LPD LaTeX Tutorial

Felix Gärtner

June 3, 2003

Goals:

- Some basics of typography and TeX operation
- Logical vs. visual markup
- Beautiful graphics with `xfig` and Metapost
- Producing pdf

This material is available at
`http://lpdwww.epfl.ch/fgaertner/latex`

Book Printing vs. Ordinary Typing [Knu90, Ch. 2]

- Adjustments, when moving from typewriter to a computer terminal (e.g., difference between digit ‘1’ and lowercase letter ‘l’).
- More adjustments when moving from computer terminal to book publishing:
- Keyboard has undirected quote mark ("), typographic quote marks are directed:

```
      ``I understand.''
```

yields: “I understand.”

Book Printing vs. Ordinary Typing (cont.)

- Four forms of hyphens:
  - normal hyphen (-) for compound words like ‘daughter-in-law’.
  - en-dash (–) for number ranges like ‘pages 1–3’.
  - em-dash (—) for punctuation in sentences—sometimes these ones are simply called dashes.
  - minus sign ($-$) in math formulas.

- Try and distinguish these hyphens:
  - for a hyphen, type a single hyphen ‘`-`’
  - for an en-dash, type two hyphens ‘`--`’
  - for an em-dash, type three hyphens ‘`---`’
  - for a minus sign, type a hyphen in math mode ‘`$-$`’

Controlling TEX [Knu90, Ch. 3]

- Keyboard is too limited to be able to encode all typographic commands directly.
- Escape character let's you switch to "instruction mode" of TEX.
- In TEX, escape character is the backslash '\'
- Typesetting instruction: '\⟨typesetting command⟩'
- Example: '`\TeX`' means 'typeset the TEX logo'
- Example: '`\"`' means 'put accent over following character'
- First type is a *control sequence*, second type is a *control symbol*.

Controlling TEX (cont.)

- Control symbols: backslash plus one additional character.
  - Examples: accents ‘`\'`’ or ‘`` \` ``’
- Control sequences: backslash plus sequence of letters `A..Z` and `a..z`
  - Control sequence ends at first non-letter character. If this character is a space, it is eaten up.
    `\TeX ignores spaces after control words.`
    gives: TEXignores spaces after control words.
  - But: ‘`` the logo `\TeX' ``’
  - If you need a space, write ‘`\TeX\ is good`’
  - Non-breakable space: ‘`\TeX~is good`’

Grouping [Knu90, Ch. 5]

- Special characters ‘`{`’ and ‘`}`’ can be used for *grouping*, similar to a *scope*.

- Commands and definitions inside the group do not affect definitions outside of the group.

- Example font switching: ‘`{\large larger text} and smaller`’ instead of ‘`\large larger text \normalsize and smaller`’

- Also holds for type changing (bold, italics, etc.).

- Empty group can be used to end control sequences: ‘`\TeX{}`’

- Remark: we’re silently switching from TeX to LaTeX now; LaTeX is just a macro package using plain TeX commands (size switching commands are only in LaTeX).

Grouping (cont.)

- Grouping also used for defining the reach of control sequences.
- Example: `\textit{This is italics.}`
- If a control sequence needs an argument, it either takes the next letter, control sequence or the next group.
  - ‘`\textit{This is \textbf{bold}.}`’
  - ‘`\textit\TeX{}`’
  - But: ‘`\textit This is italics`’
  - and: ‘`\textitThis is italics`’
- Same rules: use of grouping in math mode.

How TEX reads what you type [Knu90, Ch. 7]

- This is for people who use a text editor (like emacs) for editing manuscripts.
- Rules:
  - A ⟨return⟩ is like a space.
  - Two spaces in a row count as one space.
  - A blank line denotes the end of a paragraph.
- A comment character '%' escapes the return (like a backslash in many programming tools).
- You can use spacing to structure your file (example follows).

How TEX reads what you type (cont.)

```
You can insert linebreaks at any point in a paragraph without
ending it. If you need a paragraph, insert one (or more) blank
lines.

You can use the rules to structure the input text. If you have
a displayed math formula, you can write
  %
$$x + y = z$$
  %
to visually separate it in the input file. If necessary, you
can also avoid spaces at the end of line like in th%
is example.  You can also indent text to follow grouping:
```

```
\begin{center}
  \begin{large}
    This is the major title
  \end{large}

  and this the subtitle
\end{center}

And you can use empty lines to visually separate items in
lists:
  %
\begin{itemize}

\item Empty lines before and after items are ignored

\item So it looks much better in the input file. You can
```

```
  use indentation here too.

\end{itemize}
  %
You can visually separate the following lines without inserting
a paragraph.
```

How TEX reads what you type (cont.)

- Like the backslash, there are other special characters which don't mean what they look like:
  - Beginning and ending of group: '`{`' and '`}`'
  - Toggle math mode: '`$`'
  - Alignment and parameter: '`&`' and '`#`'
  - Superscript and subscript: '`^`' and '`_`'
  - Comment character: '`%`'
- All these characters have to be escaped to be printed, e.g., '`\&`' for '&'

Logical Markup vs. Visual Markup

- Markup are the control sequences within text (HTML is another markup language).
- Visual markup directly refers to the appearance: ‘`\textit{emphasized}`’
- Logical markup refers to logical role of text, indirectly refers to appearance: ‘`\emph{emphasized}`’
- Logical markup separates contents from layout; LaTeX was written to promote logical markup.
- FCG’s most often stated rule in using LaTeX:

Always use logical markup instead of visual!

Logical Markup vs. Visual Markup (cont.)

- Example:

  Consensus is defined using two primitive operations *propose* and *decide*. If a process invokes *propose*(v) we say that it proposed v.

- Maybe written as:

```
Consensus is defined using two primitive operations
\textit{propose} and \textit{decide}. If a process
invokes $propose(v)$ we say that it proposed $v$.
```

- Gives:

  Consensus is defined using two primitive operations *propose* and *decide*. If a process invokes $propose(v)$ we say that it proposed v.

Logical Markup vs. Visual Markup

- Two objections:
  - '`$propose$`' is the product of p, r, o, … not the identifier '*propose*' (awfull spacing). Look for example at '$definitely$' vs. '*definitely*'.
  - What if you decide to change from *italics* to *slanted*?

- The primitives *propose* and *decide* should be marked up (logically) as "primitives", not as italicized words.

```
Consensus is defined using two primitive
operations \primitive{propose} and
\primitive{decide}. If a process invokes
$\primitive{propose}(v)$ we say that it
proposed $v$.
```

Defining your own Logical Markup

- Use the LaTeX facilities to define own commands:

```
\documentclass{article}
...
\newcommand{\primitive}[1]{\textit{#1}}
...
\begin{document}
...
```

- Invoking '`\primitive{x}`' is now a macro substitution. Note separation of logical and visual roles of the text.

- Small set of well-chosen logical macros sufficient.

Popular Logical Markup for LPD

```
\usepackage{latexsym}% for \Diamond
\newcommand{\eventually}{\Diamond}
\newcommand{\textcal}[1]{{\cal #1}}
\newcommand{\perfect}{\textcal{P}}
```

- Now you can write:

```
    Solving consensus is possible using $\eventually\perfect$.
```

  yields:

  Solving consensus is possible using $\Diamond\mathcal{P}$.

Creating Graphics with `xfig`

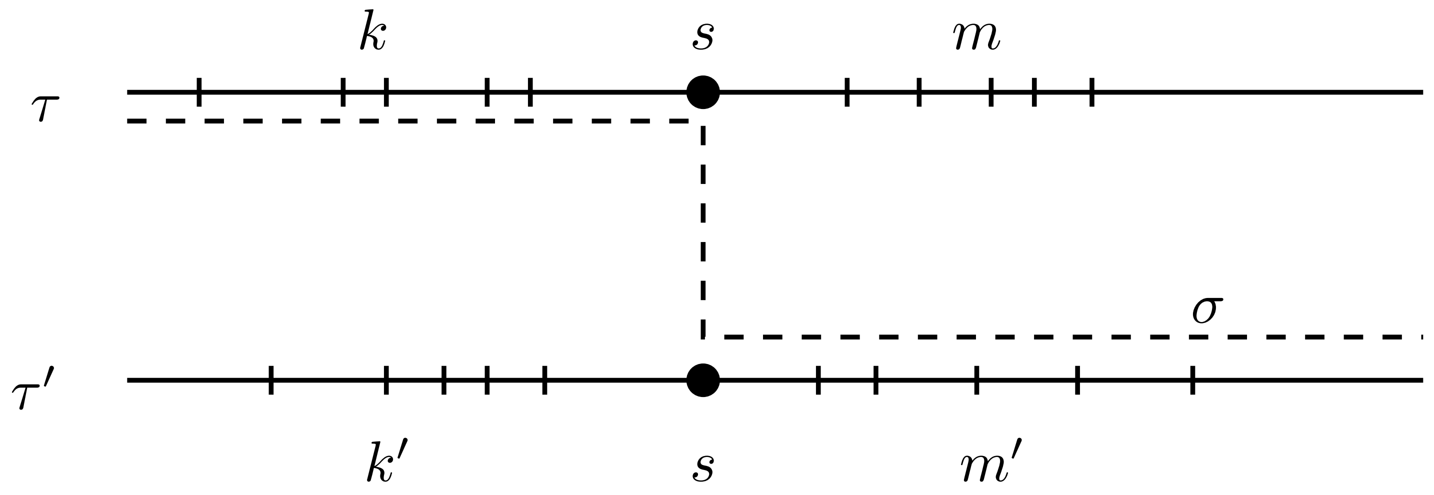
- Who has used **`xfig`**? Powerful program for creating complex figures.
- Short demonstration.
- Possible to include TEX text in figure and use LATEX fonts.
- To make this available in your LATEX file, export in Metapost format.
- File ‘**`graph.mp`**’ has to be “compiled” using Metapost **`mpost`** giving a “eps-ish” type of file ‘**`graph.0`**’.

Including Metapost Figures

- File can now be included in LaTeX document:

```
\documentclass{article}
\usepackage[dvips]{graphicx}% note the ‘cx’
...
\begin{document}
...
\begin{center}% figure can be scaled etc.
  \includegraphics[scale=0.7]{graph.0}
\end{center}
...
```

Example Figure



Going PDF

- Instead of invoking `latex` you can simply invoke `pdflatex` (it's that simple).
  - You will directly get pdf output (without having to convert Postscript to pdf).
  - Works seamlessly with Metapost if you load `graphicx` like this:

```
% to make Metapost figures useable in pdflatex
% and normal latex (include as ‘file.0’)
\ifx\pdftexversion\undefined
  \usepackage[dvips]{graphicx}
\else
  \usepackage[pdftex]{graphicx}
  \DeclareGraphicsRule{*}{mps}{*}{}
\fi
```

Going PDF (cont.)

- Does not work with eps files and `epsfig` package.
- Switch to `graphicx` package (`epsfig` is outdated anyway).
  - Using normal `latex` you can replace calls of '`\epsfig`' with calls to '`\includegraphics`'
- Leave away extension, then '`\includegraphics`' will choose the "right" file.
  - `pdflatex` can't handle eps file, but eps files can be converted to pdf using `epstopdf`.
  - If you have `graph.eps` and `graph.pdf`, then '`\includegraphics{graph}`' will automatically choose the right file depending whether you invoke `latex` or `pdflatex`.

Other Useful Things

- `cite`: handle bibliographic labels nicely (sort them, etc.)
- See the "LaTeX Companion" [GMS93] for more.
- For general rules on language, wording, abbreviations, typesetting etc. see the "Chicago Manual of Style" [Chi93]
- Indispensable AucTeX mode for emacs: `http://www.gnu.org/software/auctex/`
- See also: Knuth's booklet on "Mathematical Writing" `http://www-cs-faculty.stanford.edu/~knuth/klr.html`

References

[Chi93] *The Chicago Manual of Style.* The University of Chicago Press, forteenth edition, 1993.

[GMS93] Michael Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion.* Addison-Wesley, Reading, MA, Reading, MA, USA, 1993.

[Knu90] Donald E. Knuth. *The TeXbook.* Addison-Wesley, Reading, MA, 1990.