Under consideration for publication in J. Functional Programming

Luis Brandao
Porto University, Portugal
(*e-mail:* up201504440@fc.up.pt)

Abstract

Research on the implementation of Linear Type systems in Haskell

Contents

ZU064-05-FPR output 20 May 2018 14:38

1 Introduction

2 Background

2.1 Linear Haskell

3 Linear Type Systems

4 Correctness

5 Implementation

$$
\begin{array}{lcl}
data\ Type & = & TypeArrow\ Multiplicity\ Type\ Type \mid TBoolean\ deriving(Eq, Show) \\
dataMultiplicity & = & One \mid Omega\ deriving\ (Show, Eq) \\
exp & :: & Exp \rightarrow Arr \rightarrow Val \\
exp\ (Var\ i)\ a & = & index\ i\ a \\
exp\ (Const\ v)\ a & = & v \\
exp\ (Plus\ e_1\ e_2)\ a & = & exp\ e_1\ a + exp\ e_2\ a \\
 & & \\
com & :: & Com \rightarrow Arr \rightarrow Arr \\
com\ (Asgn\ i\ e)\ a & = & update\ i\ (exp\ e\ a)\ a \\
com\ (Seq\ c_1\ c_2)\ a & = & com\ c_2\ (com\ c_1\ a) \\
com\ (If\ e\ c_1\ c_2)\ a & = & \text{if}\ exp\ e\ a == 0\ \text{then}\ com\ c_1\ a\ \text{else}\ com\ c_2\ a \\
 & & \\
prog & :: & Prog \rightarrow Val \\
prog\ (Prog\ c\ e) & = & exp\ e\ (com\ c\ (newarray\ 0))
\end{array}
$$

6 Conclusion

A new environment exists for creating Proofs, e.g.

Proof
Use K_λ and S_λ to translate combinators into λ-terms. For the converse, translate $\lambda x \ldots$ by $[x] \ldots$ and use induction and the lemma. □

This was produced by the following code:

```
\begin{proof}
  Use $K_\lambda$ and $S_\lambda$ to...
\end{proof}
```

The end of proof marker □ is produced automatically. If you wish to omit this, use the `proof*` environment instead.

If a proof ends with a display equation, then it is customary for the proofbox to be positioned at the end of equation finishing the proof. *e.g.*

Proof
Use K_λ and S_λ to translate combinators into λ-terms. For the converse, translate $\lambda x \ldots$ by $[x] \ldots$ and use induction and the lemma.

$$a_1 \equiv (2\Omega M^2/x) \quad \square$$

Was produced with:

```
\begin{proof*}
  Use $K_\lambda$ and $S_\lambda$ to...
  \[ a_1 \equiv (2\Omega M^2/x) \mathproofbox \]
\end{proof*}
```

Notice the use of `proof*` to turn off the automatic proofbox.

The proof environment will also take an optional argument which allows you to produce 'special' proofs. e.g.

Proof of Theorem 27
We define a linear isometry $A : H \to H$ by the nonlinear Schrödinger equation. It would not be hard to modify the proof to obtain an analogous result for ellipsoids rather than spheres. □

Which was produced like this:

```
\begin{proof}[Proof of Theorem 27]
  We define a linear isometry...
\end{proof}
```

Notice that once the optional argument is used, you have to type all of the text which is to appear as the heading.

6.1 Programs

JFP encourages authors to use one of two styles for typesetting programs, *mathematical* and *verbatim*.

A program typeset in the mathematical style is shown in Figure 1, and the commands used to typeset this program are shown in Figure 2. This uses the ordinary mathematics mode of LaTeX: displayed programs are surrounded by `\[` and `\]`, and the `array` command is used for alignment. However, there are two important differences. First, the `\programmath` command appears before the program text; this causes math mode to use ordinary spacing for italic identifiers, rather than math spacing. The `\unprogrammath` command returns to normal math spacing. Second, in math mode spaces are ignored, so a tilde `~` is used instead. (In LaTeX, a tilde generates a "hard space" that is never replaced by a line break.) To include program text in mathematics style inline, surround it with dollar `$` signs. For example, the input

```
See how \programmath $differ~x$
differs from \unprogrammath $differ x$.
```

produces the output

See how $differ\,x$ differs from $differx$.

A program typeset in the verbatim style is shown in Figure 3, and the commands used to typeset this program are shown in Figure 4. This uses the ordinary verbatim mode of LaTeX: displayed programs are surrounded by `\begin{verbatim}` and `\end{verbatim}`,

Table 1. *New symbol macros*

Symbol	Usage	Keyed as
`\dplus`	$abc + + xyz$	`$abc \dplus xyz$`
`\dequals`	$abc == xyz$	`$abc \dequals xyz$`
`\dcolon`	$abc :: xyz$	`$abc \dcolon xyz$`
`\dcolonequals`	$abc ::= xyz$	`$abc \dcolonequals xyz$`

and alignment is indicated with spaces in the source file (don't use tabs, which may not be processed properly). To include program text in verbatim style inline, use the `\verb` command. For example, the input

```
On a terminal, this looks like \verb"differ x".
```

produces the output

On a terminal, this looks like `differ x`.

It is recommended that programs in figures be offset from the text using the `\figrule` command, as shown in Figures 1–4.

Some new macros have been provided for a few convenient symbols in math mode. These are illustrated in Table 1.

6.2 Abstract and Capsule reviews

The JFP class provides for an abstract and/or a capsule review; the abstract is produced by the following commands:

```
\begin{abstract}
     :
\end{abstract}
```

whereas the capsule review is produced by:

```
\begin{capsule}
     :
\end{capsule}
```

Either or both of these may be used, in either order, but it is assumed that, if both are used, there will be no other material between them. In JFP the abstract should precede any capsule review.

6.3 Lists

The JFP class provides the three standard list environments.

- Numbered lists, created using the `enumerate` environment;
- Bulleted lists, created using the `itemize` environment;

$$
\begin{array}{lcl}
exp & :: & Exp \to Arr \to Val \\
exp\ (Var\ i)\ a & = & index\ i\ a \\
exp\ (Const\ v)\ a & = & v \\
exp\ (Plus\ e_1\ e_2)\ a & = & exp\ e_1\ a + exp\ e_2\ a \\
 & & \\
com & :: & Com \to Arr \to Arr \\
com\ (Asgn\ i\ e)\ a & = & update\ i\ (exp\ e\ a)\ a \\
com\ (Seq\ c_1\ c_2)\ a & = & com\ c_2\ (com\ c_1\ a) \\
com\ (If\ e\ c_1\ c_2)\ a & = & \text{if } exp\ e\ a == 0 \text{ then } com\ c_1\ a \text{ else } com\ c_2\ a \\
 & & \\
prog & :: & Prog \to Val \\
prog\ (Prog\ c\ e) & = & exp\ e\ (com\ c\ (newarray\ 0))
\end{array}
$$

Fig. 1. Example program in mathematical style.

```
\begin{figure}
\figrule
\programmath
\[
\begin{array}{lcl}
exp                   & :: & Exp \rightarrow Arr \rightarrow Val \\
exp~(Var~i)~a         & =  & index~i~a \\
exp~(Const~v)~a       & =  & v \\
exp~(Plus~e_1~e_2)~a  & =  & exp~e_1~a+exp~e_2~a \\
\\
com                   & :: & Com \rightarrow Arr \rightarrow Arr \\
com~(Asgn~i~e)~a      & =  & update~i~(exp~e~a)~a \\
com~(Seq~c_1~c_2)~a   & =  & com~c_2~(com~c_1~a) \\
com~(If~e~c_1~c_2)~a  & =  & \textrm{if } exp~e~a \dequals 0 \textrm{ then }
                                com~c_1~a \textrm{ else } com~c_2~a \\
\\
prog                  & :: & Prog \rightarrow Val\\
prog~(Prog~c~e)       & =  & exp~e~(com~c~(newarray~0))
\end{array}
\]
\unprogrammath
\caption{Example program in mathematical style.}\label{mathfigure}
\figrule
```

Fig. 2. Typesetting the example program in mathematical style.

- Labelled lists, created using the `description` environment.

The `enumerate` environment numbers each list item with an arabic numeral; alternative styles can be achieved by inserting a redefinition of the number labelling command after the `\begin{enumerate}`. For example, a list numbered with roman numerals inside parentheses can be produced by the following commands:

```
\begin{enumerate}[(iii).]
  \renewcommand{\theenumi}{(\roman{enumi})}
```

```
exp                 :: Exp -> Arr -> Val
exp (Var i) a       =  index i a
exp (Const v) a     =  v
exp (Plus e1 e2) a  =  exp e1 a + exp e2 a

com                 :: Com -> Arr -> Arr
com (Asgn i e) a    =  update i (exp e a) a
com (Seq c1 c2) a   =  com c2 (com c1 a)
com (If e c1 c2) a  =  if  exp e a == 0  then  com c1 a  else  com c2 a

prog                :: Prog -> Val
prog (Prog c e)     =  exp e (com c (newarray 0))
```

Fig. 3. Example program in verbatim style.

```
\begin{figure}
\figrule
\begin{center}
\begin{verbatim}
exp                 :: Exp -> Arr -> Val
exp (Var i) a       =  index i a
exp (Const v) a     =  v
exp (Plus e1 e2) a  =  exp e1 a + exp e2 a

com                 :: Com -> Arr -> Arr
com (Asgn i e) a    =  update i (exp e a) a
com (Seq c1 c2) a   =  com c2 (com c1 a)
com (If e c1 c2) a  =  if  exp e a == 0  then  com c1 a  else  com c2 a

prog                :: Prog -> Val
prog (Prog c e)     =  exp e (com c (newarray 0))
\end{verbatim}
\end{center}
\caption{Example program in verbatim style.}\label{verbfigure}
\figrule
\end{figure}
```

Fig. 4. Typesetting the example program in verbatim style.

```
  \item first item
        :
\end{enumerate}
```

This produces the following list:

(i). first item
(ii). second item
(iii). *etc.*

Notice that an optional argument "`(iii).`" has been given to the `enumerate` environment, specifying the *widest label* used in the list. This is because roman numerals are wider than

the arabic numerals normally used by `enumerate`, and so the labels would otherwise have been pushed out into the margin.

7 User-defined macros

If you define your own macros, you must ensure that their names do not conflict with any existing macros in LaTeX (or AMS LaTeX if you are using this). You should also place them in the preamble of your input file, between the `\documentclass` (but after any `\usepackage` commands) and before the `\begin{document}` command.

Apart from scanning the indexes of the relevant manuals, you can check whether a macro name is already used by using `\newcommand`, which will check for the existence of the macro you are trying to define. If the macro exists LaTeX will respond with:

```
! LaTeX Error: Command ... already defined.
```

In this case you should choose another name, and try again.

Such macros must be in a place where they can easily be found and modified by the journal's editors or typesetter. They must be gathered together in the preamble of your input file, or in a separate `macros.tex` file with the command `\input{macros}` in the preamble. Macro definitions must not be scattered about your document where they are likely to be completely overlooked by the typesetter.

The same applies to font definitions that are based on Computer Modern fonts. These must be changed by the typesetter to use the journal's correct typeface. In this case, you should draw attention to these font definitions on the hard copy that you submit for publication and by placing a comment in your input file just before the relevant definitions, for example `% replace font!`

8 Some guidelines for using standard facilities

The following notes may help you achieve the best effects with the JFP class file.

8.1 Sections

LaTeX provides five levels of section headings and they are all defined in the JFP class file:

Heading A – `\section{...}`
Heading B – `\subsection{...}`
Heading C – `\subsubsection{...}`
Heading D – `\paragraph{...}`
Heading E – `\subparagraph{...}`

Section numbers are given for sections, subsection and subsubsection headings.

8.2 Figures and tables

The `figure` and `table` environments are implemented as described in the LaTeX Manual to provide consecutively numbered floating inserts for illustrations and tables respectively. The standard inserts and their captions are formatted centred. Line breaks in captions can be inserted as required using `\\`.

Fig. 5. An example figure with space for artwork.

8.2.1 Illustrations (or figures)

The JFP class will cope with most positioning of your illustrations and you should not normally use the optional positional qualifiers on the `figure` environment which would override these decisions. Figure captions should be below the figure itself, therefore the `\caption` command should appear after the figure or space left for an illustration.

Figures in JFP will frequently illustrate programs, as shown in section 6.1 of this guide. Figure 5 shows an example of space left above a caption for artwork to be pasted in. This was produced with the following commands:

```
\begin{figure}
  \vspace{5cm} % the vertical depth of the artwork
  \caption{An example figure with space for artwork.}
  \label{sample-figure}
\end{figure}
```

The vertical depth should correspond roughly to the artwork you will submit; it will be adjusted to fit the final artwork exactly.

If your illustration extends over two pages, you can use the `\continuedfigure` facility. To use this, you key the figure caption for the second figure as follows:

```
\begin{figure}
  \continuedfigure
  \vspace{80pt}
  \caption{First figure, continued.}
  \label{continued}
\end{figure}
```

This ensures that the figure counter does not get incremented, and at the same time adds the word (cont.) to the caption. You may still use labels and references for this figure.

8.2.2 Tables

The JFP class file will cope with most positioning of your tables and you should not normally use the optional positional qualifiers on the `table` environment which would

override these decisions. Normal journal style sets the table caption first, followed by a double rule, the table body and a double rule at the bottom. Single rules and spanner rules (`\cline`) can be used to separate headings from the columns. For example, Table 2 is produced using the following commands:

```
\begin{table}
  \caption{Results of Overloading for 3 Experimental Setups}
  \label{sample-table}
  \begin{minipage}{\textwidth}
    \begin{tabular}{lcrrrrr}
      \hline\hline
      Program& Expt.&
      CPU\footnote{Seconds of elapsed time on an unloaded Sun 3/50.}&
      RelCPU\footnote{CPU Time relative to experiment (a).}& GC&
      Mem\footnote{Bytes of heap used over the duration of the program.}&
      RelMem\footnote{Memory usage relative to experient (a).}\\
      \hline
      8 Queens& (a)&   2.88&  1.00&    6&   1.7M&  1.00\\
      &         (b)&  32.51& 11.29&  193&  48.9M& 28.76\\
      &         (c)&   7.90&  2.74&   42&  11.3M&  6.65\\
      \noalign{\vspace {.5cm}}
      Primes&   (a)&   4.89&  1.00&   19&   5.3M&  1.00\\
      &         (b)&  47.54&  9.72&  204&  54.5M& 10.28\\
      &         (c)&  10.08&  2.06&   47&  13.0M&  2.45\\
      \noalign{\vspace {.5cm}}
      Nfib&     (a)&  21.65&  1.00&  161&  40.4M&  1.00\\
      &         (b)& 221.65& 10.24& 1382& 349.0M&  8.64\\
      &         (c)&  21.30&  0.98&  161&  42.0M&  1.03\\
      \noalign{\vspace {.5cm}}
      KWIC&     (a)&   7.07&  1.00&   15&   6.3M&  1.00\\
      &         (b)&  34.55&  4.89&  109&  47.8M&  7.59\\
      &         (c)&  31.62&  4.47&   53&  45.0M&  7.14\\
      \hline\hline
    \end{tabular}
    \vspace{-2\baselineskip}
  \end{minipage}
\end{table}
```

Notice the use of the '`\vspace{-2\baselineskip}`' command to remove the unwanted vertical space from above the table footnotes in this example.

Captions for 'continued' tables can be generated (in the same way as for figures) using the `\continuedtable` command. These should be positioned just before the `\caption` command in the appropriate table environment.

The `tabular` environment should be used to produce ruled tables; it has been modified for the JFP class in the following ways:

1. Additional vertical space is inserted above and below a horizontal rule (produced by `\hline`);
2. Tables are centred, and span the full width of the page; that is, they are similar to the tables that would be produced by `\begin{minipage}{\textwidth}`.

Table 2. *Results of Overloading for 3 Experimental Setups*

Program	Expt.	CPU[a]	RelCPU[b]	GC	Mem[c]	RelMem[d]
8 Queens	(a)	2.88	1.00	6	1.7M	1.00
	(b)	32.51	11.29	193	48.9M	28.76
	(c)	7.90	2.74	42	11.3M	6.65
Primes	(a)	4.89	1.00	19	5.3M	1.00
	(b)	47.54	9.72	204	54.5M	10.28
	(c)	10.08	2.06	47	13.0M	2.45
Nfib	(a)	21.65	1.00	161	40.4M	1.00
	(b)	221.65	10.24	1382	349.0M	8.64
	(c)	21.30	0.98	161	42.0M	1.03
KWIC	(a)	7.07	1.00	15	6.3M	1.00
	(b)	34.55	4.89	109	47.8M	7.59
	(c)	31.62	4.47	53	45.0M	7.14

[a] Seconds of elapsed time on an unloaded Sun 3/50.
[b] CPU Time relative to experiment (a).
[c] Bytes of heap used over the duration of the program.
[d] Memory usage relative to experient (a).

Because of this reformatting, vertical rules should not be used; furthermore, commands to redefine quantities such as `\arraystretch` should be omitted. If the old tabular facilities are needed, there is a new environment, `oldtabular`, which has none of the reformatting; it should be used in exactly the same way.

8.3 Appendices

You should use the standard LaTeX `\appendix` command to place any Appendices, normally, just before any references. From that point on `\section` will produce an appendix, which are numbered A, B etc., equations as (A1), (B1) etc. Figures and tables also number A 1, B 1 etc.

8.4 References

As with standard LaTeX, there are two ways of producing a list of references; either by using BibTeX with the JFP bibliography style `jfp.bst`, or by compiling a list of references by hand (using a `thebibliography` environment).

8.4.1 Using BibTeX

If you have BibTeX installed on your system, the following is a brief description on how to automatically generate a bibliography (`.bbl` file) for your article. Your article should contain at least the following elements:

```
% sample.tex
\documentclass{jfp}
\bibliographystyle{jfp}
\begin{document}
  \cite{citations}
  \bibliography{biblio database files}
\end{document}
```

Where '*biblio database files*' may be one or more filenames of bibliographic database files (without the `.bib` extension) separated by commas. First, LaTeX the file `sample.tex`. Second, run BibTeX by typing:

```
bibtex sample
```

This creates the file `sample.bbl`. Third, re-LaTeX your document, and the newly-created `sample.bbl` will be read in and typeset. You will then need to LaTeX the document once more to resolve any unresolved citation references.

8.4.2 Typesetting the references by hand

The following listing shows some references prepared in the style of the journal; this code produces the references at the end of this guide.

```
\begin{thebibliography}{}
 \bibitem[\protect\citename{Augustsson and Johnsson, }1987]{AJ187}
   Augustsson,~L. and Johnsson,~T. (1987) LML users' manual. PMG
   Report, Department of Computer Science, Chalmers University of
   Technology, Goteborg, Sweden.
 \bibitem[\protect\citename{Butcher, }1981]{Butcher}
   Butcher,~J. (1981) Copy-editing: the Cambridge handbook.
   Cambridge University Press.
 \bibitem[\protect\citename{Chicago, }1982]{Chicago}
   The Chicago manual of style. University of Chicago Press.
 \bibitem[\protect\citename{Conklin, }1987]{JC87}
   Conklin,~J. (1987) Hypertext: an introduction and survey.
   \emph{IEEE Computer}, 20~(9): pp.~17--41.
 \bibitem[\protect\citename{Dijkstra, }1976]{EWD76}
   Dijkstra,~E.~W. (1976) \emph{A Discipline of Programming}.
   Prentice-Hall.
 \bibitem[\protect\citename{Knuth, }1984]{DEK84}
   Knuth,~D.~E. (1984) Literate programming. \emph{BCS Comput. J.}
   27~(2): 97--111 (May).
 \bibitem[\protect\citename{Lamport, }1986]{LaTeX}
   Lamport,~L. (1986) \LaTeX: a document preparation system
   (2nd edition). Addison-Wesley, New York.
 \bibitem[\protect\citename{Reynolds, }1969]{JCR69}
```

```
   Reynolds,~J.~C. (1969) Transformation systems and the
   algebraic structure of atomic formulas. In B. Meltzer and
   D. Michie (editors), \emph{Machine Intelligence 5}, pp.~135--151.
   Edinburgh University Press.
 \bibitem[\protect\citename{Toyn \emph{et al.}, }1987]{TDR87}
   Toyn,~I., Dix,~A. and Runciman,~C. (1987) Performance
   polymorphism. In \emph{Functional Programming Languages and
   Computer Architecture, Lecture Notes in Computer Science, 274},
   pp.~325--346. Springer-Verlag.
\end{thebibliography}
```

The above list is typeset at the end of this guide. Each entry takes the form

```
\bibitem[\protect\citename{Author(s), }Date]{tag}
  Bibliography entry
```

where `Author(s)` should be the author names as they are cited in the text (note the space before the closing `}` of the `\citename` command is vital), `Date` is the date to be cited in the text, and `tag` is the tag that is to be used as an argument for the `\cite{}` and `\shortcite{}` commands. `Bibliography entry` should be the material that is to appear in the bibliography, suitably formatted. This rather unwieldy scheme makes up for the lack of an author-date system in LaTeX.

8.4.3 Multiple references

References should be listed alphabetically by author name(s) and then by year if the same author has several papers. If some papers by the same author(s) also fall in the same year, their dates should be in the form (1993a), (1993b), *etc.*

Formatting for italic etc. should be avoided unless you are sure you understand the style of references; please concentrate on giving full and clear information.

8.4.4 References in the text

References in the text are given by author and date. Whichever method is used to produce the bibliography, the references in the text are done in the same way. Each bibliographical entry has a key, which is assigned by the author and used to refer to that entry in the text. There is one form of citation – `\cite{key}` – to produce the author and date, and another form – `\shortcite{key}` – which produces the date only. Thus, Augustsson and Johnsson (1987) is produced by

```
Augustsson and Johnsson \shortcite{AJ187},
```

while (Toyn *et al.*, 1987) is produced by

```
\cite{TDR87}.
```

To allow further flexibility of the `\cite` (and `\shortcite`) commands the function of the optional argument has been changed. For example, the output of the command

`\cite{TDR87}` gives (Toyn *et al.*, 1987), but you may want to list all of the author names (instead of just the first author and *et al.*). In this case we would use:

```
\cite[(Toyn, Dix and Runciman, 1987)]{TDR87}
```

in the text to produce the desired effect.

A Special commands in `jfp1.cls`

The following is a summary of the new commands, optional arguments and environments which have been added to the standard LaTeX user-interface in creating the JFP class file.

New commands

`\authorbreak`	allows a list of authors to be broken in to separate lines without starting an affiliation.
`\continuedfigure`	adds the word (cont.) to the next figure caption, and also stops the figure counter from being stepped.
`\continuedtable`	as for `\continuedfigure`, except this command achieves the same effect for tables.
`\email`	used to typeset an authors e-mail address (should only be used in the `\author` command).
`\figrule`	adds a rule, used around programs set in figure environments.
`\ls`, `\ns`	used to letter-space the heading in a alternate 'Pearl' style (when used with the `\maketitle[o]` argument).
`\programmath`	gives normal spacing for verbatim math mode.
`\proofbox`	typesets a proof box □ (this is normally put in automatically at the end of the `proof` environment). If you need to insert a `\proofbox` manually, you should add a '`\quad`' of space before it in the output.
`\removebrackets`	removes the '( )' brackets from the optional argument of environments created by the `\newtheorem` command. Should be placed just before the appropriate environment.
`\unprogrammath`	reverts to math spacing.
`\shortcite`	typesets the 'year' part of the bibliographic entry only. *e.g.* (1987).
`\mathproofbox`	as `\proofbox`, except this version is intended for use in equations ending proof's (it typesets the proof box using `\rlap` with 1em of extra space).

New environments

`capsule`	used to typeset an articles Capsule Review.
`oldtabular`	preserves the orginal tabular environment, which has been modified to insert additional space above and below an `\hrule`. The body of the environment is centred with rules full out across the text measure.
`proof`	to typeset mathematical proofs, the $*$-form omits the proof box.

New optional arguments

`[<short title>]`	in the `\title` command: to define a right running headline that is different from the article title. The `\shorttitle` command also achieves the same effect.
`[<short author>]`	in the `\author` command: to define a left running headline that is different from the authors' names as typeset at the article opening. The `\shortauthor` command also achieves the same effect.
`[<pearl style>]`	Optional argument to the `\maketitle` command which allows Functional or Theoretical pearls to be typeset.
`[<widest label>]`	in `\begin{enumerate}`: to ensure the correct alignment of numbered lists.

References

Augustsson, L. and Johnsson, T. (1987) LML users' manual. PMG Report, Department of Computer Science, Chalmers University of Technology, Goteborg, Sweden.

Butcher, J. (1981) Copy-editing: the Cambridge handbook. Cambridge University Press.

The Chicago manual of style. University of Chicago Press.

Conklin, J. (1987) Hypertext: an introduction and survey. *IEEE Computer*, 20 (9): pp. 17–41.

Dijkstra, E. W. (1976) *A Discipline of Programming*. Prentice-Hall.

Knuth, D. E. (1984) Literate programming. *BCS Comput. J.* 27 (2): 97–111 (May).

Lamport, L. (1986) LaTeX: a document preparation system (2nd edition). Addison-Wesley, New York.

Reynolds, J. C. (1969) Transformation systems and the algebraic structure of atomic formulas. In B. Meltzer and D. Michie (editors), *Machine Intelligence 5*, pp. 135–151. Edinburgh University Press.

Toyn, I., Dix, A. and Runciman, C. (1987) Performance polymorphism. In *Functional Programming Languages and Computer Architecture, Lecture Notes in Computer Science, 274*, pp. 325–346. Springer-Verlag.