Proposal for a LyX Algorithm Style

Benoît Guillon

January 10, 2001

Contents

List of Algorithms

1 Introduction

This file describes the use of a layout that allows to write algorithms without using any LaTeX command in a lyx document. The layout is built in order to fulfill the WYSIWYM principles, and uses the algorithm and algorithmic packages.

2 Document class

The `algorithm.inc` file contains the Algorithm style definition. To be available in a document class, it must be included in the related layout. The `article-algo.layout` is an example that supports the Algorithm style. The `algorithm.inc` file needs the commands provided by the latex package `algolyx.sty`, that must be installed such that LaTeX can find it. When this layout is installed (in the `~/.lyx/layout` directory), the algolyx.sty package installed, and lyx is reconfigured, the `Article (algo)` class should be available. This is the current class of this document.

3 Version

The package version detailed in this document is 0.3.

4 Using the `Algorithm` style

4.1 Common rules

To use the lyx algorithm style, simply select the `Algorithm` style, and respect the following rules:

- The style is derived from the Description style, and thus the first word of each item has a special meaning. The first word you write (bold typed) must be one of the predefined algorithmic keywords (if, else, etc.).
- If the first word is not a predefined keyword, then it is assimilated to the algorithmic environment \STATE keyword. In this document, the “*“ character is used to define a state item.
- For the keywords that need an extra parameter (such as: if, else if, until, while), the parameter is considered to be the words following the bold keyword, until a comment starts (detected by a starting comment delimiter), or until the end of the line.

- If the item levels are used (only for screen viewing purpose), they must be consistent, i.e. at the end of the algorithm, the depth level must come to zero.

4.2 Customising the screen aspect

The screen viewing of the algorithms you write can be customised as follow:

- Use the item levels to indent the algorithm blocks. Besides the screen aspect, the item levels are not used (see section 4.4).
- Customise the algorithm keywords with your own language keywords (see section 6).
- Customise the comments delimiters used (see section 4.5).

4.3 Customising the output

The output result can be customised by:

- Using the `Algorithm (num)` style, to print the algorithm line numbers (see section 4.7).
- Setting the global option noend, to omit the end statements in the output (see section 5).
- Customising the output algorithm keywords (see section 6).

4.4 Indenting the algorithm

The screen algorithm indenting is useful to see directly on the lyx document the algorithm block levels, without exporting it to any output format. However, this feature is optional, and has no influence on the output aspect.

Algorithm 1 is an example that does not use the indent capabilities and algorithm 2 uses the indent feature.

4.5 Using comments

Comments can be defined when enclosed in the appopriate delimiters, placed after the bold keyword and its extra parameter. Such as for the original algorithmic commands, comments are supported for the following keywords: if, elseif, else, while, for, forall, repeat and loop. In addition the layout allows to put comments after endif, endfor, endwhile, untill and endloop.

A comment can be on several lines, but in any case the closing comment delimiter must be the last word of an item. For example, the following line causes the layout to fail:

if here is the extra parameter {this is the comment} unexpected words here!!

Algorithm 1 Example without (screen) indentation

```
if something is true then
do action 1
do action 2
else if (something else is true) then
perform the specific action
for all (items in a group) do
do a special action
end for
repeat
run action 3
until (everything is done)
else
loop
we are sticked here
end loop
end if
```

Algorithm 2 Example with (screen) indentation

Require: a pre-condition must be met
Ensure: the algorithm output consistency

```
if (something is true) then
do action 1
do action 2
else if (something else is true) then
perform the specific action
repeat {comment here}
run action 3
until (everything is done)
else
do action 4
if (another thing is true) then
for (i = 0; i < 10; i++) do
process the iteration
end for
while (it stills true) do
run action 5
end while
end if
end if
```

The following is expected:

if here is the extra parameter {the comment ends the line}

By default, the comments delimiters are "{" and "}". These are the delimiters used in this document. The delimiters can be redefined by using the following latex command in the preamble:

```
\keycomment{<begin>}{<end>}
```

The delimiters can be either a single character or several characters, but cannot be defined through macros. For instance, the following does not work:

```
\def\begincom{\#<}
\def\endcom{>\#}
\keycomment{\begincom}{\endcom}
```

The following works:

```
\keycomment{\#<}{>\#}
```

Algorithm 3 is an example using comments.

4.6 Safety

Because the lyx users are not supposed to know accurately the latex algorithmic command syntax, the layout is built in order to be as safe as possible, by forgetting the unexpected words to avoid latex errors.

The unexpected words are suppressed as follow:

- A bold keyword not recognised is considered to be a state item definition, and does not appear to the output.
- For keywords without an extra parameter that support comments (else) the characters between the keyword and the valid comment are suppressed.

Here is an example containing unexpected characters. Check the difference between what you see on the screen and what is produced to the output.

if it is true **then** {but is it?}
  do the right action {a comment here}
else {here is the real comment}
  do the other action
end if {ending comment!}

4.7 The `Algorithm (num)` style

This style is derived from `Algorithm`, and has no behavioural difference. This style allows to produce algorithms with line numbers. Algorithm 4 is an example using this style.

Algorithm 3 Example with comments

```
repeat {first comment}
  if something true then {second comment}
    action 1 {third comment}
  else if something else is true then {fourth comment}
    action 2 {fifth comment}
    for (i = 0; i < max; i++) do {iterations to do}
      action 3
    end for {max is reached now}
  else {sixth comment}
    last possible action
    loop {seventh comment}
      let's stay here {another comment}
      while a condition is met do {does it need a comment?}
        action 3 {this comment is very long in order to show the very-long-comment support. Whatever long the comment is, no word should be written after the closing comment delimiter.}
      end while {now the condition is not met anymore}
      for all items in a group do {too many comments}
        do something for the item
      end for
    end loop {end of loop}
  end if {end of the testing block}
until it is still true {the big loop condition}
```

Algorithm 4 Algorithm with line numbers

```
1: first thing to do
2: second thing to do
3: if it is true then {we hope so}
4:    action 1
5: else {other possibility}
6:    action 2
7: end if
```

5 Options setting

The algorithm package options are not supported by the layout, because it is an internal LyX stuff (algorithm floats). The single algorithmic package option (noend) is supported by the layout. It can be set by writing the following in the latex preamble:

```
\algoption{noend}
```

6 Customisation

The lyx algorithm style can be used in another language by:

- redefining by hand in the latex preamble all the commands used by the algorithm style, when the language translation is not directly supported or doesn't match your need,
- specifying the language to support, if the language translation is available.

6.1 Full macro redefinition

6.1.1 Screen keywords

It deals with the words written in the lyx document, and that appears on the screen. In the document, the tokens that define \IF, \ELSE, etc. used in the algorithms are "if", "else", etc. These are redefinable macros. Their default definitions are:

```
\newcommand{\keyif}{if}
\newcommand{\keyelseif}{elseif}
\newcommand{\keyelse}{else}
\newcommand{\keyendif}{endif}
\newcommand{\keyfor}{for}
\newcommand{\keywhile}{while}
\newcommand{\keyrepeat}{repeat}
\newcommand{\keyuntil}{until}
\newcommand{\keyendfor}{endfor}
\newcommand{\keyendwhile}{endwhile}
\newcommand{\keyloop}{loop}
\newcommand{\keyendloop}{endloop}
\newcommand{\keyrequire}{Require:}
\newcommand{\keyensure}{Ensure:}
```

6.1.2 Output keywords

To customise the algorithm words to the output, redefine the algorithmic environment macros. The default definitions of these macros are:

```
\newcommand{\algorithmicrequire}{\textbf{Require:}}
\newcommand{\algorithmicensure}{\textbf{Ensure:}}
\newcommand{\algorithmiccomment}[1]{\{#1\}}
\newcommand{\algorithmicend}{\textbf{end}}
\newcommand{\algorithmicif}{\textbf{if}}
\newcommand{\algorithmicthen}{\textbf{then}}
\newcommand{\algorithmicelse}{\textbf{else}}
\newcommand{\algorithmicelsif}{\algorithmicelse\ \algorithmicif}
\newcommand{\algorithmicendif}{\algorithmicend\ \algorithmicif}
\newcommand{\algorithmicfor}{\textbf{for}}
\newcommand{\algorithmicforall}{\textbf{for all}}
\newcommand{\algorithmicdo}{\textbf{do}}
\newcommand{\algorithmicendfor}{\algorithmicend\ \algorithmicfor}
\newcommand{\algorithmicwhile}{\textbf{while}}
\newcommand{\algorithmicendwhile}{\algorithmicend\ \algorithmicwhile}
\newcommand{\algorithmicloop}{\textbf{loop}}
\newcommand{\algorithmicendloop}{\algorithmicend\ \algorithmicloop}
\newcommand{\algorithmicrepeat}{\textbf{repeat}}
\newcommand{\algorithmicuntil}{\textbf{until}}
```

6.2 Language support

Some languages are directly supported by the algorithm style. To specify the language to use enter the following command in the LaTeX preample:

```
\algolang{<language>}
```

At the moment the available languages are:

- french

7 Using the algorithm style in other document layouts

To make the algorithm textclass available in another document class layout than article-algo, you just need to do as follow:

1. Copy the layout to enrich in your local configuration layout directory. Example that customises the report layout:

   ```
   > cp /usr/local/share/lyx/layouts/report.layout \
        $HOME/.lyx/layouts/report-algo.layout
   ```

2. Edit the copied layout, and:

   (a) Change the the first line:

   ```
   # \DeclareLaTeXClass[report]{report (algo)}
   ```

(b) Add the following lines:

```
# Input lyx algorithm definitions
Input algorithm.inc
```

3. Save the new layout, and reconfigure lyx to make the new layout available.

8 Changes

Differences between the current release and the previous one (0.2):

- Comments after endif, endfor, endwhile, endloop, untill are now supported.
- The package definition is splitted in two files: `algolyx.sty` provides all the latex commands, and `algorithm.inc` provides the new LyX styles.
- Long items (i.e. on several lines) are now supported. It includes state items, and items using an extra parameter.
- No words must be written after a closing comment. This new constraint seems to be acceptable, and is a condition to have long items available.
- The comments delimiters can be redefined (by using `\keycomment` in the preamble).
- The global algorithmic option noend is supported (by using `\algoption` in the preamble).
- Line numbering in algorithms is available by using the Algorithm (num) style.

Differences between version 0.2 and release 0.1:

- Some languages translations are supported.
- The typewriter font previously used to display algorithms is removed: it is not useful, and it makes algorithms too big on screen.

Differences between version 0.1 and the beta version:

- The commands are more robust: unexpected words are simply lost (not shown to the output).
- The extra parameters (for: if, for, while, etc.) do not need to be enclosed in parenthesis anymore.
- The require, ensure, loop, endloop forall keywords are now supported.
- Comments are supported.

9 Limitations – Bugs

This LyX algorithm style has (at least) the following limitations:

- The line numbering interval is hard coded to 1, for the `Algorithm (num)` style. It means that every line number is printed. There is no user option to change this.
- Using a list environment (itemize, description, etc.) in an algorithm does not work.
- The algorithm options are not supported.