luabidi
Bidirectional typesetting in LuaLaTeX

Vafa Khalighi Udi Fogiel Arthur Reutenauer
Jürgen Spitzmüller*

2026/06/01 v1.1

(pdf file generated on June 1, 2026)

Contents

*Please submit bug reports and feature requests to https://codeberg.org/polyglossia/luabidi/issues.

1 Objectives

Luabidi is an attempt to provide bidirectional writing support for the LuaTEX engine in the same vein as the bidi package[1] enables bidirectional writing with XETEX. The most prominent user of this package is polyglossia[2] which uses luabidi with RTL languages and LuaTEX output (as opposed to bidi with XETEX).

The package tries to do so with the least amount of patches needed, and thus does not patch counters to be displayed correctly. If you want short LTR text, such as arabic numbers, which are hidden inside other macros and cannot be marked explicitly to have correct direction you can use the unibidi-lua[3] package which implements the unicode bidirectional algorithm.

2 Package Options

arabmaths By default, luabidi generates left-to-right maths. If you would like to have right-to-left maths, use this option.

textwidthfootnoterule expands the footnote rule to the whole text width.

autofootnoterule sets the footnote rule right or left aligned, depending on the direction of the first footnote that follows the rule (*i.e.,* that comes on the current page).

3 User Commands

3.1 Main Text Direction

By default, the main directionality of the document is left-to-right. To change it to right-to-left, use the switch

[1] See https://ctan.org/pkg/bidi.
[2] See https://ctan.org/pkg/polyglossia.
[3] See https://ctan.org/pkg/unibidi-lua.

\setRTLmain `\setRTLmain`

This is advisable if your document consists mainly of right-to-left text.

3.2 Paragraph Text Direction

To change directionality for paragraphs, you can use the following switches:

\setRTL **`\setRTL`** (alias: `\setRL`, `\unsetLTR`) changes paragraph direction to right-to-left.

\setRL
\unsetLTR **`\setLTR`** (alias: `\setLR`, `\unsetRTL`) changes paragraph direction to left-to-right.

\setLTR Alternatively,you can also use the environments

\setLR
\unsetRTL
RTL
LTR

```
\begin{RTL}
 ...
\end{RTL}
```

or

```
\begin{LTR}
...
\end{LTR}
```

3.3 Inline Text Direction

To change directionality for text chunks inside paragraphs, use

\RLE **`\RLE{...}`** (alias: `\RL`) changes the directionality locally to right-to-left.

\RL
\LRE **`\LRE{...}`** (alias: `\LR`) changes the directionality locally to left-to-right.

\LR

3.4 Footnotes

3.4.1 Horizontal Footnote Position

Luabidi provides two additional footnote commands in addition to standard `\footnote`: `\RTLfootnote` ← and `\LTRfootnote` ←. The standard `\footnote` command thereby places the footnote always on the side that is currently the origin of direction: on the left side of the page in LTR paragraphs and on the right in RTL paragraphs.

\RTLfootnote v0.5

\LTRfootnote v0.5

`\LTRfootnote`, in contrast, always places the footnote on the left side, notwithstanding the current directionality. `\RTLfootnote` always places it on the right side. Like `\footnote`, `\RTLfootnote` and `\LTRfootnote` have an optional argument to customize the number.

3.4.2 Footnote Rule Length and Position

By default, the placement of the footnote rule depends on the main text directionality:

- In default mode (*i.e.,* if `\setRTLmain` is not used), the footnote rule is always set left-aligned (as usual in LTR documents).
- If the main direction is RTL (*i.e.,* if `\setRTLmain` is used), the footnote rule is always set right-aligned (as usual in RTL documents)

`\leftfootnoterule` v0.5
`\rightfootnoterule` v0.5

However, with the switch `\leftfootnoterule` ←, all subsequent footnote rules are always placed on the left. Likewise, `\rightfootnoterule` ← causes all subsequent footnote rules to be always placed on the right.

`\autofootnoterule` v0.5

The switch `\autofootnoterule` ← and the corresponding package option advise luabidi to automatically determine the rule position, depending on the directionality of the first footnote on the page.

`\textwidthfootnoterule` v0.5

If you want a footnote rule that spans the whole text width, you can use the switch `\textwidthfootnoterule` ← or the respective package option.

`\footenoterulewidth`

The length of left and right footnote rules can be adjusted via

```
\setlength\footenoterulewidth{⟨length⟩}
```

The predefined `⟨length⟩` is `0.4\columnwidth`.

4 Commands for Package Authors

The following tests are provided to be used in packages:

`\if@RTL` **`\if@RTL`** determines whether the current paragraph direction is right-to-left.

`\if@RTLmain` **`\if@RTLmain`** determines whether the main direction is right-to-left.

The following macros are provided:

`\hboxR` v0.5 **`\hboxR{...}`** ← Puts the material between { and } in a RTL `\hbox`. It is similar to `\hbox bdir 1` but also set `\if@RTL`. An explicit opening brace is needed, it cannot be used with `\bgroup`, but verbatim can be used inside.

`\hboxL` v1.0 **`\hboxL{...}`** ← Similar to `\hboxR` but with LTR `\hbox`.

`\vboxR` v1.0 **`\vboxR{...}`** ← Similar to `\hboxR` but with `\vbox`.

`\vboxL` v1.0 **`\vboxL{...}`** ← Similar to `\hboxL` but with `\vbox`.

`\vtopR` v1.0 **`\vtopR{...}`** ← Similar to `\hboxR` but with `\vtop`.

`\vtopL` v1.0 **`\vtopL{...}`** ← Similar to `\hboxL` but with `\vtop`.

5 Notes on LuaTEX's Direction Model

5.1 Direction Registers

LuaTEX provides five direction registers that control text flow and layout at different scopes:

\pagedirection Controls the location of the origin of the page. For RTL pages, the page origin shifts to `\pagerightoffset` to the right of the top right edge (there is also `\pageleftoffset` which is simply `\hoffset` minus 1in). This register should match the direction of the box being shipped out to avoid misplacement warnings (the warning message mentions `\bodydirection` but what actually matters is the box direction itself). If for example a RTL box is shipped when `\pagedirection` is 0 (LTR) then the right edge of the box will be placed on the left edge of the media box, which will probably result with the text being mostly outside of the media box.

\bodydirection Used as the default direction for boxes created in vertical mode when no explicit direction is specified via `dir` or `bdir` keywords. This applies to both `\vbox`/`\vtop` and `\hbox` when typeset in vertical mode. It is also used by `\vcenter` and `\valign` for determining their box direction, and the direction of inserts and the output box. It used to have more roles during the build page and shipout phases, but only with directions that were dropped in LuaTEX.

\pardirection Determines the paragraph direction. When a paragraph starts (i.e., when entering horizontal mode), the current value of `\pardirection` is captured in a `local_par_node` that is inserted at the beginning of the paragraph. This captured value is then used during line breaking to determine the direction of the line boxes created from the paragraph.

\textdirection Specifies the direction for boxes created in horizontal mode when no explicit direction is given. It is also used by `\halign` for determining the direction of row and cell boxes. Additionally, changes to `\textdirection` updates a special direction stack that is mostly used to make `\textdirection` respect grouping. If used within horizontal mode then direction nodes are inserted into the horizontal list. If there was a direction in the stack, `\textdirection` will first insert an enddir node to close the previous direction, then a begindir node to start the new one. `\linedirection` is pretty similar to `\textdirection`, but if there is a glue node before `\linedirection` is used, the enddir node is inserted before that glue node.

\mathdirection Controls the direction for mathematics.

5.2 Directions In Paragraphs

When LuaTEX starts a new paragraph (entering horizontal mode), it creates a `local_par_node` at the beginning of the paragraph's horizontal list. This node captures the current value of `\pardirection` in its dir field.

Then, LuaTEX compares `\pardirection` with the current text direction stack. If they differ, direction nodes are automatically inserted at the beginning of the paragraph to ensure the text direction matches the paragraph direction.

Specifically, the `new_graf` function walks through `text_dir_ptr` and inserts begindir nodes for any text directions that differ from `\pardirection`.

These dir nodes comes before the indent box is added, thus the indent location is affected by them (which means the indent location is affected by `\textdirection`).

Later, when the paragraph is broken into lines by the line breaking algorithm, the paragraph direction is read from the `local_par_node`. This captured direction determines the direction of the line boxes created during line breaking.

When a paragraph contains direction changes (dir nodes), and these direction changes span across line breaks, the line breaking algorithm automatically handles the direction nesting properly:

1. During line breaking, as the algorithm scans through the paragraph, it maintains a direction stack that tracks all currently active (unclosed) direction changes.

2. After break points are chosen and lines are being constructed (in the post-line-break phase):
   - At the beginning of each line: If there are unclosed directions from the previous line, new begindir nodes are automatically inserted at the start of the line to re-establish those directions.
   - During line construction: The code scans through the line's content, tracking direction nodes and updating the direction stack.
   - At the end of each line: For each unclosed direction in the stack, a canceldir node is automatically inserted at the end of the line to properly close it.

These nodes comes after `\leftskip` and before `\rightskip` and `\parfillskip` so the location of these glues is determined by the line box direction.

5.3 Math Mode

The `\mathdirection` register controls the direction for mathematics. When entering math mode (both inline and display), LuaTEX performs the following operations:

1. Saves the current `text_dir_ptr` (the current direction node state)
2. Creates a new `text_dir_ptr` based on `\mathdirection`
3. Synchronizes all direction registers: Inside the math group, all of `\bodydirection`, `\pardirection`, and `\textdirection` are set to the value of `\mathdirection`

If `\mathdirection` differs from the surrounding `\textdirection`, direction nodes are automatically inserted:

- A begin-dir node is inserted before the math content
- The math list is converted to an hlist in the math direction
- A cancel-dir node (enddir) is inserted after the math content

This allows inline math to have a different direction from the surrounding text. These nodes are added inside the equation (after mathon and before mathoff nodes or inside the equation box (probably not needed for display)).

5.3.1 Display Math

When entering display math the `\predisplaydirection` register is set to -1 if `\mathdirection` and `\pardirection` differs, otherwise it is set to 0.

Later, when the display equation is finalized, if there is an equation number, if `\predisplaydirection` is negative then `\eqno` will put the equation number on the left, and `\leqno` on the right.

This means that if you don't manually set `\predisplaydirection` inside the equation, `\eqno` will put the equation number on the same side of `\parfillskip` of the paragraph before the display.

Finally when the equation and equation number are boxed, the box direction is determined by the value of `\textdirection`.

5.4 Box Directions

5.4.1 Direction Specification

Box direction can be specified in two ways:

1. Explicit specification: Using the `dir` or `bdir` keywords with box commands:

```
\hbox dir TRT {...}
\vbox bdir 1 {...}
```

2. Implicit inheritance: When no direction is explicitly given, boxes inherit from the appropriate direction register based on the current mode:
   - In vertical mode: uses `\bodydirection`

- In horizontal mode: uses `\textdirection`
- In math mode: uses `\mathdirection`

Note that `\vcenter`, `\halign`, and `\valign` do *not* accept `dir` or `bdir` keywords:

- `\vcenter` always uses `\bodydirection`
- `\halign` always uses `\textdirection` for row and cell boxes
- `\valign` always uses `\bodydirection` for column and cell boxes

5.4.2 Boxes Created From Lua

- `\oken.scan_list` is ran in horizontal mode, so the above rules apply
- `\ode.hpack` use `\textdirection` if a dir argument is not specified
- `\ode.vpack` use `\bodydirection` if a dir argument is not specified

5.4.3 Direction Registers Inside Boxes

When entering `\hbox`, `\vbox`, or `\vtop`, the registers `\bodydirection`, `\pardirection` and `\textdirection` are set inside the box to match the box's direction (whether specified explicitly via `dir`/`bdir` or inherited implicitly from the current mode).

This synchronization does not occur for `\vcenter`, `\halign`, and `\valign`, which use the simpler scanning mechanism and do not modify the direction registers upon entry.

5.5 Output and Shipout

During the shipout phase, box directions critically affect the final page layout:

5.5.1 Page Origin

The `\pagedirection` determines the page origin (From LuaTeX's point of view):

- For LTR pages, the reference point is positioned with the standard offset from the left edge: `refpoint.h = h_offset`
- For RTL pages, the reference point shifts to account for the right edge: `refpoint.h = page_width - page_right_offset - 1in`

5.5.2 Box Traversal

When outputting boxes to PDF/DVI, the direction of each box determines:

1. Starting position:
    - In LTR vboxes, nested boxes start at horizontal position 0 (left edge), positive shift (`\moveright`) is right offset.

- In RTL vboxes, nested boxes start at the box width (right edge), positive shift (`\moveright`) is left offset.

2. Content flow:
   - In LTR hboxes, the horizontal cursor advances rightward as content is placed
   - In RTL hboxes, coordinate transformations ensure content flows right-to-left

3. Rule positioning:
   - In LTR contexts, rules are drawn from the current position extending rightward
   - In RTL contexts, the position is first shifted left by the rule width, then the rule is drawn

5.5.3 Direction Nodes

Direction nodes (`\begindir` ... `\enddir`) create local direction changes within a box. They establish a nested coordinate system:

- The begindir node saves the current state and switches to a new direction with reset coordinates
- Content within the direction pair is processed in the new direction
- The enddir node restores the previous direction and position

During output, when a begindir node is encountered, the system calculates the width to the matching enddir node and stores the current position and reference point information in the enddir node itself. The direction is then switched, and the horizontal and vertical positions are reset to zero. Content is output in the new direction. When the matching enddir node is later encountered during normal list traversal, it retrieves the pre-stored position, reference point, and direction values to restore the previous state.

5.5.4 PDF Glyph Positioning

The directions affects how glyphs are positioned in the final PDF output. For example

```
\parindent=0pt
foo

\textdirection=1
foo
```

gives something like

```
BT
```

```
/F1 9.96264 Tf
1 0 0 1 72 759.927 Tm [(fo)-27(o)]TJ
1 0 0 1 82.239 747.972 Tm [(f)805(o)1028(o)]TJ
ET
```

For LTR text, glyphs are output in reading order, as in `[(fo)-27(o)]`. For RTL text, we start at the end position (82.239), then glyphs are output with large positive spacing values that shift each subsequent glyph leftward, as in `[(f)805(o)1028(o)]`, which positions the glyphs as o-o-f on the page.

5.5.5 Paragraph Shapes

When directions are involved, `\parshape` and `\hangindent` might bring surprising results. One need to remember that in RTL vlists, boxes are aligned to the right, positive shift means shift to the left and `\hangindent` and `\parshape` adds indent to line boxes by setting their shift field.

This means the side of the indent is determined by the direction of the box containing the line box, not by the paragraph direction.

LuaTEX provides the `\shapemode` primitive, which when set to some values, `\parshape` and `\hangindent` will be 'mirrored', which means for `\hangindent` the indent (the shift of the line box) will be negated, and for `\arshape` it will be calculated as `hsize_par - width - indentation`.

In fact having a couple of LTR paragraphs in RTL vbox with different hsize might give surprising result, as the narrower paragraph will be aligned to the right, this fact `\shapemode` cannot change.

5.6 Changing Directions In Different Stages

LuaTEX provides several ways to set the direction of objects. Up until now, we discussed what happens from the TEX side naturally, but what happens when the direction is changed after typesetting? Or from Lua?

For example LuaTEX provides the `\boxdirection0` primitive, which can set the direction of a saved box. Consider the following (assuming vertical mode and `\bodydirection=0`):

```
\setbox0\hbox bdir 1{foo $abc$}
\box0
\setbox0\hbox{foo $abc$}
\boxdirection0=1
\box0
```

which gives
abc oof
cba oof

As mentioned, when entering an hbox `\textdirection` is set to the box direction, so in the first box at the time the equation is typeset `\textdirection` is 1 while `\mathdirection` is 0 so LuaTEX adds dir nodes around the equation to ensure it has correct direction, while in the second box `\textdirection` and `\mathdirection` are 0 so no such dir nodes are added. Then in the output the equation is inside RTL coordinates from LuaTEX's point of view so it is typeset from right to left.

LuaTEX could have work around that by always adding dir nodes around equation I guess, but there are more such subtleties. What about nested boxes? Boxes (`\vbox`,`\vtop` and `\hbox`) inside the first box will be RTL by default, and LTR by default in the second.

Another example can be changing the direction of the `local_par_node` in the `insert_local_par` callback to RTL, what will happen to the following paragraph?

```
foo $abc$
```

5.7 Mode Primitives

After LuaTEX was declared as stable, some changes were still desired, so for backwards compatibility these changes requires setting a register. Some of these are related to directions:

`\matheqdirmode` By default LuaTEX use short skip before a display if the equation number is on the right side (assuming short last line), regardless of the paragraph direction before the display. Setting `\matheqdirmode` to one will make LuaTEX use short skip if the equation number is on the side of the `\parfillskip` before the display.

`\breakafterdirmode` By default dir nodes prevent line breaking, setting `\breakafterdirmode` to one allows line breaks after dir nodes.

`\mathemptydisplaymode` As described above, when `\pardirection` and `\textdirection` differs when entering horizontal mode, dir nodes are added to the paragraph. When TEX see an empty paragraph he discards it. But in LuaTeX things are a bit more complicated, even simply using `\noindent` can add a local par node and maybe dir nodes. Initially if a paragraph contained dir nodes it was not considered empty. Then it was decided to consider such paragraphs, as empty but the change accidentally was not made for paragraphs created before display equation (which internally use different function). setting `\mathemptydisplaymode` to one will make such paragraphs be considered empty.

`\fixupboxesmode` When LuaTEX encounters a closing group in the main loop, it adds closing dir nodes if needed and update the directions stack.

By default this is done only for groups that are not associated with boxes. This can create some problems with unboxing. For example with `\setbox0\hbox{\textdirection=1 foo}\unhbox0 bar` the box contains only begindir node, and when it is unboxed the begindir node is closed only at the end of the line (from the line breaking algorithm) and thus the direction switch affects the whole paragraph. Setting `\fixupboxesmode` to one will make LuaTeX fix directions at the end of hbox as well.

6 Revision Log

v. 1.1 (2026/06/01) support `\alignat*`, remove tabular patch that is no longer needed, ensure math direction (see section 5.6).

v. 1.0 (2026/04/10) Setting `\mathemptydisplaymode` and `\fixupboxesmode` to 1, fixing tabulars, ensure `\bodydirection` is equal to `\pagedirection` in the outputroutine in a more general way, adding more box variants, support tabularray, fix equation numbers, support text in math using amsmath, fix `\underline`, improve `\autofootnoterule`. Most of the internals were revised and rewritten.

v. 0.6 (2023/10/01) Fixing a bug in `\RLE` and `\LRE`; Switching `\bodydir` inside long RTL/LTR text; ensuring `\bodydir` and `\pagedir` are equal at shipout; patching lists to use a correct value of `\shapemode`; Setting `\breakafterdirmode` and `\matheqdirmode` to 1. Patch kindly provided by Udi Fogiel.

v. 0.5 (2019/10/27) Add `\RTLfootnote`, `\LTRfootnote`, and `\hboxR`; fix `autofootnoterule` option; add `\autofootnoterule`, `\leftfootnoterule`, `\rightfootnoterule` and `\textwidthfootnoterule`; add manual.

v. 0.4 (2019/08/24) Fix `\@ensure@RTL`.

v. 0.3 (2019/07/10) Fix compatibility with recent LuaTeX (this version was never released to CTAN).

v. 0.2 (2013/05/27) Fix additional files.

v. 0.1 (2009/04/01) Initial release.