Syntax diagram

Syntax diagrams (or **railroad diagrams**) are a way to represent a context-free grammar. They represent a graphical alternative to Backus–Naur form, EBNF, Augmented Backus–Naur form, and other text-based grammars as metalanguages. Early books using syntax diagrams include the "Pascal User Manual" written by Niklaus Wirth[1] (diagrams start at page 47) and the Burroughs CANDE Manual.[2] In the compilation field, textual representations like BNF or its variants are usually preferred. BNF is text-based, and used by compiler writers and parser generators. Railroad diagrams are visual, and may be more readily understood by laypeople, sometimes incorporated into graphic design. The canonical source defining the JSON data interchange format provides yet another example of a popular modern usage of these diagrams.

Contents

Principle of syntax diagrams

The representation of a grammar is a set of syntax diagrams. Each diagram defines a "nonterminal" stage in a process. There is a main diagram which defines the language in the following way: to belong to the language, a word must describe a path in the main diagram.

Each diagram has an entry point and an end point. The diagram describes possible paths between these two points by going through other nonterminals and terminals. Historically, terminals have been represented by round boxes and nonterminals by rectangular boxes but there is no official standard.

Example

We use arithmetic expressions as an example, in various grammar formats.

BNF:

```
<expression> ::= <term> | <term> "+" <expression>
<term>       ::= <factor> | <factor> "*" <term>
<factor>     ::= <constant> | <variable> | "(" <expression> ")"
<variable>   ::= "x" | "y" | "z"
<constant>   ::= <digit> | <digit> <constant>
<digit>      ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
```

EBNF:

```
expression = term , [ "+" , expression ];
term       = factor , [ "*" , term ];
factor     = constant | variable | "(" , expression , ")";
variable   = "x" | "y" | "z";
constant   = digit , { digit };
digit      = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9";
```

ABNF:

```
expression = term ["+" expression]
term       = factor ["*" term]
factor     = constant / variable / "(" expression ")"
variable   = "x" / "y" / "z"
constant   = 1*digit
DIGIT      = "0" / "1" / "2" / "3" / "4" / "5" / "6" / "7" / "8" / "9"
```

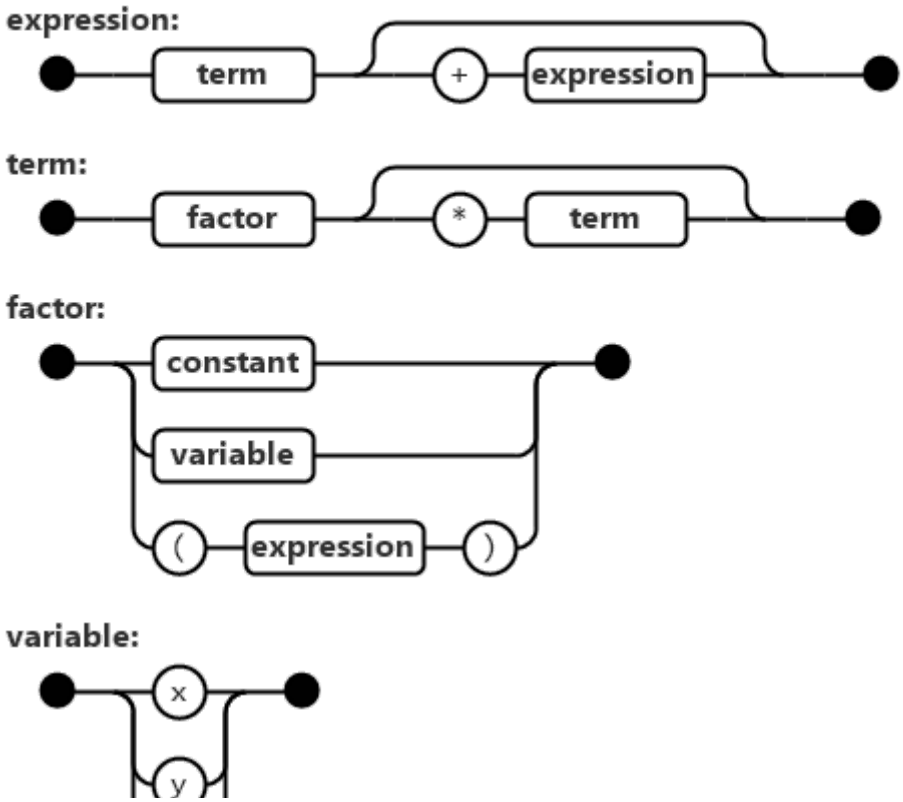
Note that ABNF also supports ranges, e.g. `DIGIT = %x30-39`, but it is not used here for consistency with the other examples.

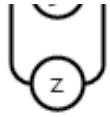
Red (programming language) Parse Dialect:

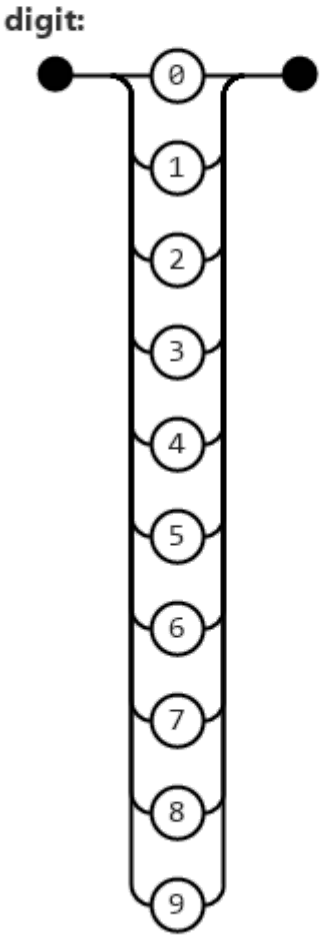
```
Red [Title: "Parse Dialect"]
expression: [term opt ["+" expression]]
term:       [factor opt ["*" term]]
factor:     [constant | variable | "(" expression ")"]
variable:   ["x" | "y" | "z"]
constant:   [some digit]
digit:      ["0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"]
```

Note that this format also supports ranges, e.g. `digit: charset [#"0" - #"9"]`, but it is not used here for consistency with the other examples.

One possible syntax diagram for the example grammars is below. Note that while the syntax for the text-based grammars differs, the syntax diagram for all of them can be the same because it is a metalanguage.







See also

- Recursive transition network
- Extended Backus–Naur form (EBNF)

References

1. Niklaus Wirth: *The Programming Language Pascal.* (July 1973) (http://e-collection.library.ethz.ch/eserv/eth:3059/eth-3059-01.pdf)
2. Burroughs B6700/B7700 *Command AND Edit (CANDE) Language: Information Manual* (http://bitsavers.org/pdf/burroughs/B6500_6700/5000318_B6700_CANDE_Oct72.pdf)

Note: the first link is sometimes blocked by the server outside of its domain, but it is available on archive.org (https://web.archive.org/web/20180909121538/https://www.research-collection.ethz.ch/bitstream/handle/20.500.11850/68910/eth-3059-01.pdf). The file was also mirrored at standardpascal.org (http://www.standardpascal.org/The_Programming_Language_Pascal_1973.pdf).

External links

- JSON website including syntax diagrams (https://www.json.org/)
- Generator from EBNF (http://dotnet.jku.at/applications/Visualizer/)
- From EBNF to a postscript file with the diagrams (http://www.informatik.uni-freiburg.de/~thiemann/haskell/ebnf2ps/)
- EBNF Parser & Renderer (https://karmin.ch/ebnf/index)
- SQLite syntax diagram generator for SQL (https://www.sqlite.org/docsrc/finfo?name=art/syntax/bubble-generator.tcl)
- Online Railroad Diagram Generator (https://bottlecaps.de/rr)
- Augmented Syntax Diagram (ASD) grammars (https://www.yorku.ca/jmason/asdgram.htm)
- (ASD) **Augmented** Syntax Diagram Application Demo Site (http://www.asd-networks.com)
- SRFB Syntax Diagram representation by Function Basis + svg generation (https://github.com/gbrault/railroad-diagrams/blob/gh-pages/live/doc/readme.md/)