The formal-grammar package*

Martin Vassor
bromind+ctan@gresille.org

February 9, 2022

Contents

1 Introduction

The notion of formal language is one of the most important in theoretical computer science. Intuitively, it is defined as follow: we are given a set Σ of *letters* or *symbols* (the *alphabet*). For instance, we can consider $\Sigma = \{a, b, \dots, z\}$: our symbols is the set of lowercase characters. A *word* is a tuple of letters. For instance, $\langle a, b, c\rangle$ is a word of three letters over Σ. Therefore, Σ^k is the set of words of k letters over Σ. Notice in particular that Σ^0 is the set $\{\langle\rangle\}$. $\langle\rangle$ is the unique word that contains 0 letters. It is often noted ε.

The set of all words over Σ (independently of their length) is defined as $\Sigma^\star = \bigcup_{i\in\mathbb{N}} \Sigma^i$.

An important operation is the *concatenation*, noted $\cdot$, which consists in sticking two words together. For instance $\langle a, b\rangle \cdot \langle c\rangle = \langle a, b, c\rangle$. For the sake of this quick introduction, we do not specify further this operation.

A (formal) language over Σ is a subset of $\Sigma^\star$: $\mathbb{L} \subseteq \Sigma^\star$. That is, it consists in picking some of the words of $\Sigma^\star$. For instance we can define the language $\mathbb{L}_a$ which contains all words that begin with an a: $\mathbb{L}_a = \{\langle a\ \rangle \cdot w, w \in \Sigma^\star\}$.

We can see that describing languages by the mean of equations is quite tedious. Therefore, we most often use *grammars*, which are a set of rules that characterise

*This document corresponds to formal-grammar v1.2, dated 2022/02/09.

a language. In particular, one of the standard way to define a grammar is what we call the BNF, for *Backus-Naur form* (or *Backus normal form*). Such grammars are defined from two sets of elements: *terminal* and *non-terminal* (by convention, in this document, non-terminal are in calligraphic font, except when explicitely stated otherwise). Basically, *terminal* correspond to the alphabet, and *non-terminal* are names of rules.

A rule has the form $\mathcal{R} ::= a$, where $\mathcal{R}$ is the name of the rule, and a is the production. The name of the rule is a non-terminal, and the production is a sequence of terminals and non-terminal. If a rule has multiple possible productions, we separate them as follow: $\mathcal{S} ::= a \mid a\mathcal{S}$.

Finally, a grammar is a set of rules.

A production defines a set of words. Without going into formalities, a production produces the words described by the terminal, and where non-terminal are replaced by productions of the corresponding rule. For instance, the rule $\mathcal{S}$ above produces $\{a, aa, aaa, \ldots\}$. Notice that rules can be mutually recursive.

This package provides a new environment (`grammar`) and associated commands to typeset BNF grammars. It allows to easily write formal grammars. For instance, the syntax of the λ-calculus is given in Grammar 1.

$\mathcal{T}$	::=	$\mathcal{V}$	*Variable*
	|	$(\mathcal{T}\ \mathcal{T})$	*Application*
	|	$\lambda\mathcal{V} \cdot \mathcal{T}$	*Abstraction*
$\mathcal{V}$	::=	$x, y, \ldots$	*Variables*

Grammar 1: λ-calculus syntax

2 Usage

2.1 Loading the package (and loading options)

This package accepts a single option when loading: `center`. If the option is set, the initial | of non-initial lines of multi-line rules is centered with respect to the ::= of the initial line. If unset, the | is aligned to the right.

2.2 Basic usage

Creating a grammar. We first start creating a grammar using the `grammar` environment.

`grammar` This is the main environment to write your grammar. `grammar` accepts 3 optional arguments: the first one is a possible caption; the second is a positionning option; and the third is a label.

If none of the optional arguments are provided, the grammar is inlined (i.e. not in a float environment. If the first argument is set (the optional caption), the grammar is typeset in a float, captionned with the provided caption. The second optional argument is a positionning option (one of `t`, `b`, `p`, `h`, etc.). The default is

$$\begin{array}{rcll} \mathcal{A} & ::= & () & \textit{Parenthesis} \\ & | & \{\} & \textit{Curly brackets} \end{array}$$

Grammar 2: A simple grammar

$$\begin{array}{rcll} \mathcal{B} & ::= & (\mathcal{B}) \mid \{\mathcal{B}\} & \textit{Nested parenthesis or brackets} \\ \mathcal{C}_1 & ::= & \mathcal{B} & \textit{Example of subtil non-terminal} \\ \mathcal{D} & ::= & \mathcal{B} & \textit{An interesting line} \\ & | & \mathcal{D} & \textit{An uninteresting line} \\ & | & \mathcal{D} + \mathcal{D} \mid \mathcal{A} & \textit{Important item} \end{array}$$

Grammar 3: A more advanced grammar

`p`. The last argument is a label, used to reference the grammar elsewhere in the document.

The grammar can then be populated using two basic constructs: `firstcase` and `otherform`.

`\firstcase` The `firstcase` command creates a new non-terminal of the grammar. It takes 3 mandatory arguments: the letter(s) of the non-terminal, the definition,

`\otherform` and an explanation. On the other hand, `otherform` create an alternative for the preceding non-terminal, on a new line. It takes two arguments: the definition of the alternative, and an explanation. For instance, the following grammar typesets as the grammar in Grammar 2.

```
\begin{grammar}[A simple grammar][t][gr:simple_grammar]
\firstcase{A}{()}{Parenthesis}
\otherform{\{\}}{Curly brackets}
\end{grammar}
```

`\nonterm` **Referencing non-terminals.** This allows you to typeset a symbol as a non-terminal. In the current version, the default typesetting is to wrap in a `\mathcal` command. This allow to reference those non-terminals, both in grammar rules and elsewhere in the document. Notice that, since the typesetting is just a wrapper over `\mathcal`, it should be used in a math environment. For instance, the only non-terminal of Grammar 2 is $\mathcal{A}$ (`\(\nonterm{A}\)`).

2.3 Advanced capabilities

In this subsection, we will explain the more advanced capabilities of the package. These would allow to typeset more complex grammars such as the one displayed in Grammar 3.

`\gralt` **Variants on the same line.** When variants are short and simple, it is possible to display multiple of them on the same line using `\gralt`. For instance, the first line of Grammar 3 is typeset with the following command:

```
\firstcase{B}{(\nonterm{B})\gralt \{\nonterm{B}\}}{Nested parenthesis or brackets}
```

Subtle typesetting of non-terminals. Since nonterminals are, by default, typeset using `\mathcal`, it can lead to the usual issues of `\mathcal` (typically, for lowercases). Therefore, we provide *subtle* variants of `\firstcase` and `\nonterm`, in which the non-terminal symbol is not typeset (i.e. as the user, you have to typeset it manually).

`\nontermsubtil` This is equivalent to `\nonterm`, but where typesetting is left to the user. In the current implementation does nothing. However users are encouraged to use it for future modifications of the package. For instance, it is possible to typeset a non-terminal with a number index as follow $\mathcal{C}_1$ with the following command: `\(\nontermsubtil{\nonterm{C}_1}\)`

`\firstcasesubtil` The subtle variant of `\firstcase`. It works similarly, except that the non-terminal (i.e. the first argument) is not embedded in a `\mathcal` macro. For instance, the $\mathcal{C}_1$ in Grammar 3 is typeset with the following command:

```
\firstcasesubtil{\(\nonterm{C}_1\)}{\nonterm{B}}{Example of subtil non-terminal}
```

Highlighting and downplaying variants. Three commands are provided to highlight or downplay some parts of a grammar. `\highlight` highlights a whole line, `\loghighlight` highlights a part of a line, while `\downplay` downplays a line.

`\downplay` `\highlight` The two commands `\highlight` and `\downplay` work similarly: when used before a `\firstcase`, `\firstcasesubtil`, or `\otherform`, the next line is highlighted in blue, or printed in light grey. For instance, in Grammar 3, the rule for non-terminal $\mathcal{D}$ is typeset with:

```
\highlight
\firstcase{D}{\nonterm{B}}{An interesting line}
\downplay
\otherform{\nonterm{D}}{An uninteresting line}
```

`\lochighlight` For more local highlighting, it is possible to use `\lochighlight`, which prints some part of a rule in red. The last line of Grammar 3, which contains such local highlight, is typeset with the following command:

```
\otherform{\lochighlight{\nonterm{D} + \nonterm{D}}
 \gralt \nonterm{A}}{Important item}
```

Customizing the ::= symbol. At the end of the preamble (i.e. before the `\begin{document}`), the package checks if a command `\Coloneqq` is defined. If that is the case, it is used instead of ::=. Typically, packages mathtools, txfonts and pxfonts define this command, but you can also define it manually if you use the symbol elsewhere in the document.

3 Implementation

We declare an option `center` for aligning definition symbol (::=) and separator symbol (|) in center. This is done by create a new conditional, and assign corresponding values depending on the option.

```
\newif\if@formalalignsymbol\@formalalignsymbolfalse
\DeclareOption{center}{
  \@formalalignsymboltrue
}
```

Now we process options.

```
\ProcessOptions\relax
```

`floatgrammar` This is a new float that contains floating grammars. This is needed so that they are labeled with 'Grammar'.

```
\DeclareFloatingEnvironment[
name=Grammar,
listname={List of Grammars},
placement=tbhp,
]{floatgrammar}
```

`\formal@rowstyle` The default `rowstyle` is empty.

```
\newcommand*{\formal@rowstyle}{}
```

`\rowstyle` An command used to set the style of a row. In addition, we add column types to reset the style (`=`) and to keep the style from one column to the other (`+`). As of today, it is not advised for the user to use `\rowstyle` to define their own style (i.e. I have not tested it), although I hope it will someday be possible.

```
\newcommand*{\rowstyle}[1]{% sets the style of the next row
  \gdef\formal@rowstyle{#1}%
  \formal@rowstyle\ignorespaces%
}

\newcolumntype{\formal@reset}{% resets the row style
  >{\gdef\formal@rowstyle{}}%
}

\newcolumntype{\formal@add}{% adds the current row style to the next column
  >{\formal@rowstyle}%
}
```

`grammar` This is the implementation of the `grammar` environment. The main difficulty is to check whether optional arguments are provided. If the first is provided, we embed the grammar into a `floatgrammar`; then if the second argument is provided, we use it as the position, (otherwise, we use `p`). Finally, if the third argument is provided, we use it as a label. Notice that, if the grammar is not a float (is inline), we do *not* break line before and after the grammar.

Depending on whether the option `center` is set, we align the symbols accordingly. This is done via an auxiliary column type.

```
    \if@formalalignsymbol
      \newcolumntype{\formal@symbol}{c}
    \else
      \newcolumntype{\formal@symbol}{r}
    \fi
\ExplSyntaxOn
%% 1st argument: caption (makes it float)
%% 2nd argument: positionning option (‘p‘ by default)
%% 3rd argument: label
\NewDocumentEnvironment{grammar} {o O{p} o}
{
\IfNoValueF{#1}{
\begin{floatgrammar}[#2]
\centering
}

\begin{tabular}{\formal@reset l \formal@add \formal@symbol \formal@add l \formal@add l}
}{
\end{tabular}

\IfNoValueF{#1}{
\caption{#1}
\IfNoValueF{#3}{
\label{#3}
}
\end{floatgrammar}
}
}
\ExplSyntaxOff
```

`\firstcase` The `\firstcase` is typeset as a new line in the array, which first cell is the symbol of the non-terminal, the second cell is just ::=, the third cell is the rule (it is directly printed, without any modification), and the last cell is the description of the rule, in greyish color.

```
\newcommand{\firstcase}[3]{\(\mathcal{#1}\) & \(\formal@Coloneqq\) & \(#2\) & {\itshape \color{
```

`\firstcasesubtil` The `\firstcasesubtil` is implemented similarly to `\firstcase`, except that the first argument is not surrounded by `\mathcal`.

```
\newcommand{\firstcasesubtil}[3]{#1 & \(\formal@Coloneqq\) & \(#2\) & {\itshape \color{gray!90!
```

`\otherform` Adds a line with an empty first cell, and which second cell is just a pipe. The third and fourth cells are similar to `\firstcase`.

```
\newcommand{\otherform}[2]{& \(|\) & \(#1\) & {\itshape \color{gray!90!black} #2}\\}
```

`\nonterm` Typesets in `\mathcal`.

```
\newcommand{\nonterm}[1]{\mathcal{#1}}
```

`\nontermsubtil` Does nothing right now.

```
\newcommand{\nontermsubtil}[1]{#1}
```

`\gralt` `\gralt` is simply a pipe surrounded by large spaces.

```
\newcommand{\gralt}[0]{\quad |\quad }
```

`\highlight` We simply set the row color to LightCyan.

```
\newcommand{\highlight}[0]{\rowcolor{LightCyan}}
```

`\lochighlight` We simply surround the argument with red.

```
\newcommand{\lochighlight}[1]{{\color{red} #1}}
```

`\downplay` We simply apply a style that write in light grey for the row.

```
\newcommand{\downplay}[0]{\rowstyle{\color{white!80!black}}}
```

Finally, we check, at the end of the preamble, if there already exist a `::=` symbol. We search for a command called `Coloneqq`, e.g. defined in the mathtools.

```
\AtBeginDocument{%
\ifdefined\Coloneqq
\let\formal@Coloneqq\Coloneqq
\else
\newcommand{\formal@Coloneqq}{::=}
\fi
}
```