l2r
r2l

a few tips

hans hagen

Table of contents

Introduction

With ConTeXt you can typeset in two directions: from left to right and from right to left. In fact you can also combine these two directions, like this:

```
There are many {\righttoleft \maincolor \it scripts in use} and some run into the
other direction. However, there is {\righttoleft \maincolor \it no fixed relation
{\lefttoright \black \it between the} direction of the script} and cars being
driven left or right of the road.
```

There are many *esu ni stpircs* and some run into the other direction. However, there is *dexfi on tpircs eht fo noitcerid between the noitaler* and cars being driven left or right of the road.

Even someone not familiar with right to left typesetting can see what happens here, or not? In fact Luigi Scarso pointed out that the `fixed` reversed into `dexif` but not in the example where **fixed** becomes **dexfi**. This signals an important property of the way the text gets processed: you input something, at some points font features get applied (like ligatures) and in the end the resulting glyph stream is reversed. By that time the combination of **f+i** has become **fi**! So, be prepared for surprises.

This manual is written by a left to right user so don't expect a manual on semitic typesetting. Also don't expect a (yet) complete manual. I'll add whatever comes to mind. This is not a manual about Hebrew or Arabic, if only because I can't read any of those scripts (languages). I leave that to others to cover.

This is work in progress and might always be! So expect errors and typos. As with anything related to typesetting the truth about how it should be done and what looks best is not absolute. So, the most we can offer is flexibility and the way ConTeXt is setup permits that.

Of course this is not possible without input. When we moved to ConTeXt LMTX, the bidi thread was picked up by Mohammad Hossein Bateni, Idris Samawi Hamid, Wolfgang Schuster and myself. So, expect more!

Hans Hagen
Hasselt, NL

1 Setting up fonts

So let's see how Arabic and Hebrew come out:

```
The sentence \quotation {I have no clue what this means.} is translated (by
Google Translate) into \quotation {\ar \righttoleft ا م ع  ة ر ك ف  ي أ  ي د ل  س ي ل
  ا ذ ه  ه ي ن ع ي.}
which is then translated back to \quotation {I have no idea what this means.} so
maybe arabic has no clue what a clue is. The suggested Arabic pronunciation is
\quotation {lays laday 'ayu fikrat eamaa yaenih hadha}. Hebrew also likes ideas
more: \quotation {\he \righttoleft ר מ ו א  ה ז  ה מ  נ ש ו מ  י ל  ן י א}.
```

The sentence "I have no clue what this means." is translated (by Google Translate) into "ليس لدي أي فكرة عما يعنيه هذا." which is then translated back to "I have no idea what this means." so maybe arabic has no clue what a clue is. The suggested Arabic pronunciation is "lays laday 'ayu fikrat eamaa yaenih hadha". Hebrew also likes ideas more: "אין לי מושג מה זה אומר".

According to Idris Hamid the Arabic should actually be this: "لَيْسَ لَدَيَّ أَيُّ فِكْرَةٍ عَمَّا يَعْنِيهِ هٰذَا" and the transliteration "Laysa ladayya ayyu fikratin ʿammā yaʿnihi hādhā".

The ConTeXt (or any TeX) ecosystem deals with languages and fonts. Languages (that relate to scripts) have specific characteristics, like running from right to left, and fonts provide a repertoire of glyphs and features. There is no real (standard) relationship between these. In for instance browsers, there are automatic fallback systems for missing characters in a font: another font is taken. These fallbacks are often not easy to tweak.

In this document we use Dejavu and although that font has Arabic shapes in its monospace variant, the serifs come without them (at least when I write this down). Before we actually define the bodyfont we hook in some fallbacks. The typescript for Dejavu has lines like this:

```
\definefontsynonym
  [SerifBoldItalic]
  [name:dejavuserifbolditalic]
  [features=default,
   fallbacks=SerifBoldItalic]
```

This permits us to do this:

```
\definefontfallback
  [Serif] [scheherazaderegular*arabic]
  [arabic] [force=yes,rscale=1.5]
\definefontfallback
  [SerifBold] [scheherazadebold*arabic]
  [arabic] [force=yes,rscale=1.5]
\definefontfallback
  [SerifItalic] [scheherazaderegular*arabic]
  [arabic] [force=yes,rscale=1.5]
\definefontfallback
  [SerifBoldItalic] [scheherazadebold*arabic]
```

```
  [arabic] [force=yes,rscale=1.5]

\definefontfallback
  [Serif] [sileot*hebrew]
  [hebrew] [force=yes]
\definefontfallback
  [SerifBold] [sileot*hebrew]
  [hebrew] [force=yes]
\definefontfallback
  [SerifItalic] [sileot*hebrew]
  [hebrew] [force=yes]
\definefontfallback
  [SerifBoldItalic] [sileot*hebrew]
  [hebrew] [force=yes]

\definefontfallback
  [Mono] [almfixed*none]
  [arabic] [force=yes]

\definefontfallback
  [Mono] [sileot*none]
  [hebrew] [force=yes,factor=1] % factor forces a monospace

\setupbodyfont
  [dejavu,10pt]
```

In addition we set up the languages:

```
\setuplanguage[ar][font=arabic,bidi=right]
\setuplanguage[he][font=hebrew,bidi=right]
```

The following example demonstrates what the effects of these commands are:

```
{لَ يْ سَ دَ لَ يَّ أَ يُّ فِ كْ رَ ةٍ عَ مَّ ا يَ عْ نِ يهِ هٰ ذَ ا.
  }
{רמוא הז המ גשומ יל ן י א.}
{\righttoleft لَ يْ سَ دَ لَ يَّ أَ يُّ فِ كْ رَ ةٍ عَ مَّ ا يَ عْ نِ يهِ
  هٰ ذَ ا.}
{\righttoleft רמוא הז המ גשומ יל ן י א.}
{\ar \righttoleft لَ يْ سَ دَ لَ يَّ أَ يُّ فِ كْ رَ ةٍ عَ مَّ ا يَ عْ نِ يهِ
  هٰ ذَ ا.}
{\he \righttoleft רמוא הז המ גשומ יל ן י א.}
{\ar لَ يْ سَ دَ لَ يَّ أَ يُّ فِ كْ رَ ةٍ عَ مَّ ا يَ عْ نِ يهِ
  هٰ ذَ ا.}
{\he רמוא הז המ גשומ יל ן י א.}
```

لَيْسَ لَدَيَّ أَيُّ فِكْرَةٍ عَمَّا يَعْنِيهِ هٰذَا.
רמוא הז המ גשומ יל ן י א.

لَيْسَ لَدَيَّ أَيُّ فِكْرَةٍ عَمَّا يَعْنِيهِ هٰذَا.

אין לי מושג מה זה אומר.
لَيْسَ لَدَيَّ أَيُّ فِكْرَةٍ عَمَّا يَعْنِيهِ هٰذَا.
אין לי מושג מה זה אומר.
لَيْسَ لَدَيَّ أَيُّ فِكْرَةٍ عَمَّا يَعْنِيهِ هٰذَا.
אין לי מושג מה זה אומר.

In principle you can also rely on automatic direction changes, for instance by using the following command:

```
\setupdirections
  [bidi=global,
   method=three]
```

But that doesn't do a font switch for you, nor does it do any of the other language related settings. It really helps if you properly tag your document content, as in:

```
{\ar لَيْسَ لَدَيَّ أَيُّ فِكْرَةٍ عَمَّا يَعْنِيهِ هٰذَا.}
{\he אין לי מושג מה זה אומר.}
```

One reason to set the `font` parameter for a language is that it will activate the right features in a font. Instead of falling back on some default, we can be very specific in what we want to enable.

2 A mixed layout

The typesetting engine normally works from left to right and top to bottom. Going from right to left actually involved two decisions:

- the direction of the display elements, the paragraphs
- the direction of the inline text, the lines

The first one is kept track of in a state variable. Every paragraph starts with a node that carries, among other information, that state. This node is added automatically and does not interfere with the typesetting. The inline direction is more intrusive as it is marked by nodes that indicate the beginning and end of a reversed strip. This mechanism is rather reliable and normally works out well. Take this:

```
left {\righttoleft right} left
left{ \righttoleft right} left
left {\righttoleft right }left
left{ \righttoleft right }left
```

You can see that we need to be careful with spaces as they can end up inside or outside a substream and by swapping next to each other:

left thgir left
left thgir left
left thgirleft
left thgirleft

We can wrap the lines in boxes as in:

```
\hbox{left\space{\bf\righttoleft right}\space left}
\hbox{left{\bf\space \righttoleft right}\space left}
\hbox{left\space{\bf\righttoleft right\space}left}
\hbox{left{\bf\space\righttoleft right\space}left}
```

When visualize the spaces we get this:

The space of a normal and bold font in the same family normally is the same but let's mix with a larger size:

```
\hbox{left {\bfa\righttoleft right} left}
\hbox{left{\bfa\space \righttoleft right} left}
\hbox{left {\bfa\righttoleft right }left}
\hbox{left{\bfa\space\righttoleft right }left}
```

Now we get the following. As you can see, it really matters where we put the braces.

Figure 2.1 Watch your spaces!

Once you are accustomed to tagging and TEX you will probably not fall into these traps. In figure 2.1 we show a large version.

The `\righttoleft` command actually has two meanings. This can best be seen from an example.

```
\righttoleft \bf How will this come out?
```

?tuo emoc siht lliw woH

```
And \righttoleft \bf how will this come out?
```

And **?tuo emoc siht lliw woh**

When we start a paragraph (or in TEX speak: when we are still in vertical mode) the paragraph direction as well as the inline direction is set. Otherwise only the inline direction is set. There are low level TEX commands (primitives) to set the direction but you can best *not* use these because we need to do a bit more than that.

There are quite some low level commands related to changing directions. Some deal with the layout, some with boxes. We might provide more in the future.

`\lefttoright`	l2r dir node or paragraph property
`\righttoleft`	r2l dir node or paragraph property
`\checkedlefttoright`	l2r dir node or paragraph property (unless already set)
`\checkedrighttoleft`	r2l dir node or paragraph property (unless already set)
`\synchronizeinlinedirection`	pickup a (possibly) reset state
`\synchronizelayoutdirection`	pickup a (possibly) reset state
`\synchronizedisplaydirection`	pickup a (possibly) reset state
`\righttolefthbox`	r2l `\hbox`
`\lefttorighthbox`	l2r `\hbox`
`\righttoleftvbox`	r2l `\vbox`
`\lefttorightvbox`	l2r `\vbox`
`\righttoleftvtop`	r2l `\vtop`
`\lefttorightvtop`	l2r `\vtop`
`\leftorrighthbox`	l2r or r2l `\hbox`
`\leftorrightvbox`	l2r or r2l `\vbox`
`\leftorrightvtop`	l2r or r2l `\vtop`
`\autodirhbox`	l2r or r2l `\hbox` (a bit more clever)
`\autodirvbox`	l2r or r2l `\vbox` (a bit more clever)
`\autodirvtop`	l2r or r2l `\vtop` (a bit more clever)
`\bidilre`	character `U+202A`, enforce l2r state
`\bidirle`	character `U+202B`, enforce r2l state
`\bidipop`	character `U+202C`, return to last state
`\bidilro`	character `U+202D`, override l2r state
`\bidirlo`	character `U+202E`, override r2l state
`\lefttorightmark` `\lrm`	character `U+200E`, l2r indicator
`\righttoleftmark` `\rlm`	character `U+200F`, r2l indicator
`\dirlre`	switch to l2r mode using `\bidilre` or
`\dirrle`	switch to r2l mode using `\bidirle` or
`\dirlro`	enforce l2r mode using `\bidilro` or
`\dirrlo`	enforce r2l mode using `\bidirlo` or
`\naturalhbox`	a normal l2r `hbox`
`\naturalvbox`	a normal l2r `vbox`
`\naturalvtop`	a normal l2r `vtop`
`\naturalhpack`	a normal l2r `hpack`
`\naturalvpack`	a normal l2r `vpack`

When we talk about layout, we mean the overall layout, concerning the document as a whole. We can have a dominantly l2r, dominantly r2l or mixed setup. In a next chapter we will give more details on the dominant setup. Here we stick to mentioning that the document flow direction is set with

```
\setupalign[r2l] % or r2l
```

When a command to setup an environment has a `align` parameter, the same keywords can be uses as part of the specification.[1]

[1] We haven't tested all situations and possible interferences. Just report anomalies to the mailing list.

3 Numbering and positioning

todo: columns (direction key), numbers (conversionsets), margins (begin/end), etc

4 Tables

Because tables represent a lot of the issues we have to deal with we start with a little introduction. This chapter can therefore also be seen a bit more in depth discussion of the kind of problems that we have to deal with.

It makes no sense to simplify bi-directional typesetting to two variants: left-to-right and right-to-left. Among the reasons are that till now there never was a consistent model presented. Over the years several individual features were introduced but there is no simple truth here. This is quite understandable in the perspective of TEX where all is about flexibility and user control. Here are some reason why we settled on the approach that is now present in CONTEXT LMTX.

A document layout is normally dominated by a main direction. The main language can be (for instance) English so all goes from left to right but snippets can be Arabic or Hebrew. When they are inline, they need to run from left to right for sure. But what about a paragraph, the order of cells in a table, the placement of images at the margin, binding to the edges of the text area in a double sided layout, footnotes? A mixed direction document might want to set up the dominate direction and then specific preferences per structural element.

In CONTEXT we can use the alignment related keywords to deal with the inline and display directions. At the document level that is done with `\setupalign` and in commands we use the `align` key(s). When we have a composite structural element the `direction` key controls the order or flushing the sub-elements. With each combination of these options there can be interesting side effects, hopefully most are predictable and intentional but some might be the result of interaction between features and of course some can be bugs (that then need to be fixed).

Because in tables the cells are basically boxes glued together in rows, it helps to know a bit about what happens at the lowest level. When we enter a snippet of text eventually it will get boxed, either as (part of) a line or as a (more or less) hard coded box, think of something framed, a floating body or a table cell. Inside a box (we're talking a list then) the stream of glyphs, rules, skips, kerns and whatever else can be mixed with directional directives that push and pop a directional state. At a pretty low level this can be done as follows (keep in mind that CONTEXT has more robust interfaces that work better with other features). First we see a normal l2r box with an embedded r2l snippet.

```
\hbox \bgroup
    abc%
    \bgroup
        \textdirection\directionrighttoleft
        DEF%
    \egroup
    ghi%
\egroup
```

abcFEDghi

The next example explicitly makes a r2l box that has an opposing snippet:

```
\hbox direction \directionrighttoleft \bgroup
    abc%
    \bgroup
```

```
        \textdirection\directionlefttoright
        DEF%
    \egroup
    ghi%
\egroup
```

ihgDEFcba

You can imagine that global settings influence this process and the more explicit boxing takes place the more care has to be taken. Often the typesetting completely reverses with a right-to-left switch but just as often we don't want that. This is why naive switching doesn't work well: there has to be some integration in the whole system.

We now repeat these two examples but reverse the content. This reversal feature is to be used with care: the node list is literally reversed. This is different from the directional directive because these only pass information to the backend about the way the result is positioned. The `reverse` keyword can also be given to an alignment command which is quite safe because an alignment row is just a list of packaged cells. Their content is left untouched.

```
\hbox reverse \bgroup
    abc%
    \bgroup
        \textdirection\directionrighttoleft
        DEF%
    \egroup
    ghi%
\egroup
```

ihgDEFcba

```
\hbox reverse direction \directionrighttoleft \bgroup
    abc%
    \bgroup
        \textdirection\directionlefttoright
        DEF%
    \egroup
    ghi%
\egroup
```

abcFEDghi

The table related mechanisms in ConTeXt have (l2r and r2l) alignment support in the cells and (reverse) direction support too. We have traditional tables:

```
\starttable[|l|r|]
    \NC left \NC right \NC \NR
    \NC l    \NC r     \NC \NR
\stoptable
```

Running text tabulation:

```
\starttabulate[|l|r|]
```

```
    \NC left \NC right \NC \NR
    \NC l    \NC r     \NC \NR
\stoptabulate
```

And natural tables:

```
\bTABLE
    \bTR
        \bTD[align=flushleft]  left  \eTD
        \bTD[align=flushright] right \eTD
    \eTR
    \bTR
        \bTD[align=flushleft]  l \eTD
        \bTD[align=flushright] r \eTD
    \eTR
\eTABLE
```

We've put these definitions in buffers, and call these with:

```
\setuptables  [direction=normal]  \getbuffer[table:table]    \blank
\setuptables  [direction=reverse] \getbuffer[table:table]    \blank
\setuptabulate[direction=normal]  \getbuffer[table:tabulate] \blank
\setuptabulate[direction=reverse] \getbuffer[table:tabulate] \blank
\setupTABLE   [split=yes]         \getbuffer[table:TABLE]    \blank
\setupTABLE   [direction=normal]  \getbuffer[table:TABLE]    \blank
\setupTABLE   [direction=reverse] \getbuffer[table:TABLE]    \blank
```

In a lefttoright layout we get:

left right
l r

right left
r l

left right
l r

left right
l r

left	right
l	r

left	right
l	r

left	right
l	r

In a righttoleft one we get:

tfel thgir
l r

thgir	tfel
r	l

thgir	tfel
r	l

thgir	tfel
r	l

tfel	thgir
l	r

tfel	thgir
l	r

tfel	thgir
l	r

Table environments are kind of complicated because we can have free standing tables and wrapped ones. They can be part of a float and get split if needed. In most cases using a tabulate makes most sense because that one is meant to be part of the text flow. It can handle paragraphs well too.

```
\starttabulate[|l|p|]
    \dorecurse {2} {
        \NC sapolsky \NC \samplefile{sapolsky} \NC \NR
    }
\stoptabulate

\starttabulate[|l|l|l|]
    \dorecurse {2} {
        \NC sapolsky \NC robert \NC \darkred \tabulateparameter{direction} \NC \NR
    }
\stoptabulate
```

sapolsky	Agriculture is a fairly recent human invention, and in many ways it was one of the great stupid moves of all time. Hunter-gatherers have thousands of wild sources of food to subsist on. Agriculture changed that all, generating an overwhelming reliance on a few dozen domesticated food sources, making you extremely vulnerable to the next famine, the next locust infestation, the next potato blight. Agriculture allowed for stockpiling of surplus resources and thus, inevitably, the unequal stockpiling of them — stratification of society and the invention of classes. Thus, it allowed for the invention of poverty. I think that the punch line of the primate-human difference is that when humans invented poverty, they came up with a way of subjugating the low-ranking like nothing ever seen before in the primate world.
sapolsky	Agriculture is a fairly recent human invention, and in many ways it was one of the great stupid moves of all time. Hunter-gatherers have thousands of wild sources of food to subsist on. Agriculture changed that all, generating an overwhelming reliance on a few dozen domesticated food sources, making you extremely vulnerable to the next famine, the next locust infestation, the next potato blight. Agriculture allowed for stockpiling of surplus resources and thus, inevitably, the unequal stockpiling of them — stratification of society

and the invention of classes. Thus, it allowed for the invention of poverty. I think that the punch line of the primate-human difference is that when humans invented poverty, they came up with a way of subjugating the low-ranking like nothing ever seen before in the primate world.

sapolsky robert normal
sapolsky robert normal

sapolsky Agriculture is a fairly recent human invention, and in many ways it was one of the great stupid moves of all time. Hunter-gatherers have thousands of wild sources of food to subsist on. Agriculture changed that all, generating an overwhelming reliance on a few dozen domesticated food sources, making you extremely vulnerable to the next famine, the next locust infestation, the next potato blight. Agriculture allowed for stockpiling of surplus resources and thus, inevitably, the unequal stockpiling of them — stratification of society and the invention of classes. Thus, it allowed for the invention of poverty. I think that the punch line of the primate-human difference is that when humans invented poverty, they came up with a way of subjugating the low-ranking like nothing ever seen before in the primate world.

sapolsky Agriculture is a fairly recent human invention, and in many ways it was one of the great stupid moves of all time. Hunter-gatherers have thousands of wild sources of food to subsist on. Agriculture changed that all, generating an overwhelming reliance on a few dozen domesticated food sources, making you extremely vulnerable to the next famine, the next locust infestation, the next potato blight. Agriculture allowed for stockpiling of surplus resources and thus, inevitably, the unequal stockpiling of them — stratification of society and the invention of classes. Thus, it allowed for the invention of poverty. I think that the punch line of the primate-human difference is that when humans invented poverty, they came up with a way of subjugating the low-ranking like nothing ever seen before in the primate world.

sapolsky robert reverse
sapolsky robert reverse

taerg eht fo eno saw ti syaw ynam ni dna ,noitnevni namuh tnecer ylriaf a si erutlucirgA ykslopas
-bus ot doof fo secruos dliw fo sdnasuoht evah srerehtag-retnuH .emit lla fo sevom diputs
nezod wef a no ecnailer gnimlehwrevo na gnitareneg ,lla taht degnahc erutlucirgA .no tsis
txen eht ,enimaf txen eht ot elbarenluv ylemertxe uoy gnikam ,secruos doof detacitsemod
sulprus fo gnilipkcots rof dewolla erutlucirgA .thgilb otatop txen eht ,noitatsefni tsucol
yteicos fo noitacfiitarts — meht fo gnilipkcots lauqenu eht ,ylbativeni ,suht dna secruoser
eht taht kniht I .ytrevop fo noitnevni eht rof dewolla ti ,suhT .sessalc fo noitnevni eht dna
yeht ,ytrevop detnevni snamuh nehw taht si ecnereffid namuh-etamirp eht fo enil hcnup
eht ni erofeb nees reve gnihton ekil gniknar-wol eht gnitagujbus fo yaw a htiw pu emac
.dlrow etamirp

taerg eht fo eno saw ti syaw ynam ni dna ,noitnevni namuh tnecer ylriaf a si erutlucirgA ykslopas
-bus ot doof fo secruos dliw fo sdnasuoht evah srerehtag-retnuH .emit lla fo sevom diputs
nezod wef a no ecnailer gnimlehwrevo na gnitareneg ,lla taht degnahc erutlucirgA .no tsis
txen eht ,enimaf txen eht ot elbarenluv ylemertxe uoy gnikam ,secruos doof detacitsemod

sulprus fo gnilipkcots rof dewolla erutlucirgA .thgilb otatop txen eht ,noitatsefni tsucol yteicos fo noitacfiitarts — meht fo gnilipkcots lauqenu eht ,ylbativeni ,suht dna secruoser eht taht kniht I .ytrevop fo noitnevni eht rof dewolla ti ,suhT .sessalc fo noitnevni eht dna yeht ,ytrevop detnevni snamuh nehw taht si ecnereffid namuh-etamirp eht fo enil hcnup eht ni erofeb nees reve gnihton ekil gniknar-wol eht gnitagujbus fo yaw a htiw pu emac .dlrow etamirp

lamron	trebor	ykslopas
lamron	trebor	ykslopas

taerg eht fo eno saw ti syaw ynam ni dna ,noitnevni namuh tnecer ylriaf a si erutlucirgA sulprus fo gnilipkcots rof dewolla erutlucirgA .thgilb otatop txen eht ,noitatsefni tsucol yteicos fo noitacfiitarts — meht fo gnilipkcots lauqenu eht ,ylbativeni ,suht dna secruoser eht taht kniht I .ytrevop fo noitnevni eht rof dewolla ti ,suhT .sessalc fo noitnevni eht dna yeht ,ytrevop detnevni snamuh nehw taht si ecnereffid namuh-etamirp eht fo enil hcnup eht ni erofeb nees reve gnihton ekil gniknar-wol eht gnitagujbus fo yaw a htiw pu emac .dlrow etamirp		ykslopas
taerg eht fo eno saw ti syaw ynam ni dna ,noitnevni namuh tnecer ylriaf a si erutlucirgA sulprus fo gnilipkcots rof dewolla erutlucirgA .thgilb otatop txen eht ,noitatsefni tsucol yteicos fo noitacfiitarts — meht fo gnilipkcots lauqenu eht ,ylbativeni ,suht dna secruoser eht taht kniht I .ytrevop fo noitnevni eht rof dewolla ti ,suhT .sessalc fo noitnevni eht dna yeht ,ytrevop detnevni snamuh nehw taht si ecnereffid namuh-etamirp eht fo enil hcnup eht ni erofeb nees reve gnihton ekil gniknar-wol eht gnitagujbus fo yaw a htiw pu emac .dlrow etamirp		ykslopas

esrever	trebor	ykslopas
esrever	trebor	ykslopas

So called natural tables are also popular. They are coded like HTML tables and formatting is mostly automatic.

```
\bTABLE
    \bTR
        \bTD[align=flushleft]     left     \eTD
        \bTD[align=flushright]    right    \eTD
        \bTD[align=flushforward]  forward  \eTD
        \bTD[align=flushbackward] backward \eTD
    \eTR
    \bTR
        \bTD[align=flushleft]     l \eTD
```

```
        \bTD[align=flushright]    r \eTD
        \bTD[align=flushforward]  f \eTD
        \bTD[align=flushbackward] b \eTD
    \eTR
\eTABLE
```

left	right	forward	backward
l	r	f	b

left	right	forward	backward
l	r	f	b

tfel	thgir	drawrof	drawkcab
l	r	f	b

tfel	thgir	drawrof	drawkcab
l	r	f	b

5 The LUA interface

We assume that you run CONTEXT MKIV in combination with LUATEX. Direction support in this engine has been improved over time. Originally the OMEGA (ALEPH) direction model was used but in the meantime it has been stripped to the basics, and what used to be so called whatsits (extension nodes) are now first class nodes. Of the many directions only four are kept in LUATEX and they are indicated by three letters:

0	`TLT`	left to right
1	`TRT`	right to left
2	`LTL`	not used in context (obsolete)
3	`RTT`	not used in context (obsolete)

In LUAMETATEX, and therefore CONTEXT LMTX we only have the first two. Therefore in LMTX you normally don't have to worry about checking for them at the LUA end because they are irrelevant for calculations (the vertical ones swapped the horizontal and vertical progression). Also, when really needed, we use the `direction` keys with numerical indicators, so zero for `l2r` and one for `r2l`. These values are used for local par nodes as well as direction nodes. In addition a direction node has a subtype:

0	`normal`	comparable to `+`
1	`cancel`	comparable to `-`

6 Going vertical

Normally the term bidi is reserved for horizontal direction swapping but there is no reason to limit our view to that. So, here I will spend some words on how we can deal with vertical directions.

I will move some (not yet public) explanation from elsewhere to here in due time.