The luaprogtable package: programmatic table interface for LuaLaTeX

Ziyue "Alan" Xiang

ziyue.alan.xiang@gmail.com

August 13, 2020

Contents

1 Introduction

The LaTeX3 project provides LaTeX users with a handful of macros to interpret and manipulate various types of objects (e.g. integers, floating point numbers, token lists, sequences, ...) However, there is few existing function that allows users to interact with tables in a programmatic fashion. For example, if a user needs to modify the content of the cell on 20^{th} row and 8^{th} column, he/she needs to navigate the correct cell location among a pile of `&`'s and `\\`'s, which is very inefficient and error-prone. It is very difficult for someone to modify a cell based on the content within it using LaTeX macros.

luaprogtable aims to tackle these problems by providing a series of *programmatic* interface for tables. The `\LPTGetCellData` and `\LPTSetCell` commands allow one to access and alter the content of a single cell. The `lptview` environment allows one to modify a small sub-view of a larger table.

2 Basic Usage

2.1 Concepts

- Coordinate system

  This package uses a 1-indexed row-column-order coordinate system.

(1, 1)	(1, 2)	(1, 3)	(1, 4)
(2, 1)	(2, 2)	(2, 3)	
(3, 1)	(3, 2)		
(4, 1)			

- Index expressions

  This package adopts a *Pythonic* indexing convention: each *component* is delimited by '`,`'; within each component, a *range* is separated by a colon ('`:`'). For example, the expression `1,2,3:5` has three components, where the first and second components contain a single index 1, 2, respectively; the third component holds the range $[3, 5]$.

  To comply with LaTeX conventions, both ends of a range are *inclusive*. Therefore, the range `3:5` is made up of three indices, namely 3, 4 and 5. Negative indices indicate reverse access. For example, `-1` refers to the last element; the range `3:-1` contains all indices between 3 and the last element (both ends included).

- View expressions

  In LaTeX, tables are usually constructed with '`&`' and '`\\`'. However, it is very difficult to write a simple parser for this syntax, because these special symbols can appear in other environments with different meanings. For simplicity, luaprogtable builds table using *view expressions*. In view expressions, each column is enclosed with braces (`{}`); rows are broken with `\\`. For example, the view expression on the right is equivalent to traditional LaTeX table notation on the left. Note that **one cannot abbreviate outermost braces in view expressions**.

```
a & b & c & d \\      {a} {b} {c} {d} \\
e & f & g & h \\      {e} {f} {g} {h} \\
i & j & k & l         {i} {j} {k} {l}
```

Sometimes table cells can span across multiple rows/columns. Because luaprogtable cannot parse LaTeX content, one needs to specify the shape of the cell explicitly for these scenarios. The shape of a cell can be altered by adding `[]` after the closing brace. If a cell spans n columns, one can type `[-n]`; if a cell spans n rows, one can type `[|n]`. For example, view expression (2) is equivalent to traditional LaTeX table (1).

(1)
```
\multirow{3}{*}{a} & b & c & d             \\
                   & \multicolumn{3}{c}{e} \\
                   & f & g & h
```

(2)
```
{\multirow{3}{*}{a}}[|3] {b} {c} {d}                  \\
                         {\multicolumn{3}{c}{e}}[-3] \\
                         {f} {g} {h}
```

2.2 Creating a new table

\LPTNewTable

`\LPTNewTable {⟨table name⟩} {⟨num cols⟩} {⟨table preamble⟩} [⟨table options⟩]`

Creates a new table named after ⟨*table name*⟩. The name of the new table must not be the same as existing ones. The number of columns specified in ⟨*num cols*⟩ needs to match ⟨*table preamble*⟩ for the table to work correctly. In general, ⟨*table name*⟩ should not contain comma and back slash. Starting and trailing white spaces in ⟨*table name*⟩ will be ignored.

Description of ⟨*table options*⟩

`backend = {tabular}`

Specifies the table environment to be used for this table. Apart from `tabular`, one can also use `longtable`, `tabu` and so on. However, the corresponding package must be loaded manually.

`default before line = {}`

Specifies the default line style before each row.

`default after line = {}`

Specifies the default line style after each row.

`default after spacing = {}`

Specifies the default additional spacing after each row. This is achieved by appending `[<dim expr>]` to each row.

`input method = file,stringbuffer`

Specifies how LaTeX will read the table source generated by luaprogtable.

`file` The constructed table source will be saved to file system as:

`\jobname_<table name>.table`

It is then read into LaTeX by `\input` macro. This is ideal for debug purposes because the source is visible to the user.

`stringbuffer` The constructed table source will be fed into LaTeX directly, without the need of file system operations. On LaTeX side, this is still achieved by calling `\input`. However, the corresponding file callback functions on Lua side are changed, which allows LaTeX to read from Lua string buffers directly.

`nrows = {0}`

Specifies the number of rows in the table.

2.3 Selecting current table

\LPTSetCurrentTable `\LPTSetCurrentTable {⟨table name⟩}`

For many subsequent commands, there is no need to specify ⟨*table name*⟩ repeatedly: they fetch this information from a global variable. This macro sets the global variable for current table.

\LPTGetCurrentTable `\LPTGetCurrentTable`

Get the Lua-escaped name of current table.

2.4 Modifying table rows

\LPTAddRow `\LPTAddRow [⟨row options⟩]`

This command appends one more row to the current table. If an option is not specified in ⟨*row options*⟩, the default value is taken from ⟨*table options*⟩ of the table.

Description of ⟨*row options*⟩

`before line = {}`

Specifies the line before this row.

`after line = {}`

Specifies the line after this row.

`after spacing = {}`

Specifies the extra spacing after this row.

\LPTSetRowProp `\LPTSetRowProp {⟨index expr⟩} {⟨row options⟩}`

This command modifies the properties of rows specified in ⟨*index expr*⟩. In this case, ⟨*index expr*⟩ can point to multiple rows. For example, the index expression `:3,4,6` will trigger the modification of 5 rows, namely row 1, 2, 3, 4 and 6.

Description of ⟨*row options*⟩

`before line = {}`

Specifies the line before this row.

`after line = {}`

Specifies the line after this row.

`after spacing = {}`

Specifies the extra spacing after this row.

2.5 Modifying table contents

lptview

```
\begin{lptview} {⟨index expr⟩}
{⟨view expr⟩}
\end{lptview}
```

This environment creates a sub-view of current table. The sub-view region is specified by ⟨*index expr*⟩, and the content of this sub-view is specified by ⟨*view expr*⟩. The index expression should always consist of two components, where the first defines the range of rows and the second defines the range of columns.

lptfill

```
\begin{lptfill} {⟨index expr⟩}
{⟨content⟩}
\end{lptfill}
```

This environment fills table region specified by ⟨*index expr*⟩ with ⟨*content*⟩. When ⟨*index expr*⟩ is empty, the entire table is filled. Even when ⟨*index expr*⟩ is empty, the braces surrounding it cannot be abbreviated.

2.6 Using table source

\LPTUseTable `\LPTUseTable`

Reads the source of current table into input stream.

2.7 Deleting existing tables

\LPTDeleteTable `\LPTDeleteTable {⟨table name⟩}`

Remove a table from Lua storage to save memory.

3 Advanced Usage

3.1 Internal design

In Lua, each cell is a *class* with three attributes: `data`, `shape` and `parent`. Apparently, `data` holds the content of the cell. By default, the `shape` of all cells is `{1,1}`; the parent of all cells is `nil`. When there is a cell that spans multiple rows/columns, the top-left cell will be considered *parent cell*, and the remaining cells in the region would have `shape` set to `nil` and `parent` set to the coordinates of parent cell.

3.2 Programmatic interface

The following macros allow one to access and modify table cells programmatically.

\LPTSetCell `\LPTSetCell {⟨index expr⟩} [⟨shape⟩] {⟨content⟩}`

Set the content of the cell specified in ⟨*index expr*⟩ to ⟨*content*⟩. The index expression should always consist of two components, where each component only points to one integer. The shape is given by `<row span>, <column span>`. By default, the shape of a cell is `1,1`. When a cell occupies more then one cell space, the shape of its children cells will be set to `nil` automatically.

Author's note `\LPTSetCell` is not completely identical to `lptview`. More concretely, `lptview` and `lptfill` override LuaLaTeX's `process_input_buffer` callback, which allows Lua side to receive verbatim content as is. However, the ⟨*content*⟩ of `\LPTSetCell` needs to be processed by `\tl_to_str:n` before being passed to Lua. While the outcome is the same most of the time, `\tl_to_str:n` does append an empty space after macros, which stops verbatim commands from working properly.

\LPTFill `\LPTFill {⟨index expr⟩} {⟨content⟩}`

Fills table region specified by ⟨*index expr*⟩ with ⟨*content*⟩. When ⟨*index expr*⟩ is empty, the entire table is filled.

\LPTGetTableNames `\LPTGetTableNames`

Returns a comma-separated string containing the names of all tables.

\LPTGetTableShape `\LPTGetTableShape`

Returns the shape of the current table as a token list string. The number of rows is stored in first group; the number of columns is stored in second group.

\LPTGetCellData `\LPTGetCellData {⟨index expr⟩}`

Returns the data of the cell specified by ⟨*index expr*⟩.

\LPTGetCellShape `\LPTGetCellShape {⟨index expr⟩}`

Returns the shape of the cell specified by ⟨*index expr*⟩ as a token list string. The number of rows is stored in first group; the number of columns is stored in second group. When the shape is `nil`, the macro returns `\c_novalue_tl`.

\LPTGetCellParent `\LPTGetCellParent {⟨index expr⟩}`

Returns the coordinates of the parent of the cell specified by ⟨*index expr*⟩ as a token list string. The row index is stored in first group; the column index is stored in second group. When the parent is `nil`, the macro returns `\c_novalue_tl`.

4 Examples

4.1 Creating and filling a table

```
\LPTNewTable{oruVVAVhbMD0}{3}{|c|c|c|}[
    default after line=\hline,
    nrows=3]
\LPTSetCurrentTable{oruVVAVhbMD0}
\LPTSetRowProp{1}{before line=\hline}
\begin{lptfill}{}
\verb|#&_^|
\end{lptfill}
\LPTUseTable
```

#&_^	#&_^	#&_^
#&_^	#&_^	#&_^
#&_^	#&_^	#&_^

4.2 Changing sub-table

```
\LPTNewTable{IAs5OwqBcv0R}{4}{|c|c|c|c|}[
    default after line=\cline{2-4},
    nrows=4]
\LPTSetCurrentTable{IAs5OwqBcv0R}
\LPTFill{:-2,:-2}{Lorem}
\LPTFill{-1,2:-2}{Sit}
\LPTFill{:,-1}{Dolor}
\LPTSetRowProp{1}{before line=\hline}
\LPTSetRowProp{-1}{after line=\hline}
\begin{lptview}{:, 1}
{ \multirow{4}{*}{\rotatebox{90}{Ipsum}} }[|4] \\ \\ \\
\end{lptview}
\LPTUseTable
```

Ipsum	Lorem	Lorem	Dolor
	Lorem	Lorem	Dolor
	Lorem	Lorem	Dolor
	Sit	Sit	Dolor

```
\LPTNewTable{4Fz0h0ES2zU9}{6}{|c|c|c|c|c|c|}[
    default after line=\hline,
    nrows=6]
\LPTSetCurrentTable{4Fz0h0ES2zU9}
\begin{lptfill}{}
\verb|Lorem|
\end{lptfill}
```

```
\LPTSetRowProp{1}{before line=\hline}
\LPTSetRowProp{3}{after line=\cline{1-2}\cline{5-6}}
\begin{lptview}{3:4, 3:4}
{ \multicolumn{2}{c|}{\multirow{2}{*}{Ipsum}} }[-2]\\
{ \multicolumn{2}{c|}{} }[-2]
\end{lptview}
\LPTUseTable
```

Lorem	Lorem	Lorem	Lorem	Lorem	Lorem
Lorem	Lorem	Lorem	Lorem	Lorem	Lorem
Lorem	Lorem	Ipsum		Lorem	Lorem
Lorem	Lorem			Lorem	Lorem
Lorem	Lorem	Lorem	Lorem	Lorem	Lorem
Lorem	Lorem	Lorem	Lorem	Lorem	Lorem

4.3 Modifying row spacing

```
\LPTNewTable{KSDJ4C6wUgXL}{4}{|c|c|c|c|}[
    default after line=\hline,
    nrows=4]
\LPTSetCurrentTable{KSDJ4C6wUgXL}
\LPTFill{}{$\int_a^b f(x) dx$}
\LPTSetRowProp{1}{before line=\hline}
\LPTSetRowProp{-2:-1}{after spacing=1em}
\LPTUseTable
```

$\int_a^b f(x)dx$	$\int_a^b f(x)dx$	$\int_a^b f(x)dx$	$\int_a^b f(x)dx$
$\int_a^b f(x)dx$	$\int_a^b f(x)dx$	$\int_a^b f(x)dx$	$\int_a^b f(x)dx$
$\int_a^b f(x)dx$	$\int_a^b f(x)dx$	$\int_a^b f(x)dx$	$\int_a^b f(x)dx$
$\int_a^b f(x)dx$	$\int_a^b f(x)dx$	$\int_a^b f(x)dx$	$\int_a^b f(x)dx$

4.4 Sequentially constructing a table

```
\LPTNewTable{yTvLL6PEYgoE}{3}{ccc}
\LPTSetCurrentTable{yTvLL6PEYgoE}
\LPTAddRow[before line=\toprule, after line=\midrule]
\begin{lptview}{-1,:}
{Field 1} {Field 2} {Field 3}
\end{lptview}
\LPTAddRow
\begin{lptview}{-1,:}
{Data 1} {Data 2} {Data 3}
```

```
\end{lptview}
\LPTAddRow[after line=\bottomrule]
\begin{lptview}{-1,:}
{Data 4} {\multicolumn{2}{c}{Data 5}}[-2]
\end{lptview}
\LPTUseTable
```

Field 1	Field 2	Field 3
Data 1	Data 2	Data 3
Data 4	Data 5	

4.5 Change the color of cells based on value

```
\LPTNewTable{hJXHnablQ14y}{8}{cccccccc}[nrows=8]
\LPTSetCurrentTable{hJXHnablQ14y}
\begin{lptview}{:,:}
{20} {90} {43} {36} {73} {72} {77} {68} \\
{60} {48} {41} {52} {39} {31} {90} {65} \\
{81} {47} {58} {62} {67} {35} {49} {51} \\
{85} {41} {59} {69} {46} {77} {46} {39} \\
{24} {64} {69} {64} {89} {90} {64} {67} \\
{27} {75} {47} {40} {43} {63} {29} {27} \\
{86} {21} {40} {79} {55} {40} {36} {40} \\
{71} {63} {65} {53} {74} {58} {75} {63}
\end{lptview}
\ExplSyntaxOn
\int_step_inline:nn {8} {
    \int_step_inline:nn {8} {
        \str_set:Nx \l_tmpa_str {\LPTGetCellData{#1,##1}}
        \tl_set:Nx \l_tmpa_tl {\int_eval:n {100 - \l_tmpa_str}}
        \str_set:Nx \l_tmpb_str {\exp_not:N\cellcolor{black!\l_tmpa_str}
            \exp_not:N\color{blue!\l_tmpa_tl}\l_tmpa_str}
        \exp_args:Nno \LPTSetCell {#1,##1} {\l_tmpb_str}
    }
}
\ExplSyntaxOff
\LPTUseTable
```

20	90	43	36	73	72	77	68
60	48	41	52	39	31	90	65
81	47	58	62	67	35	49	51
85	41	59	69	46	77	46	39
24	64	69	64	89	90	64	67
27	75	47	40	43	63	29	27
86	21	40	79	55	40	36	40
71	63	65	53	74	58	75	63

4.6 Listing the name and shape of all tables

```
\ExplSyntaxOn
\clist_set:Nx \l_tmpa_clist {\LPTGetTableNames}
\LPTNewTable{oOnXsQcb7f8j}{2}{cc}
\LPTSetCurrentTable{oOnXsQcb7f8j}
\LPTAddRow
\begin{lptview}{1,:}
{Table~Name} {Shape}
\end{lptview}
\clist_map_inline:Nn \l_tmpa_clist {
    \LPTAddRow
    \LPTSetCell{-1,1}{\texttt{#1}}
    \LPTSetCurrentTable{#1}
    \tl_set:Nx \l_tmpa_tl {\LPTGetTableShape}
    \LPTSetCurrentTable{oOnXsQcb7f8j}
    \tl_set:Nx \l_tmpb_tl {$(\tl_item:Nn \l_tmpa_tl {1},
        \tl_item:Nn \l_tmpa_tl {2})$}
    \exp_args:Nno \LPTSetCell {-1,2} {\l_tmpb_tl}
}
\LPTSetRowProp{1}{before~line=\toprule, after~line=\midrule}
\LPTSetRowProp{-1}{after~line=\bottomrule}
\LPTUseTable
\ExplSyntaxOff
```

Table Name	Shape
4Fz0h0ES2zU9	$(6, 6)$
IAs5OwqBcv0R	$(4, 4)$
KSDJ4C6wUgXL	$(4, 4)$
hJXHnablQ14y	$(8, 8)$
oruVVAVhbMD0	$(3, 3)$
yTvLL6PEYgoE	$(3, 3)$

5 Test cases

Testing robustness of table modification

```
\LPTNewTable{mnEfCpDkN3OL}{6}{|c|c|c|c|c|c|}[
    default after line=\hline,
    nrows=6]
\LPTSetCurrentTable{mnEfCpDkN3OL}
\begin{lptfill}{}
\verb|Lorem|
\end{lptfill}
\LPTSetRowProp{1}{before line=\hline}
\LPTSetRowProp{3}{after line=\cline{1-2}\cline{5-6}}
\begin{lptview}{3:4, 3:4}
```

```
{ \multicolumn{2}{c|}{\multirow{2}{*}{Ipsum}} }[-2]\\
{ \multicolumn{2}{c|}{} }[-2]
\end{lptview}
\LPTUseTable\par\vspace*{1em}
\begin{lptview}{3:4, 3:4}
{ \multicolumn{2}{c|}{Change} }[-2]\\
{ \multicolumn{2}{c|}{Change} }[-2]
\end{lptview}
\LPTUseTable\par\vspace*{1em}
\begin{lptview}{3:4, 3:4}
{A} {B}\\
{C} {D}
\end{lptview}
\LPTUseTable
```

Lorem	Lorem	Lorem	Lorem	Lorem	Lorem
Lorem	Lorem	Lorem	Lorem	Lorem	Lorem
Lorem	Lorem	Ipsum		Lorem	Lorem
Lorem	Lorem			Lorem	Lorem
Lorem	Lorem	Lorem	Lorem	Lorem	Lorem
Lorem	Lorem	Lorem	Lorem	Lorem	Lorem

Lorem	Lorem	Lorem	Lorem	Lorem	Lorem
Lorem	Lorem	Lorem	Lorem	Lorem	Lorem
Lorem	Lorem	Change		Lorem	Lorem
Lorem	Lorem	Change		Lorem	Lorem
Lorem	Lorem	Lorem	Lorem	Lorem	Lorem
Lorem	Lorem	Lorem	Lorem	Lorem	Lorem

Lorem	Lorem	Lorem	Lorem	Lorem	Lorem
Lorem	Lorem	Lorem	Lorem	Lorem	Lorem
Lorem	Lorem	A	B	Lorem	Lorem
Lorem	Lorem	C	D	Lorem	Lorem
Lorem	Lorem	Lorem	Lorem	Lorem	Lorem
Lorem	Lorem	Lorem	Lorem	Lorem	Lorem

```
\LPTNewTable{869CviFSrnEy}{6}{|c|c|c|c|c|c|}[
    default after line=\hline,
    nrows=6]
\LPTSetCurrentTable{869CviFSrnEy}
\begin{lptfill}{}
\verb|Lorem|
\end{lptfill}
\LPTSetRowProp{1}{before line=\hline}
\LPTSetRowProp{3}{after line=\cline{1-2}\cline{5-6}}
\LPTSetCell{3,3}[1,2]{\multicolumn{2}{c|}{\multirow{2}{*}{Ipsum}}}
\LPTSetCell{4,3}[1,2]{\multicolumn{2}{c|}{}}
\LPTUseTable\par\vspace*{1em}
```

```
\LPTSetCell{3,3}[1,2]{\multicolumn{2}{c|}{Change}}
\LPTSetCell{4,3}[1,2]{\multicolumn{2}{c|}{Change}}
\LPTUseTable\par\vspace*{1em}
Notice how \verb|Lorem| appears because we haven't changed
the values of cell $(3,4)$ and $(4,4)$.\par
\LPTSetCell{3,3}{A}
\LPTSetCell{4,3}{C}
\LPTUseTable\par\vspace*{1em}
```

Lorem	Lorem	Lorem	Lorem	Lorem	Lorem
Lorem	Lorem	Lorem	Lorem	Lorem	Lorem
Lorem	Lorem	Ipsum		Lorem	Lorem
Lorem	Lorem			Lorem	Lorem
Lorem	Lorem	Lorem	Lorem	Lorem	Lorem
Lorem	Lorem	Lorem	Lorem	Lorem	Lorem

Lorem	Lorem	Lorem	Lorem	Lorem	Lorem
Lorem	Lorem	Lorem	Lorem	Lorem	Lorem
Lorem	Lorem	Change		Lorem	Lorem
Lorem	Lorem	Change		Lorem	Lorem
Lorem	Lorem	Lorem	Lorem	Lorem	Lorem
Lorem	Lorem	Lorem	Lorem	Lorem	Lorem

Notice how `Lorem` appears because we haven't changed the values of cell $(3,4)$ and $(4,4)$.

Lorem	Lorem	Lorem	Lorem	Lorem	Lorem
Lorem	Lorem	Lorem	Lorem	Lorem	Lorem
Lorem	Lorem	A	Lorem	Lorem	Lorem
Lorem	Lorem	C	Lorem	Lorem	Lorem
Lorem	Lorem	Lorem	Lorem	Lorem	Lorem
Lorem	Lorem	Lorem	Lorem	Lorem	Lorem

Testing other functions

```
\LPTNewTable{kCYKq42SyQtf}{5}{|c|c|c|c|c|}[
    default after line=\hline,
    nrows=6]
\LPTSetCurrentTable{kCYKq42SyQtf}
\LPTSetRowProp{1}{before line=\hline}
\ExplSyntaxOn
\int_step_inline:nn {6} {
    \int_step_inline:nn {5} {
        \tl_set:Nx \l_tmpa_tl {\LPTGetCellShape{#1,##1}}
        \tl_set:Nx \l_tmpb_tl {shape=\tl_item:Nn \l_tmpa_tl {1},
            \tl_item:Nn \l_tmpa_tl {2}}
        \exp_args:Nno \LPTSetCell {#1,##1} {\l_tmpb_tl}
```

```
    }
}
\ExplSyntaxOff
\LPTUseTable\par\vspace*{1em}
\ExplSyntaxOn
\int_step_inline:nn {6} {
    \int_step_inline:nn {5} {
        \tl_set:Nx \l_tmpa_tl {\LPTGetCellParent{#1,##1}}
        \tl_if_eq:NNTF \l_tmpa_tl \c_novalue_tl {
            \LPTSetCell {#1,##1} {NoP}
        }{
            \LPTSetCell {#1,##1} {HasP}
        }
    }
}
\LPTUseTable
\ExplSyntaxOff
```

shape=1,1	shape=1,1	shape=1,1	shape=1,1	shape=1,1
shape=1,1	shape=1,1	shape=1,1	shape=1,1	shape=1,1
shape=1,1	shape=1,1	shape=1,1	shape=1,1	shape=1,1
shape=1,1	shape=1,1	shape=1,1	shape=1,1	shape=1,1
shape=1,1	shape=1,1	shape=1,1	shape=1,1	shape=1,1
shape=1,1	shape=1,1	shape=1,1	shape=1,1	shape=1,1

NoP	NoP	NoP	NoP	NoP
NoP	NoP	NoP	NoP	NoP
NoP	NoP	NoP	NoP	NoP
NoP	NoP	NoP	NoP	NoP
NoP	NoP	NoP	NoP	NoP
NoP	NoP	NoP	NoP	NoP