

# Getting Started With Agda

Anton Setzer\*

March 7, 2007

- Installation of Agda in the Linux Lab:
  - When using XEmacs (preferred), add the content of `~csetzer/.xemacs/init.el` to your file `~/.xemacs/init.el`. (At time of modifying this file, xemacs is not yet installed in the linux lab, but that should happen soon – check on a shell whether the command “xemacs &” works).
    - \* If you have no such file (and xemacs is installed), you can create one by using the menu “Options → Edit Init File” from the top menu bar of XEmacs.
  - When using Emacs, add the content of the file “`~csetzer/agdainstallation/emacs/codeToBeAddedToDotEmacsFile.el`” to your your `~/.emacs` file, or create such a file in your home directory.
- After having saved this file, we test whether the installation was successful. Start XEmacs (or Emacs) again and open from it a new file called `test1.agda`.
  - Type into this file

```
postulate A :: Set
id (!a:: A)
  :: A
  = ?
```
  - Select from the Agda menu at the top of your screen the items “(Re)Start Buffer” and then “Load Buffer” and press return when the system asks for the buffer to be loaded.
  - Now the goal `{! !}` should have changed colour and be in a special mode.
  - You can type into it `a` and by choosing from the menu with the right mouse button “Give”.  
Now there should be no longer any goals left. If this worked, then your installation should have been successful.

- A few hints on using Agda:

---

\*Dept. of Computer Science, University of Wales Swansea, Singleton Park, Swansea SA2 8PP, UK. Email: [a.g.setzer@swan.ac.uk](mailto:a.g.setzer@swan.ac.uk)

- When typing in something into a goal, you should start typing between the two exclamation marks or on the right most exclamation mark.
  - \* Make sure that (X)Emacs is in “insertion mode”, which can be toggled by pressing the “Insert” key.
  - \* Make sure as well that the brackets `{!` and `!}` indicating a goal are preserved after you have typed in whatever you wanted to type in.
- When filling in a solution in a goal sometimes “give” is the right option, sometimes “refine”.
  - \* “Refine” allows partial solutions. For instance, if you type in goal which requires something of type  $B$  an element  $f$  of type  $A \rightarrow B$  and use menu “refine”, the goal becomes  $f \{! \}$ , since  $f$  applied to something of type  $A$  (and the type of the new goal is  $A$ ) has result  $B$ .
  - \* But there are cases when “refine” doesn’t work. Then one should use “give” instead.
- XEmacs and Emacs have excellent tutorials which can be started by typing in “C-h t” (Control-h t).
- Note that XEmacs and Emacs have several buffers. There is sometimes one special buffer called the “Minibuffer”, which appears on the bottom line and has the height of one line only. There one can enter for instance the name of a file to be opened, or in case of Agda an expression to be evaluated. One can switch between the buffers by using “C-x o”.
- The fastest way of using Cut and paste in (X)Emacs is to mark the beginning of the text to be cut by typing in “C-Space” (where Space is the key for space) at the beginning, then moving to the end and then using “C-w” for deleting it. Then move to wherever you want the text to be and type C-y for pasting the killed text.
- It is recommended as well to go through some of the examples given in the lectures. They are in
  - Sect. 2, starting from slide 2-6 (the page numbers might change slightly)
  - Subsect. 3 (c).
  - More material to come in later sections (e.g. 3 (e)).
- Once you have gone through a few examples, you can start solving the questions related to Agda in the coursework.
- At time of writing this, Alfa, which was a graphical user interface for Agda with some additional features, hasn’t been adapted to the new syntax of Agda yet, but you might still try it out. But for coursework only solutions following the new syntax of Agda will be accepted.
  - If you want you can try out Alfa, by starting `~csetzer/Alfa/bin/Alfa`

- \* At present, you have to enlarge one of the two windows opening, before you can see what is displayed (it was compiled for Sun work stations, and doesn't show well under Linux).
- \* There are some other problems which have to do with the fact that it is compiled for Sun work stations.