The hebpatch Package
A Compatibility Patch for babel's Original Implementation of Hebrew Language Support

Zvi Gilboa
zgilboa@virginia.edu

v0.1 from 2012/06/01

Abstract

The current package aims to render babel's Hebrew implementation (last updated in March 2005) compatible with a variety of contemporary macro packages, including, but not limited to biblatex, tabu, pst-text, and Arabi. As Hebrew's incompatibility with such packages is primarily the result of kernel macros being redefined in rlbabel.def (in itself a necessity since neither LaTeX nor babel provide native bidirectional support), the task undertaken by hebpatch is to patch the relevant macros in a way that would retain all existing features on the one hand, yet be compatible with the rest of LaTeX on the other.

Acknowledgments

I am deeply indebted to Maxim Iorsh, Youssef Jabri, Philipp Lehman, James Clawson, and Vafa Khalighi, whose feedback and ideas gave direction to, as well as helped me set the goals of a larger project that strives to provide extended Hebrew support in LaTeX2$_\varepsilon$, and of which hebpatch makes the first part. The current package was created with sincere gratitude to the many developers of LaTeX Hebrew support throughout the years, and out of deep appreciation to lasting contributions by Michail Rozman, Boris Lavva, and the late Rama Porrat.

1 Introduction

Background

With the advent of XeTeX as well as continuing development of LuaTeX, many users have met the decision to switch – at least when composing new documents – to the above TeX engines, not the least since they both include, among many other exciting and innovative features, the native support of OpenType fonts.

In view of the growing number of high-quality OpenType fonts that contain the glyphs needed by semitic alphabets and are available in the public domain, using the `bidi` package by Vafa Khalighi in conjunction with either the `Polyglossia` package by François Charette (if using XeTeX) or `babel` (if using LuaTeX) has thus become many TeX users' preferred method for typesetting (Biblical) Hebrew or (Classical) Arabic.

There are, however, various instances where one cannot (yet) use XeTeX or LuaTeX, at least not in an intuitive manner. For anyone interested in typesetting *text along path*, for instance, the `\pstextpath` macro from the `pst-text` package is truly indispensable. Then again, if one wishes to revise or expand an old document that includes Hebrew or Arabic, switching to XeTeX or LuaTeX will not only require a considerable amount of debugging effort (consider for a minute all those `\sethebrew`, `\unsethebrew`, `\L` and `\R` occurrences), but will also inevitably result in a document that, for better or for worse, looks somehow different.[1] In light of the above, then, it is my belief that in order for true choice between LaTeX, LuaLaTeX, and XeLaTeX to exist – at least as far as Hebrew goes – we must guarantee that the functionality of a language's special features, and likewise compatibility of its support files with the major macro packages used in one's system, be retained with respect to as many typesetting engines as possible.

"Off-Shore" Hebrew

A Brief Historical Account

As of June 2012, the source files that comprise `babel`'s Hebrew support[2] date back to March 2005. To be sure, development of Hebrew (font) support for LaTeX2$_\varepsilon$ did not cease at that time. On the contrary, development of the Culmus Hebrew fonts[3], most notably by Maxim Iorsh and Yoram Gnat, has continued – and still continues – to take place, and there emerged also several loosely related LaTeX packages that made Culmus fonts accessible to TeX, specifically by providing the re-

[1] When switching a document to LuaTeX or XeTeX we would either replace all (or some) fonts with their OpenType counterparts, or stick to the document's original fonts. If changing fonts, differences in appearance (minor as they might be) are most likely unavoidable. Holding on to the document's original (Type1) fonts, on the other hand, will result in a typesetting mishmash that counts on the novel bi-directional and multilingual support mechanisms, as currently implemented in XeTeX or LuaTeX, to handle right-to-left paragraph building in general, and Hebrew or Arabic language features in particular, just as TeX--XeT and `babel` would. There are many factors that render the latter impossible, most notably the fact that neither XeTeX nor LuaTeX provide perfect equivalences to `\beginR`, `\beginL`, `\endR`, and `\endL`.

[2] These files are `hebinp.dtx`, `hebrew.dtx`, and hebrew.fdd. To perform the installation manually (note, however, that both TeXLive and MiKTeX install the entire `babel` package by default), one would also need the driver file `hebrew.ins`. If you are curious enough to visit the CTAN directory which contains the source files of `babel` you'd notice the presence of another Hebrew-related file, specifically `heb209.dtx`. As its name suggests, that file contains code that has long been considered obsolete.

[3] `culmus.sourceforge.net`

quired font-metrics (tfm), virtual-font (vf), encoding (enc), and finally font-mapping (map) files.

As a single roof under which all "post-babel" code could be found has not yet been created, it seems useful to provide here a brief, not to say crude historical account of the developemnt that took place since 2004, that is, since the point in time when Hebrew support in babel was about to become stable.[4]

The initial location at which independent development took place was Itai Levi's Nikud Project[5]. Implementing Sivan Toledo's technique for vowel-placement by means of Type-3 fonts[**?**], the Nikud project provided a set of files that consisted of the raw Culmus fonts, the custom Type-3 fonts, and the corresponding virtual fonts, thus making it possible to use these fonts in bi-directional LaTeX documents.

Using both the Nikud Project and `Ivritex` as its starting point, the culmus-latex project[6] – geared towards Linux users and led by Baruch Cohen, Guy Rutenberg, Tzafrir Cohen, and Yotam Medini – provided enhanced font-definition as well font-encoding files for use with Culmus, and also included a modified version of `hebrew.ldf`, setting cp1255 as the default Hebrew input encoding and HE8 as the default Hebrew font encoding, respectively.

Aiming to provide MiKTeX users with similar support, Iddo Samet created a variant of culmus-latex[7] in which font definition files exort greater flexibility with respect to font scaling.

Last but not least, Amit Aronovitch had identified a discrepancy between `rlbabel.def`'s handling of the `book` class and the way the class is treated elsewhere in LaTeX, for which he also posted a working patch.

For one or the other reason, none of the above packages had made it to CTAN. What's more, as the Culmus support packages for Linux and MiKTeXcontain many overlapping files, and since both also include an alternate `hebrew.ldf`, incorporating any of these packages in its current form into any of the main TeX distributions will most likely generate unpredictable, or otherwise inconsist behavior of the system. As a result, Hebrew support in LaTeX does not fully work "out of the box," and instead requires that the Culmus fonts, along with all files needed in order to use them, must be downloaded and installed seperately.

`hebpatch` in Relation to Other Hebrew Support Packages

In light of the above, the current package aims to rememdy, by way of a hot patch, the most accute problems encountered when using the above legacy support files for Hebrew, while yet keeping these files

[4] The details provided here are based on personal email communication, study of the relevant developer websites, study of the IvriTeXmailing list archives, and a comparison of the source files included in each of the packages – all necessary steps for `hebpatch` to provide users with maximum backward compatibility. Any and all comments regarding the accuracy and/or completeness of this rather rough historical account are more than welcome, and will be reflected in the next version of this package's documentation.

[5] nikud.berlios.de

[6] http://sourceforge.net/projects/ivritex/

[7] http://www.ma.huji.ac.il/~sameti/tex/culmusmiktex.html

intact. Accordingly, users should consider one of the three following approahces: 1) use the current package along with the legacy packages and font files; 2) use the current package in conjunction with the `culmusx` package, thereby gaining access to enhanced support of the culmus fonts; or 3) use the `hebrewx` package (rather than the current package) along with the `culmusx` package. As neither `hebrewx` nor `culmusx` uses any of the legacy Hebrew support files, this last approach should be preferred for all new documents.

2 Usage

Put text here.

3 Implementation

Compatability with babel and Friends

Our first task is to restore babel's original language selection mechanism. The rationale here is that instead of redefining babel's sequence of macro calls, we only make a small addition to the portion of the kernel responsible for content tables, while otherwise leaving babel's logic and flow algorithm intact. Then again, several language-related packages (most notably Philipp Lehman's `biblatex`) depend on babel's original language-selection macros, and accordingly break (or have some features break) as soon as the legacy `rlbabel.def` has been loaded by babel.

`\selectlanguage` We begin by restoring the definitions of `\selectlanguage` and all of its variants.[8]

```
1  \edef\selectlanguage{%
2    \noexpand\protect
3    \expandafter\noexpand\csname selectlanguage \endcsname
4    }
```

`*` Now follows `\languagename` , also known as the variant with the trailing space in its name.

```
5  \expandafter\def\csname selectlanguage \endcsname#1{%
6    \bbl@push@language
7    \aftergroup\bbl@pop@language
8    \bbl@set@language{#1}}
```

`\bbl@set@language` Having restored the definitions of the two `\selectlanguage` macros, we now need to gurantee that support of bi-directional content tables retains its behaviour as currently defined in `rlbabel.def` and `hebrew.ldf`. To achieve this we amend the definition of `\bbl@set@language`—the kernel macro which (beginning with babel version 3.7f) handles language

[8] For a detailed description of these macros, see the babel package documentation.

track-keeping in auxiliary files—so that it also treats the three right-to-teft content tables as appropriate.

As should be noted, `\bbl@set@language` calls `\select@language` prior to adding the relevant lines to the various auxiliary files. We may thus replace the reference to `#1` (as used in `rlbabel.def`'s `\@@selectlanguage`), with a more meaningful reference to `\languagename`.

```
\def\bbl@set@language#1{%
  \edef\languagename{%
    \ifnum\escapechar=\expandafter`\string#1\@empty
    \else \string#1\@empty\fi}%
  \select@language{\languagename}%
  \if@filesw
    \protected@write\@auxout{}{\string\select@language{\languagename}}%
     \if@rl%
       \addtocontents{cot}{\xstring\select@language{\languagename}}%
       \addtocontents{fol}{\xstring\select@language{\languagename}}%
       \addtocontents{tol}{\xstring\select@language{\languagename}}%
     \else%
       \addtocontents{toc}{\xstring\select@language{\languagename}}%
       \addtocontents{lof}{\xstring\select@language{\languagename}}%
       \addtocontents{lot}{\xstring\select@language{\languagename}}%
     \fi%
  \fi}
```

The Right-to-Left Implementation

In a Nutshell: TEX, LATEX, and Right-to-Left Languages

To typeset Right-to-Left documents in TEX or LATEX
need our typesetting engine to support bi