Agda Commands

[Agda-documentation team at AIST CVS]

August 15, 2005

Abstract

Contents

A List of commands

All `Agda` commands can be invoked by key operations, or by selecting items in menus. The commands which are effective in the whole of the code are found in Agda menu in the menu bar. On the other hand, the commands for goals are found in the popup menu by right-clicking on a goal. Most of items in goal menu depend on the context.

Commands are classified to four categories roughly.

Necessary commands you must know.

Important commands used very often.

Often commands which help you use `Agda` effectively. You can do without them.

A.1 Agda menu

Restart
key: `C-c C-x C-c`
category: *often*
(Re-)initializes the type-checker.

Quit
key: `C-c C-q`
category: *necessary*
Quits and cleans up after agda. If you do not want Emacs to warn in quiting Emacs, then you should invoke this command every time.

Goto error
key: `C-c ‘`
category: *important*
Jumps to the line the first error occurs.

Load
key: `C-c C-x C-b`
category: *often*
Reads and type-checks the current buffer.

Chase-load
key: `C-c C-x RET`
category: *necessary*
Reads and type-checks the current buffer and included files.

Show constraints
key: `C-c C-e`
category: *often*
Shows all constraints in the code. A constraint is an equation of two goals or of a goal and an expression.

Compute
key: `C-c C-x >`
category: *often*
Compute a closed top-level expression. Does not reduce under lambda.

Suggest

key: `C-c C-x C-s`
category: *often*

Suggests suitable expressions.

Show goals

key: `C-c C-x C-a`
category: *often*

Shows all goals in the current buffer.

Next goal

key: `C-c C-f`
category: *often*

Moves the cursor to the next goal, if any.

Previous goal

key: `C-c C-b`
category: *often*

Moves the cursor to the previous goal, if any.

Undo

key: `C-c C-u`
category: *important*

Cancels the last `Agda` command or typing.

Text state

key: `C-c '`
category: *necessary*

Resets agda to the state that the current buffer is loaded.

Check termination

key: `C-c C-x C-t`
category: *often*

Runs termination check on the current buffer. You will need to retype-check the buffer.

Submitting bug report

key:
category:

(not implemented)

A.2 Goal commands.

When a goal is replaced with a new expression by the commands below, we know that it is type-correct.

Give

key: `C-c C-g`
category: *often*

Substitute a given expression in the goal.

Intro

key: `C-c TAB`
category: *often*

Introduces the canonical expression of the type the goal i.e. an abstraction, a record, or a constructor if only one possible exists

Refine

key: `C-c C-r`
category: *important*

Given an expression, e, this command will apply the minimum number of meta-variables needed for the expression e ? ? ..? to be type-checkable

Refine(exact)

key: `C-c C-s`
category: *often*

Given a expression with arity n it applies n meta-variables to the given expression.

Refine(projection)

key: `C-c C-p`
category: *often*

Refines the goal with a expression in a given package. For example, a function `cat` is defined in the package `OpList`. When a goal is filled with "`OpList cat`", the Refine (projection) command accepts it and refine the goal but refine command fails. (In case a goal is filled with "`OpList.cat`", refine command works.)

Case

key: `C-c C-c`
category: *often*

Makes a template of a case expression with a given formal parameter.

Let

key: `C-c C-l`
category: *often*

Makes a template of a let expression with given formal parameters.

Abstraction

key: `C-c C-a`
category: *often*

Makes a template of a function expression with given formal parameters.

Goal type

key: `C-c C-t`
category: *often*

Shows the type of the goal.

Goal type(unfolded)

key: `C-c C-x C-r`
category: *often*

Shows the reduced type of the goal.

Context

key: `C-c |`
category: *important*

Shows context (names already defined) of the goal.

Infer type

key: `C-c :`

category: *important*

Prompts an expression and infers the type of it, under the current context.

Infer type(unfolded)

key: `C-c C-x :`

category: *often*

Prompts an expression and infers the reduced type of it, under the current context.

References

[1] Programming Logic Team at Chalmers and AIST. Agda, 2000. http://www.coverproject.org/AgdaPage/.

[2] Lena Magnusson and Bengt Nordstrm. The alf proof editor and its proof engine. In *TYPES '93: Proceedings of the international workshop on Types for proofs and programs*, pages 213–237. Springer-Verlag New York, Inc., 1994.

[3] Nordström, B., Petersson, K. and Smith, J.M., *Programming in Martin-Löf's Type Theory*, available at
http://www.cs.chalmers.se/Cs/Research/Logic/book/.

[4] Nordström, B., Petersson, K. and Smith, J.M., *Martin-Löf's Type Theory*, pp. 1 - 37 in Handbook of Logic in Computer Science, vol. 5 (2000), Oxford Science Publication.