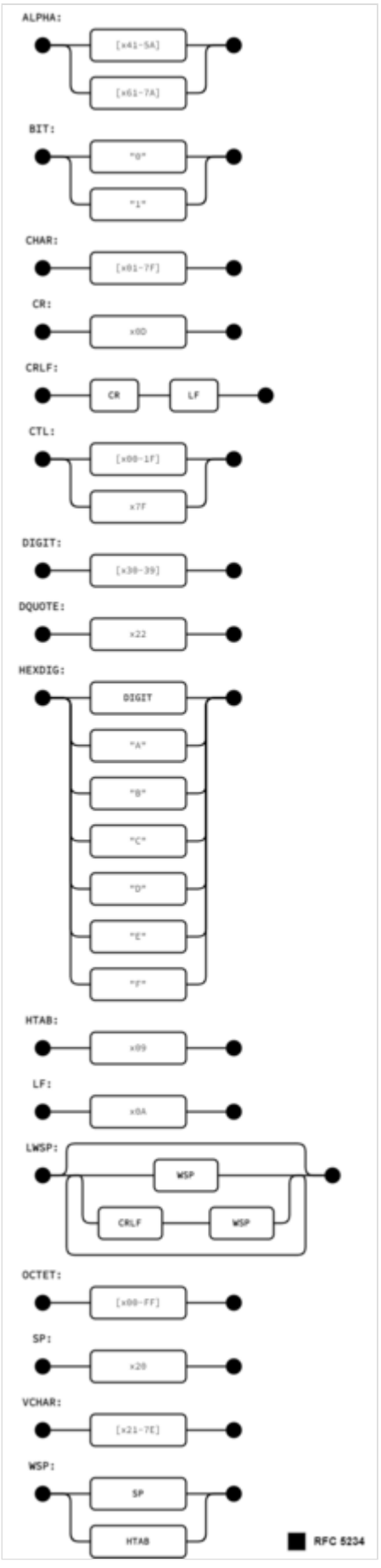
Note that in the core rules diagram the **CHAR2** charset is inlined in **char-val** and **CHAR3** is inlined in **prose-val** in the RFC spec. They are named here for clarity in the main syntax diagram.

Example

The (U.S.) postal address example given in the augmented Backus–Naur form (ABNF) page may be specified as follows:

```
postal-address   = name-part street zip-part

name-part        = *(personal-part SP) last-name [SP suffix] CRLF
name-part        =/ personal-part CRLF

personal-part    = first-name / (initial ".")
first-name       = *ALPHA
initial          = ALPHA
last-name        = *ALPHA
suffix           = ("Jr." / "Sr." / 1*("I" / "V" / "X"))

street           = [apt SP] house-num SP street-name CRLF
apt              = 1*4DIGIT
house-num        = 1*8(DIGIT / ALPHA)
```

```
street-name         = 1*VCHAR

zip-part            = town-name "," SP state 1*2SP zip-code
CRLF
town-name           = 1*(ALPHA / SP)
state               = 2ALPHA
zip-code            = 5DIGIT ["-" 4DIGIT]
```



ABNF syntax diagram of core rules

Pitfalls

RFC 5234 (http://www.ietf.org/rfc/rfc5234.txt) adds a warning in conjunction to the definition of LWSP as follows:

> Use of this linear-white-space rule permits lines containing only white space that are no longer legal in mail headers and have caused interoperability problems in other contexts. Do not use when defining mail headers and use with caution in other contexts.

References

1. "Official Internet Protocol Standards" (http://www.rfc-editor.org/rfcxx00.html). RFC Editor. 2010-02-21. Archived (https://web.archive.org/web/20100209035909/http://www.rfc-editor.org/rfcxx00.html) from the original on 9 February 2010. Retrieved 2010-02-21.
2. Crocker, D.; Overell, P. (January 2008). "Augmented BNF for Syntax Specifications: ABNF" (http://ftp.rfc-editor.org/in-notes/std/std68.txt) (plain text). RFC Editor. p. 16. Retrieved 2010-02-21.
3. "RFC Index" (http://www.rfc-editor.org/rfc-index2.html). RFC Editor. 2010-02-19. Archived (https://web.archive.org/web/20100209041834/http://www.rfc-editor.org/rfc-index2.html) from the original on 9 February 2010. Retrieved 2010-02-21.
4. RFC Errata 3096 (http://www.rfc-editor.org/errata_search.php?rfc=5234&eid=3096).