The `backnaur` package

Adrian P. Robson[*]

Version 3.1
18 June 2019

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:

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

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*}
```

---

[*] `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 §3.4 below. Terminal (`\bnfts{;}`) and nonterminal (`\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
`tsrm`	Terminal string typeface is roman
`altpo`	Production operator is ::=

The defaults are: the empty string symbol is λ, the production operator is $\models$, and the terminal string typeface is typewriter.

3.2 Environments

`bnf`
`bnf*`

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

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

3.3 Productions

`\bnfprod`
`\bnfprod*`

A production is defined by `\bnfprod` or `\bnfprod*`, which respectively give a numbered or unnumbered line in the `bnf` environment. They have identical unnumbered behaviour in the `bnf*` enviromment. They take two arguments:

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

`\bnfmore`
`\bnfmore*`

A production can be continued on addition lines by `\bnfmore` or `\bnfmore*`, which respectively give a numbered or unnumbered line in the `bnf` environment. They are treated the same in the `bnf*` environment. They take one arguments:

```
\bnfmore{<production definition>}
\bnfmore*{<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. By default literal terminal strings are printed in typewriter font, but this can be changed as a package option (see §3.1). 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 to ϵ or $\perp$ 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 should be 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

`\bnfpn` `\bnfpo`

The `\bnfprod` and `\bnfmore` macros cannot be used inline, so the `\bnfpn` and `\bnfpo` macros are provided to support typeseting productions inline using maths mode. The production’s name can be typeset with `\bnfpn{name}` and the production operator with `\bnfpo`. By default the production operator is $\models$, but it can be changed to ::= with a package option (see §3.1). The right side of the production can be defined with the usual macros (see §3.4). So `$\bnfpn{name} \bnfpo \bnftd{description}$` gives ⟨name⟩ $\models$ *description.*

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
`\bnprod`	production line	<name> $\models$ *def*
`\bnmore`	extra line	$\models$ *def*
`\bnfor`	OR operator	$\mid$
`\bnfsk`	skip	…
`\bnfsp`	space/AND operator	
`\bnfes`	empty string	λ
`\bnfts{}`	terminal string	`terminal`
`\bnftd{}`	terminal description	*description*
`\bnfpn{}`	production name	⟨name⟩
`\bnfpo`	production operator	$\models$

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}
      {\bnfts{-}}
   \bnfprod{schar}
      {\bnfts{‘\hspace{1em}’} \bnfor \bnfts{!} \bnfor \bnfts{"}
       \bnfor \bnfts{(}  \bnfor \bnfts{)} \bnfor \bnfts{\{}
       \bnfor \bnfts{\}} \bnfor }\\
   \bnfmore{\bnfts{:} \bnfor \bnfts{;} \bnfor \bnfts{?} \bnfor
            \bnfts{,} }
\end{bnf*}
```

This creates the following BNF definition:

$$\langle\text{sentence}\rangle \quad \models \quad \langle\text{start}\rangle\ \langle\text{rest}\rangle\ \mathtt{.} \tag{1}$$

$$\langle\text{start}\rangle \quad \models \quad \langle\text{space}\rangle \mid \lambda \tag{2}$$

$$\langle\text{rest}\rangle \quad \models \quad \langle\text{word}\rangle\ \langle\text{space}\rangle\ \langle\text{rest}\rangle \mid \langle\text{word}\rangle \mid \lambda \tag{3}$$

$$\begin{aligned}
\langle\text{word}\rangle &\models \langle\text{wchar}\rangle\ \langle\text{word}\rangle \mid \langle\text{wchar}\rangle \tag{4}\\
\langle\text{space}\rangle &\models \langle\text{schar}\rangle\ \langle\text{space}\rangle \mid \langle\text{schar}\rangle \tag{5}\\
\langle\text{wchar}\rangle &\models \langle\text{cchar}\rangle \mid \langle\text{ichar}\rangle \tag{6}\\
\langle\text{cchar}\rangle &\models \texttt{A...Z} \mid \texttt{a...z} \mid \texttt{0...9} \mid \texttt{'} \tag{7}\\
\langle\text{ichar}\rangle &\models \texttt{-} \tag{8}\\
\langle\text{schar}\rangle &\models \text{‘\hspace{1em}’} \mid \texttt{!} \mid \texttt{"} \mid \texttt{(} \mid \texttt{)} \mid \texttt{\{} \mid \texttt{\}} \mid \\
&\quad\ \texttt{:} \mid \texttt{;} \mid \texttt{?} \mid \texttt{,} \tag{9}
\end{aligned}$$

Notice the kludge in production 9. We use `\textrm{‘\hspace{1em}’}` to typeset a representation for a space character. This is needed because we do not want to print in typewriter font, which would imply the quotes were part of an actual terminal string. The `\textrm` is needed because are in maths mode.

5 Terminal string characters

The characters used with `\bnfts{}` (terminal string) are just standard LaTeX that is typeset in either a roman or typewriter font. This means we might have to use some escape pairs and a few special characters. Apostrophes and speech marks can be confusing. There are some of the possibilities:

alpha	`\bnfts{abcdABCD}`	`abcdABCD`	abcdABCD
numeric	`\bnfts{01234}`	`01234`	01234
simple	`\bnfts{<>[]()*+-=}`	`<>[]()*+-=`	<>[]()*+-=
simple	`\bnfts{@!?/,.;:}`	`@!?/,.;:`	@!?/,.;:
escaped	`\bnfts{\{\}\$\%\&\_\#}`	`{}$%&_#`	{}$%&_#
quotes	`\bnfts{’ ‘ " ‘‘ ’’}`	`’ ‘ " " "`	’ ‘ " “ ”
quotes	`\bnfts{\textquotesingle}`	`'`	'
pound	`\bnfts{\pounds}`	`£`	£
hat	`\bnfts{\textasciicircum}`	`^`	^
backslash	`\bnfts{\textbackslash}`	`\`	\
tilde	`\bnfts{\textasciitilde}`	`~`	~

The `\textquotesingle` symbol needs the `textcomp` package, which provides lots of other interesting symbols. Consult the excellent *The Comprehensive LATEX Symbol List* by Scott Pakin for more information.