QCheck:

BoB question checker application

v. 001

1 Main Program Information

1.1 Program identification

1.1.1 **QCheck** (Question checker) GUI application.

1.1.2 **Version 1.0**

1.2 Brief description of an application

1.2.1 **QCheck** is a tool with a GUI interface that aims to check matching of regular expression against a user question using one of language-dependant abbreviation file (English or German or Italian). At the end **QCheck** shows the user the “expanded” regular expression tree with all the abbreviations substituted.

1.2.2 **QCheck** provides a user a GUI interface for abbreviation file specification, regular expression and the related question strings input. First, when the user specifies abbreviation file, application determines its language, preloads it and checks its validity with the help of `BobHelper.java` class from chatterbot project.

When regular expression and the related question both are specified by the user and button “check” is clicked, **QCheck** loads three “Bob ANTLR classes” from chatterbot project(`Bob_Parser, Bob_Lexer` and `Bob_TreeParser)` and does regular expression validation (`Bob_Parser.java`), builds and displays AST tree (regular expression expaned tree) and finally performs question matching against the tree (`Bob_TreeParser.java`)

“`Bob ANTLR` classes” belongs to chatterbot project and are generated automaticaly from `ANTLR` parser grammar (ANother Tool for Language Recognition).

2 Program functions

QCheck allows the user to perform following actions:

- Abbreviation file loading dialog

- Checking regular expression input format between **macro-file format** (which supports only a basic Perl regular expression syntax) and **topic-tree format** (an extended regular expression, i.e. with Boolean connectivity operands ("**&&**", "**!**"))

3 Program Layout

3.1 Among the types that are used in **QCheck**, the most significant one are the "`Bob ANTLR` classes" (`MyAST.java`, `BobHelper.java`, `Bob_Parser.java`, `Bob_Lexer.java` and `Bob_TreeParser.java`). They are provided by **chatterbot** project and implement the main logic.

Classes:

`MyAST.java` contains a modification abstract syntax tree, which is the output format for ANTLR parse. Modification adds to the basic set of parameters also line number information.

`Bob_Lexer.java` class is automatically generated by **ANTLR**. It scans the extended regular expression string for the relevant tokens.

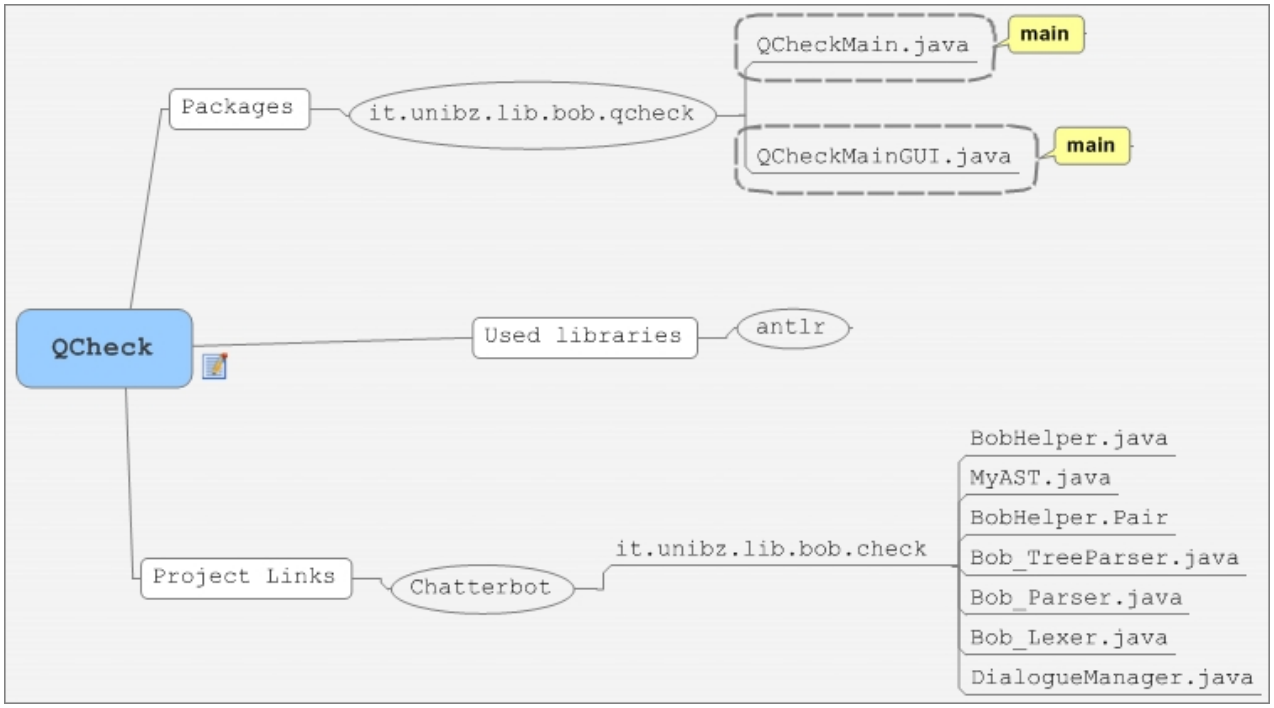
`Bob_Parser.java` is automatically generated from `Bob.g` by ANTLR. It builds a parsing tree out of tokens, extracted by Lexer from regular expression string and thus validates Regular Expressions(`bExpression()` procedure that in case throws recognition exceptions).

`Bob_TreeParser.java` is also generated from `Bob.g`. It evaluates whether some user question are matched by the tree that was built by the parser (`bExpression(tree, stringToMatch)` Boolean function). Algorithm works as following: it applies the required Boolean logic and executes calls to `regex.match()` for the atomic (non-extended) regular expression patterns with the user question.

`BobHelper.java` defines the abbreviation file checking routine (by Boolean function `checkAllRegexesInMacroMap`).

`ASTFrame` ANTLR library class is used for regular expression tree (ASTTree) visualization (tree comes from Bob_Parser `getAST` method).

3.2 **Program structure (see attached bob_qcheck_v002.jpeg)**



3.2.1 Packages and classes

`it.unibz.lib.bob.qcheck` contains following classes:

- `QCheckMainGUI.java` – contains GUI description, action listeners for interaction with user and main checking logic with calls to external classes.
- `QCheckMain.java` – test main method. Can be easily deleted.

3.2.2 External JAR libraries:

`antlr.jar`: extended Boolean regular expressions parser, regular expression tree (AST – Abstract Syntax Tree) classes, AST tree visualization frame.

3.2.3 Project dependences:

`Chatterbot`: Matching regular expressions against question phrase, regular expression validation.

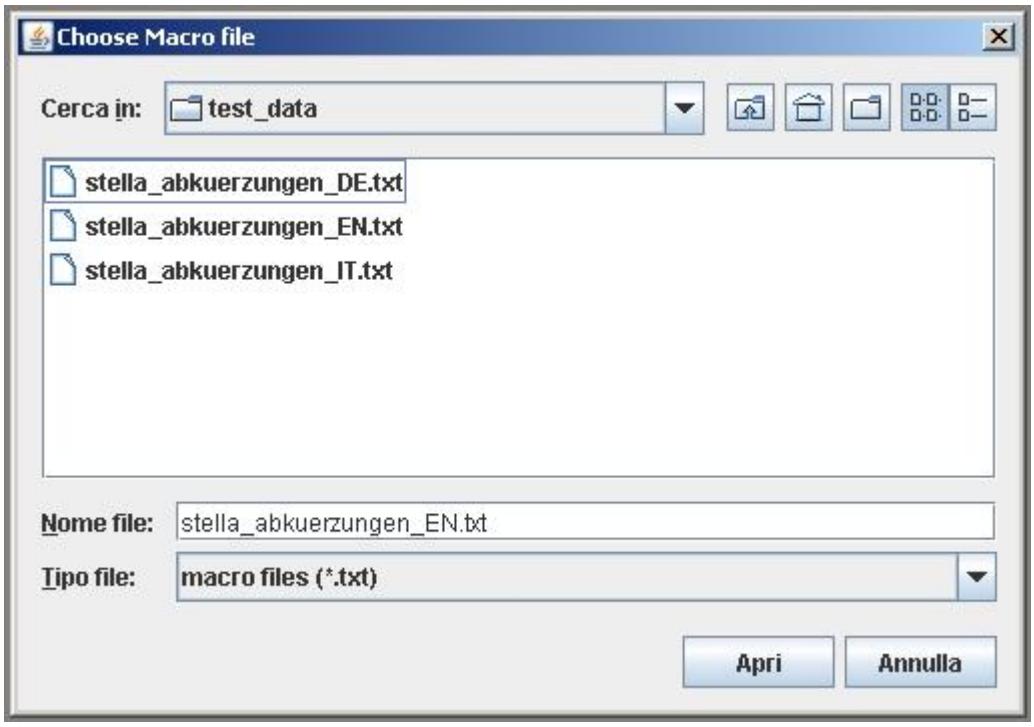
3.3 Compiling and linking

Linking: export as a single runnable jar file.

4 Program Flow

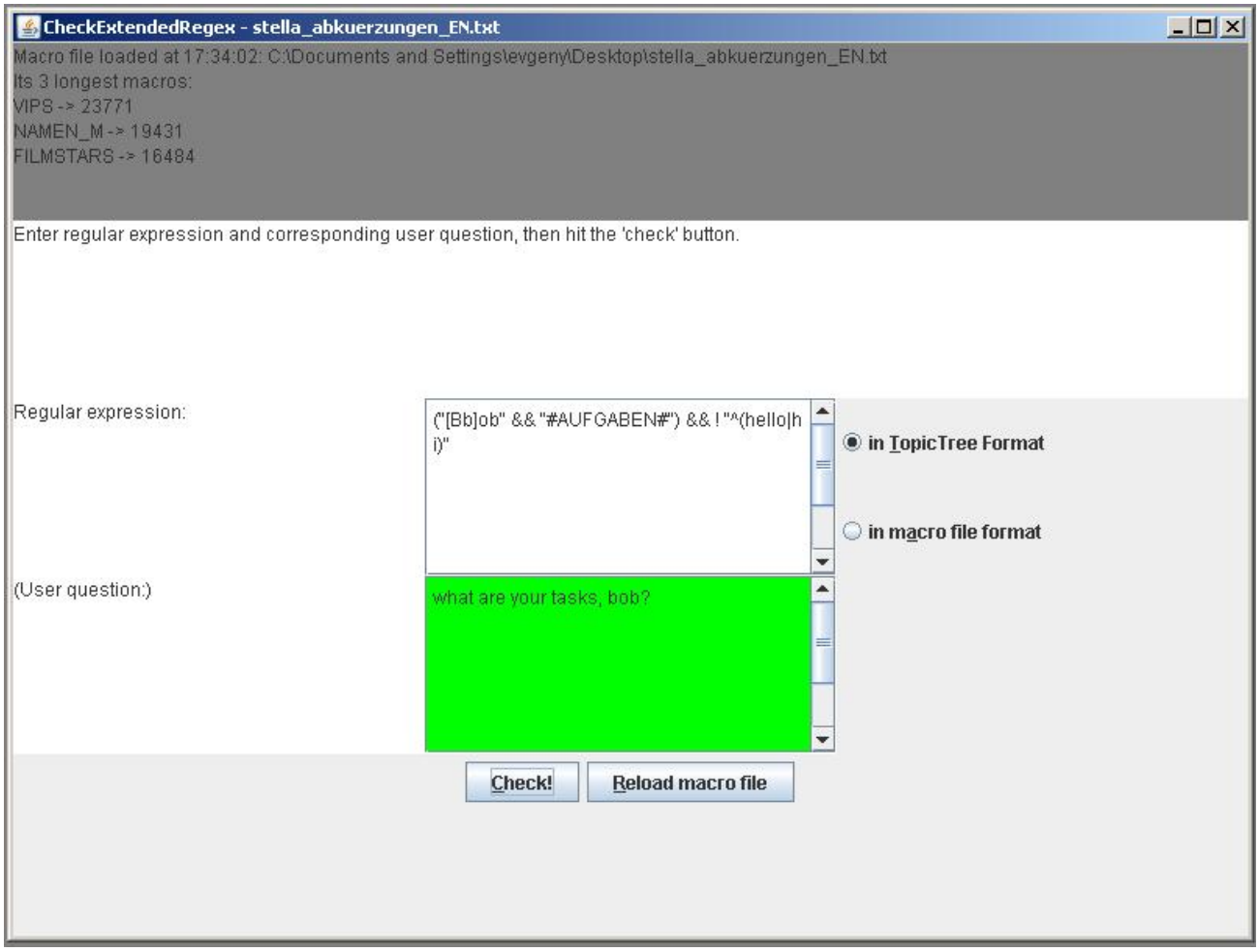
4.1 Dialogs description

"Select abbrefiation file" dialog: user can select abbreviation file in one of 3 languages(German, English and Italian)



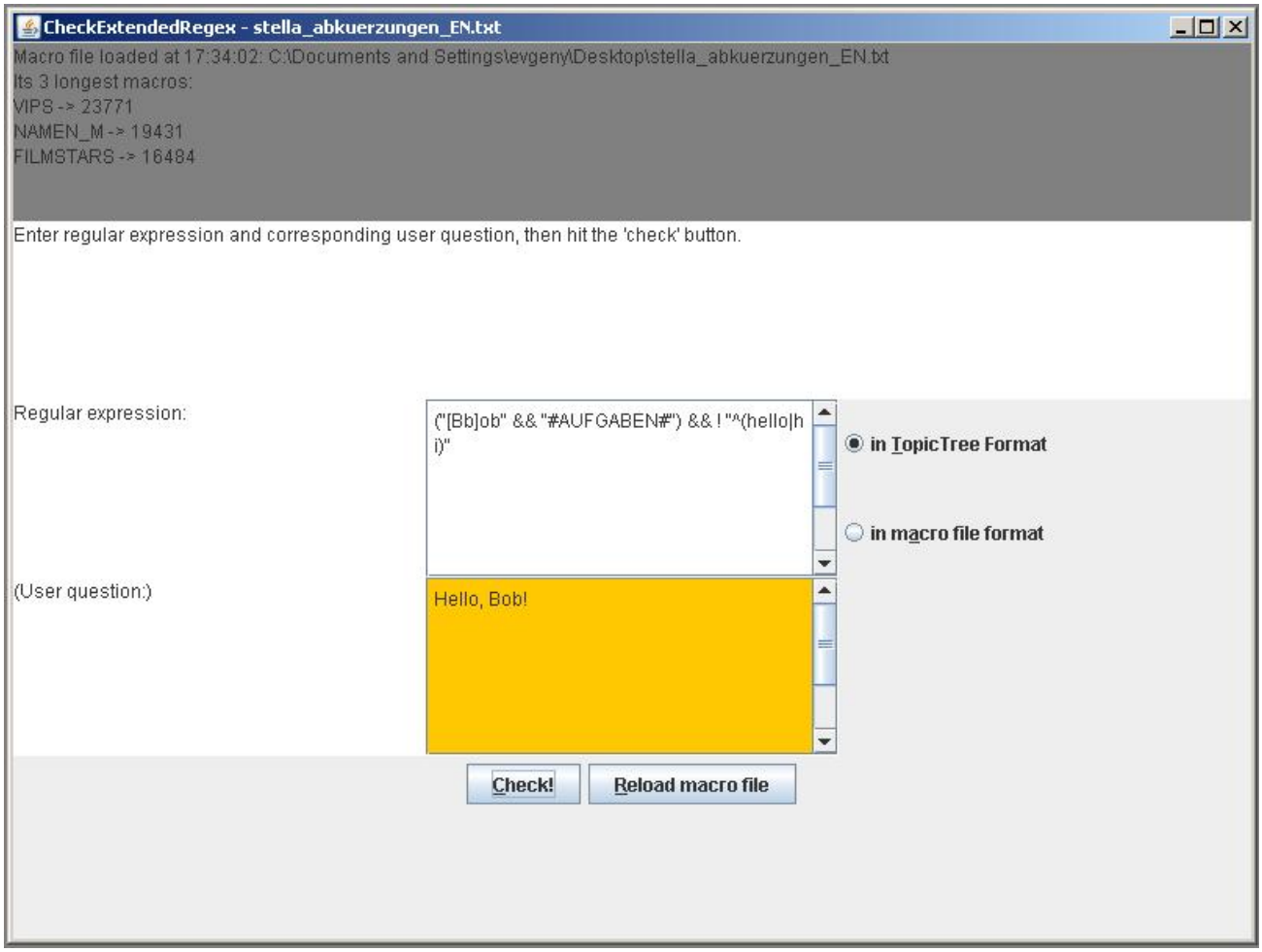
Select abbreviation file

"Main application window": provides fields for regular expressions and user question. Contains area at the top for macro-file (abbreviations file) information and a frame under it for system messages (errors) output.



RE matches the question

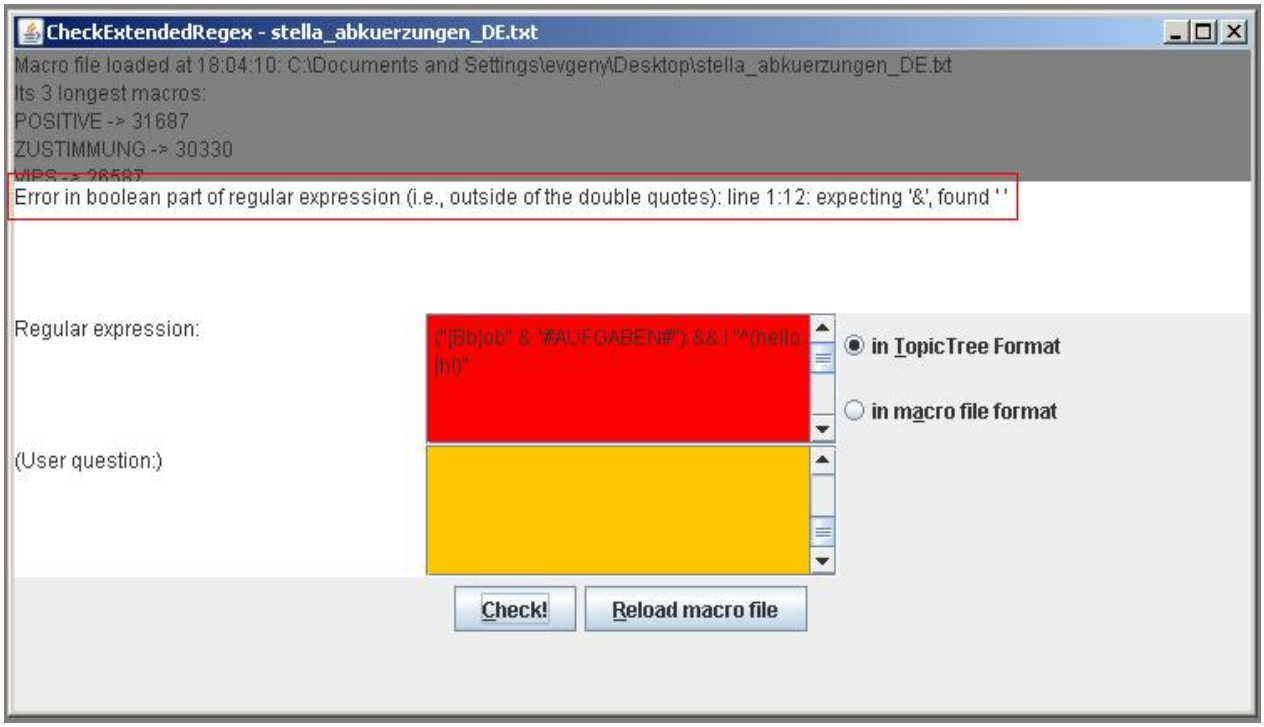
User question field can be 3 colors: white (no checking performed jet), green (matching succeeded) and orange (matching failed).



RE does not match the Question

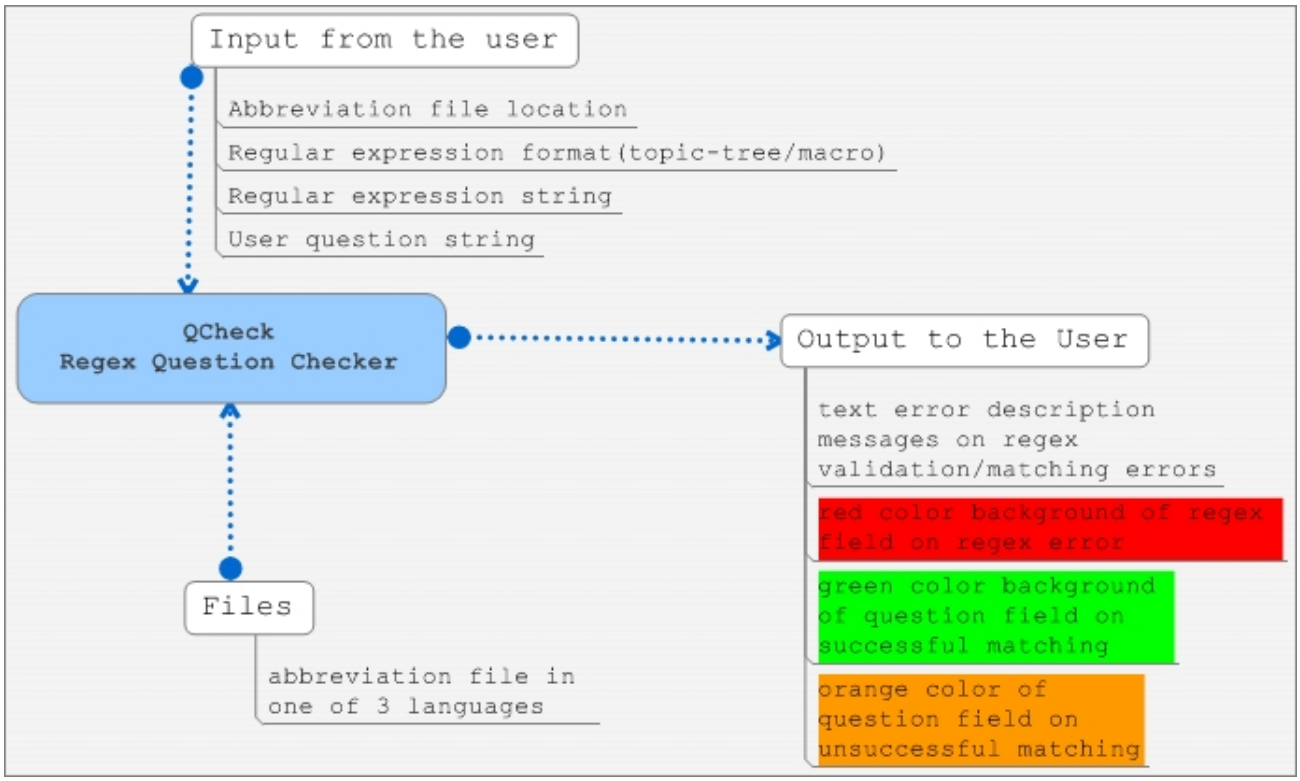
Regular expression field also might change the color depending on if regular expression validates or not (in this case it becomes red and in the message area (red rectangle) appears text with error description).

If otherwise, the RE you entered is correct and the system is able to parse it you will see an additional frame with an extended tree structure appearing (see the picture in the chapter 6.1).



Error in RE syntax

5 Data Flow



6 Data organization

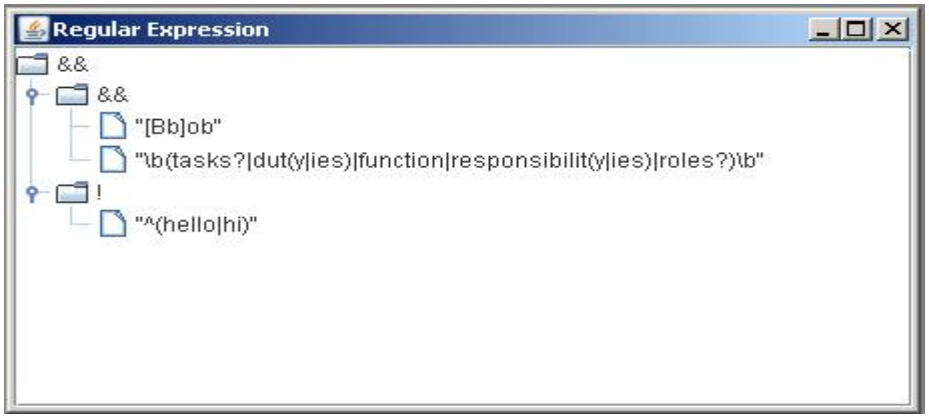
6.1 Extended regular expressions structure

Extended regular expression syntax is described in `Bob.g` file of chatterbot project (package it.unibz.lib.bob.check). It differs from the "ordinal" Perl regular expressions by introducing Boolean operators:

`OR : "||"; AND: "&&"; NOT: "!";` And their parenthesis () grouping.

```
("[Bb]ob" && "#AUFGABEN#") && ! "^(hello|hi)"
```

In this case will be matched all phrases that consist of 2 parts: first part should contain the words "Bob" or "bob" and the second one should match the "`AUFGABEN`" abbreviation (which looks like following "`(tasks?|dut(y|ies)|function|responsibilit(y|ies)|roles?)`"). And one more condition: the matched phrase should not start from words "hello" or "hi". Order of parts is irrelevant, so both of "tasks bob?" and "bob tasks?" will be matched. The whole parsing tree structure of an extended regular expression could look like this:



Regular expression tree

6.2 Abbreviation files structure

It's a text file with a list of regular expression's abbreviations (one per row). Structure of abbreviation is following: `ABBREVIATION_NAME = "\b"+ +REGULAR_EXPRESSION+"\b"`, where "\b" a word boundary symbol used in RegEx. Example (for English version):

```
WARUM = \b(why|what for|wherefore|for (what|which) reasons?)\b
```

In this case expressions that are matching this abbreviation will be:

“why”, “what fore”, “wherefore”, “for what reasons?”, “for which reasons?”

7 Testing

Make sure that you are using the standard English abbreviation file (actually, it has only to contain the following line)

```
AUFGABEN = \b(tasks?|dut(y|ies)|function|responsibilit(y|ies)|roles?)\b
```

and topic-tree input format selected. Use this extended regular expression for test:

```
("[Bb]ob" && "#AUFGABEN#") && ! "^(hello|hi)"
```

1) Positive test: What are your tasks, Bob?
2) Negative test: Hello bob, what are you your tasks?

8 Software Licence

In order to provide maximum flexibility as to how the code can be used by customers we distribute the project to the research community under the LGPL (GNU Lesser General Public License) v3 open source licence (http://www.gnu.org/licenses/lgpl-3.0.html). Bob’s source code is available for downloading on google code site where other researchers can contribute their changes and improvements: http://code.google.com/p/chatterbot-bob/

Note that we distinguish chatterbot source code, which is published under the LGPL licence via the given link from Bolzano-specific hand-crafted parts of Bob (such as: topic-tree xml file, abbreviation files, BoB’s face images). These parts of the project are the property of the “Library of the Free University of Bozen-Bolzano” and must not be passed on to third parties without it’s consent.

Why do we use the LGPL:

The Chatterbot project is licensed to everyone under the LGPL. The official name of the LGPL is “GNU Lesser General Public License,” reflecting its heritage as a derivation of the “GNU General Public License,” the GPL. The LGPL is different from the GPL in important ways. Since we don’t license the Chatterbot project under the GPL, we won’t bother explaining those differences or describing the GPL. Only the LGPL license applies to the Chatterbot project software. We use the LGPL for the Chatterbot

project because it promotes software freedom without affecting the proprietary software that sits alongside and on top of the Chatterbot project.

The LGPL is an open source project licence:

1. The LGPL allows you to do the following things with the Chatterbot project software: use the Chatterbot project software as a component of your business
applications in any way you wish, including linking to the Chatterbot project from your own or other proprietary software.
2. Make unlimited copies of the Chatterbot project software without payment of royalties or license fees.
3. Distribute copies of the Chatterbot project software, although it will be much easier to refer anyone who wants copies directly to our website, which we will publish on http://code.google.com/p/chatterbot-bob/
4. Make changes to the Chatterbot project if you need to do so for use within your own company. The LGPL license does not require you to share those internal changes with the rest of the community.
5. Distribute changed versions of the Chatterbot project to others, but if you distribute such changed versions you are required to share those changes with the rest of the community by publishing that changed source code under the LGPL.(see http://www.jboss.com/pdf/Why_We_Use_the_LGPL.pdf for further information)