The backnaur package*

Adrian P. Robson†

12 December 2012

1 Introduction

The backnaur package typesets Backus-Naur Form (BNF) definitions. It creates aligned lists of productions, with numbers if required. It can also print in line BNF expressions using math mode.

Backus-Naur Form is a notation for defining context free grammars. It is used to describe such things as programming languages, communication protocols and command syntaxes, but it can be useful whenever a rigorous definition of language is needed.

2 BNF Definitions

The following is a BNF definition of a semicolon separated list:

$$\begin{aligned} \langle \text{list} \rangle &\models \langle \text{listitems} \rangle \mid \lambda \\ \langle \text{listitems} \rangle &\models \langle \text{item} \rangle \mid \langle \text{item} \rangle \,;\, \langle \text{listitems} \rangle \\ \langle \text{item} \rangle &\models \textit{description of item} \end{aligned}$$

Here, $\models$ signifies *produces*, $\mid$ is an *or* operator, $\langle ... \rangle$ are *production names*, and λ represents the *empty string*. However, some BNF users prefer alternative terminologies, where $\models$ stands for *is defined as*, $\langle ... \rangle$ is a *category name* or *nonterminal*, and λ is refered to as *null* or *empty*.

The above definition was created with the following code:

```
\usepackage{backnaur}
...
\begin{bnf*}
   \bnfprod{list}
           {\bnfpn{listitems} \bnfor \bnfes}\\
   \bnfprod{listitems}
           {\bnfpn{item} \bnfor \bnfpn{item}
            \bnfsp \bnfts{;} \bnfsp \bnfpn{listitems}}\\
   \bnfprod{item}
           {\bnftd{description of item}}
\end{bnf*}
```

*This work replaces *Writing BNF Notation in LaTeX*, which described a non-package method of BNF typesetting. This document corresponds to backnaur 1.1, dated 2012/12/12.

†adrian.robson@nepsweb.co.uk

Each BNF production is defined by a `\bnfprod` command, which has two arguments giving its left and right sides. The right hand side of each production is specified with the commands described in §3. Terminal (`\bnfts{;}`) and non-terminal (`\bnfpn{item}`), elements are separated by spaces (`\bnfsp`) and OR symbols (`\bnfor`). The `\bnfes` command gives the symbol for the empty string.

3 Package Commands

3.1 Loading and options

The package is loaded with

```
\usepackage{backnaur}
```

or

```
\usepackage[<options>]{backnaur}
```

Possible options are

`perp`	The empty string symbol is $\perp$
`epsilon`	The empty string symbol is ϵ
`tstt`	Terminal string typeface is typewriter

The defaults are: the empty string symbol is λ, and the terminal string typeface is normal (roman).

3.2 Environments

`bnf` `bnf*` BNF productions are defined in a `bnf` or `bnf*` environment, which respectively give numbered and unnumbered lists of productions.

```
\begin{bnf}
  <list of productions>
\end{bnf}
```

```
\begin{bnf*}
  <list of productions>
\end{bnf*}
```

3.3 Productions

`\bnfprod` A production is defined by `\bnfprod`, which takes two arguments:

```
\bnfprod{<production name>}{<production definition>}
```

3.4 Production definitions

The following commands are used to compose the right hand side of a production. They are deployed in the second argument of the `\bnfprod` command.

`\bnfpn` The `\bnfpn` command generates a production name. It takes a single argument that is the name. It is used as follows:

```
\bnfpn{list item}
```

⟨list item⟩

`\bnftm` `\bnftd` `\bnfes` There are three types of terminal item: a literal string, a descriptive phrase and an empty string. A literal terminal string is specified by the `\bnftm` command, which takes a single argument. The `\bnftd` command generates a descriptive phrase, as an alternative to a literal string. The `\bnfes` command generates a token that represents the empty string. This is normally λ, but it can be changed as a package option (see §3.1).

`\bnfts{terminal}`	terminal
`\bnftd{description}`	*description*
`\bnfes`	λ

`\bnfsk` Some literal terminal strings can be abbreviated with the 'skip' token, which is generated by the `\bnfsk` command. This substitutes for a sequence of terminal characters. It is used like this:

`\bnfts{A} \bnfsk \bnfts{Z}`	A . . . Z

`\bnfor` `\bnfsp` All items are separated by an OR or a space. The `\bnfor` command generates the OR symbol, and the `\bnfsp` command introduces a space. A space can be considered equivalent to an AND operator.

`\bnfpn{abc} \bnfor \bnfts{xzy}`	⟨abc⟩ | xzy
`\bnfpn{abc} \bnfsp \bnfts{xzy}`	⟨abc⟩ xzy

3.5 Inline expressions

The package's definition commands can be typeset inline using maths mode, so the expression `$\bnfpn{name}$` will give ⟨name⟩.

`\bnfpo` The `\bnfpo` command is provided so that the production operator ⊨ can be printed independently from the `bnf` environment if required. The `\bnfprod` command cannot be used inline.

`\bnfpo` The `\bnfpo` command is provided so that the production operator ⊨ can be printed independently from the `bnf` environment if required. The `\bnfprod` command cannot be used inline.

`\bnfpo` The `\bnfprod` command cannot be used inline. So the `\bnfpo` command is provided so that the production operator ⊨ can be printed independently from the `bnf` environment if required.

3.6 Command summary

The commands that can be used to define a BNF production in a `bnf` or `bnf*` environment are as follows:

Command	Operator	Outcome
`\bnfpn{}`	production name	⟨name⟩
`\bnfor`	OR operator	|
`\bnfsk`	skip	. . .
`\bnfsp`	space/AND operator	
`\bnfes`	empty string	λ
`\bnfts{}`	terminal string	terminal
`\bnftd{}`	terminal description	*description*
`\bnfpo`	production operator	⊨

4 Example

A more significant example is the following definition of a ⟨sentence⟩, where ⟨cchar⟩ are countable characters, and ⟨ichar⟩ are characters that should be ignored:

```
\begin{bnf*}
   \bnfprod{sentence}
      {\bnfpn{start} \bnfsp \bnfpn{rest} \bnfsp \bnfts{.}}\\
   \bnfprod{start}
      {\bnfpn{space} \bnfor \bnfes}\\
   \bnfprod{rest}
      {\bnfpn{word} \bnfsp \bnfpn{space} \bnfsp \bnfpn{rest}
       \bnfor \bnfpn{word} \bnfor \bnfes}\\
   \bnfprod{word}
      {\bnfpn{wchar} \bnfsp \bnfpn{word} \bnfor \bnfpn{wchar}}\\
   \bnfprod{space}
      {\bnfpn{schar} \bnfsp \bnfpn{space} \bnfor \bnfpn{schar}}\\
   \bnfprod{wchar}
      {\bnfpn{cchar}  \bnfor \bnfpn{ichar} }\\
   \bnfprod{cchar}
      {\bnfts{A} \bnfsk \bnfts{Z} \bnfor \bnfts{a} \bnfsk \bnfts{z}
       \bnfor \bnfts{0} \bnfsk \bnfts{9} \bnfor
       \bnfts{\textquotesingle}}\\
   \bnfprod{ichar}
      {-}\\
   \bnfprod{schar}
      {\bnfts{‘\hspace{1em}’} \bnfor \bnfts{!} \bnfor \bnfts{"}
       \bnfor \bnfts{(} \bnfor \bnfts{)} \bnfor \bnfts{\{} \bnfor
       \bnfts{\}} \bnfor \bnfts{:} \bnfor \bnfts{;} \bnfor \bnfts{?}
       \bnfor \bnfts{,}}\\
\end{bnf*}
```

This creates the following BNF definition:

$$
\begin{aligned}
\langle \text{sentence} \rangle &\models \langle \text{start} \rangle \ \langle \text{rest} \rangle \ . \\
\langle \text{start} \rangle &\models \langle \text{space} \rangle \mid \lambda \\
\langle \text{rest} \rangle &\models \langle \text{word} \rangle \ \langle \text{space} \rangle \ \langle \text{rest} \rangle \mid \langle \text{word} \rangle \mid \lambda \\
\langle \text{word} \rangle &\models \langle \text{wchar} \rangle \ \langle \text{word} \rangle \mid \langle \text{wchar} \rangle \\
\langle \text{space} \rangle &\models \langle \text{schar} \rangle \ \langle \text{space} \rangle \mid \langle \text{schar} \rangle \\
\langle \text{wchar} \rangle &\models \langle \text{cchar} \rangle \mid \langle \text{ichar} \rangle \\
\langle \text{cchar} \rangle &\models \text{A} \ldots \text{Z} \mid \text{a} \ldots \text{z} \mid 0 \ldots 9 \mid \text{'} \\
\langle \text{ichar} \rangle &\models - \\
\langle \text{schar} \rangle &\models \text{‘} \quad \text{’} \mid ! \mid \text{"} \mid (\mid) \mid \{ \mid \} \mid : \mid ; \mid ? \mid ,
\end{aligned}
$$