l2r
r2l

a few tips

hans hagen

1 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.

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! So expect errors and typos.

Hans Hagen
Hasselt, NL

2 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 ليس لدي أي فكرة
عما يعنيه هذا.} 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 {\ar lays laday 'ayu fikrat eamaa yaenih hadha}. Hebrew also likes ideas
more: \quotation {\hr \righttoleft אין לי מושג מה זה אומר}.
```

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 "hineay aamae tarkif uya' yadal syal ahdah". Hebrew also likes ideas more: "אין לי מושג מה זה אומר".

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 sa 1.5]
  [arabic] [force=yes]
\definefontfallback
  [SerifBold] [scheherazadebold*arabic sa 1.5]
  [arabic] [force=yes]
\definefontfallback
  [SerifItalic] [scheherazaderegular*arabic sa 1.5]
  [arabic] [force=yes]
\definefontfallback
  [SerifBoldItalic] [scheherazadebold*arabic sa 1.5]
  [arabic] [force=yes]

\definefontfallback
  [Serif] [sileot*hebrew sa 1.0]
```

```
  [hebrew] [force=yes]
\definefontfallback
  [SerifBold] [sileot*hebrew sa 1.0]
  [hebrew] [force=yes]
\definefontfallback
  [SerifItalic] [sileot*hebrew sa 1.0]
  [hebrew] [force=yes]
\definefontfallback
  [SerifBoldItalic] [sileot*hebrew sa 1.0]
  [hebrew] [force=yes]

\definefontfeature[fakemono][mode=node,fakemono=yes]

% \definefontfallback
%   [Mono] [scheherazaderegular*fakemono sa 1.5]
%   [arabic] [force=yes,factor=1] % factor forces a monospace

\definefontfallback
  [Mono] [sileot*fakemono sa 1.0]
  [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:

```
{ليس لدي أي فكرة عما يعنيه هذا.}
{רמוא הז המ גשומ יל ויא.}
{\righttoleft ليس لدي أي فكرة عما يعنيه هذا.}
{\righttoleft רמוא הז המ גשומ יל ויא.}
{\ar \righttoleft ليس لدي أي فكرة عما يعنيه هذا.}
{\he \righttoleft רמוא הז המ גשומ יל ויא.}
{\ar ليس لدي أي فكرة عما يعنيه هذا.}
{\he רמוא הז המ גשומ יל ויא.}
```

.هذا يعنيه عما فكرة أي لدي ليس
.רמוא הז המ גשומ יל ויא

ليس لدي أي فكرة عما يعنيه هذا.
.אין לי מושג מה זה אומר
ليس لدي أي فكرة عما يعنيه هذا.
.אין לי מושג מה זה אומר
ليس لدي أي فكرة عما يعنيه هذا.
.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 ليس لدي أي فكرة عما يعنيه هذا.}
{\he זה לא משנה לי בכלל.}
```

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.

3 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.

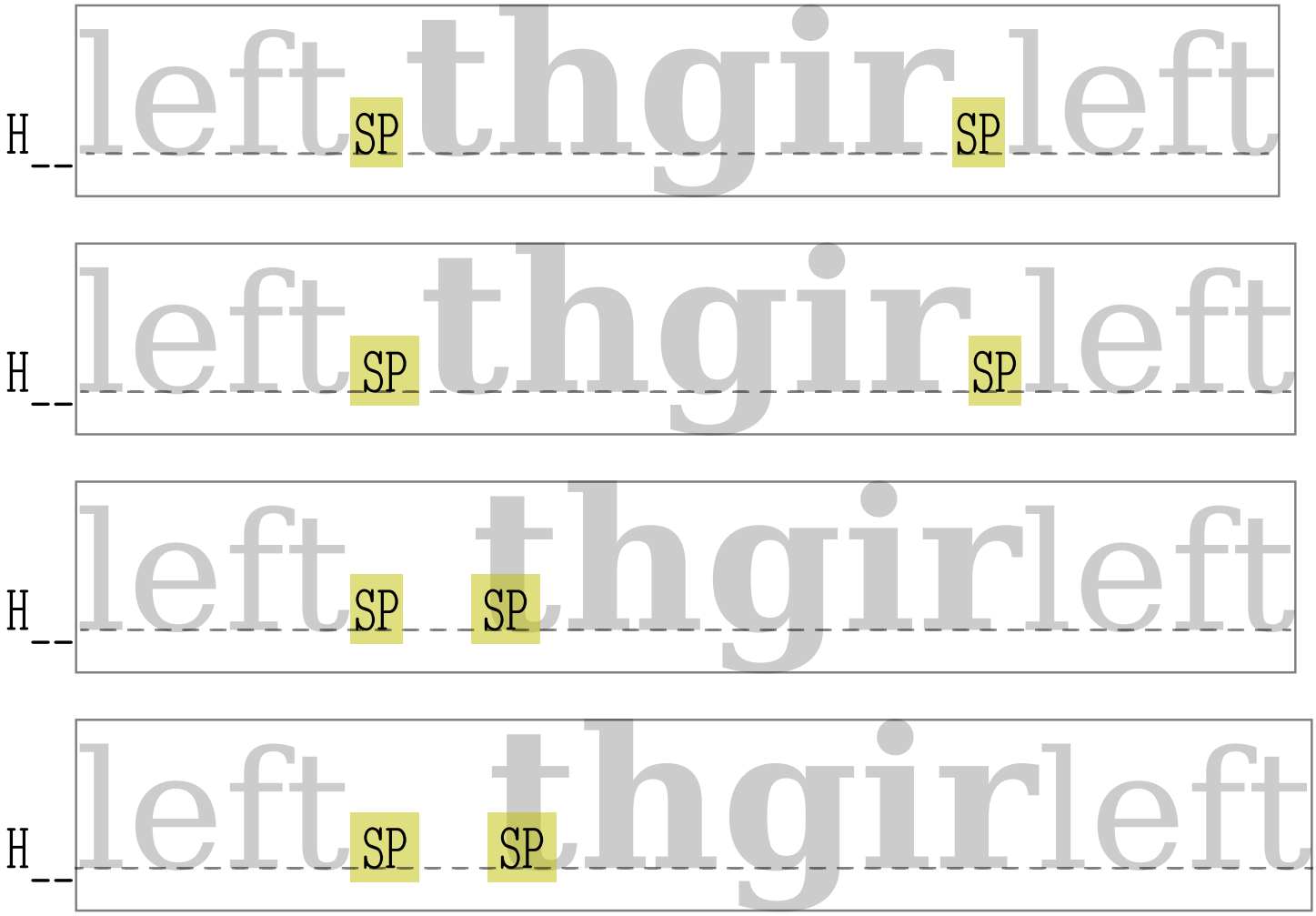

Figure 3.1 Watch your spaces!

Once you are accustomed to tagging and TEX you will probably not fall into these traps. In figure 3.1 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.

4 Numbering and positioning

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

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. In the LUA interface we still support the:

- `LTR` and `RTL` keywords for local par nodes
- `+LTR`, `-LTR`, `+RTL` and `-RTL` keywords for direction nodes

However, because we use dedicated nodes, and because nodes actually store numbers and not strings we also expose the numeric model:

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

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 `-`

The `dir` field uses the strings, the `direction` field the number (both of course share the same internal node variable).