Makor: Typesetting Hebrew with Omega

Alan Hoenig
City University of New York
ahoenig@suffolk.lib.ny.us

Abstract
It’s relatively easy to typeset a language using a keyboard customized for that language. A more interesting problem arises when trying to set a ‘foreign’ language (say, Russian, Hebrew, or Arabic) using a native keyboard (American, for example). This leads to the problem of transliteration: how to represent some language, call it $A$, using the conventions of a different language $B$?

This paper concerns the author’s attempted solution to one such problem: to create quality Hebrew typesetting using the conventions of an English language keyboard. Apart from the different alphabet, which invokes a different set of sounds than does its English counterpart, Hebrew can involve as many as two distinct sets of diacritics, uses special glyph forms (sometimes) at word endings, and is, of course, typeset from right to left. The solution involves using the Omega extension of $\TeX$.

$\text{Makor}$, the name for this Hebrew typesetting system, consists of a user manual, fonts from seven distinct font families, and a special set of macros and conventions. Many examples of its use will be shown. All this software is publicly and freely available.

1 Introduction
We English speakers and readers are lucky — $\TeX$ or its equivalent would have been quite different, and arguably more difficult to create, had Don Knuth needed to typeset different scripts with different conventions. I never fully realized this until I turned my attention to typesetting Hebrew. Today, we all realize how robust $\TeX$ is, and how it can be coerced into doing stuff totally undreamt of by its author, but certain foreign languages break $\TeX$’s back.

Hebrew, in fact, can not be handled by the original $\TeX$. Just in case you’ve never seen Hebrew, here’s what we expect — at a bare minimum — from a Hebrew typesetting system; see figure 1 for pure Hebrew and figure 2 for mixed Hebrew-English text. ($\text{Makor}$ produced these samples, and indeed all examples of Hebrew in this article.)

Discussing ways in which $\TeX$ would fail will also deepen the reader’s understanding of Hebrew (or at least how to typeset it!).

2 Why $\TeX$ can typeset Hebrew
Here are some things which are not a problem. First off, fonts are not a problem. It’s easy enough to define a Hebrew font for use within a $\TeX$ document.

As is well known, Hebrew is an RTL (right-to-left) language, whereas $\TeX$ is an LTR (left-to-right) typesetter. This is not really a problem at all. Early on, $\TeX$ was extended to handle RTL. These early versions, $\TeX$--$\mathrm{XeT}$ and so on, early provided this capability. In recent years, this RTL-capability has been subsumed in the various extended $\TeX$’s that have appeared. Mixed Hebrew/English text should look something like figure 2.

Hebrew and Yiddish, like other Semitic languages, demand that certain letterforms be used only in word-initial and word-final positions. (Actually, only Yiddish has word-initial glyphs.) We’d like to design an input convention so that the typesetting engine makes the decisions as to which letterform is appropriate depending on context. This, too, $\TeX$ can handle, by means of virtual fonts.

3 Why $\TeX$ can’t typeset Hebrew
So what can’t $\TeX$ do? One of the many fascinating things about Hebrew (and Arabic too) is that texts normally contain only the consonants of the words. Vowels are viewed as adjuncts, and are indicated solely by means of diacritical marks rather than by full-fledged letters. The trouble is, unlike English diacrits, each Hebrew letter has its own axis around which we need to (horizontally) center the vowel mark. So, for example, figure 3 shows two letters with the same vowel mark. You don’t need to be a bona fide Hebrew reader to see that vowels
Rabbinic Hebrew (RH) does not differ greatly from Biblical Hebrew (BH) in its inflection of the noun, although the neutralization of final mem and nun means that the masculine plural is often, as in Aramaic, ָו. Apart from the more frequent use of the archaic feminine suffix ג as in ִּוָּא ‘priest’s wife’ and ֱֵָל ‘dumb woman’, RH also employs the suffixes ג- and ג for example גָּר ‘Aramaic’ and ג ‘servitude’. RH developed distinctive feminine plural suffixes in ג (Babylonian) or ג (Palestinian), for example גלְל ‘bath-houses’ and ג, as in ג ‘kingdoms’ for BH, ג ‘houses’ for nouns ending in ג- in the singular. Masculine plural forms sometimes differ from those that would be expected, or are normally found, in BH, for example, ג from ג ‘damage’, ג from ג ‘marker’, ג from ג ‘side’, ג from ג ‘half’, and ג from ג ‘envoy’. The same is true of feminine nouns, for example ג from ג (of alphabet), ג from ג ‘covenant (without plural in BH)’, and ג from ג ‘mother’.

Some masculine nouns take the feminine plural suffix ג, for example, ג from ג ‘favour’, ג from ג ‘rule’, ג from ג ‘city’, ג from ג ‘water’. Similarly, there are some feminine nouns which take the masculine plural suffix ג—ג from ג ‘dove’, ג from ג ‘ant’, and ג from ג ‘egg’, for example. Occasionally, both types of plural are evidenced, as with ג from ג ‘day’ or ג from ג ‘year’, with each form having a slightly different shade of meaning and the ‘feminine’ variant only used with suffixes. In BH we sometimes find plurals of nouns only attested in the singular in RH, for example ג from ג ‘limb’, ג from ג ‘grass’, and ג from ג ‘daily sacrifice’. Likewise, there are singular forms of nouns only attested in the plural in BH, for example ג ‘coral-wood’, ג ‘egg’, and ג ‘onion’. The dual is used more than in BH, with existing forms retained and new ones created, for example ג ‘scissors’ and ג ‘meanwhile’. (1993: A. S’aenz-Badillos, A History of the Hebrew Language, Cambridge University Press, pp.188-89.)

Figure 2: Mixed Hebrew/English text.

are positioned in very different places. Actually, the situation is even worse than that, for each letter contains two such axes, one to be used for vowel marks appearing below the letter, and another for those above the letter. In theory, the typesetting engine has to be able to keep track of axis placements for each individual letter. (In practice, though, many letters share the same axis placement.) As far as I can see, there is no really robust way to encode this axis information within a Hebrew font.
Speaking of vowel marks, there is a second set of diacritics we should be concerned with. To be sure, they are only necessary in Biblical texts, but \TeX typesetters tend to be neurotically completist about things like these, and so for the sake of completeness, we'd like our system to contain this capability. (This alternative set of diacritics provides information on how to chant the words in the sacred texts.) \TeX is hard enough pressed to typeset normal vowels without worrying about this second set. See figure 4 for Hebrew containing both sets of diacritics. Even if you don't know how to read Hebrew, a quick comparison with figure 1 shows which accents belong to the second set.

Another interesting aspect of Hebrew typography is that of alternative conventions. One or two letters, such as the *lamed* (with an 'l' sound), might appear in two distinct forms. Also, there are alternative choices for diacritic placement in certain instances. In addition, the presence or absence of the vocal diacritics and the cantorial diacritics themselves count as alternatives. Since, in the compulsive manner common to a certain class of \TeX users, we would like to enable an aspiring author to use any selection of these alternatives with any other, it's not clear how a \TeX solution for this could arise. Different fonts? We'd need $2^4 = 16$ for each base font. Macros? That would involve too much author markup. Active characters? Way too dangerous. In my view, no good pure \TeX solution exists.

The hitherto unspoken assumption on my part up to now is that we're typing at an American keyboard. As a result, we'll need a really swell input scheme to lessen the possibility of making typing errors. \TeX certainly makes it possible to get, say, the Hebrew equivalent of 'l' by typing \texttt{l}. One problem involves letters with sounds that don't occur in English. To be sure, virtual font virtuosity allows us to type \texttt{ch} to get the Hebrew letter corresponding to a throat-clearing gutteral, which is what 'ch' corresponds to (in German, at any rate). However there are additional keyboard entry issues that would require stretching virtual font definitions to the limit, so much so as to put them out of the reach of essentially any \TeX user. (The *Makor* manual describes these keyboard entry conventions.)

Another problem with proper typesetting Hebrew (and Arabic too, for that matter) has to do with numbers. Oddly enough, numbers appear in standard LTR order in a Hebrew document. Suppose \texttt{h_1} and \texttt{h_2} represent strings of number-free input which typeset the proper Hebrew text \texttt{h_1} and \texttt{h_2}. Suppose \texttt{\[ and \]} are the markup switches that enter and exit Hebrew typesetting modes. Then, we expect to be able to enter

\texttt{\[h_{-1} 12345 h_{-2}\]}

in order to typeset the fragment

\texttt{h_212345h_1}.

\TeX, though, will typeset \texttt{h_254321h_1}. You might think we could get the proper text if we exit and enter Hebrew mode before and after typesetting the number. But think about it — if you do typeset

\texttt{\[h_{-1}\] 12345 \[h_{-2}\]}

what you get is the opposite-of-correct \texttt{h_112345h_2}. Of course, you could design, using recursion, a (hypothetical) \texttt{\HebrewNumber} macro to do the job, but somehow you should expect to be able to key in numeric data in an input file without requiring special markup.

For these reasons, I have deemed it unrealistic to aspire to perfect Hebrew typesetting using \TeX or \e-\TeX.

4 Omega

Omega is a superset of \TeX originally created (and still being developed) by Yannis Haralambous and John Plaice. It was developed to handle typesetting idiosyncrasies in all the world's languages. I cannot testify about its success in other languages, but it does a splendid job with Hebrew.

Although Omega contains \TeX (and therefore all of \TeX's capabilities) at its heart, Omega differs from \TeX in several well-defined ways. I should mention that as a matter of course, it includes RTL typesetting.

More significantly, its registers have been extended to 32-bits, so, for example, it can handle large Unicode fonts. Next, it includes the capability of analyzing patterns in the input and modifying the input stream, before sending this stream to Omega for typesetting. For example, an Omega text filter could check input for the string \texttt{ffi} and replace it by the appropriate ligature. (This is a silly example, because \TeX's ligature mechanism already does that quite nicely.) However, the modifications to the input stream could also be inclusion of a macro call, so for the first time in the history of \TeX, it's now possible to modify the course of typesetting by means of macros which the typesetting engine itself inserts for you within the text of your input.
In my opinion, Omega represents a truly significant extension to \TeX. But I don’t really want to stand before you as Omega booster. Yannis and John are forceful and articulate advocates of their own work, and I encourage the interested author to explore the large base of Omega literature and to join the Omega list.

5 Makor

Makor is my name for the system I created for typesetting Hebrew.

\texttt{sh\textasciitilde{1}a\textasciitilde{1}u\textasciitilde{1}0m, \textquoteleft Ola\textquoteleft um!}

\texttt{\textquoteleft Shalom! Shalom!}

\emph{Hello, world!}

\textbf{Figure 5}: Makor input and output.

I’m not going to talk about specific methods for using Makor, nor about any of the underlying tricks I used in the Makor macros, thereby doing my part to uphold a longstanding tradition at these meetings of banning audience-unfriendly discussion. This package comes with a user manual, \texttt{mkr2man.pdf}, and I invite interested and masochistic authors to dip into the macro file \texttt{mkr2.tex} and the Omega .otp files that are part of the package. The package also includes \texttt{refcard.tex}, which is a reference card for all Makor conventions; figure 6 displays part of that for anyone who’s interested.

If you need to typeset Hebrew and quality of output and ease of input is your concern, here’s why you should use Makor:

- It’s \emph{easy} to enter consonants and vowels into a document. For example, the Hebrew equivalent of ‘Hello, world!’ might be casually transliterated as \texttt{shalom}, ‘olam!: see figure 5 for the Makor equivalent input.

- \texttt{Makor} automatically decides if a final form for a letter is necessary. If you know Hebrew, you’ll appreciate that you get these forms automatically in figure 5.

- \texttt{Makor} takes care to position the vowel symbols properly with respect to different letters, as we’ve discussed.

- It’s easy to finagle these and other aspects—altering placement of a vowel, forcing or suppressing the final form of a letter, and so on.

- \texttt{Makor} adopts the view that the diacritics we’ve mentioned are part of the logical structure of the document. It’s a good idea to include them.
Figure 6: Part of the Makor reference card.
in the input, even if you don’t want them in the output, because it makes it easier to proofread the source document. Consequently, Makor has a software switch for including or suppressing these vowel markers.

- The Makor package comes with over twenty different fonts, as you see in figure 7.
- Authors can enter cantorial diacritics (trope) into the text, as we’ve discussed. See figure 4.
- Makor also supports Yiddish with a separate input convention and special Yiddish characters.

Makor is also Ladino-ready, but as I’ve been unable to find a reliable explanation of Ladino typographic conventions, I have not (yet) implemented a Ladino input scheme. (Ladino bears a similar relationship to Hebrew and Spanish as does Yiddish to Hebrew and German. There are, in addition, other dialects that use the Hebrew alphabet, and Makor could support these conventions as well.)

- Alternative typographic conventions are supported, as we’ve already discussed.
Figure 8: Traditional Hebrew typography from the Talmud.

Figure 9: Another example of complex Hebrew typography.
• Fonts include oddball special characters so that it’s possible to typeset the Hebrew Bible with *Makor*.

• Numbers are entered normally; *Makor* takes care to typeset them properly within the Hebrew (that is, the numbers are LTR even though the surrounding text is RTL).

• *Makor* understands the conventions of *ArabTeX*, so you can process *ArabTeX* Hebrew documents in *Makor*. This is actually a consequence of Omega’s filtering mechanism. It’s just a question of prepending to the filter a sub-filter that translates *ArabTeX*’s input into *Makor* input.

• *Makor* understands the conventions of BHS, so you can process *Biblia Hebraica Stuttgartensia* in *Makor*. This downloadable ASCII file contains the full text of the Hebrew Bible, vowels, vocal diacritics, and special symbols included, but using a vastly different input convention from that of *Makor*.

• One of *Makor*’s fonts allows scholars to typeset Old Hebrew (see the last line of figure 7).

• Authors and scholars can typeset using the archaic Palestinian or Babylonian vowel systems. These were systems of vocal diacritics that died out of use about one thousand years ago or so.

• When you revise the document, either by changing your text or altering layout parameters (say, a column width), these changes automatically propagate into your text.

• Because *TeX* is Omega’s underlying typesetting engine, layouts of arbitrary complexity are possible. Figures 8 and 9 show some of the complex Hebrew typography that, over the centuries, has become traditional.

• All of the versatility that’s part of *TeX* and of Omega is always available to the author using this system.

• …And much, much more!

6 Getting *Makor*

The latest version of *Makor* is always available from the CTAN archives. You should find it at
tex-archive/language/hebrew/makor
but this directory may be a bit fouled up. To reliably locate the Makor software, simply visit your local CTAN site, and search for the file mkr2man.pdf. Then, pick up all the files in that directory and all sub-directories.

Please feel free to contact me with any questions or comments about *Makor*. You can reach me via email at ahoenig@suffolk.lib.ny.us