Intelligent brackets
The ibrackets package

Antoine Missier
antoine.missier@ac-toulouse.fr
2023/07/26, v1.2

1 Introduction

Open intervals are commonly represented with parenthesis, e.g. \((0, +\infty)\) but sometimes square brackets are used, especially in French mathematics: \([0, +\infty[.\) In that specific case, the space around the square brackets is often inappropriate, as in the expression \(x \in ]0, +\infty[.\) This small package address this issue and redefines brackets symbols [ and ] for mathematical mode to get correct spacing: \(x \in ]0, +\infty[.\)

Originally implemented in the mismath package [1] and also in frenchmath [2] since version 2.1, our previous redefinitions produce however incorrect spacing when the left bound of the interval begins with a sign - or +, which was then interpreted as a binary operation. As a result, blank spaces surrounding the sign would have been too large. This issue was pointed out by Jean-François Burnol, and an easy solution, that has been documented, consists of nesting the operator or the left bound within a pair of braces, e.g. \(x \in \{-\}\infty, 0\}\), or using \texttt{\left} and \texttt{\right} or even \texttt{\mathopen{]}}.\)

Inspired by Walter Schmidt’s icomma package [3], we now provide an improved bracket definition that works correctly without the need for these pairs of braces.

Let’s also mention other approaches, such as the \texttt{\DeclarePairedDelimiter}s macro from the mathtools package [4], or the interval package [5] with its \texttt{\interval} macro. However our solution is more lightweight.

2 Usage

With the ibrackets package, you can easily type intervals. For example the code

\$x \in ]0, \pi[ \cup ]2\pi, 3\pi[\$

yields

\(x \in ]0, \pi[ \cup ]2\pi, 3\pi[\) with ibrackets,

instead of \(x \in ]0, \pi[ \cup ]2\pi, 3\pi[\) without ibrackets.
For the example in the introduction the spacing is now correct with the following simple code: $x \in ]-\infty, 0]$, which gives $x \in ]-\infty, 0]$.

In \texttt{ibrackets}, the symbols \texttt{[} and \texttt{]} are not defined by default as delimiters. Therefore, a line break could occur between the two brackets. However, it is always possible to transform them into delimiters using \texttt{\left} and \texttt{\right}.

Actually, brackets are set as “active” characters, behaving like ordinary characters in most cases. However, when a bracket is immediately followed by a + or - character, it becomes an open delimiter. Therefore, when the left bound contains an operator sign, \textit{you don’t have to leave a space between the first bracket and the sign}, otherwise, the spaces surrounding the operator will be too large. For example if you write $x \in ]-\infty, 0]$ it yields $x \in ]-\infty, 0]$ instead of $x \in ]-\infty, 0]$. Conversely, when dealing with algebraic expressions involving intervals, \textit{you must leave a space between the second bracket and the +/\- operations} to maintain proper spacing. For instance $[a, b] + [c, d]$ yields $[a, b] + [c, d]$ while $[a, b] + [c, d]$ would yield $[a, b]+[c, d]$.

\section{Implementation}

At \texttt{\begin{document}}, we store the original \texttt{\mathcode} of the brackets, in the \texttt{\math...bracket} macros, and then we make the brackets active in math mode.

\begin{verbatim}
\AtBeginDocument{\mathcode[=\active
\gdef[{uturelet\@next\sm@rtopenbracket}
\def\sm@rtopenbracket{\ifx\@next- \mathopen \else
\ifx\@next+ \mathopen \else
\mathord\fi\fi \mathopenbracket
\mathcode]\=\active
\gdef\]{\futurelet\@next\sm@rtclosebracket}
\def\sm@rtclosebracket{\ifx\@next- \mathopen \else
\ifx\@next+ \mathopen \else
\mathord\fi\fi \mathclosebracket
\end{verbatim}

The active brackets check the next input character. If the next character is a - or a +, the active brackets return \texttt{\mathopen} with the saved \texttt{\math...bracket}, so that no space will be added after the bracket. Otherwise, \texttt{\mathord} is returned.

\begin{verbatim}
\catcode[=\active
\gdef[{uturelet\@next\sm@rtopenbracket}
\def\sm@rtopenbracket{\ifx\@next- \mathopen \else
\ifx\@next+ \mathopen \else
\mathord\fi\fi \mathopenbracket
\mathcode]\=\active
\gdef\]{\futurelet\@next\sm@rtclosebracket}
\def\sm@rtclosebracket{\ifx\@next- \mathopen \else
\ifx\@next+ \mathopen \else
\mathord\fi\fi \mathclosebracket
\end{verbatim}
We could have use the internal \texttt{\LaTeX} command \texttt{\@ifnextchar} to skip blank spaces after the bracket and look if there is a + or - after, but then it would become tricky when you really want to follow an interval with an operation plus or minus.

**References**


[5] *The interval package*. Lars Madsen, CTAN, v0.4 2019/03/06.