KEMLG RESEARCH GROUP (UPC)

The WizArg Software

A generic-extension based argumentation semantics solver

Ignasi Gómez-Sebastià

09/06/2010

Extension-based argumentation semantics have shown to be a suitable approach for performing practical reasoning. WizArg is a generic extension-based argumentation semantics solver. The WizArg project consists in two main components: The WizArg Argumentation library (that can be used by any generic software to solve argumentation frameworks and answer questions about extension-based argumentation semantics) and the WizArg front-end (a generic software component that allows users to create, save, and load their own argumentation frameworks).

Contents

1 – The WizArg Project

The WizArg software aims to provide generic computational process (i.e. Java processes) with argumentation-based reasoning capabilities. The WizArg project allows generic software processes to use argumentation meta-interpreters, effectively enabling them to solve argumentation frameworks using a wide range of argumentation semantics.

By solving argumentation frameworks, generic software components can show intelligent behavior, such as: a) negotiation including sceptical and credulous reasoning b) knowledge refination, including justification of information and processes to deal with dynamic on incomplete information.

TheWizArg project consists in two main components:

- TheWizArg Argumentation library, that can be used by any generic software to solve argumentation frameworks and answer questions about them, such as: is an argument credously/ sceptically accepted in a given framework for a given semantics?
- The WizArg front-end, a generic software component that allows users to create, save, and load their own argumentation frameworks. Argumentation frameworks can be processed by the Argumentation library, enabling users to visualize the framework solutions and interactively ask questions about them.

2 – The WizArg front-end

The WizArg front-end makes use of the WizArg Argumentation library and the Jung {http://jung.sourceforge.net/index.html} libraries. The WizArg library provides means to effectively solve the argumentation frameworks specified by the user, and answer questions about them. The Jung libraries allow the user to graphically design their own argumentation frameworks (or modify the ones provided along with the WizArg software) and show the results of the solving process.

The WizArg front-end takes an approach based on labelling for representing the argumentation frameworks graphically. Given an argumentation framework *AF=<AR,attacks>*, WizArg is drawing a directed graph of the form *Graph=<Nodes,Edges>* where the nodes are the arguments (i.e. *Nodes=AR*) and there is a directed edge between two arguments when there is an attack relation between them. Notice the first node in the *Edges* set represents the origin node (corresponds to attacker argument on Argumentation Framework) and the second one, destination node (corresponds to attacked argument on Argumentation Framework).

When solving argumentation frameworks, the WizArg front-end will paint the nodes following a labelling approach, applying the following colours to each of the nodes:

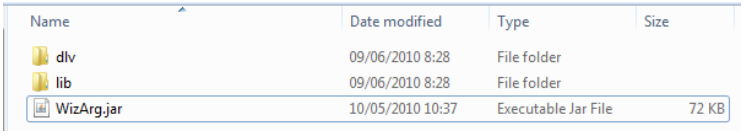
- red if the argument represented by the node is defeated (corresponds to OUT label)
- green if the argument represented by the node is accepted (corresponds to IN label)
- yellow if the argument represented by the node is undecidable (corresponds to UNDEC label).

When designing new argumentation frameworks or modifying existing ones, user can add and delete nodes or attack relations, and move existing nodes (making the graph that represents the framework easier to understand visually). When visualizing results, user is not allowed to add or delete elements, but can freely move existing ones. What's more, user can make use of a wide range of graph drawing algorithms to automatically order the elements on the graph.

2.1 – Introduction

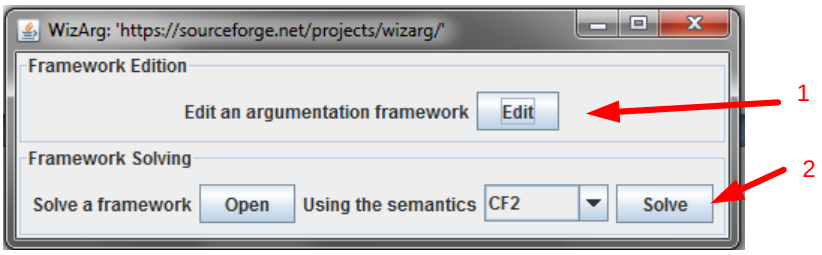
The application is compatible with Linux and Windows systems. It should work properly with MAC systems as well, although it has not been tested on such systems. To run the application a Java Runtime environment is required (http://www.java.com/es/download/manual.jsp).

To run the application first, de-compress the zip archive file. Free programs such as 7-zip (http://www.7-zip.org/) can be used to de-compress it. Once the archive file is decompressed, the application folder is available. To start the application, go to the application folder and double-click on the jar file.



Jar file on the application folder

The application starts and the main window shows up. From here, you can access the framework edition (1) and framework solving (2) windows by pressing the respective buttons. Details on these windows are provided on the next sub-sections.



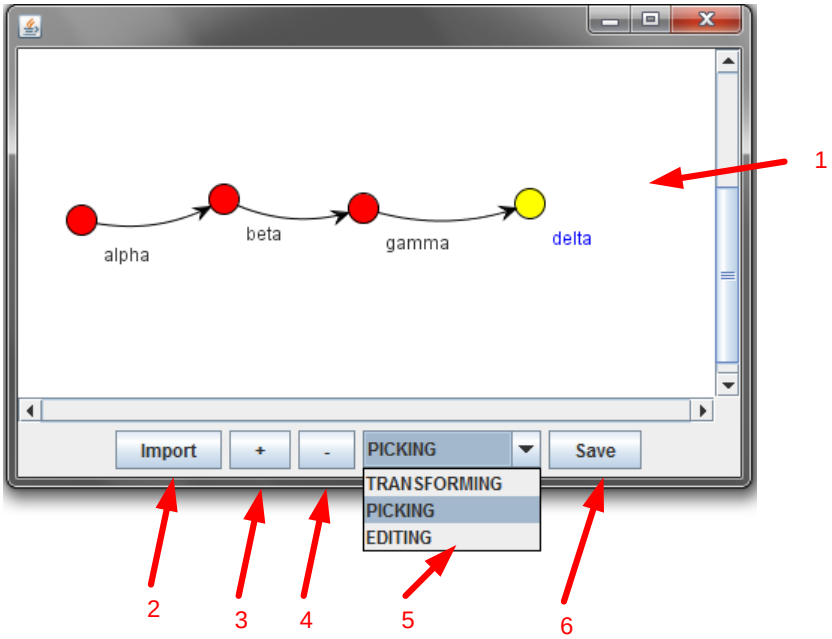
Application's main window

2.2 – Editing frameworks

The framework editing window effectively allows users to define their own argumentation frameworks, setting the arguments and the attack relations between them. The main component of the window is the "editing zone"(1). Users can right-click on the zone to add or remove arguments to the framework. Right-clicking on an empty zone, allows to create a new argument. A prompt, asking for the name of the argument to be created, appears. Users can specify their own names (application makes sure the name provided has not been already assigned to another argument on the framework) or leave the name empty, resulting in automatic name assignation. Right clicking on an argument allows to delete the argument, or create arches between arguments. Arches between arguments can also be created by right-clicking in an argument and dragging-dropping the arrow to another argument.

The created frameworks can be saved using the "Save"(6) button. Once save, frameworks can be edited or solved. Edition of saved frameworks is performed via the "Import"(2) command. This will effectively load an argumentation framework (i.e. nodes and arches) on the editing window. Then, user is free to add and remove nodes or arches. Once done, user can save the framework on the same file (overwriting the imported framework, after confirming the changes) or on a new file, creating a new framework based on the one that has been imported (no confirmation is required then).

Users can zoom-in(3) and out(4) the current framework, and switch between editing, transforming and picking modes (5). Editing mode allows to add and remove nodes and arches on the framework. Picking, allows to select nodes and move them around the editing area, re-ordering the framework, and making it easier to understand visually. Finally, transforming allows to move the framework as a whole along the editing area.



Framework editing window

2.3 – Solving frameworks

The framework solving allows to visually solve an argumentation framework previously edited by the user. A wide range of pre-created argumentation framework files is shipped along with the application (available on *WizArg\dlv\TestFrameworks* folder), so users can easily test this feature.

To solve an argumentation framework, first choose one via the "Open" button. This will show a menu that allows to select the file containing the framework. Trying to solve an argumentation framework without selecting one before results in the application showing a pop-up menu that reminds the user of the correct procedure to be followed in order to solve argumentation frameworks.

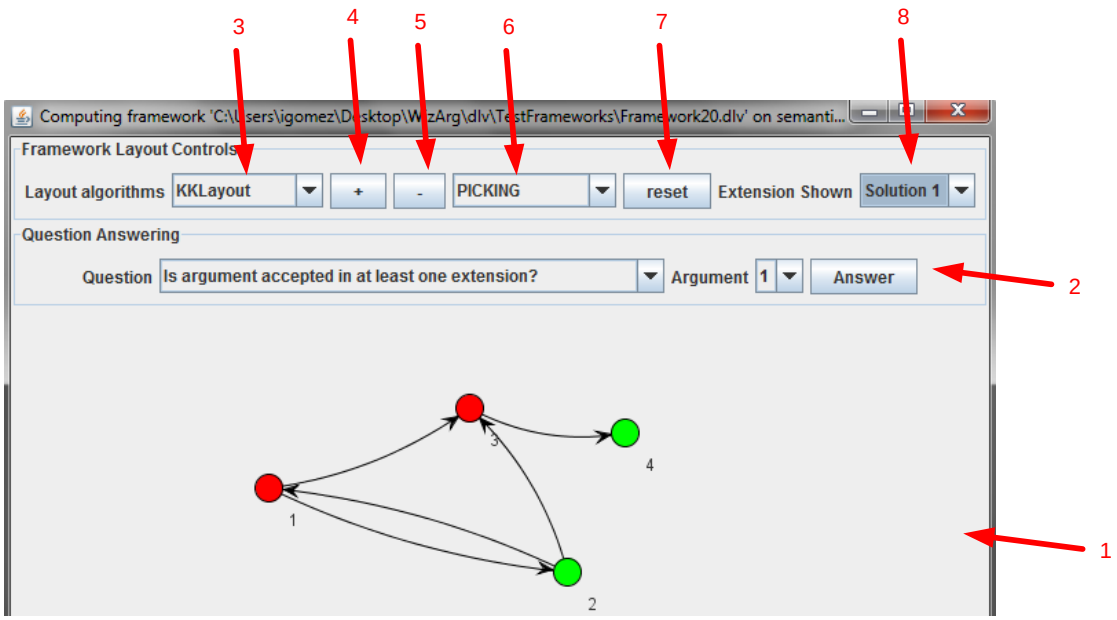
Once the file containing the framework has been selected, pressing the "Solve" button will effectively result in the argumentation framework contained on the file being solved, using the logic programs contained on the DLV meta-interpreters. The logic program to be used is determined by the semantics chosen by the user among the set of available ones. User can use the drop-list to switch among the semantics available. As soon as the framework is solved, the application shows visually the solving results on the framework-solving window.

The main component of the framework solving window is the solving area(1), where the results are visually shown. This area shows the arguments and their attack relations. A colour is applied to each argument depending on its state.

The framework solving window allows to answer questions about the solved framework (2). User has to choose a given question, and the argument it refers to and press the "Answer" button. Application will show a pop-up message with the answer and proof to support it, if required.

The way the information is shown can also be controlled by the user. User can choose among a wide range of graph-ordering algorithms to make the framework easier to understand visually(3), zoom-in(4) or out(5) or manually edit the framework, moving the nodes around, using the picking and transforming options (6). If the user does not like the visual modifications applied, he/she can go back to the initial visual representation via the "reset" button.

Finally, in the case where several possible solutions are available for the framework-semantics chosen, user can switch among them (7).



Framework Solving Window

3- The WizArg library

Section under construction