

jInfer AutoEditor 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, specifically alter displaying of automata .

Responsible developer:	Mário Mikula
Required tokens:	org.openide.windows.WindowManager
Provided tokens:	none
Module dependencies:	Base JUNG
Public packages:	cz.cuni.mff.ksi.jinfer.autoeditor cz.cuni.mff.ksi.jinfer.autoeditor.automatonvisualizer cz.cuni.mff.ksi.jinfer.autoeditor.automatonvisualizer.layouts cz.cuni.mff.ksi.jinfer.autoeditor.gui.component

1 Introduction

This is an implementation of a *AutoEditor*. Using JUNG library, it provides an API to display and user interactively modify automata, so the process of inference can be easily made user interactive.

2 Structure

Structure of *AutoEditor* can be divided into following four main parts.

- API - API to display automaton in GUI.
- Base classes - Classes providing basic functionality that can be extended and combined to achieve desired visualization of an automaton.
- Derived classes - Classes derived from the base classes that are used in existing modules and simultaneously serve as examples.
- Layout creation - System of creating Layouts.

First, Layouts and use of base classes to create a visualization of automaton will be described.

2.1 Layout

Layout is a JUNG interface responsible primarily of representation of automaton and positions of its states. JUNG library provides several implementation of Layout interface. However, none of them is convenient for automatic automaton displaying, *AutoEditor* provides two additional implementations. Layout by *Julie Vyhnanovska*, used in her master thesis and Layout which is using external *Graphviz* software. TODO odkazy v predchadzajucej vete.

Class providing creation of Layout instances is named *LayoutHelperFactory*.

2.1.1 Vyhnanovska Layout

As mentioned above, this Layout was implemented by *Julie Vyhnanovska* as a part of her master thesis. TODO link? It positions automaton states to a square grid. This Layout gives good results for relatively small automata (about 10 states of less) but for larger ones, the results are quite disarranged and confused.

Source codes resides in package `cz.cuni.mff.ksi.jinfer.autoeditor.automatonvisualizer.layouts.vyhnanovska`.

2.1.2 Graphviz Layout

Graphviz Layout uses *Graphviz*, third-party graph visualization software, to create positions of automaton states. [TODO link?](#) To use this Layout, *Graphviz* has to be installed and path to *dot* binary has to be set in options. This Layout gives nice results even on large automata.

Source codes resides in package `cz.cuni.mff.ksi.jinfer.autoeditor.automatonvisualizer.layouts.graphviz`.

2.1.3 LayoutHelperFactory

2.2 Base classes

Main two classes representing visualization of automaton are *Visualizer* and *AbstractComponent*. *Visualizer* is a graphical representation of automaton and *AbstractComponent* is a panel (extends *JPanel*) containing the *Visualizer* which will be displayed in GUI. *TODO* obrazok ako AC dedi od *JPanelu* a obsahuje *Visualizer*.

2.2.1 Visualizer

Visualizer class extends JUNG *VisualizationViewer* class, so it provides all its methods and adds support for saving contained automaton to an image file. Responsible methods are `saveImage()` and `getSupportedImageFormatNames()`. However, to save an image of automaton it is not necessary to call this methods directly. *AutoEditor* GUI contains button to save a save an image of displayed automaton. For information on how to to this, see *TODO* ref.

Constructor has one argument, instance of *Layout* class. *Layout* class is described in *TODO* ref.

TODO obrazok ako *Visualizer* dedi od *VisualizationVieweru* a obsahuje *Layout*.

2.2.2 PluggableVisualizer

Tried *PluggableVisualizer* je rozšírením triedy *Visualizer*, ktorá poskytuje navyše jednoduchý spôsob ako zapájať mouse pluginy.

Mouse plugin je trieda JUNGu, ktoré rozširuje možnosti *Visualizera* o prácu s mysou (vyberanie stavov, zoom, ...). Spomenutý už existujúce pluginy a ref.

TODO obrazok ako *PluggableVisualizer* dedi od *Visualizeru*

Metody:

`addGraphMousePlugin()` `setVertexLabelTransformer()` `replaceVertexLabelTransformer()` `setEdgeLabelTransformer()` `replaceEdgeLabelTransformer()`

By default obsahuje 2 pluginy, jeden pre zoom a jeden pre posuvanie canvasu. V prípade potreby je možné ich odstrániť pomocou metód *VisualizationVieweru*. *TODO* dopísať.

2.2.3 AbstractComponent

TODO translate

Trieda *AbstractComponent* je panel v ktorom bude vykreslený automat, presnejšie *Visualizer* reprezentujúci nejaký automat. Dedi od triedy *JPanel*, takže poskytuje všetky jej metódy a správanie. Navyše poskytuje metódy `setVisualizer()` `getVisualizer()` `waitForGuiDone()` `guiDone()` `guiInterrupt()` `guiInterrupted()` a abstraktnú metódu `getAutomatonDrawPanel()`

Purpose tejto triedy je rozšíriť ju a poskladať si panel aký sa hodi (tlacítka, nápisy, ...) s tým, že musí obsahovať aspoň jeden *JPanel*, v ktorom bude vykreslený nastavený *Visualizer*. Účel metódy `getAutomatonDrawPanel()` je vrátiť tento *JPanel*, aby *AutoEditor* vedel, kam má ten *Visualizer* vykresliť.

Ak je žiadaný user interaktívnosť, je nutné si podporu pre ňu zahrnúť práve do tejto triedy. Pre viac informácií viz *TODO* ref.

Visualizer sa nenastavuje v konštruktorovi z toho dôvodu, že často je žiaduce, aby sa na rovnakom paneli kreslilo postupne viac rôznych automatov. Na to nie je nutné vyrábať novú instanciu, ale stačí na jednej instancii volať `setVisualizer()`.

2.3 API

API AE je veľmi jednoduché. Trieda *AutoEditor* poskytuje tieto 3 statické metódy.

`drawComponentAsync()` `drawComponentAndWaitForGUI()` `closeTab()`

2.4 Derived classes

Popis tried pouzitych v inych moduloch, ktore sluzia zaroven ako priklad.

StatePickingVisualizer StatesPickingVisualizer

2.5 Layout factory

TODO

2.6 GUI

TODO

tlacitka

2.7 Preferences

TODO

All settings provided by *BasicXSDExporter* are project-wide, the preferences panel is in `cz.cuni.mff.ksi.jinfer.basicxsd.properties` package. As mentioned above, it is possible to set the following.

- Turn off generation of global element types. Turning off this feature is not recommended as it may cause certain problems with validity of resulting XSD. See ??.
- Minimal number of occurrences of element to define its type globally. (Only if generation of global elements is active.)
- Number of spaces in output per one level of indentation.
- Global type name prefix. It is a string which will be inserted before a name of a type, which is derived from element's name. Can be also an empty string. (Only if generation of global elements is active.)
- Global type name suffix. It is a string which will be appended after a name of a type, which is derived from element's name. Can be also an empty string. (Only if generation of global elements is active.)

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.rpbouret.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 log4jTM. <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.