The TEX backend for *Jade* and the JadeTEX macros

Sebastian Rahtz
Elsevier Science Ltd
`s.rahtz@elsevier.co.uk`

August 1998

1 Introduction

DSSSL is one of the great frustrations of the SGML world. On the one hand, it is the eagerly-awaited result of years of work, which finally seems to have produced a genuinely useful model of multi-lingual text transformation and formatting (Figure 1). On the other hand, its very complexity and completeness means that

- there are no full implementations of the standard;
- there are no formatting engines capable of delivering its requirements;
- the XML community has been forced to develop a new lighter-weight style language for the Web (XSL).

In addition, the style of the language used for writing specifications (more or less, but not exactly, Scheme) has had an unfortunate off-putting effect on those more used to Omnimark or C++.

However, the (publicly visible) DSSSL community is slowly developing, thanks in the main part to two things: James Clark's partial implementation, *Jade*[1], and the considerable effort put by Norm Walsh into DSSSL specifications for formatting documents marked up against the Docbook DTD. The latter effort is targeted at HTML and RTF output, and has effectively demonstrated that the lack of the DSSSL transformation language in *Jade* is no barrier to very useable.

But there is more to *Jade* than RTF and HTML. What if we need *real* typesetting, beyond the capabilities of Microsoft? Then we can turn to the TEX backend. This has many advantages

1. It is free, well-understood, and available for all machines;
2. It is designed for rule-based batch typesetting;

---

[1] Standing for **J**ames' **A**wesome **D**SSSL **E**ngine, by some accounts from James; and since this is the most charming, lets believe it.

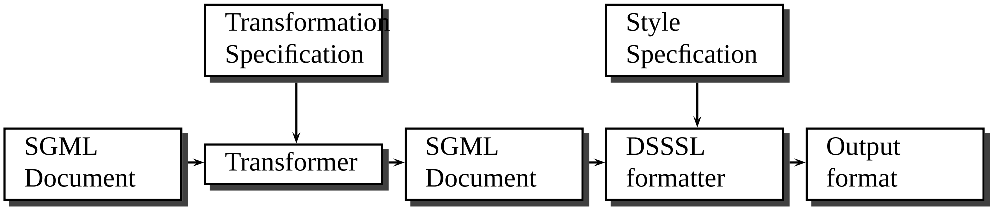


Figure 1: The DSSSL process

3. It is (pretty) good at page makeup, and very good at paragraph makeup;
4. It understands the full range of typesetting minutiae (hyphenation, fonts, math, etc);
5. It has a variant (pdfTEX) which produces PDF directly, making it more congruent with modern pre-press;
6. It is up to date with respect to Unicode (Omega).

For many years, of course, SGML practioners have transformed their files to the input format of various formatting engines, including TEX, but now we have a chance to write device independent specifications and use TEX's power to instantiate them.

2 TEX as a *Jade* backend

Jade's TEX backend (originally written by David Megginson, since modified by Sebastian Rahtz and Kathleen Marszalek) has a very simple model: it emits a TEX command for the start and end of every flow object, defining any changed characteristics at the start of the command. This abstract TEX markup can then be fleshed out by writing definitions for each of the flow object commands, and this is what the JadeTEX macro package provides. It is implemented on top of the widely used LATEX macro package, for a variety of reasons:

- LATEX has proper support for fonts, similar to DSSSL's (the New Font Selection System);
- It has standardized multi-lingual, color, graphics inclusion, hypertext and tabular support;
- It has lots of 'functions' that one can borrow.

This means that it provides a good short cut to an implementation, to see whether TEX can in fact meet the demands of DSSSL. It is important, however, for regular LATEX users to realize that no use is made of LATEX high-level constructs. There are no familiar sections, lists, cross-references, or bibliographies; everything is expressed in terms of vertical and horizontal space, font changes etc, explicit in the specification. Only page and line breaking is left to TEX: the rest is up to the DSSSL code.

3 Installation and usage

Jade's TEX backend is available by default. The JadeTEX macros are delivered (at `ftp://ftp.tex.ac.uk/tex-archive/macros/jadetex/`) in a packed format; they must first be expanded, and then used to build a new TEX format file. The sequence of command might look like this, using a modern TEX system based on Web2c 7.2:

```
tex jadetex.ins
pdftex -ini "&pdflatex" -progname=pdfjadetex pdfjadetex.ini
tex -ini "&hugelatex" jadetex.ini
```

which produces format files `pdfjadetex.fmt` and `jadetex.fmt` which can be moved to where TEX looks for such things. In practice, you will find a working system set up ready to go on the TEX Live CD-ROM (see `http://www.tug.org/texlive/`).

Assuming we have a working system, usage can be as simple as

```
jade -t tex -d article.dsl article.sgml
pdfjadetex article.tex
```

which process the SGML file `article.sgml` with the DSSSL specification `article.dsl` and writes `article.tex`; this is then run through pdfTEX, which will write `article.pdf`, which you can view or print.

4 Some simple examples

Let us look at what goes in and what comes out. If the DSSSL specification looks like this:

```
(root (make simple-page-sequence
            right-header: (literal "DSSSL Test")
            center-footer: (page-number-sosofo)
            font-family-name: body-font-family
            page-n-columns: 2
            page-column-sep: 16pt
            header-margin: .5in
            footer-margin: .5in
            left-margin: 1in
            right-margin: 1in
            top-margin: 1in
            bottom-margin: 1in
            page-width: 211mm
            page-height: 297mm))
```

then the intermediate TEX file (which is *not* meant to be edited bu humans!), looks like this:

```
\SpS{\def\fFamName{iso-serif}
 \def\PageNColumns{2}
 \def\PageColumnSep{16\p@}
 \def\HeaderMargin{36\p@}
 \def\FooterMargin{36\p@}
 \def\LeftMargin{72\p@}
 \def\RightMargin{72\p@}
 \def\TopMargin{72\p@}
 \def\BottomMargin{72\p@}
 \def\PageWidth{598.11\p@}
 \def\PageHeight{841.889\p@}
}
```

which clearly demonstrates the way *Jade* simply writes a macro name for the flow objects, and a series of `\def` commands for the characteristics.

Now consider some simple SGML markup

```
and <it>Uncle Tom Cobbley</it> and all
```

processed by this DSSSL

```
(element it
  (make sequence
    font-posture: 'italic
    (process-children-trim)))
```

from which *Jade* will write

```
and \Node{\def\Element{11}}%
\Seq{\def\fPosture{italic}}%
Uncle Tom Cobbley
\endSeq{}\endNode{} and
all.\endSeq{}\endNode{}
```

Here we see as a side effect that almost every object that comes out of *Jade* has an 'Elememt' identifier, used for cross-referencing.

What about mathematics? This is TEX's traditional strength, and something that few typesetting systems handle well. The intent of the following SGML markup should be fairly clear (to render as

$$\frac{X}{Y}$$

):

```
<fd><fr><nu>X<de>Y</fr></fd>
```

The DSSSL specification might look like this:

```
; displayed equation
(element fd
 (make display-group
 (make math-sequence
   math-display-mode: 'display
   min-leading: 2pt
   font-posture: 'math
   (process-children-trim))))
```

```
; fraction
(element fr
 (make fraction
   (process-children-trim)))

(element nu
      (make math-sequence
         label: 'numerator
   (process-children-trim)))

(element de
      (make math-sequence
         label: 'denominator
   (process-children-trim)))
```

and that results in the (slightly simplified) TEX code:

```
\DisplayGroup{}
\MathSeq{
 \def\MathDisplayMode{display}
 \def\MinLeading{2\p@}
 \def\MinLeadingFactor{0}
 \def\fPosture{math}
}
\FractionSerial{}
\insertFractionBar{}
\FractionNumerator{}
\MathSeq{}
X
\endMathSeq{}
\endFractionNumerator{}
\FractionDenominator{}
\MathSeq{}
Y
\endMathSeq{}
\endFractionDenominator{}
\endFractionSerial{}
\endMathSeq{}
\endDisplayGroup{}
```

For TEX aficionadoes, the implementation of these macros is as follows (simplified):

```
\def\FractionSerial#1{#1\bgroup}
\def\endFractionSerial{\egroup}
\def\FractionDenominator{}
\def\endFractionDenominator{}
\def\FractionNumerator{}
\def\endFractionNumerator{\over }
\def\insertFractionBar{}
```

5 DSSSL extensions supported in JadeTEX

The subset of DSSSL supported by *Jade* only covers 'simple page sequences', which do not allow such stables for the scientific publishing community as floating figures, footnotes, and multiple columns. To work around this, the TEX backend of *Jade* supports the following extra flow objects and characteristics:

```
(declare-flow-object-class page-float
      "UNREGISTERED::Sebastian Rahtz//Flow Object Class::page-float")
(declare-flow-object-class page-footnote
      "UNREGISTERED::Sebastian Rahtz//Flow Object Class::page-footnote")
(declare-characteristic page-n-columns
      "UNREGISTERED::James Clark//Characteristic::page-n-columns" 1)
(declare-characteristic page-column-sep
      "UNREGISTERED::James Clark//Characteristic::page-column-sep" 4pt)
```

(the RTF backend also supports the last two.) These allow the specification author to produce simple multicolumn pages, with footnotes and floating figures.

Numbered equations are still an unresolved issue, since they too require more complex objects than *Jade* supports

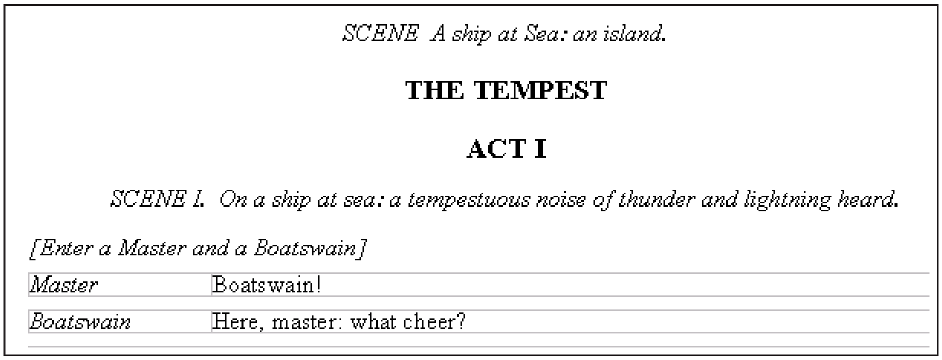
6 Is JadeTEX useable in practice?

It is not hard to process simple texts with *Jade* and see more or less identical output from the RTF and the TEX backends (Figures 2 and 3). The pages displayed in Figures 5 and 6 are more interested, as they demonstrate that a DSSSL specification, *Jade*, and JadeTEX can produce plausible pages of a scientific article. Figure 4 shows a portion of the math in Figure 5 as displayed in Microsoft Word, demonstrating the inadequacy of the math support in RTF (though the spacing can be adjusted for a somewhat better display).

7 Conclusions

The potential power of SGML/XML, DSSSL and TEX working together is fairly awesome. Unfortunately, there are some downsides to what we have today:

- There is (perhaps) no formatter capable of dealing with the DSSSL page model;
- There is no implementation of the full DSSSL transformation language;
- There is no implementation of the full specification language;
- You cannot easily tweak the TEX output;
- DSSSL has no equivalent of the integrated graphical languages we are beginning to appreciate in the LATEX world;
- DSSSL may be sidelined by the emerging XSL standard.



SCENE A ship at Sea: an island.

THE TEMPEST

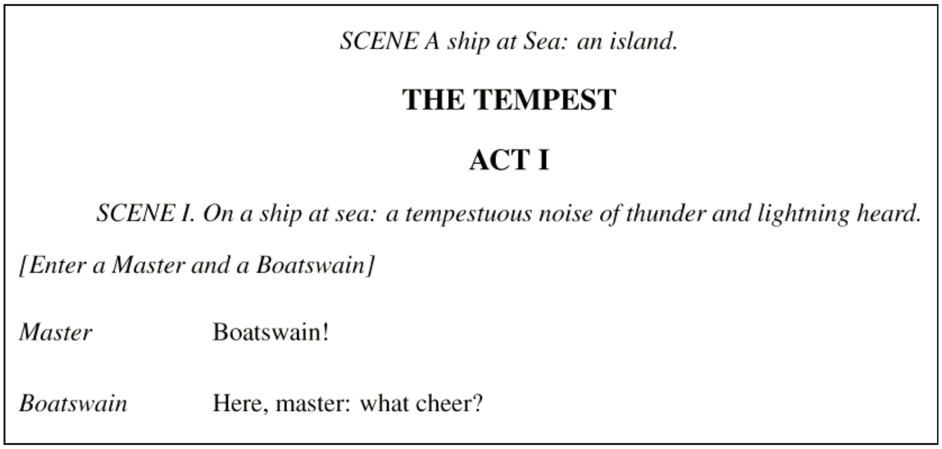
ACT I

SCENE I. On a ship at sea: a tempestuous noise of thunder and lightning heard.

[Enter a Master and a Boatswain]

Master	Boatswain!
Boatswain	Here, master: what cheer?

Figure 2: The Tempest, formatted by Microsoft Word



SCENE A ship at Sea: an island.

THE TEMPEST

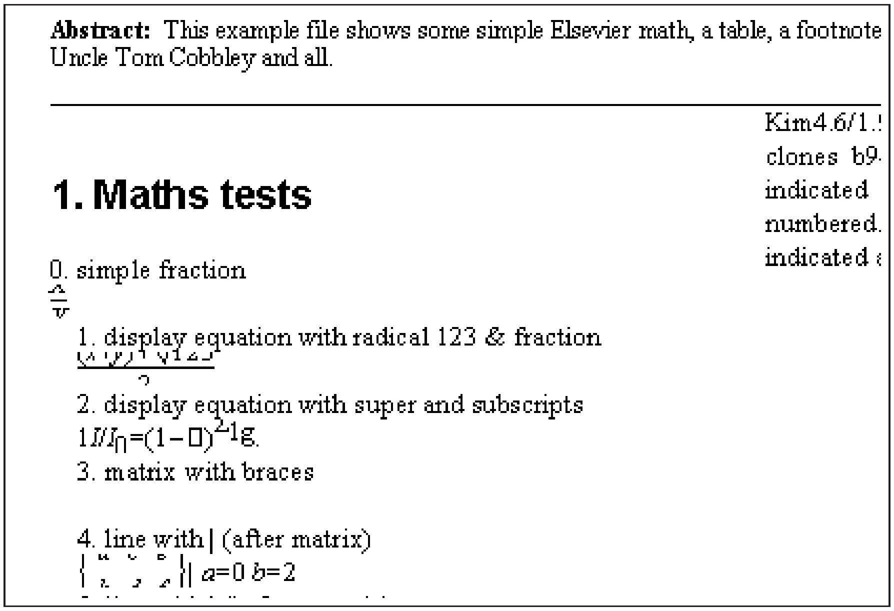
ACT I

SCENE I. On a ship at sea: a tempestuous noise of thunder and lightning heard.

[Enter a Master and a Boatswain]

Master	Boatswain!
Boatswain	Here, master: what cheer?

Figure 3: The Tempest, formatted by TEX



Abstract: This example file shows some simple Elsevier math, a table, a footnote
Uncle Tom Cobbley and all.

Kim4.6/1.
clones b9
indicated
numbered.
indicated

1. Maths tests

0. simple fraction

1. display equation with radical 123 & fraction

2. display equation with super and subscripts

$1I/I_{\cap}=(1-\square)^{21}$g.

3. matrix with braces

4. line with | (after matrix)

a=0 b=2

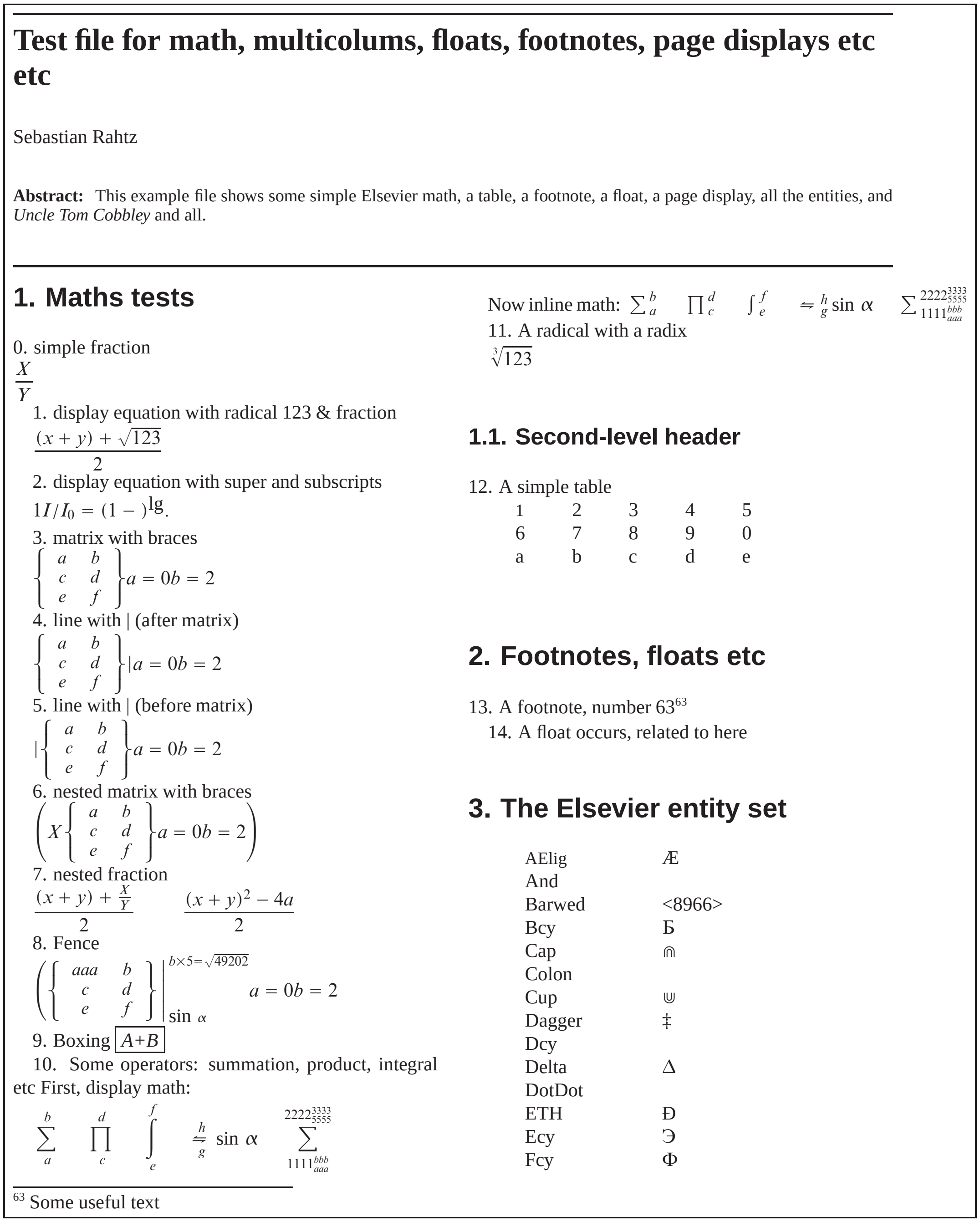
Figure 4: RTF math in Microsoft Word

In addition, JadeTEX has some problems of its own:

1. The table support (while distinctly improved since its initial release) is not complete
2. The handling of white space and line-endings is hard to get right in all circumstances
3. The penalties for paragraph breaking are complicated, and not necessarily right, while DSSSL's hyphenation characteristics have not even been looked at yet.

We also have to consider what will happen if we get a full DSSSL implementation, where the front end will provide parallel streams of input (for the body text, footnotes, floats etc), along with information about how items in the streams have to be synchronized (e.g. appear on the same page), and each stream will have its own independent stack for inherited characteristics. The TEX backend currently handles flow objects with multiple streams by serializing the streams, i.e. giving you them each in sequence. This would not work well for `column-set-sequence`. You would get the main body text for a chapter followed by all the footnotes for the chapter, followed by all the floats for the chapter, plus information about which point in the body text was to be synchronized with each float/footnote. This would almost certainly be a monumental task to program in TEX, and really needs a complete rethink of how the backend works.

All this does not mean that we should despair. The *Jade* DSSSL implementation already supports a huge amount of useful transformation and specification code, and



Test file for math, multicolums, floats, footnotes, page displays etc etc

Sebastian Rahtz

Abstract: This example file shows some simple Elsevier math, a table, a footnote, a float, a page display, all the entities, and *Uncle Tom Cobbley* and all.

1. Maths tests

0. simple fraction

$$\frac{X}{Y}$$

1. display equation with radical 123 & fraction

$$\frac{(x + y) + \sqrt{123}}{2}$$

2. display equation with super and subscripts

$$1I/I_0 = (1 -)^{\lg}.$$

3. matrix with braces

$$\left\{\begin{matrix} a & b \\ c & d \\ e & f \end{matrix}\right\} a = 0b = 2$$

4. line with | (after matrix)

$$\left\{\begin{matrix} a & b \\ c & d \\ e & f \end{matrix}\right\} | a = 0b = 2$$

5. line with | (before matrix)

$$| \left\{\begin{matrix} a & b \\ c & d \\ e & f \end{matrix}\right\} a = 0b = 2$$

6. nested matrix with braces

$$\left(X \left\{\begin{matrix} a & b \\ c & d \\ e & f \end{matrix}\right\} a = 0b = 2 \right)$$

7. nested fraction

$$\frac{(x + y) + \frac{X}{Y}}{2} \qquad \frac{(x + y)^2 - 4a}{2}$$

8. Fence

$$\left(\left\{\begin{matrix} aaa & b \\ c & d \\ e & f \end{matrix}\right\} \right|_{\sin\alpha}^{b\times5=\sqrt{49202}} \qquad a = 0b = 2$$

9. Boxing $\boxed{A+B}$

10. Some operators: summation, product, integral etc First, display math:

$$\sum_{a}^{b} \quad \prod_{c}^{d} \quad \int_{e}^{f} \quad \overset{h}{\underset{g}{\leftrightharpoons}} \sin \alpha \quad \sum_{1111_{aaa}^{bbb}}^{2222_{5555}^{3333}}$$

Now inline math: $\sum_{a}^{b}$ $\prod_{c}^{d}$ $\int_{e}^{f}$ $\leftrightharpoons_{g}^{h} \sin \alpha$ $\sum_{1111_{aaa}^{bbb}}^{2222_{5555}^{3333}}$

11. A radical with a radix

$$\sqrt[3]{123}$$

1.1. Second-level header

12. A simple table

1	2	3	4	5
6	7	8	9	0
a	b	c	d	e

2. Footnotes, floats etc

13. A footnote, number 63[63]

14. A float occurs, related to here

3. The Elsevier entity set

AElig	Æ
And	
Barwed	<8966>
Bcy	Б
Cap	⋒
Colon	
Cup	⋓
Dagger	‡
Dcy	
Delta	Δ
DotDot	
ETH	Đ
Ecy	Э
Fcy	Φ

---

[63] Some useful text

Figure 5: Sample pages, part 1

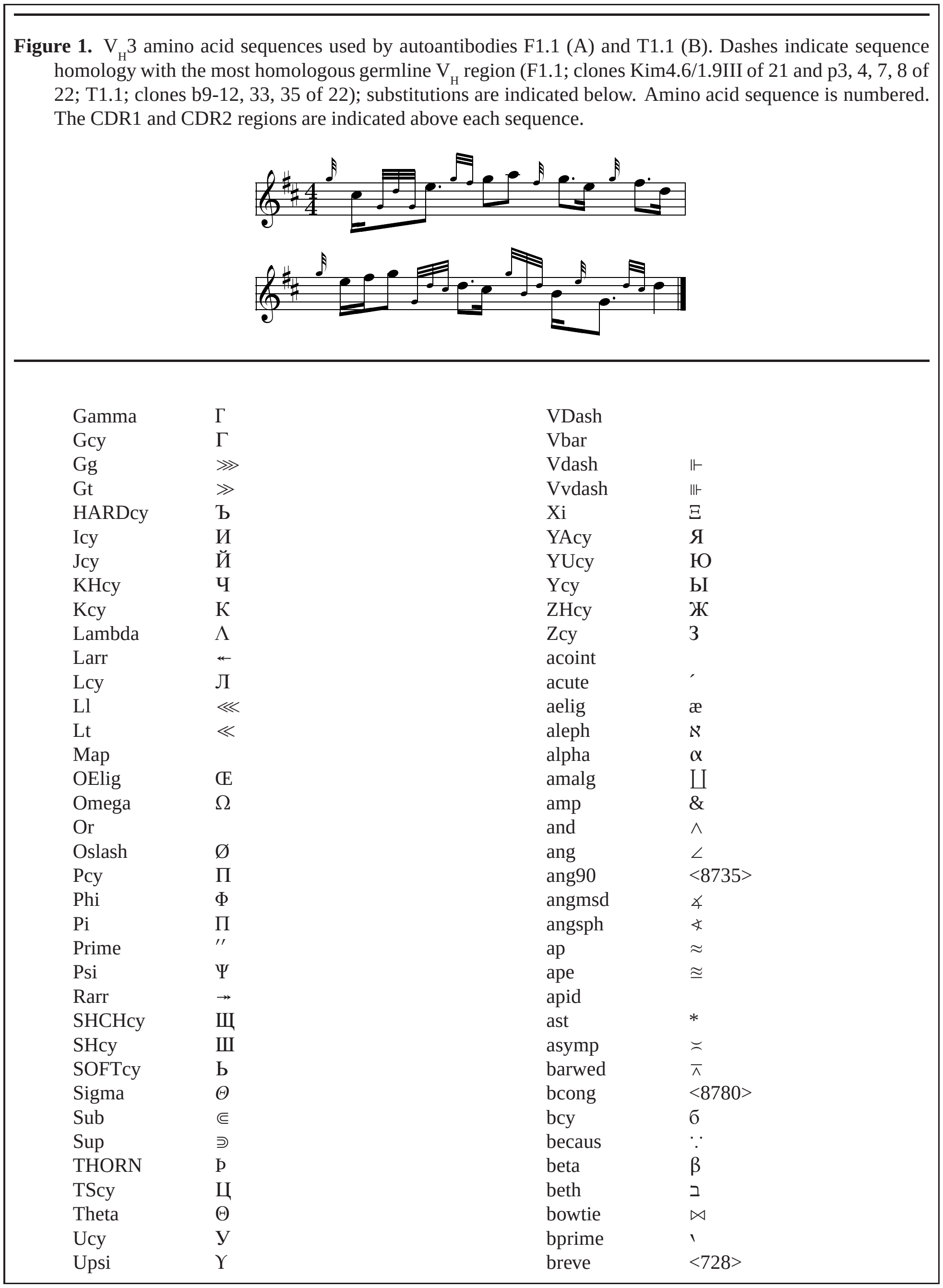


Figure 1. V_H3 amino acid sequences used by autoantibodies F1.1 (A) and T1.1 (B). Dashes indicate sequence homology with the most homologous germline V_H region (F1.1; clones Kim4.6/1.9III of 21 and p3, 4, 7, 8 of 22; T1.1; clones b9-12, 33, 35 of 22); substitutions are indicated below. Amino acid sequence is numbered. The CDR1 and CDR2 regions are indicated above each sequence.

Gamma	Γ
Gcy	Г
Gg	⋙
Gt	≫
HARDcy	Ъ
Icy	И
Jcy	Й
KHcy	Ч
Kcy	К
Lambda	Λ
Larr	↞
Lcy	Л
Ll	⋘
Lt	≪
Map	
OElig	Œ
Omega	Ω
Or	
Oslash	Ø
Pcy	П
Phi	Φ
Pi	Π
Prime	″
Psi	Ψ
Rarr	↠
SHCHcy	Щ
SHcy	Ш
SOFTcy	Ь
Sigma	𝛩
Sub	⋐
Sup	⋑
THORN	Þ
TScy	Ц
Theta	Θ
Ucy	У
Upsi	ϒ
VDash	
Vbar	
Vdash	⊩
Vvdash	⊪
Xi	Ξ
YAcy	Я
YUcy	Ю
Ycy	Ы
ZHcy	Ж
Zcy	З
acoint	
acute	´
aelig	æ
aleph	ℵ
alpha	α
amalg	∐
amp	&
and	∧
ang	∠
ang90	<8735>
angmsd	∡
angsph	∢
ap	≈
ape	≊
apid	
ast	*
asymp	≍
barwed	⊼
bcong	<8780>
bcy	б
becaus	∵
beta	β
beth	ב
bowtie	⋈
bprime	‵
breve	<728>

Figure 6: Sample pages, part 2

TEX is close to being a DSSSL-capable formatter. Since the TEX world knows about Unicode (in the shape of the Omega project, see `http://www.ens.fr/omega`) we are closer than many systems to dealing effectively with true multi-script typesetting.

In the medium term, it will be necessary to rewrite the font handling inside the backend, for speed, and to optimize the handling of labels and references (so many things are labelled at present that TEX can run out of memory for potential cross-references). In the longer term, it would nice to rewrite the JadeTEX macros to be independent of LATEX, and reimplement it to use Omega and native Unicode.

DSSSL is not perfect, and neither is TEX; but they do make a very nice combination. . .