jInfer TreeRuleDisplayer Module Description

Michal Klempa, Mário Mikula, Robert Smetana, Michal Švirec, Matej Vitásek
Advisors: RNDr. Irena Mlýnková, Ph.D., Martin Nečaský, Ph.D.

Praha, 2011

Target audience: developers willing to extend jInfer, looking for ways to visualize grammars.

Responsible developer:	Michal Švirec
Required tokens:	none
Provided tokens:	cz.cuni.mff.ksi.jinfer.base.interfaces.RuleDisplayer
Module dependencies:	Base JUNG
Public packages:	none

1 Introduction

This rule displayer uses JUNG graph implementation to draw rules as a forest of trees, where each tree represents one rule from the whole set. In each tree, root vertex represents left side of the rule, inner vertices represents concatenation, alternation or permutation and leafs are elements, simple data or attribute. As in basic rule displayer, tree rule displayer creates a component with multiple tabs: each one for a new grammar to display.

2 Structure

The main class implementing `RuleDisplayer` inference interface and simultaneously registered as its service provider is `TreeRuleDisplayer`. Main method of this class is `createDisplayer()`, which looks up the component, adds a new panel to it and renders the specified grammar in it.

The diagram of classes involved in rule display is in figure 1. Main window of rule displayer may have multiple tabs in it: each contains its own `RuleDisplayer` responsible for rendering specified grammar using *`JUNG`* library. Each tab in rule displayer window is represented by instance of `RulePanel` which contains button linking to options and the main panel (canvas) with rendered rules. Canvas is created by `VisualizationViewer` instance with `TreeLayout`

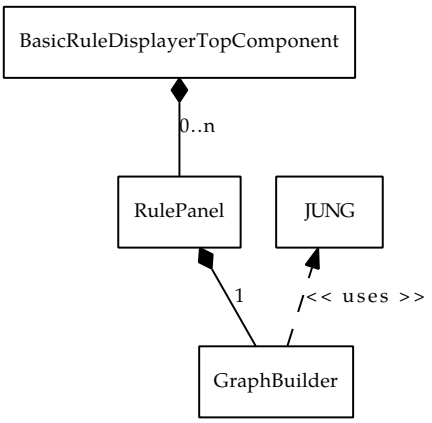

Figure 1: TreeRuleDisplayer class diagram

layout (which are both classes from JUNG library). For `VisualizationViewer`, `GraphMouse` is set with transforming mode to allows zooming and moving of graph in canvas. To allow the user to set custom appearance of rules, we set various `Transformers` in `RenderContext` of `VisualizationViewer`. These transormers change color, shape and vertex size in each rule tree according to properties set in options.

All the trees representing rules are created in the `GraphBuilder` class using the `buildGraphPanel()` method. This method creates `DelegateForrest` of `DelegateTrees` where each tree of the forrest represents one rule. Each vertex of the tree is represented by `Regexp<? extends AbstractNamedNode>` and edge by `RegexpInterval`. While the code building tree structure is a nice recursion programming exercise, there is nothing of a special interest in it. Finally the method appends forrest into layout and VisualizationViewer which were described above. All code responsible for building rule trees is contained in the `cz.cuni.mff.ksi.jinfer.treeruledisplayer.logic`. For more details about JUNG library please see [jun].

All graphics used by *`TreeRuleDisplayer`* is contained in the `cz.cuni.mff.ksi.jinfer.treeruledisplayer.graphics` package.

2.1 Settings

All settings provided by *`TreeRuleDisplayer`* are NetBeans-wide. The options panel along with all the logic is in the `cz.cuni.mff.ksi.jinfer.treeruledisplayer.options` package. Available options include setting the color of the background, horizontal and vertical distance between vertices. Also for each type of vertex is possible to set the size, the shape and the color.

References

[Aho96] H. Ahonen. *Generating grammars for structured documents using grammatical inference methods*. PhD thesis, Department of Computer Science, University of Helsinki, Series of Publications A, Report A-1996-4, 1996.

[Bou] Ronald Bourret. Dtd parser, version 2.0. `http://www.rpbourret.com/dtdparser/index.htm`.

[HMU01] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation (2nd Edition)*. Addison-Wesley, 2001.

[HW07] Yo-Sub Han and Derick Wood. Obtaining shorter regular expressions from finite-state automata. *Theor. Comput. Sci.*, 370(1-3):110–120, 2007.

[jun] Java universal network/graph framework. `http://jung.sourceforge.net/`.

[KMS+a] Michal Klempa, Mário Mikula, Robert Smetana, Michal Švirec, and Matej Vitásek. *jInfer Architecture*.

[KMS+b] Michal Klempa, Mário Mikula, Robert Smetana, Michal Švirec, and Matej Vitásek. *jInfer AutoEditor automaton visualization and editor module*.

[KMS+c] Michal Klempa, Mário Mikula, Robert Smetana, Michal Švirec, and Matej Vitásek. *jInfer Base Module Description*.

[KMS+d] Michal Klempa, Mário Mikula, Robert Smetana, Michal Švirec, and Matej Vitásek. *jInfer BasicDTDExporter Module Description*.

[KMS+e] Michal Klempa, Mário Mikula, Robert Smetana, Michal Švirec, and Matej Vitásek. *jInfer BasicIGG Module Description*.

[KMS+f] Michal Klempa, Mário Mikula, Robert Smetana, Michal Švirec, and Matej Vitásek. *jInfer BasicRuleDisplayer Module Description*.

[KMS+g] Michal Klempa, Mário Mikula, Robert Smetana, Michal Švirec, and Matej Vitásek. jinfer javadoc. `http://jinfer.sourceforge.net/javadoc`.

[KMS+h] Michal Klempa, Mário Mikula, Robert Smetana, Michal Švirec, and Matej Vitásek. *jInfer TwoStep simplifier design and implementation*.

[log] Apache log4j™. `http://logging.apache.org/log4j/`.

[loo] org.openide.util.class lookup. `http://bits.netbeans.org/dev/javadoc/org-openide-modules/org/openide/modules/doc-files/api.html`.

[mod] Module system api. `http://bits.netbeans.org/dev/javadoc/org-openide-modules/org/openide/modules/doc-files/api.html`.

[Nor] Theodore Norvell. A short introduction to regular expressions and context free grammars. `http://www.engr.mun.ca/~theo/Courses/fm/pub/context-free.pdf`.

[VMP08] Ondřej Vošta, Irena Mlýnková, and Jaroslav Pokorný. Even an ant can create an xsd. In *DASFAA'08: Proceedings of the 13th international conference on Database systems for advanced applications*, pages 35–50, Berlin, Heidelberg, 2008. Springer-Verlag.

[wik] Regular expression. `http://en.wikipedia.org/wiki/Regular_expression`.