LaTeX authoring environment

Jan Lahoda

May 27, 2004

1 Introduction

The LaTeX authoring environment is an Integrated Development Environment (IDE) for LaTeX, written in Java[1]. Powerful features supporting editing LaTeX code are provided.

1.1 What It Is and What It Is Not

This editor supports editing of plain LaTeX code. It has quite advanced features.

It is NOT a WYSIWYG (or WYSIWYM — What You See Is What You Meant) editor.

1.2 Requirements

- min. 128MB RAM (256MB recommended)
- approx. 80MB free disk space
- JDK1.4.1 or later

1.3 Features

syntax highlighting for LaTeX text LaTeX commands, arguments and many more

integrated spell checker and corrector words not in the dictionary are underlined

interactive code completion is provided for a wide range of text elements:

- (subset of) LaTeX commands and arguments of these
- natural language words
- `\ref` argument
- `\cite` argument (parses BiBTeX file)

[1] In fact, the authoring tool is only an add-on module for an existing Java IDE NetBeans.

- `\documentclass` and `\usepackage` arguments

help on LaTeX commands provided instantly in the editor

reverse search

LaTeX model and parser with error recovery, providing fast error info

automata GUI editor editor for Vaucanson

code folding

LaTeX structure

mathematical symbols list

1.4 Installation

2 Project System

The LaTeX authoring environment uses a simple project system. This project system is based on so called *main file* (sometimes called also the master file). This is the file you run LaTeX on. Each main file includes a set (possibly empty) of included files. Any number of files (main and included) can be opened.

If you open a file for the first time, the IDE will try to find out whether it is the main file or not. If it looks like a main file (it has the LaTeX header), it is marked as the main file and opened. If the file does not look like a main file, user is asked to provide appropriate main file.

2.1 Compiling

2.2 Showing the result

2.3 Setting Paths to Executables

In the LaTeX Build Settings dialog, paths to executables of LaTeX, BibTex, DVIPS and DVIPDF can be set. Also default argument for these programs can be set. The main file (the file to be compiled) should not be set in this dialog, as it is added automatically during build process.

3 Code completion

Code completion can be invoked using `Ctrl-Space` shortcut. The content of the code completion vary depending on the actual context. In each context a prefix is defined, and only possibilities beginning with the appropriate prefix are shown.

Basically, there are following situations when the code completion offers results:

- plain text
- command
- command argument

The first two situations offer the list words and the list of commands, respectively.

The last situation is much more complicated. If the command corresponding to the argument is one of special commands (`\cite`, `\ref`, `\usepackage`, , `\documentclass`), special results (as described below) are shown. If there is a list of allowed values of this arguments, this list is offered. Otherwise, no code completion is provided.

3.1 Citations

The code completion for the `\cite` command provides all publications in all requested BiBTeX databases. The BiBTeX database is added to the search, if it is mentioned in an `\bibliography` command.

The code completion for `\cite` currently does not recognise `thebibligoraphy` environment.

3.2 References

The code completion for the `\ref` command are all found labels altogether with the guessed captions.

3.3 Document Classes and Packages

The code completion for the first (non-mandatory) argument of the `\usepackage` and `\documentclass` command is the list of all known options for given package or document class, respectively.

The code completion for the second (mandatory) argument of the `\usepackage` and `\documentclass` command is the list of all known packages and document classes.

It is recommended to fill out the second mandatory argument of these commands at the beginning and after fill the first non-mandatory argument.

4 Spell Checker

The spell checker is integrated in the LaTeXeditor. Each word in the document is checked against the dictionary and underline if it is not valid. Interactive spell-correction is also available.

4.1 Spell Checker

Spell Checker checks spelling of words against the dictionary. There are three possible results of the check:

- the word is correct. In this case, it is not underlined.
- the word is a prefix of another word in the dictionary. In this case, it is underlined by the gray waveline.
- the word is not correct (it does not fall into previous categories). In this case, it is underlined by the red waveline.

4.2 Spell Correction

The editor also allows to correct spelling errors interactively. Simply point the mouse over an incorrectly spelled word, press right mouse button and select "Spelling" in the pop-up menu. A list of words similar to the incorrect one will appear. When a word is chosen from this list, it immediately replaces the original incorrect word.

4.3 Selecting dictionary

For each document, a dictionary is chosen according to locale of the document. By default, each document has set *default* locale, which maps to the system default locale. The locale of a document can be changed using *LaTeX→Change Language* menu item. Under this menu, the locales of all installed dictionaries, the *Default* locale and the *Other* are shown. If a dictionary is found for a given locale, it is used for spell checking (see Sections 4.1 and 4.2). If none dictionary is found, the spell checking is turned off the document.

4.4 Adding dictionaries

5 Structural View

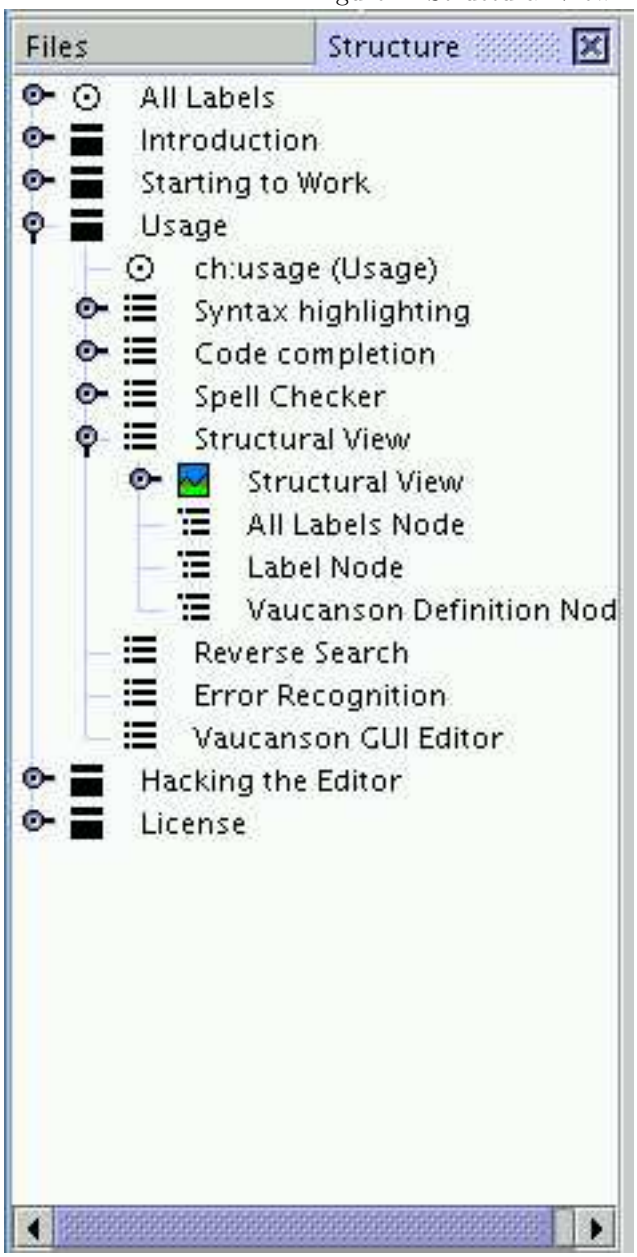
The Structural View shows an overview on the structure of the document. The structure is presented as a tree (see Figure 1). All nodes has the Go To Source action, which jumps to the corresponding point in the source.

There are some special types of nodes in the tree, described below.

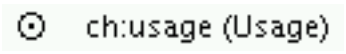
5.1 All Labels Node

Under this node, all labels found in the document are listed.

Figure 1: Structural View



5.2 Label Node



Each Label node is corresponding to one `\label` statement in the document. It shows the label, guessed caption corresponding to the label. The children of the label node are references (`\ref`) to this label.

5.3 Vaucanson Definition Node

Node corresponding to a code defining the automaton using the Vaucanson plug-in. This node allows opening of the GUI editor (see Section 6).

6 Automata Visual Editor

The Automata Visual Editor allows interactive editing of finite automata diagrams. The native storage format for this editor are LaTeX commands defined by the Vaucanson-G package.

Warning: This piece is of alpha quality and may cause problems.

6.1 Opening of the Automata Visual Editor

The Automata Visual Editor can be opened for existing figure using the Structural View. Locate the node corresponding to the automaton in the Structural View, show pop-up menu (by right-click) and chose `Open`. If the `Open` item is disabled, the Vaucanson code cannot be parsed correctly (it either uses a non-supported feature or is invalid) and therefore cannot be edited visally.

6.2 Saving Resulting Automaton

It is not necessary to save the resulting automaton. The content of the visual editor is dumped into the LaTeX code each time the visual editor looses focus. While the visual editor is opened, the corresponding code in the editor is marked as guarded, meaning that it cannot be changed.