Typesetting Chess

Wolfgang Appelt
Gesellschaft für Mathematik und
Datenverarbeitung, Sankt Augustin

On a recent Sunday morning I was sitting in my garden reading a book on the latest chess world championship between Karpow and Kasparow. Though the book was surely good, giving a lot of explanations concerning the (probable) reasoning of the players, for me as a rather poor chess player, reading chess literature only casually, it presented some problems. There were rather few diagrams in the book showing the position on the board and sometimes for more than twenty moves only the usual textual chess notation (e.g., c2–c4, Kh8×g7) was used to describe the progress of the game.

Of course, for a chess expert it is no problem to read such a notation and to have always a precise visual picture of the position on the board in his mind, but for me as an untrained reader it was rather difficult. Usually, when reading such a book, you are supposed to have a chess board at hand and to perform the moves on the board to watch the progress of the play. However, such a procedure is not always completely satisfactory since it is often much more appropriate to compare simultaneously different positions on the board to understand what is really happening.

In other words, what I really would like was a lot more diagrams in chess books and I started to consider if TEX could be used for such a job. I went to my PC and two hours later I had a set of macros which were at least a starting point for such a task.

The general principles of these macros are described in this article. I shall not show all the details since, as it will be seen later on, there are still some problems left and the rather specific solutions I have chosen might not be of general interest.

The user interface of the macro set should be as close as possible to the conventional chess notation, i.e, you should be allowed to type for example

```
\move e2-e4  \move d7-d5
\move e4xd5  \move Sg8-f6
\showboard
```

and receive a diagram of the actual position on the board. (I shall use the so–called algebraic notation, which is commonly found in German chess literature, and I shall also use the German abbreviations for the pieces, e.g., "S" stands for "Springer" which means knight. However, a "translation" of the macros into the notation scheme of English or any other language should cause no major problems.)

Obviously, the chess board can be typeset as a table where each table entry represents exactly one field on the board. A macro for printing a chess board might therefore look like this:

```
\def\showboard{$$\vbox{\offinterlineskip
  \halign{\vrule\field##.&\field##.&...
           ...&\field##.\vrule\vrule\cr
    \noalign{\hrule}\noalign{\hrule}
    .\@a8&*.\@b8&.\@c8&*.\@d8&...&*.\@h8\cr
    \noalign{\hrule}
    *.\@a7&.\@b7&*.\@c7&.\@d7&...&.\@h7\cr
    \noalign{\hrule}
    .\@a6&*.\@b6&.\@c6&*.\@d6&...&*.\@h6\cr
    \noalign{\hrule}
    ⋮
    \noalign{\hrule}
    *.\@a1&.\@b1&*.\@c1&.\@d1&...&.\@h1\cr
    \noalign{\hrule}\noalign{\hrule}
    }}$$}
```

Each field on the chess board is represented by a macro, called \@a1, \@a2, ... \@a8, \@b1, ... \@h8, as it is the usual convention for naming the fields on a chess board. (Please note that the numbers 1...8 and the at-sign (@) have the *category code* 11, and therefore, e.g., \@b1 is a valid macro name.) Each of these "field macros" must either hold the value "void" or the value of the piece which is on this field at any time in the game. That means we have to keep track of an 8×8 matrix representing the position of each piece on the board. At the start of the game these definitions read as follows:

```
\def\@a8{\ST}\def\@b8{\SS}\def\@c8{\SL}
   \def\@d8{\SD}\def\@e8{\SK}\def\@f8{\SL}
   \def\@g8{\SS}\def\@h8{\ST}
\def\@a7{\SB}...\def\@h7{\SB}
\def\@a6{\void}\def\@b6{\void}
⋮
   \def\@g3{\void}\def\@h3{\void}
\def\@a2{\WB}...\def\@h2{\WB}%
\def\@a1{\WT}\def\@b1{\WS}...
   \def\@g1{\WS}\def\@h1{\WT}}
\def\void{}
```

The next step is providing a macro for performing a single move on the board. A move requires two actions:

— The "start position" of the moved piece must be updated, i.e., the macro \@xn (x ∈ {a...h}, n ∈ {1...8}), must be "cleared", and

— the "target position" of the moved piece must be redefined, i.e., the macro \@xn (x ∈ {a...h}, n ∈ {1...8}), must receive the value of the corresponding piece.

This can be achieved by the following macro:

```
\def\@move#1#2#3#4#5#6{% Syntax:
  % [KDTLSB][a-h][1-8][-x][a-h][1-8]
  % [...] means: select one
  \expandafter
      \def\csname @#2#3\endcsname{\void}%
  \ifx\colour\whitecolour\expandafter
      \def\csname @#5#6\endcsname
          {\csname W#1\endcsname}%
  \else\expandafter
      \def\csname @#5#6\endcsname
          {\csname S#1\endcsname}\fi}
```

The required syntax of the arguments is given as a comment in the macro. The macro first redefines the macro for the starting position to \void and then defines the value of the macro for the target position to the value of the moved piece. The value of this macro depends also on the fact if the move is performed by a white piece or by a black piece. For example, if white is the next player, the two consecutive moves e4×e5 and Sg8–f6, represented by the macro calls `\@move Be4xe5` and `\@move Sg8-f6` will produce the following definitions:

```
\def\@e4{\void} \def\@e5{WB}
\def\@g8{\void} \def\@f6{BS}
```

The next step is providing the macros for the user interface. The conventional algebraic chess notation first gives the value of the piece (in German either K, D, L, S or T), the start position (e.g., f2), followed by a dash or a "×" (if the move removes a piece from the board), followed by the target position. If the piece is a pawn the value of the piece is not given, i.e., a move may be denoted by "e2–e4", "e4×d5", "Sf2–g4" or "Sf2×g4". This is a very simple syntax, only the case of the missing value in the case of a pawn must be handled specially. The following macro can be used:

```
\def\move#1#2#3#4#5#6 {% Syntax:
  % {KDTLS}[a-h][1-8][-x][a-h][1-8]
  % {...} means: select zero or one
  \if#3-\@move B#1#2#3#4#5%
     \else\if#3x\@move B#1#2#3#4#5%
            \else\@move #1#2#3#4#5#6\fi\fi
  \ifx\colour\whitecolour\def\colour{S}%
      \else\def\colour{W}\fi
   }
```

Please note that the last parameter is a *delimited parameter*, i.e., the end of the argument sequence is denoted by a space. This makes the macro work with both five or six arguments.

Now we have assembled all together except the macros for the actual display of the fields of the board. The best way to print a field with or without a piece on it would be by using a special font, containing an empty white field, an empty black field, a white pawn on a white field, a white pawn on a black field, a black pawn on a white field etc. This would sum up to 26 different symbols which should be created by METAFONT. However, this would exceed a Sunday afternoon's entertainment and I therefore used the following, admittedly rather quick and dirty "approximation".

The macro `\showboard` uses the macro `\field` in the preamble of the `\halign`, and this `\field`–macro has two parameters as you may guess by looking at the table lines in the `\halign`. The first one is either empty or a "*", the latter case indicating that the field is a black one (see the "checkered" distribution of the asteriskes in the table lines). The second argument is one of the 64 "field macros". Having no special chess font available, the basic idea is the following:

— The displays of the chess pieces are created by putting together symbols from existing *Computer Modern* fonts.

— Black fields are created by constructing a quadratic pattern of characters from the (usually also available) *gray* font. (Making fat black squares out of `\vrules` or `\hrules` looks really ugly; I tried this first.)

There is only one complication which must be taken care of: If a piece has to be put on a black field, this field must not be completely filled with the background pattern, but there must be left some white space in the middle where the piece is displayed. The `\field`–macro may there look as follows:

```
\newif\ifblackfield
\def\field#1.#2.{\def\next{#1}%
   \ifx\next\empty\blackfieldfalse
       \else\blackfieldtrue\fi
   \ifblackfield\edef\next{#2}%
       \ifx\next\empty\vrule\fieldstrut
           \hbox to \fieldwidth{\hfill
                \emptyblackfield\hfill}%
       \else\vrule\fieldstrut\blackborder
            \setpiece{#2}\fi
   \else\vrule\fieldstrut\setpiece{#2}\fi}
\def\fieldstrut{\vrule height\fieldheight
```

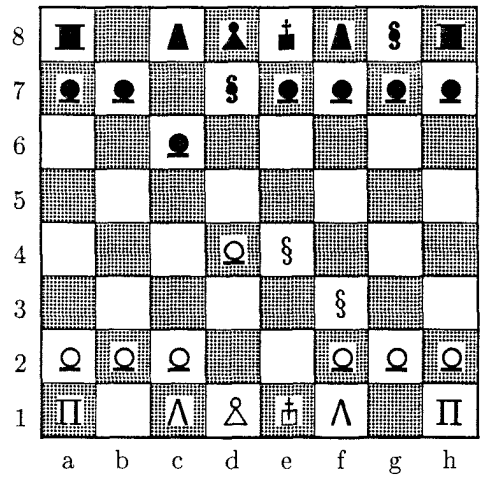
```
                depth\fielddepth width0pt}
\def\setpiece#1{\hbox to \fieldwidth
                    {\hfill#1\hfill}}
```

The macro `\emptyblackfield` which appears in the `\field`-macro is basically a `\vbox`, containing several lines of characters from the *gray* font, and the macro `\blackborder` is rather similar, but it leaves some white space in the middle. Instead of describing these two macros and the macros displaying the different chess pieces in detail, I shall give a small example, whereby you might guess what symbols of the *Computer Modern* fonts I used for the pieces. For example, the input text

```
\move e2-e4  \move c7-c6
\move d2-d4  \move d7-d5
\move Sb1-d2 \move d5xe4
\move Sd2xe4 \move Sb8-d7
\move Sg1-f3
\showboard
```

will give the the following diagram:

Board after White's 5. move



Please note, that the `\showboard`-macro has been slightly extended to print a number and a character, respectively, on the left side and on the bottom of the board. Furthermore, a heading for the diagram is printed, telling the number of the move and the name ("Black" or "White") of the last player. This requires also a small extension of the `\move`-macro to keep track of this information.

Just to give you an impression what the macros for the display of the different chess pieces look like, one example: The definition for a white pawn is:

```
\def\WB{\together{\bbbsym\char14 }%
     {\kern -1pt\hbox{\vrule
      height 1.4pt depth 0pt width 8pt}}}
```

where `\together` is a macro with two parameters which are symbols, characters or rules, which are to be printed atop of each other, and `\bbbsym` is the symbol font at `\magstep3`.

If there is a special chess font available the definitions of the macros displaying the pieces are just selections of characters from this font, e.g.

```
\def\WB{{\chessfont
  \ifblackfield\char11 \else\char39 \fi}}
```

and the definition of the `\field`-macro would become a bit simpler.

Though the macros shown above will not give a professional environment for typesetting chess books they may be used as a good starting point for such a task. The most obvious improvement is, of course, the creation of a set of special symbols by METAFONT. This should be a rather simple task, even for people with a rather limited experience in typographic design.

Furthermore, the macros `\move` and `\@move`, respectively, have to be extended to handle the so–called *castling*, denoted by 0–0 or 0–0–0, and the special pawn move called *capture en passant*, denoted often by, e.g., f5×g6(e.p.). These extensions are rather straightforward.

It is also a simple extension to the `\move`–macro to make it print its arguments. For example, before printing the board the above shown input could give an additional listing of the game in the form:

1.	**e2–e4**	**c7–c6**
2.	**d2–d4**	**d7–d5**
3.	**Sb1–d2**	**d5×e4**
4.	**Sd2×e4**	**Sb8–d7**
5.	**Sg1–f3**	

Making an even more ambitious step, it should also be possible to extend the macros shown above to check if a move is legal or not. For example, if you enter `\move Sg3xf5` you should get an error message if there is no "Springer" on g3 or no piece on f5. In other words, it should be possible to develop a set of TEX macros which know the legal moves of a chess game and which detect typos.

To summarize: TEX can be used for an integrated typesetting of chess games (in conventional notation) and of chess diagrams. Exploiting TEX's macro facilities it should be possible to eliminate typos which can be a great embarassment to the readers. Even if typos are less frequent in well-typeset chess books due to careful proofreading — you will probably find typos more frequently in the

chess columns of newspapers — such an approach might be able to improve the quality and to reduce the costs of chess literature.

Editor's note: The gray font referred to here is normally used to test METAFONT proof characters — it is the font that appears in the character illustrations in Volume E of *Computers & Typesetting*. Unlike ordinary METAFONT fonts, the gray font is device-dependent. That is, different versions, *with different* .TFM *files*, will be used to produce output on devices with different print characteristics, including resolution.

Dr. Appelt originally prepared this article using a laser printer with 300 dots-per-inch resolution; the typesetter on which TUGboat camera copy is prepared has a final resolution of over 1000 dots per inch, although fonts for it are created at 723 dots per inch. Attempts to install a suitable typesetter-specific gray font failed, so the figure of the chessboard has been pasted in from the laser printer copy that Dr. Appelt supplied.

Anyone attempting to use the macros defined in this article, or doing anything else that requires the gray font (including METAFONT), should be aware of this restriction.

Equation Numbering in Plain TEX

J. E. Pittman

A few simple macros can provide facilities for automatic equation numbering with (limited) forward referencing. A backward (after the equation has been displayed) reference to an equation is made in the text by the use of the `\referenceequation{`*name*`}` macro, which generates the appropriate number and inserts it into the text. The `\referenceequation` macro will also work correctly if it is used 'just' before the referenced equation, i.e., as long as there are no numbered equations between the referenced equation and the point of reference.

A forward (before the equation has been displayed) reference to an equation is made by the use of the `\forwardreferenceequation{`*name*`}{`*n*`}` macro, where n is the number of numbered equations that will be displayed between the point of reference and the referenced equation.

Within displayed equations, the `\eqname{`*name*`}` macro can be used in same manner that the `\eqno` *text* macro is normally used. Note: `\eqno` is documented in chapter 19 of The TEXbook.

If an equation is to be numbered but not referenced, the `\eqnum` macro can be used in place of the `\eqname{`*name*`}` macro.

Figure 1 gives an example of the way in which these macros are normally used.

This method of equation numbering is limited due to the requirement of equation counting for forward referencing, however, it works well for most applications and does not require more than one pass through the input file(s).

The following input:

```
% --- example ---
Equation \forwardreferenceequation{byhalves}{2}
gives a simple example of a convergent
infinite series.
$$
E = mc^2        \eqname{emc2}
$$
$$
A = A           \eqnum
$$
$$
1 = \sum_{n=1}^\infty 2^{-n}
  = {1 \over 2} + {1 \over 4} +
    {1 \over 8} + \cdots    \eqname{byhalves}
$$
\TeX\ reduces the task of typesetting
Einstein's famous equation
(\referenceequation{emc2}) to pure
simplicity.
\par
```

Produces:

Equation 3 gives a simple example of a convergent infinite series.

$$E = mc^2 \tag{1}$$

$$A = A \tag{2}$$

$$1 = \sum_{n=1}^{\infty} 2^{-n} = \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots \tag{3}$$

TEX reduces the task of typesetting Einstein's famous equation (1) to pure simplicity.

Figure 1. Example of equation numbering macro use