The \texttt{linguistics} package

Andrew McKenzie
http://people.ku.edu/~a326m085

February 10, 2014

Contents

1 About linguistics

The \texttt{linguistics} package is designed to allow easier use of formal symbols used in generative linguistics. The set arose from the macros that I have been using over the years for papers and class handouts. Suggestions and additions are welcome; contact me via my website (above).

To call the package, type \texttt{\usepackage{linguistics}} in the preamble of your document. The package has no options to call.

The \texttt{linguistics} package calls for the following packages: \texttt{stmaryrd}, \texttt{ulem}, \texttt{amssymb}, and \texttt{upgreek}. My personal version also calls for various linguistics packages (\texttt{qtree}, \texttt{tipa}, etc.), but I have left them out in order to prevent option clashes. I have included their call commands in this version of the package, but they are commented out (look for \texttt{\RequirePackage{qtree}}, etc).

Using \texttt{linguistics} is pretty straightforward, since it is just a series of macros. The macros are organized by module of the grammar.

2 General linguistics

The following macros are used pretty generally throughout the subfields.

<table>
<thead>
<tr>
<th>command</th>
<th>purpose</th>
<th>source</th>
<th>in print</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{\textbackslash{nl}}</td>
<td>null symbol</td>
<td>\texttt{nl}</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>\texttt{\textbackslash{ix}{}...}</td>
<td>subscript index with upright text</td>
<td>Bill\texttt{{j}}, x\texttt{{cat}}</td>
<td>Bill$<em>j$, $x</em>{cat}$</td>
</tr>
<tr>
<td>\texttt{\textbackslash{superx}{}...}</td>
<td>superscript index with upright text</td>
<td>x\texttt{{superx}3}, y\texttt{i}</td>
<td>$x^3, y^i$</td>
</tr>
</tbody>
</table>

\footnote{You might find the title of the package to be overly broad. Call it a matter of perspective.}
The \( \texttt{m\{} \ldots \texttt{m} \} \) command sets grammatical morphemes (m) in **small caps**. This is helpful in glosses, notably using gb4e, since it makes the source easier to read. Also, if you need to change the morphemes’ typesetting globally, a simple \( \texttt{renewcommand} \) of \( \texttt{m} \) will suffice.

\[
\text{the dog.}\texttt{m\{}\texttt{masc.sg}\}\quad \text{the dog.}\texttt{MASC.SG}
\]

## 3 Phonology

OT Tableaux can be made with a number of packages, each with their own macros for symbols. To write phonological rules, however, you can use the following macros to simplify things.

<table>
<thead>
<tr>
<th>command</th>
<th>purpose</th>
<th>source</th>
<th>in print</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \texttt{underlying}{} \ldots \texttt{underlying}{+back} )</td>
<td>the input to the rule</td>
<td>( \texttt{underlying}{+back} )</td>
<td>/ +back /</td>
</tr>
<tr>
<td>( \texttt{becomes} )</td>
<td>the arrow</td>
<td>( \texttt{becomes} )</td>
<td>( \rightarrow )</td>
</tr>
<tr>
<td>( \texttt{spoken}{} \ldots \texttt{spoken}{–back} )</td>
<td>the output</td>
<td>( \texttt{spoken}{–back} )</td>
<td>[–back]</td>
</tr>
<tr>
<td>( \texttt{environ} )</td>
<td>‘in the environment of’ slash</td>
<td>( \texttt{environ} )</td>
<td>/</td>
</tr>
<tr>
<td>( \texttt{spot} )</td>
<td>the exact spot of the change</td>
<td>( \texttt{spot} )</td>
<td>( \texttt{</td>
</tr>
<tr>
<td>( \texttt{syll} )</td>
<td>syllable subscript</td>
<td>( \texttt{</td>
<td>syll} )</td>
</tr>
</tbody>
</table>

Combined, these get a source code like this, for a rule fronting a back vowel between /i/ and any consonant:

\[
\texttt{underlying}\{+back\} \texttt{becomes} \texttt{spoken}\{–back\} \texttt{environ} i \texttt{spot} C
\]

The commands \( \texttt{prule} \) and \( \texttt{iparule} \) are macros combining the above macros.

About the \( \texttt{prule}\{} \ldots \texttt{prule}\{} \ldots \texttt{command} \): The first command is the underlying form, the second the spoken form, and the third the environment. The \( \texttt{iparule}\{} \ldots \texttt{iparule}\{} \ldots \texttt{command} \) does the same, but puts everything in the rule in IPA. The \( \texttt{iparule} \) command requires the tipa package, which you should already be using if you’re typesetting phonology.

\[
\texttt{prule}\{+back\}\{–back\}\{i\texttt{spot} C\} / +back / \rightarrow [–back] / i_\texttt{C}
\]

\[
\texttt{iparule}\{2\}\{E\}\{i\texttt{spot} \texttt{*C}\} / \lambda / \rightarrow [\epsilon] / i_\texttt{C}
\]

## 4 Syntax

For syntax trees, a tree package like qtree suffices. The following macros allow quick and regular typing of some common syntactic symbols, in better looking ways than are offered by ordinary distributions and packages.
command | purpose | source | in print
---|---|---|---
\textbf{\head{...}} | the head circle | \textbf{\head{V}} | V
\textbf{xbar{...}} | the bar in X-bar | \textbf{xbar{V}} | \overline{V}
\textbf{lv} | little v | \textbf{lv} | v
\textbf{\feat{...}} | syntactic feature in trees | \textbf{\feat{fem}} | [fem]
\textbf{\textfeat{...}} | syntactic feature in text | \textbf{\textfeat{fem}} | [fem]
\textbf{\dcopy{...}} | deleted copy strike-out | \textbf{\dcopy{the car}} | the car
\textbf{\ufeat{...}} | unvalued feature in trees | \textbf{\ufeat{T}} | \textbf{[uT: ]}
\textbf{\unv{...}} | unvalued feature | \textbf{\unv{T}} | \textbf{[uT: ]}

The \textbf{\unv} command should be used inside a \textbf{\feat} or \textbf{\textfeat} command, but of course doesn’t have to be. If you want to use an upright φ symbol (φ), use the \textbf{\upphi} declaration in math mode.

5 Semantics

5.1 Operators

Operators all require math mode, and putting them in math mode makes source documents hard to read. These macros simplify the writing of operators, and make the source code more intuitive.

command | purpose | source | in print
---|---|---|---
\textbf{\lam{variable}} | lambda operator | \textbf{\lam{x}} | λx
\textbf{\landd{var.}{type}} | lambda operator with domain D | \textbf{\landd{P}{s,t}} | \lambda P \in D_{s,t}
\textbf{\all{var.}} | universal quantifier | \textbf{\all{x}} | \forall x
\textbf{\some{var.}} | existential quantifier | \textbf{\some{x}} | \exists x
\textbf{\ddet{var.}} | iota-operator (definite determiner) | \textbf{\ddet{x}} | \iota x

5.2 Other delimiters

5.2.1 Semantic types

The \textbf{\type{...}} command is used for writing semantic types. It places its argument in ordered pair brackets. It can be used inside the arguments of another \textbf{\type} command to get complex types.

\textbf{\type{e,t}} \quad \langle e, t \rangle
\textbf{\type{e,type{s,t}}} \quad \langle e, (s, t) \rangle
\textbf{\type{\type{e,t},\type{\type{e,t},t}}} \quad \langle \langle e, t \rangle, \langle e, t, t \rangle \rangle

Since \textbf{\type} places its arguments in italics, it can be used for ordinary ordered pairs as well. To get types with upright letters, use the \textbf{\uptype{...}} command.

\textsuperscript{2}Requires the \textbf{ulem} package
\textsuperscript{3}This employs the upright lambda, and requires the \textbf{upgreek} package.
5.2.2 Sets

The \set{...} and \varset{...} commands are used to write sets. The \set command is purely for making the source code more intuitive, since \{ is not exactly hard to type. The \varset command (‘variable set’) writes an abstracted set. \varset uses a vertical line for ‘such that’. For the older colon notation, use the \cvarset command.

\set{a, b, c, d} { a, b, c, d }
\varset{x}{x is happy} { x | x is happy }
\cvarset{x}{x is happy} { x : x is happy }

5.2.3 Functions

To write a function in array format requires math mode and the array environment. This is inconvenient, so the following macros simplify this.

1. The \fleft (function left) declaration gives the left bracket.

2. The \func{domain}{range} command is used for each line of the function.

3. The \fright (function right) declaration gives the right bracket.

To write the function \{ ⟨a, 1⟩, ⟨b, 2⟩ \}:

\fleft%
\func{a}{1}%
\func{b}{2}%
\fright%

These macros can be used recursively.

\fleft%
\func{a}{% range of a
\fleft \func{c}{1}%
\func{d}{2}%
\fright}% end of range of a
\fright%

To find macros to write functions in an explicit notation, see section ??.

5.3 Denotation brackets

Several commands involve double brackets for denotations (or interpretation functions). These all require the \texttt{stmaryrd} package, which is called by the \texttt{linguistics} package.

Many commands involve assignment modifications. These modifications are already in math mode, so any use of $ in them will lead to an error.
5.4 Writing out functions

There is also a command \funcnot, which allows quick writing of functions in an explicit functional notation (hence the name). The first argument is the domain of the function, the second is the variable, and the third is the condition. The \funcnotx command has two arguments, the domain and the condition; it only employs the variable x.

\funcnot{G}{y}{y is a bandit} \quad f : G \to \{1,0\} 

for all y ∈ G, f(y) = 1 iff y is a bandit

\funcnotx{G}{x is a bandit} \quad f : G \to \{1,0\} 

for all x ∈ G, f(x) = 1 iff x is a bandit