LaTeX study group 2

Aaron Ecay

February 24, 2010

1 Leftovers from last time

- `itemize`, which produces bulleted lists
- like this one
  - nested bullets
  - are handled automatically

1. `enumerate`, for numbered lists
   (a) which is also capable
   (b) of nested lists
2. see?

1.1 Footnotes

Footnotes are very easy to create in LaTeX. Simply use the command `\footnote` with the footnote text as the argument.[1]

1.2 Sections

You can divide your document into sections using the `\section` commands. In addition to sections, there are subsections and subsubsections. If you use these commands, you can get an automatically generated table of contents, like so:

Contents

[1] LaTeX will then handle the numbering and positioning for you.

You can also use commands to automatically insert section numbers and the pages they begin on (we will cover this today).

2 Some useful packages

In the preamble of this document are some packages that may be useful. They are:

- **tikz** – this package is a powerful drawing language for LaTeX. We will only be scratching the surface of what it can do today. (Maybe we will return to it in a future lesson.)
- **geometry** – last time I told you about the fullpage package. Turns out that this is a more flexible way of getting the same results. You can specify a uniform margin, or tweak individual sides by specifying the options *top*, *bottom*, *left*, *right*.
- **hyperref** – this one is just cool. If you have any in-document references, this package automagically turns them into clickable links, which take you to where the example was defined. This works for sections, footnotes (not so useful, as you just go to the bottom of the same page), numbered example sentences, and bibliography items (among other things). It also loads the url package, which lets you use the `\url` command to insert clickable links in your document: `http://google.com`. The default is to draw ugly bokes around the link text. By giving the option *colorlinks=true*, you get the slightly less ugly behavior of making the text itself hideous colors. (You can fix this, it just takes more fiddling – check the package documentation for all the gory details.)
- **setspace** – this package allows you to control the line spacing with the commands `\doublespacing` and `\singlespacing`, as well as the environments `singlespace` and `doublespace`. This is better than fiddling with the LaTeX-internal length settings (by far!)
- **xltxtra** – This one doesn't provide any useful commands. Instead, it "patches" some of LaTeX's internal commands to work better in the XeLaTeX engine. It also automatically loads the useful fontspec package.
- **fontspec** – this package lets you set the font of your document. It provides the command `\setmainfont` for this purpose, which takes the name of a font installed on your system as an argument. There are also `\setsansfont` and `\setmonofont` for setting the sans-serif and monospace font variants respectively. This command also takes an

optional argument with font options. One we care about is `Mapping=tex-text`, this keeps the old TeX shortcuts for certain punctuation (quotes, dashes, …) active. If you are fussy about typography, there are many more options you can pass.

- **booktabs** – This package makes tables better, and gives you a few commands for drawing lines and such in them. Examples given below.
- **graphicx** – for including pictures in LaTeX. This is a better package than the graphics package, hence the "x." There are a couple commands that go with this package. The first is `\DeclareGraphicsExtensions`, which takes as an argument a comma-separated list of file extensions your graphics files might have, dot included. This is useful in case you want to switch between two sets of graphics, say low-quality color ones for on-screen display and high-quality B&W for printing. You refer to the graphics in the document by their names only, and change the extensions you declare to switch between the two sets. The command `\graphicspath` lets you place your images in subfolders, so that they don't get mixed in with all the extra files LaTeX generates for its nefarious purposes.
- **qtree-tikz** – for drawing syntax trees. It is based on tikz (above), and is an improvement on an older package called qtree.
- **gb4e** – this is for numbered examples. We'll talk about it shortly.

3 Defining commands and environments

It is easy to define commands in LaTeX. The command to do so is `\newcommand`. This command takes, in order, the name of the new command (with backslash), optionally (i.e. in square brackets) the number of arguments to the new command (blank for no arguments), then the value of the new command. In the value portion, the hash mark (#) plus a number 1–9 is replaced with that argument. So this code will create a new command that typesets foreign words in italics: `\newcommand{\foreign}[1]{\textit{#1}}`

Ich bin ein Berliner. "I am a Berliner."

To redefine a command (either a LaTeX built in command or one you defined previously), use `\renewcommand`, with the same arguments:

Ich bin ein Berliner.

The need to specify *re*-new command is so you don't accidentally overwrite an important command defined by LaTeX.

To define an environment, the command to use is `\newenvironment`. This takes the name of the environment, (optional) number of arguments, and then two more arguments. The first of these is placed before text in the new environment, the second after. One new environment I like to define is as follows: `\newenvironment{exes}{\begin{exe}\singlespacing}{\end{exe}}` This keeps linguistic examples single-spaced regardless of whether the document they appear in is single or double spaced.

Just like `\renewcommand`, there exists `\renewenvironment`.

4 Glosses, trees, and arrows

The following code to produce linguistic examples should be self-explanatory:

(1) I am an example.

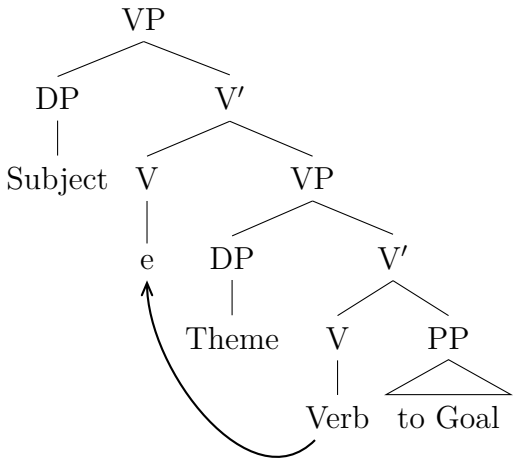
(2) a. This is example one.
b. This is example two.

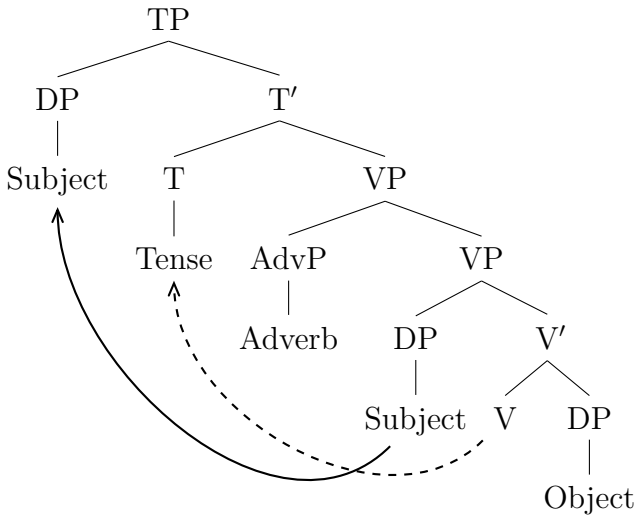
(3) * This are an ungrammatical example.

(4) **adibide hau gloseatuta ba dago**
example this glossed is
This is an example.

ɯɵɲʷ
foobar
“ ”

The example example is 1





5 Floats

5.1 Tables

LaTeX has two parts to its table-drawing code, implemented as environments. The outer one is concerned with positioning the table in the document. LaTeX has an algorithim for deciding what looks "pretty" on a page. It tries to align so-called "floating elements" to accord with these guidelines. (The two main types of floating elements we will deal with are tables and figures.) Things it considers:

- floats look best at the top or bottom of a page (so they don't break up the flow of the text)
- there shouldn't be more than a certain percentage of the page devoted to floats
- if a lot of floats pile up, such that it is impossible to satisfy the requirement immediately above, it is better to make a page of all floats than try to squeeze a little text onto a page with some floats as well

It is possible to tweak this algorithm if you don't like it, but in general trying to get a particular float into a particular place in the document is a fool's errand. If you are going to publish a book or important journal article, the publishing company has editors to do this for you.

There is one useful command for dealing with floats: `\clearpage`. This command forces a page break, and forces LaTeX to print out any floats it was saving up. This is useful at the end of a section in a longer paper, for example, to make sure that any tables in that section are printed before the next section begins.

The other half of the LaTeX table-drawing facility is for drawing the table itself. Let's see an example, and then discuss its parts.

	Yes	**No**
Own a dog	0.65	0.35
Own a cat	0.40	0.60

Table 1: A fictitious table

The outermost environment, `table`, is responsible for positioning. It takes an optional argument indicating where the float should be placed. Possible values are "Here," "Top of page," "Bottom of page," or "on a Page of all floats," indicated by the capital letters in their names. You can specify any combination of these options. Their order indicates the order of your preference. These should be interpreted as gentle hints to the placement algorithm – LaTeX will do what it wants. Inside the environment, we use the command `\centering`, for the purpose of making the table centered on the page.

The `tabular` environment is responsible for laying out the table itself. It takes an olbigatory argument specifying the columns the table is to have. These can be right, left, or center aligned, indicated by the first letter of each of these words. (There are more complex ways of doing this, but I shan't address them now.) Inside the environment, each row of the

table is specified. The ampersand character signals the end of a cell, and a double backslash the end of a row. Table rows can span multiple lines, as the first row in this table does. The built-in LaTeX command for making a line across a table is `\hline`, but it looks ugly. Far better to use `\midrule` from the booktabs package. If you want to put a table into a document without worrying about LaTeX moving it around, you can just use a `tabular` environment.

Next come two commands that make referring to this table easy. The first is the caption, which will be printed below the table with an automatically generated number. The second is a label, which makes referring to the table by number possible: Table 1. It is also possible to refer to the page the table is on: 5. (Hyperref will make both of these links.)

For a more comprehensive intro to tables in LaTeX, see http://en.wikibooks.org/wiki/LaTeX/Tables.

5.2 Figures

The `figure` environment is quite similar to the `table` one.

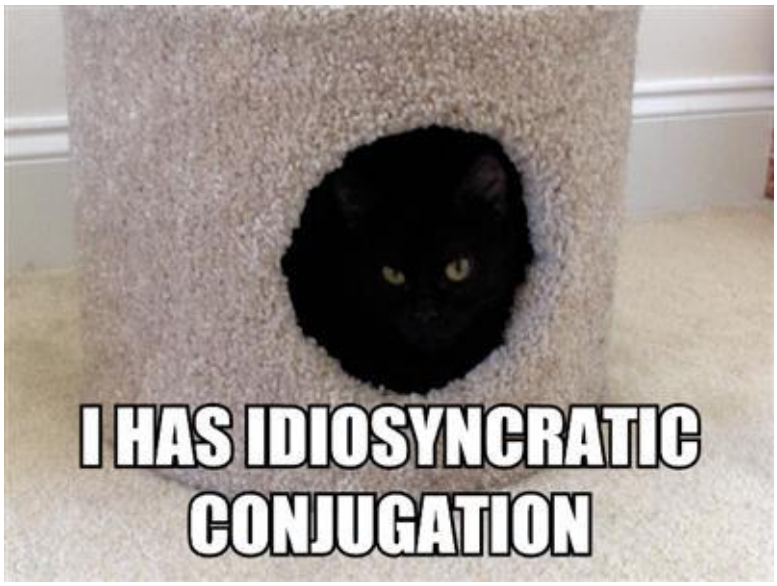


Figure 1: Morphologists beware!

5.3 Managing floats

There are two packages for managing float positioning in a document. The first is **float**. After usepackage'ing this package, put the commands `\restylefloat{table}` and `\restylefloat{figure}` in your document's preamble. This will allow you to use "H" in addition to "t", "b", "p", and "h" as a position specification for floats. This communicates a stronger desire for the float to be placed *right here* (you can tell from the fact that it is a capital letter!).

The second package is **flafter**, which forces floats to appear after the point at which they are defined in a source file.