Ebnf2ps — Automatic Railroad Diagram Drawing Version 1.01

Peter Thiemann, Michael Walter
Wilhelm-Schickard-Institut, Universität Tübingen
Sand 13, 72076 Tübingen
E-mail: `{thiemann,walterm}@informatik.uni-tuebingen.de`

May 29, 1996

Ebnf2ps was written to fill an urgent need of the author: to produce lots of syntax diagrams (railroad diagrams) from a grammar in EBNF notation in short time. Over the time the program has acquired a couple of features in order to fulfill a variety of requirements. The current version produces highly readable railroad diagrams from grammars in various formats: Yacc, Happy and EBNF.
 Please send any comments, bugs, etc. to the author(s).

1 How to build it

Ebnf2ps was written in Haskell, a non–strict, purely functional programming language. To build Ebnf2ps you need a Haskell Compiler, for example the Glasgow Haskell Compilation System `ghc`[1] or the Haskell Compiler `hbc` by Lennart Augustsson.

1. Edit the Makefile to reflect your Haskell compiler, for example

   ```
   HC= ghc
   ```

2. Find out where your TeX installation stores `.afm`-files (Adobe font metric files). Then you can change the variable `AFMPATH` in the Makefile or you can set the environment variable `AFMPATH` when running the program. The `.afm`-files are absolutely necessary. If you don't find them complain to your system administrator or get them yourself from `ftp://ftp.adobe.com/pub/adobe/AFMFiles/`.

3. Change the variable `RGBPATH` to contain the directory where your X installation stores its color data base (usually `/usr/lib/X11/rgb.txt`). This can also be corrected later with the environment variable `RGBPATH`. There are fallback color definitions for the 8 *digital* colors (black, white, blue, green, cyan, red, magenta, yellow), so nothing needs be done if that's enough for you.

4. Typing `make` should produce an executable file named Ebnf2ps.

[1] GHC home page URL `http://www.dcs.glasgow.ac.uk/fp/software/ghc.html`.

2 How to use it

The syntax is

```
Ebnf2ps [options] Grammarfile Nonterminal ... [-- Nonterminal ...]
```

where the `options` are described in Sec. 2.1, the syntax of the input file `Grammarfile` in Sec. 2.2, and `Nonterminal` describes the nonterminals, for which diagrams are to be generated, as regular expressions (see Sec. 2.3). The nonterminals after the symbol `--` are used only with the option `+unfold` (see Sec. 2.1 and 2.4).

2.1 Options

The following list shows all **options** with their *parameters* and their (default values). All distances and sizes measured in 1/100 point (1 point is about 1/72 inch). Changes with respect to the last version of Ebnf2ps: some ⓝew features, extension of grammar transformations and bug fixes.

-titleFont ***font*** (Times-Roman) PostScript font used for titles of diagrams, any font goes here that your printer knows and for which you have fetched the AFM files. Figure 1 shows a sample list of font names. ⓝ

-titleScale ***int*** (10) Pointsize of typeface for titles of diagrams.ⓝ

-titleColor ***color*** (black) Color of typeface for titles of diagrams. ⓝ

-ntFont ***font*** (Times-Roman) PostScript font used for nonterminals.

-ntScale ***int*** (10) Pointsize of typeface for nonterminals.

-ntColor ***color*** (black) Color of typeface for nonterminals.

-ntBg ***color*** (white) Background color of typeface for nonterminals.ⓝ

-ntBoxColor ***color*** (black) Color used for outlines of boxes (nonterminals).ⓝ

-tFont ***font*** (Times-Roman) PostScript font used for terminal strings.

-tScale ***int*** (10) Pointsize of typeface for terminals.

-tColor ***color*** (black) Color of typeface for terminals.

-tBg ***color*** (white) Background color of typeface for terminals.ⓝ

-tBoxColor ***color*** (black) Color used for outlines of boxes (terminals).ⓝ

-borderDistX ***int*** (500) Horizontal distance of objects from their container.

-borderDistY ***int*** (500) Vertical distance of objects from their container.

-lineWidth ***int*** (30) Width of lines used for connecting lines.

-fatLineWidth ***int*** (100) Width of lines used for drawing boxes.

-lineColor ***color*** (black) Color used for connecting lines.

-arrowSize ***int*** (200) Size of (invisible) box containing an arrow.

-rgbFileName ***filename*** (rgb.txt) File name for color definitions.

-happy Accept happy format input files. Happy, written by Andy Gill and Simon Marlow, is a parser generator system (LALR(1)) for Haskell, similar to the tool yacc for C[2].

-yacc Accept yacc/bison format input files.ⓝ

+ps Produce encapsulated PostScript output (default).

+fig Produce fig output (FORMAT 3.1).

[2]The program Happy is available by FTP from `ftp.dcs.glasgow.ac.uk` in `pub/haskell/happy`

AGaramond-Bold	AGaramond-BoldItalic
AGaramond-Italic	AGaramond-Regular
AGaramond-Semibold	AGaramond-SemiboldItalic
AGaramondAlt-Italic	AGaramondAlt-Regular
AGaramondExp-Bold	AGaramondExp-BoldItalic
AGaramondExp-Italic	AGaramondExp-Semibold
AGaramondExp-SemiboldItalic	AvantGarde-Book
AvantGarde-BookOblique	AvantGarde-Demi
AvantGarde-DemiOblique	Bookman-Demi
Bookman-DemiItalic	Bookman-Light
Bookman-LightItalic	Courier-Bold
Courier-BoldOblique	Courier-Oblique
Courier	Helvetica-Bold
Helvetica-BoldOblique	Helvetica-Narrow-Bold
Helvetica-Narrow-BoldOblique	Helvetica-Narrow-Oblique
Helvetica-Narrow	Helvetica-Oblique
Helvetica	NewCenturySchlbk-Bold
NewCenturySchlbk-BoldItalic	NewCenturySchlbk-Italic
NewCenturySchlbk-Roman	Optima-Bold
Optima	Palatino-Bold
Palatino-BoldItalic	Palatino-Italic
Palatino-Oblique	Palatino-Roman
StoneInformal-Bold	StoneInformal-BoldItalic
StoneInformal-Italic	StoneInformal-Semibold
StoneInformal-SemiboldItalic	StoneInformal
StoneSans	StoneSerif-Bold
StoneSerif-BoldItalic	StoneSerif-Italic
StoneSerif-Semibold	StoneSerif-SemiboldItalic
StoneSerif	Symbol
Times-Bold	Times-BoldItalic
Times-Italic	Times-Oblique
Times-Roman	ZapfChancery-MediumItalic
ZapfDingbats	

Figure 1: Sample list of font names

+ebnf Produce grammar in EBNF notation (file extension ".BNF") for yacc or happy input format files.n

+simplify Simplify productions (experimental, default: don't).

+unfold Replace nonterminals in productions (experimental, default: don't). See Sec. 2.4.n

-verbose Prints some progress messages.

-help Prints a brief description of the command line, the options with their defaults, and the environment variables.

2.2 Syntax of EBNF Files

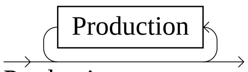
The following grammar and diagrams describe the standard input format (e.g. without **-yacc** or **-happy**) as well as the output format with the **+ebnf** option.

```
File            = {Production};
```

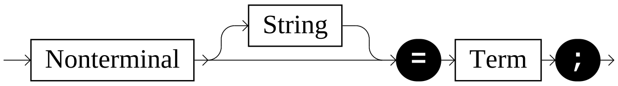
```
Production      = Nonterminal [ String ] "=" Term ";" ;
Term            = Factor / "|" ;          # alternative
Factor          = ExtAtom + ;             # sequence
ExtAtom         = Atom
                | Atom "/" Atom           # repetition through Atom
                | Atom "+";               # at least one repetition
Atom            = Nonterminal
                | String                  # terminal string
                | "(" Term ")"
                | "[" Term "]"            # an optional Term
                | "{" Term "}"            # zero or more repetitions
                ;
String          = "\"" { Character } "\"" ;
Nonterminal     = letter { letter | digit | "_" | "." } ;
Character       = character
                | ["\\"] anycharacter ;
```

Or in terms of diagrams (what would you expect):

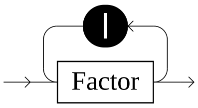
File



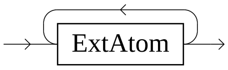
Production



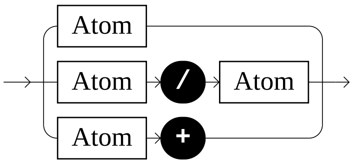
Term



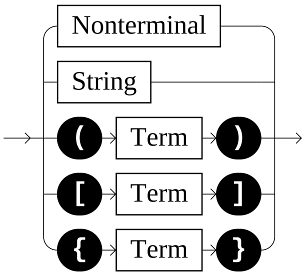
Factor



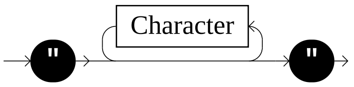
ExtAtom



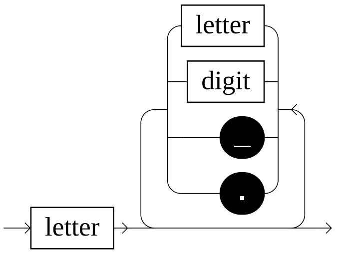
Atom



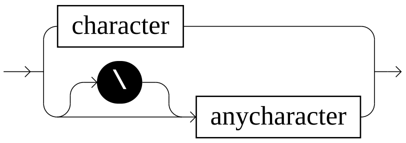
String



Nonterminal



Character



These diagrams have been produced with `Ebnf2ps -tFont Courier-Bold -tBg Black -tColor White -fatLineWidth 50 ebnf.BNF '.*'`.

2.3 Syntax of regular expressions

The regular expressions used to select nonterminal symbols have the following syntax, which is similar to the regular expression syntax of `grep`.

```
Regexp   = RFactor / "\\|" ["$"];
RFactor  = RExtAtom + ;
RExtAtom = RAtom ["*" | "+" | "?"] ;
RAtom    = character | "\\" character | "." | "\\(" Regexp "\\)" ;
```

Regular expressions `Regexp` may be joined by the infix operator `|`. The resulting regular expression matches any string matching either subexpression. A subexpression may be enclosed in parenthesis `\(`, `\)` to override precedence rules. Repetition takes precedence over concatenation, which in turn takes precedence over alternation. A regular expression matching a single character may be followed by one of several repetition operators:

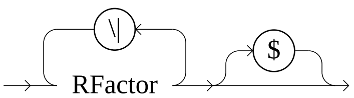
`?` The preceding item is optional and matched at most once.

`*` The preceding item will be matched zero or more times.

`+` The preceding item will be matched one or more times.

The period `.` matches any single character, and the dollar sign `$` match the empty string at the end of a line. Well, graphically, the syntax is[3]:

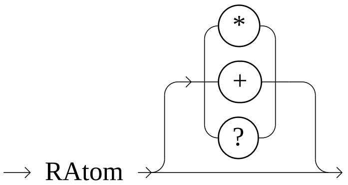
Regexp



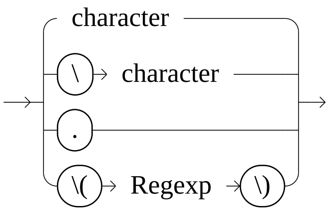
[3]The diagrams have been produced with:
`Ebnf2ps -titleFont AvantGarde-Book -ntBoxColor White -fatLineWidth 50 regular.BNF '.*'`

RFactor

RExtAtom



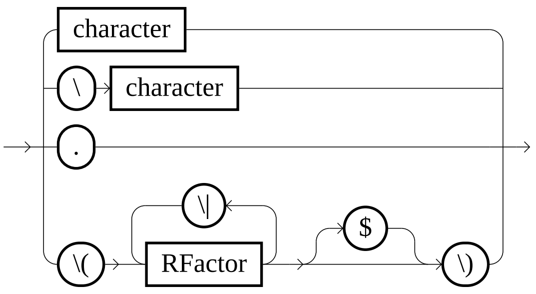
RAtom



2.4 Option +unfold

The option +**unfold** allows the replacement of nonterminal symbols in productions, through the "right side" of the production of corresponding nonterminals. The nonterminals, which follows the symbol `--` in a command line could be specified by their full names, or by regular expressions (see Sec. 2.3). An example: with the grammar `regular.BNF` for the regular expression syntax in section 2.3 should be generated a diagram for `RAtom`, with a replacement of nonterminal `Regexp`. The following diagram is been produced with `Ebnf2ps +unfold regular.BNF RAtom -- Regexp`.

RAtom



The following example illustrates the use of regular expressions with the option `+unfold`. In all productions of the grammar file `ebnf.BNF` — which describes the syntax of EBNF files (see Sec. 2.2) — the nonterminal `String` and all nonterminals that match the pattern

```
.*\(t\|T\)erm.*
```

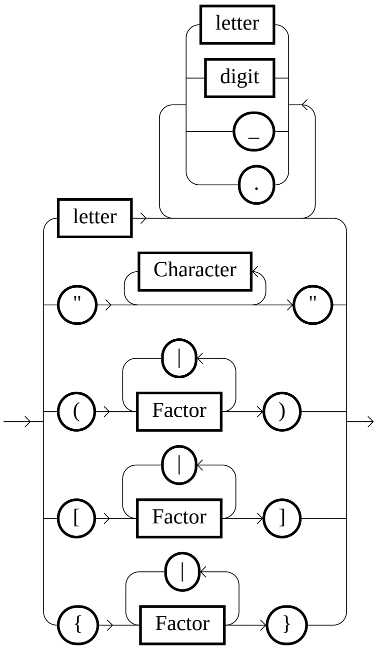
should be replaced. The next lines shows the command to produce the railroad diagrams and the progress messages of Ebnf2ps.

```
% Ebnf2ps -verbose -tScale 8 -ntScale 8 +unfold ebnf.BNF '.*' -- String '.*\(t\|T\)erm.*'
using colors:
```

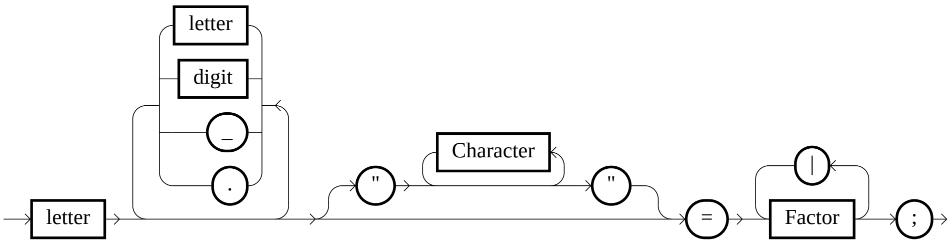
```
        ntColor          (0,0,0)
        tColor           (0,0,0)
        lineColor        (0,0,0)
        ntBoxColor       (0,0,0)
        tBoxColor        (0,0,0)
        ntBg             (255,255,255)
        tBg              (255,255,255)
        titleColor       (0,0,0)
from rgbPathDefault: ["/usr/lib/X11", "/usr/local/X11R5/lib/X11"]
generating nonterminals: [".*"]
from ebnf.BNF
using input path ["."]
using ebnfInput
unfold nonterminals:
        Nonterminal
        String
        Term
in the following productions
        Production
        Atom
produce layout and output
```

Well, graphically that is:

Atom



Production



2.5 Environment variables

The format of search pathes is a colon separated list of directories. The default path is inserted in place of an empty directory in the list. Any file name starting with /

is taken as an absolute name and any file name starting with `.` is taken relative to the current working directory.

AFMPATH search path for Adobe font metric files.

EBNFINPUTS search path for grammar input files.

RGBPATH search path for the color definitions.

3 How to change it

In order to add new options to the program you will need a modified copy of the program `Mkhprog` with kind permission of the author Nick North (`ndn@seg.npl.co.uk`) which is available from URL

`http://www-pu.informatik.uni-tuebingen.de/users/thiemann/haskell/ebnf2ps/`