A User Guide to the RAISE Tools as used in the DTU 02263 course

Anne Haxthausen, Morten Lindegaard and Torben Gjaldbæk
IMM, DTU

January 4, 2006

Contents

1 Introduction

This document contains a user guide to how RSL specifications can be type checked and printed with LaTeX using the RAISE tools developed at UNU/IIST. First, in section 2 it is described how to install the tools. Then, in sections 3–5 it is described how to use the tools in three steps:

- create a text file containing the considered specification
- type check the specification
- print the specification with LaTeX

For further details, read the full user and installation guide from UNU/IIST available at: http://www.iist.unu.edu/newrh/III/3/1/page.html.

2 How to install the tools

2.1 Windows

All files needed for installation can be found on the course cd on which there is also an installation guide.

Alternatively, use the files from http://www.imm.dtu.dk/courses/02263/ under the link "RAISE tools installation files" and use the UNU/IIST installation guide.

2.2 How to install `rsltc` for Linux

All files needed for installation can be found on the course home page http://www.imm.dtu.dk/courses/02263/ under the link "RAISE tools installation files" or http://www.iist.unu.edu/newrh/III/3/1/page.html.

If you are using Linux machines in the E-databar, you can skip the steps of this section as they have already been done for you.

The `rsltc` binary file can be found under the link "How to get the RAISE tools" at the course home page http://www.imm.dtu.dk/courses/02263/.

Put the `rsltc` binary file in some directory, *dir* (that could for instance be `/home/`*nn*`/bin`, where `/home/`*nn* is the name of your home directory), on your path.

If *dir* is not already on your path, you can ensure this by modifying the `PATH` environment variable. If you are using a bash shell this can be done by adding the following line to your `.bashrc` file:

```
export PATH=${PATH}:dir
```

2.3 How to set up emacs for Linux

To set up emacs for supporting for type checking and LaTeX formatting RSL specifications for Linux (in the E-databar or on own PC)

1. create a directory `~/.elisp`
2. put the files `rsltc.el`, `rsl-mode.el`, `rslconvert.el` and `tokenise.el` in this directory, and
3. add the following to your `.emacs` file

```
;;; Facilities for RSL support

;; Loading files for RSL support:
;; put rsltc.el, rsl-mode.el, rslconvert.el and tokenise.el in ~/.elisp
;; put that directory on the emacs load path:
(add-to-list 'load-path "~/.elisp")
;; load the files (rslconvert loads tokenise)
(load "rsltc.el")
(load "rsl-mode.el")
(load "rslconvert.el")

;; Function for converting RSL formulae in ascii syntax to latex
;; The formulae are placed between \RSLatex and \endRSLatex

(defun rsl2latex ()
  "Do rslatex on buffer"
  (interactive)
  (do-latex))

;; Function for undoing the above conversion

(defun latex2rsl ()
  "Undo rslatex on buffer"
  (interactive)
  (undo-latex))

;; Setting up function keys
;; Pressing f2 invokes 'do-latex' on the whole buffer
;; Pressing f3 invokes 'undo-latex' on the whole buffer

(global-set-key [f2] 'rsl2latex)
(global-set-key [f3] 'latex2rsl)
```

The tool *rsltc* can now be run from *emacs*, i.e. you can syntax and type check the current emacs buffer. Furthermore, you can include RSL specifications in LaTeX documents.

3 How to create RSL specifications in text files

RSL specifications are written in ordinary text files. Use `emacs` to edit a specification of an RSL module with name *X* in a file with name *X.rsl*. You must use ascii equivalents for the non-ascii RSL symbols. A table of ascii equivalents can be found on page 385 of the RSL book.

Here is an example of a specification to be stored in a file with the name `COUNTER.rsl`:

```
scheme COUNTER =
  class
    variable counter : Nat := 0
    value increase : Unit -> write counter Nat
    axiom increase() is counter := counter + 1 ; counter
  end
```

If a module depends on other modules, their names (without ".rsl") should appear as a comma separated list at the start of the file. This list is the module's context. The tool calculates the transitive closure of contexts, so if *A* depends on *B* and *B* depends on *C* then you must include `C` as the context for *B*, and you may for the context of *A* use either `B` or `B, C`. The files `A.rsl`, `B.rsl`, and `C.rsl` must all be in the same directory. When a module is checked, the context modules are checked first, so you only need to run the tool on top level modules.

4 How to type check RSL specifications using `rsltc`

To syntax and type check a specification *X*, stored in a file *X.rsl*, open the file in `emacs` and choose `RSL` → `Type check.` [1]

If there are errors, `rsltc` outputs messages in the form

```
X.rsl:m:n: text
```

which indicates that there is an error described by *text* in the file *X.rsl* at line *m* and column *n*.

As an example, consider the following erroneous specification:

```
scheme COUNTER =
  class
    variable counter : Nat := 0
    value increase : Unit -> write counter Nat
    axiom increase() is counter := counter + true ; counter
  end
```

Type checking gives the following output:

[1] Alternatively, issue the command `rsltc X` from the command prompt.

```
Checking COUNTER ...
COUNTER.rsl:5:46: Argument types Nat and Bool
incompatible with '+' type Int >< Int -> Int or
Real >< Real -> Real or
Int -> Int or
Real -> Real
Finished COUNTER
rsltc completed: 1 error(s) 0 warning(s)
```

After changing the specification to the correct version (section 3), type checking gives the following output:

```
Checking COUNTER ...
Finished COUNTER
rsltc completed: 0 error(s) 0 warning(s)
```

5 How to print RSL specifications with LaTeX

This is done in two steps:

1. create a LaTeX document as described in section 5.1
2. make a postscript file from the LaTeX document as described in section 5.2

5.1 Creating a LaTeX document

LaTeX files are plain text files having a name of the form *Y.tex*, e.g. `main.tex`. A basic LaTeX file for use with RSL specifications looks like this:

```
\documentclass{article}
\usepackage{rslenv}
\begin{document}
%here you can write text and
%include RSL specifications
%   in one of two ways as described in the following subsections
\end{document}
```

boxedminipage.sty is needed by rslenv.sty (but may already be in your LaTeX distribution). You should either put the LaTeX `rslenv.sty` file (and if necessary boxedminipage.sty) in the same directory as your LaTeX file, or you can get LaTeX to find your local `.sty` files.

rslenv.sty (and if necessary boxedminipage.sty) should be placed somewhere with your other LaTeX style files. Under Linux the latter can be achieved by doing the following:

1. In `$HOME` do: `mkdir -p texmf/tex/latex/misc/`
2. Now put `.sty` files in `$HOME/texmf/tex/latex/misc`

3. Add the following line to `.bashrc`:

```
export TEXINPUTS=$TEXINPUTS:$HOME/texmf//
```

4. Close the XTerm window and start a new one
5. Run 'texhash texmf'

You must do step 5 every time you add a `.sty` file to `misc`.

5.1.1 RSL specifications in separate text files

If you have an RSL specification in a separate text file (say, `X.rsl`) that you want to print neatly, you create a LaTeX file (say, `main.tex`) which includes the specification by using the command `\RAISEIN{X}`

A basic LaTeX file including the specification in `X.rsl` looks like this:

```
\documentclass{article}
\usepackage{rslenv}
\begin{document}
\RAISEIN{X}
\end{document}
```

To print the specification, do the following:

1. Open `main.tex` in `emacs`.
2. From `emacs`, issue the command `M-x mkdoc` (that is, press Alt-x, write 'mkdoc' and press Enter). `mkdoc` generates a file `X.tex` from `X.rsl`. If there had been several `RAISEIN` commands, then `mkdoc` would have generated a tex file for each included rsl file.

The file can now be converted to PostScript format as described in section 5.2.

5.1.2 RSL specifications within LaTeX files

RSL specifications can also be written directly within a LaTeX document. This is convenient if you e.g. want to include fractions of a specification within the ordinary text of an article or a report.

RSL formulae within a LaTeX file should be written between `\RSLatex` and `\endRSLatex`, e.g.:

```
\documentclass{article}
\usepackage{rslenv}
\begin{document}
\RSLatex
  scheme COUNTER =
    class
```

```
      variable counter : Nat := 0
      value increase : Unit -> write counter Nat
      axiom increase() is counter := counter + 1 ; counter
    end
\endRSLatex
\end{document}
```

By pressing key F2 from within emacs, the RSL formulae in ascii-format are converted into LaTeX-format:

```
\documentclass{article}
\usepackage{rslenv}
\begin{document}
%\RSLatex
%  scheme COUNTER =
%    class
%      variable counter : Nat := 0
%      value increase : Unit -> write counter Nat
%      axiom increase() is counter := counter + 1 ; counter
%    end
%\endRSLatex
\bp
\>\kw{scheme} COUNTER {\EQ}\\
\>\>\kw{class}\\
\>\>\>\kw{variable} counter : \kw{Nat} :{\EQ} 0\\
\>\>\>\kw{value} increase : \kw{Unit} {\RIGHTARROW} \kw{write} counter \kw{Nat}\\
\>\>\>\kw{axiom} increase() {\IS} counter :{\EQ} counter {\PLUS} 1 ; counter\\
\>\>\kw{end}
\ep
\end{document}
```

The file can now be converted to PostScript format as described in section 5.2.

By pressing the key F3, the LaTeX-formatted formulae are converted back to the original ascii-format.

Notice that you may need to change some of the spacing in your ascii formulae to make it appear nicely in the final PostScript file — indentation has to be done by inserting blank spaces.

5.2 Creating a PostScript file from a LaTeX document

5.2.1 Under Windows

To make a PostScript file from a LaTeX file called `main.tex`, do the following:

1. Open the file `main.tex` in `emacs`.
2. Choose `Command` → `LaTeX`. [2]

[2] Alternatively: From the command prompt, run `latex main`

3. Choose `Command` → `dvips`. [3]

The result is a file called `main.ps`, which may be viewed with *GSview*.

5.2.2 In the E-databar

To make a PostScript file from a LaTeX file called `main.tex`, issue the following commands at the command prompt:

```
latex main
dvips main.dvi -o main.ps
```

The result is a file called `main.ps`, which may be viewed with */usr/bin/gv*.

[3] Alternatively: Choose Command → File and press Enter or from the command prompt, run `dvips main.dvi -o main.ps`