Paradigms: Searching

Kees van der Laan

1 BLUe's Design VIII

Hi folks. I started using LaTeX for my hobby bridge, to typeset bid sequences and plays. Important in these kind of plays is data integrity, i.e. the system should remember that a card has been played. In TeXnical terms it comes down to update the computer memory. This is precisely what makes computer-assisted formatting different from previous formatting techniques. We can update memory.

This is a common phenomenon and the important building block is the handling of sets, that is trace an element and update the set, or extract from it, casu quo add to it.

The central macro in BLUe's format searching is called `\loc`, from locate.

Operations are to modify such as delete a card in bridge, or to extract, to copy the searched for element. The latter is applied abundantly in BLUe's format selective loading.

Below I'll discuss searching TeXniques, as used in The TeXbook, David Salomon's searching, and the various search activities which have been applied in BLUe's format.

2 Searching in TeX

Intrinsic in TeX is the searching for optimal linebreaks within a paragraph and for good pagebreaks of the main vertical list. Too complex material to be treated in this note.

3 Searching in METAFONT

A very nice feature is its capability to solve equations especially to determine intersection points of curves, specified declaratively. Searching via METAFONT will not be addressed in this paper.

4 Searching in The TeXbook

The examples of searching are about balancing columns.

On page 387 it is applied to adjusting the column widths when we have the English text in column one and the same text in another language in column two. With fixed column widths the column lengths are different.

On page 417 the manmac macro `\balancecolumns` is given to balance two columns on the last page of the index of The TeXbook. The latter TeXniques and macros have been used—and customized a little—for typesetting BLUe's format indexes.[1]

5 Salomon's procrusting

Salomon[2] looks for the optimal font size in fitting text to a box of prescribed size. The essence of the search process is given below stripped from other aspects. I modified his example into the following.

```
%Salomon's fitting text to a box
       %<TeX_Marker>
%Sept 95, cgl@rc.service.rug.nl
\begingroup
%Text
\toks0{Leentje leerde lotje lopen
       langs de lange lindenlaan.}
%Restrictions
\hsize=3cm \dimen1=20pt%\vsize=20pt
\def\showresult{\rlap{\f
    Fontsize=\the\dimen0\quad
    ht, dp=\the\ht0, \the \dp0}
    \copy0\smallbreak}
%Start: first estimate size of font
\dimen0=5pt
\loop\font\f=cmr10 at\dimen0
   \setbox0=\vbox{\tolerance10000
   \baselineskip\dimen0
   \f \the\toks0 }
\showresult
\ifdim\ht0<\dimen1 %next, linear search
   \advance\dimen0 by1pt
\repeat
%make the best fit
\advance\dimen0 by-1pt
\font\f=cmr10 at\dimen0
   \setbox0=\vbox to\dimen1{\tolerance10000
   \baselineskip\dimen0 plus 1pt minus1pt
   \f \the\toks0 }
\showresult
\endgroup

%Remark: extension to adjust for total size
%of box is left to the reader, and the
%context
\bye
```

Remark. For more advanced applications see Arsenau's macros 'Typesetting paragraphs of a specific shape,' MAPS 93.2.

6 Finding an element

The basic functionality is to locate an element in a set. The result of the process—success or failure—is stored in the 'Boolean' `\iffound`.

[1] My contribution in this area is that I process indexes on-the-ly, in one-pass.

[2] Advanced TeX, Springer, 1995, page 189.

[3] Assumed is that the set does not contain a period.

BLUe's format uses the following approach.[3]

```
\def\loc#1#2{%locate #1 in #2 a def
  \def\locate##1#1##2\end{\ifx\empty##2\empty
    \foundfalse\else\foundtrue\fi}%
  \ea\locate#2.#1\end}
```

To append the searched for element at the end of the arguments for `\locate` is related to the sentinel technique in programming.

Example (Printing vowels in bold)

Schwarz 1987 coined the problem to print vowels in bold face.[4]

The problem can be split into two parts. First, the general part of going character by character through a string, and second, decide whether the character at hand is a vowel or not.

For the first part use `\fifo`.

For the second part, combine the vowels into a string, `aeiou`, and the problem can be reduced to the question ⟨*char*⟩ ∈ `aeiou`?

With `\process` appropriately defined—locate the argument in the string of vowels—`\fifo Audacious\ofif` yields **Au**d**a**c**iou**s, with

```
\def\process#1{\uppercase{\loc#1}%
  {AEIOU}\iffound{\bf#1}\else#1\fi}
```

Example (Searching a set of accent strings)

In sorting in TEX I determine whether a control symbol belongs to the set of accents—`\def\accstr{\‘\’\"\^\c}`. In the macro `\nxtw` the relevant invoke reads as follows.

```
\ea\loc\head\accstr
```

The same approach has been applied in my BLUe-2-MAPS convertor assistant.

Example (Searching a set of Pascal reserved words)

In typesetting PASCAL fragments I determine whether a word belongs to the set of reserved ‘words.’ The set of reserved words reads

```
\reservedset{and array begin case const
 div do downto else end. end; file for
 function goto if in label mod nil not of
 or packed procedure program record repeat
 set then to type until var while with}
```

The relevant invoke in `\processw` reads

```
\loc{#1}{\the\reservedset}%
```

The Pascal environment scans the program fragment line-by-line and each line word-by-word. Each word is tested against the set of reserved words.

7 Deleting an element

The functionality is to delete an element from a set. It consists of two steps: finding and deleting. It can be coded as a variant of `\loc`, with as result not a Boolean but the modified set. A template might read as follows.

```
\def\delete#1#2{%Delete #1 from #2 a def
  \def\locate##1#1##2\end{\ifx\empty##2\empty
    \else\def#2{##1##2}\fi}%
  \ea\locate#2#1\end}
```

Addition of an element to a set can be done as follows.[5]

```
\def\add#1#2{%Add #1 to #2 a def
   \ea\def\ea#2\ea{#2#1}}
%or when #2 consists of unexpandable tokens
\def\add#1#2{%Add #1 to #2 a def
   \xdef#2{#2#1}}
```

Example (Updating a hand in bridge)

Bridge is an elimination play like most games. From the start elements are removed. The deletion of a card is a little special because we know that the card is there. The macro is called `\strip` and has been adapted. The hand is a definition instead of a token variable.

```
\def\strip#1#2{%Function:
% delete card value #1, is AKQJT9...2
% from #2, a def.
  \def\wis##1#1##2\siw{\gdef#2{##1##2}%
   \ifx#2\empty\gdef#2{--}\fi}%
  \ea\wis#2\siw}
```

Remarks. In the above example the searching (and deleting) is a side-effect of the printing. It is merely there to guarantee data integrity. If only attention is paid to the main issues, the searching would have been remained hidden, under a ‘pile of cards’. It is worth it to make some kind of library macro out of it—at least a template—ready for reuse.

Realize that the searched for element is supplied dynamically.

Example (Modifying `\dospecials`)

Inspired upon the Babel macros the following alternatives which also obey the locality character.

```
%Variant \bbl@add@specials
%No grouping, nor edef,
%assumed \dospecials, \sanitize are defined.
%
\let\sanitize\empty \let\dospecials\empty
%
\def\addspecials#1{\ea\def\ea
  \dospecials\ea{\dospecials\do#1}%
                 \ea\def\ea
  \sanitize\ea{\sanitize\makeother#1}
}
%
%Variant \bbl@remove@specials
%No grouping, nor edef,
%assumed \dodefault available
%
\let\dodefault\empty
%
```

[4] His solution mixes up the picking up of list elements and the process to be applied. Moreover, his nesting of `\if`-s in order to determine whether a character is a vowel or not, is not elegant.

[5] This is not so much about searching but added for your convenience.

```
\def\removespecials#1{%
   \def\do##1{\ifnum'#1='##1
              \else\nx\do\nx##1\fi}
   \edef\dospecials{\dospecials}
   \def\makeother##1{\ifnum'#1='##1
              \else\nx\makeother\nx##1\fi}
   \edef\sanitize{\sanitize}
\let\do\dodefault
\let\makeother\makeotherdefault
}
```

8 Dek's set macros

In The TEXbook the locating of an element—and delete it—is done as follows.

Dek provides `\remequivalent`,[6] The TEXbook 380, which uses a very general TEXnique. I'll untangle and recast the coding in the FIFO—First In First Out—notation.

Example (A set is just a list of defs)

Suppose the set reads `\def\set{\a\b\c}`. Then

```
%Assume #2 not empty
\def\remequivalent#1\from#2{%
   \def\process##1{\ifx#1##1\else\nx##1\fi}
   \xdef#2{\ea\fifo#2\ofif}}
\def\fifo#1{\ifx\ofif#1\ofif\fi
   \process{#1}\fifo}
\def\ofif#1\fifo{\fi}
```

Example (A set is a list of defs with list element tags)

Suppose the set reads `\def\set{\\\a\\\b\\\c}`. Then

```
%Assume #2 not empty
\def\remequivalent#1\from#2{%
   \def\\##1{\ifx#1##1\else\nx\\\nx##1\fi}%
   \edef#2{#2}}
```

Once we use list element tags we can also have more general elements, as Dek put it 'control sequences which are `\ifx`-equivalent to a given control sequence.'

Example (A set is a more general list of control sequences)

Suppose the set reads `\def\set{\\\a\\\b\\\c}`. Then

```
%Assume #2 not empty
\def\remequivalent#1\from#2{\let\given=#1%
\def\\##1{\ifx#1##1\else\nx\\\nx##1\fi}%
   \edef#2{\ea\fifola#2\alofif}}
\def\fifola\\#1#2{\ifx#1\given\else\nx\\\nx#1\fi
   \ifx#2\alofif\alofif\fi\fifola#2}
\def\alofif#1\alofif{\fi}
```

Finally, it can be extended by the test for the emptiness of `\set`.

9 Database use

Databases are all about easy input, conveniently storing and retrieving, and appropriately reporting.

While developing BLUe's format system and using the database idea for storing and retrieving other search TEXniques have been coded. Needed are facilities for browsing a database next to macros for selective loading of addresses, a format, pictures, references, or tools.

The browse macros are based on `\loc`, and can be associated with keyword pattern recognition.

The selective loading macros are based on the proper definition of the list element tag.

Below the searching aspects have been put together. 'BLUe's Databases' treats the use and coding of BLUe's format databases in depth.

9.1 Pattern recognition

A nice application is mail-merge, merging addresses with a letter.

Example (Match addresses for the pattern RUSSIA)

```
\input blue.tex \letter \searchfile{address}
\search{RUSSIA}
\bye
```

To send a letter to all those persons or to make all address labels insert `\makesearchletters`, respectively `\makesearchlabels` before `\bye`.

Example (Coding the search macro)

At the heart lies the earlier mentioned `\loc` macro. Because we need to do more with found entries than just knowing that they are there, their names, preceded by the list element tag for further action, are collected in the toks variable `\namelst`. For convenience the names are also written to the log file, in order that we can follow what is going on.

```
\def\search#1{\def\loc##1##2{%
   \def\locate####1##1####2\end
   {\ifx\empty####2\empty\foundfalse
   \else\foundtrue\fi}\ea\locate##2.##1\end}
   \def\lst##1##2{\loc{#1}{##2}\iffound
   \immediate\write16{\nx##1}%log file
   \namelst\ea{\the\namelst\lst##1}
   \def##1{##2}%define found element
   \fi}\input\the\searchfile.dat\relax}
```

9.2 Selective loading

This is all about organizing collections and reusing parts of it. More restricted it is about using memory space economically, just to load what is needed, trading time—selection process and loading—for space.

Selective loading is also a search activity. The file is scanned and when an entry is found it will be loaded. The details of the selective loading process depend on the entries of the database, how they are stored. I distinquish class I and class II (TEX) databases. The first class consists of formats and tools—which could have been loaded non-selectively—and the second class consists of addresses, pictures, and references, for which selective loading is essential.

[6] As part of a suite of set macros.

9.2.1 Formats and tools

The idea is that for example `\letter` only loads the letter macros from `fmt.dat`. In the examples below use is made of `\tool`, with the functionality that when the tag after is known the control sequences which follow the tag up to `\endinput` will be loaded, otherwise all in between the tag and `\endinput` will be gobbled.

```
\long\def\tool#1{\ifx#1\undefined
          \bgroup\unouterdefs
            \ea\gobbletool\fi}
\long\def\gobbletool#1\endinput{\egroup}
```

Example (Coding the loading of the letter macros)

```
%From blue.tex
\def\letter{\ifx\undefined\letterfmt
   \let\letterfmt=x\fi
   \loadformat
   \let\letterfmt\undefined}
\def\loadformat{\input fmt.dat \relax}
%from fmt.dat
\tool\letterfmt
<lettermacros>
\endinput
```

Similarly, a tool can be loaded as follows.

Example (Coding the loading of the smiley macros)

```
%From blue.tex
\newbox\smileybox
\newcount\smileysloadcount
\def\smiley{\loadsmileys\raise.5ex
   \hbox{\unitlength.01pt
   \copy\smileybox
   \eyes\mouth
   \kern1000\unitlength} }
\def\winksmiley{<similar>}
\def\sadsmiley{<similar>}
\def\loadsmileys{\ifx\undefined\smileystool
   \let\smileystool=x\fi
   \ifnum\smileysloadcount=0 \ea\loadtool
   \else\message{--- smileys already
                  loaded---}\fi
   \advance\smileysloadcount1
   \let\smileystool\undefined}
%from tools.dat
\tool\smileystool
<smileymacros>
\endinput
```

Remark. Whenever suited a load counter is maintained such that double loading is inhibited.

9.2.2 Addresses, pictures and references

The entries of these database obey the syntax

`\lst\<name>{<entryproper>}`

The selective loading comes down to a proper definition of `\lst`. Moreover, the names of the entries to be selected must be defined with whatever you wish as replacement text.[7]

```
\def\loadselectivefrom#1{%#1 lit etc.
 \def\lst##1{\ifx##1\undefined\ea\gobble
         \else \ea\gdef\ea##1\fi}
 \input #1.dat \relax%e.g. lit.dat
 }
\def\gobble#1{}
```

Because of the scanning `\outer` defs are not allowed, nor are `\par`-s. The selective loading macro is embedded in the user macros `\references` and its ilks. In detail the meaning of 'loading' is adapted to the application. For references this means that the specified entries are set in a box and their names redefined by numbers. The names can be used for cross-referencing purposes while the box can be pasted up at your place of choice. However, the underlying searching methodology is the same for addresses, references and pictures.

Example (Coding the handling of references)

```
\def\references#1{\beginreferences#1%
   \endreferences}
\def\bluereferences#1\par
   {\beginreferences#1\endreferences}
\def\loadreferences{\loadselectivefrom
   {\the\referencesfile}}
\def\beginreferences#1\endreferences{%
  \bgroup\def\process##1{\ifx\undefined##1
     \global\let##1\referenceserror\else
     \message{***\tt\string##1
             already loaded.***}
  \fi\namelst\ea{\the\namelst\lst##1}}%
  \fifo#1\ofif
  \if]#1]\else\ea\loadreferences\fi
  %formatting
  \ifstore\global\setbox\referencesbox=
    \vbox\bgroup\fi\prenum{}\postnum{}
  \lsams%Default ls
  \the\thisreferences
  \def\lst##1{\ls{##1}
    \xdef##1{\the\itemno}}
  \the\namelst\endreferences}
```

Remark. BLUe's format style of coding is centred along two-part macros with a one-part macro on top, enriched by a convenient alias such as `\bluereferences`, for the user interface.

9.2.3 Max Díaz' TEXnique

The TEXbook 382–384 mentions the fast loading TEXnique of Max Díaz, which requires that every line is preceded by a special character. The process comes down to the following.

```
%The TeXbook, Appendix D 4.Selective loading.
%The Max Diaz fast selective loading process.
%(A little simplified, and combined with
% my list element tag, \lst.)
\def\lst#1{\catcode`\ =%tilde
   \ifx#1\undefined14    %comment
             \else9 \fi}%ignore char
%We want to load the cgl part.
\def\cgl{<anything>}

\lst\name
~\ a
~\ b
~\ c
\lst\cgl
~\ aa
~\ bb
~\ cc
\lst\erik
```

[7] This approach is the opposite of preventing reloading. We tacitly want to redefine the fancy entries by the meaningful ones. My fancy replacement text is an error message.

```
~\ aaa
~\ bbb
~\ ccc
\catcode`\~=13
%
<Commonpart>
\bye
%Explanation: the list element tag toggles
%the catcode for ~ such that either the
%first character is ignored (and the rest
%of the line loaded) or the line is a
%comment line.
```

In the above one can replace `\lst\name...` by `\input <filetobeloadedfrom>`.

9.2.4 Variant document parts

The idea is that a script is marked up also with markup tags which have a selection/omission function. For example an abridged version is interspersed within the script. The idea is that the owner can either ask for an abridged or a full version. Another example is documentation with details for various computer operating systems. Given a customer with a specific operating system only the relevant parts will be printed.[8]

With the new hype HTML this functionality may enjoy a second youth.

10 Tree search

When I implemented trees in TEX especially the Pythagorean trees—to illustrate turtle graphics—I played a little longer with it and could use TEX in 'dialogue mode' to search for a name by answering questions.

10.1 Interactive path through a binary tree

The following is inspired upon Greene's 'Playing in TEX's mind.'[9]

```
%Guess what? August 1995, cgl@rc.service.rug.nl
%Idea biased by A.M.Greene's
%Playing in TeX's mind, TUG 89.
%Interactive binary tree traversal.
%Interactive TeX ing,
%TeX as an engine to play with.
\input blue.tex
\def\bintree{\message{\csname\node\endcsname}
   \ea\ifx\csname\node0\endcsname\relax\eertnib\fi
   \read0to\yorn
   \edef\node{\node\if n\yorn1\else0\fi}
   \bintree}
\def\eertnib#1\bintree{\fi\def\node{1}
   \immediate\write0{Hope this is the one
               you are looking for :-) }
   \immediate\write0{}
   \message{Another play?}
   \read0to\yorn
   \if y\yorn\ea\bintree\fi
   \immediate\write0{Thank you, bye}}
%Rules of the game
\immediate\write0{Guess game.
 The system asks questions to be answered}
\immediate\write0{by *** y or n ***}
\immediate\write0{The following play
             guesses an NTG member}
\immediate\write0{}
%
%Data (a tree structure)
\begingroup\obeylines
\def\lst#1 #2
   {\ea\def\csname#1\endcsname{#2}}
\lst 1 NTG member?
\lst 10 Plain TeX ie?
\lst 100 Honoured?
\lst 1000 Kees
\lst 1001 HH
\lst 101 On board?
\lst 1010 Chair?
\lst 10100 Erik
\lst 10101 Secretary?
\lst 101010 Gerard
\lst 101011 Treasurer?
\lst 1010110 Wietse
\lst 1010111 Dark?
\lst 10101110 Johannes
\lst 10101111 Frans
\lst 1011 Anonymous
\lst 11 Just a friend
%
%Start the play
\endlinechar-1 %TB20.18
\def\node{1}\bintree
\endgroup
%
%Typesetting the data
\onecol
Pretext
\thisbt{\xoffset{-400}}
{\obeylines
\beginbt1 NTG member?
10 Plain TeX ie?
100 Honoured?
1000 cgl
1001 HH
101 On board?
1010 Chair?
10100 Erik
10101 Secretary?
101010 Gerard
101011 Treasurer?
1010110 Wietse
1010111 Dark?
10101110 JLB
10101111 FG
1011 Anonymous
11 Just a friend
8 \endbt
}Posttext
\bye
```

A typical log file looks as follows.

```
Guess game. The system asks questions to
be answered by *** y or n ***
The following play guesses an NTG member

NTG member?
\yorn=y
 Plain TeX ie?
\yorn=y
 Honoured?
\yorn=y
 Kees
Hope this is the one you are looking for :-)

Another play?
\yorn=y
 NTG member?
\yorn=n
 Just a friend
Hope this is the one you are looking for :-)
```

[8] In real-life this is hardly done. When I buy a TV the operation booklet contains the information in several languages.

[9] Courtesy Bernd Raichle for node representation.

```
Another play?
\yorn=y
 NTG member?
\yorn=y
 Plain TeX ie?
\yorn=n
 On board?
\yorn=y
 Chair?
\yorn=y
 Erik
Hope this is the one you are looking for :-)

Another play?
\yorn=y
 NTG member?
\yorn=y
 Plain TeX ie?
\yorn=n
 On board?
\yorn=y
 Chair?
\yorn=n
 Secretary?
\yorn=y
 Gerard
Hope this is the one you are looking for :-)

Another play?
\yorn=y
 NTG member?
\yorn=y
 Plain TeX ie?
\yorn=n
 On board?
\yorn=n
 Anonymous
Hope this is the one you are looking for :-)

Another play?
\yorn=n
```

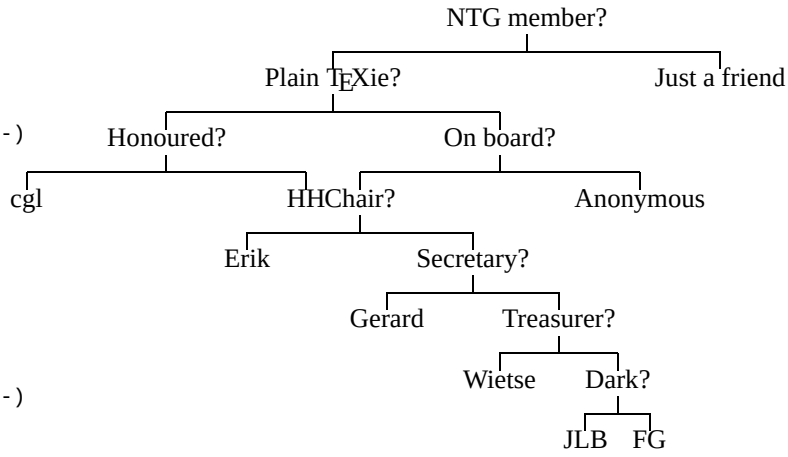
Remarks. The first version used a counter to represent the nodes with the restriction of 2^{16}.

Robustness with respect to mistypes of the user after the prompt—the user does not supply ‘y’ or ‘n’—has been treated in Paradigms: Loops.

10.2 The tree

The typesetting of the binary tree visualizes the data for your convenience. Within BLUe’s format this goes as follows

```
\thisbt{\xoffset{-400}}
\beginbt 1 NTG member?
10 Plain \TeX ie?
...
11 Just a friend
8 \endbt
```



Have fun, and all the best