Drawing Argument Maps in yEd

Introduction

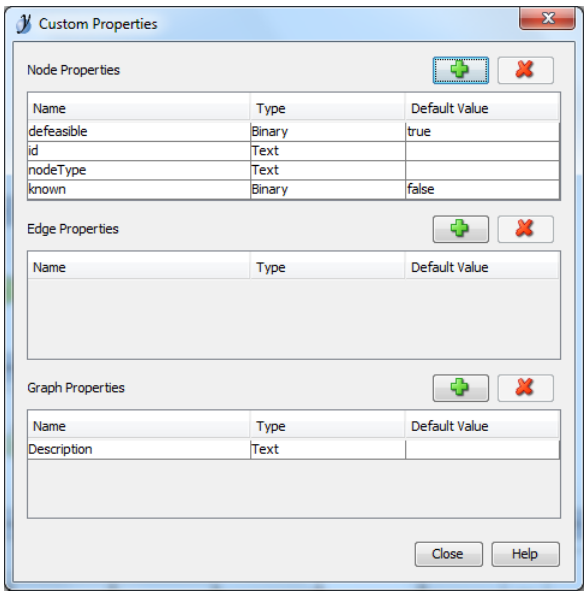
The yEd Graph Editor is a general purpose graph drawing tool that can be used to create argument maps. It stores graphs as GraphML, which is an open standard XML format. Argument Interchange Format (AIF) is an open standard for argumentation that can be expressed as OWL. An argument map drawn in yEd can be transformed into the equivalent AIF with an XSL transformation. Further, the GraphML (and hence the AIF) can be annotated with additional properties that allow direct evaluation of the argument map depicted. This document describes a proof-of-concept prototype that validates these claims.

The method

The object is to translate boxes and arrows in the argument map to AIF nodes and their linkages. This is relatively straightforward in the case of the OVA-style argument maps[1], where diagrammatic nodes specifically depict AIF nodes.

GraphML documents the nodes and edges of a graph and their properties. yEd allows *data properties* to be defined on nodes and edges in a graph; and these are stored in the GraphML along with various properties relating to visualization and layout.

It would be possible to achieve the desired effect with an XSL stylesheet that just uses stylistic elements (shape, colour, etc.) of the argument map to create AIF. However, this would restrict the utility of the transformation to using yEd as the argument map editor, and then only allow a single style of argument map within yEd. Instead, the following custom data properties on graph nodes are defined:



[1] See http://ova.arg-tech.org/

These custom properties are used alongside the *URL* and *Description* properties that yEd assigns to all nodes and edges by default.

Broadly, the transformation is as follows: Each yEd graph node is translated to an AIF node of type ***nodeType***, which will have URI set to the value of ***URL*** if that property is set, or will be an RDF blank node if not. The ***Description*** property becomes the *claimText* of the AIF node. The ***known*** and ***defeasible*** properties translate into custom boolean RDF properties[2] that extend AIF to provide hints for an evaluation of the argument. If *known* is true, then the AIF-node is presumably an I-node that is a premise of the argument. If *defeasibl*e is also true (or not specified) then it is an ordinary premise. If *defeasible* is false, then it is an axiom. For an RA-Node or CA-Node[3], the *defeasible* property indicates whether the corresponding rule is defeasible (the default) or not.

The *defeasible* and *known* properties provide the information needed to create ASPIC+[4] from the generated AIF – or to create ASPIC+ directly from GraphML. The ***id*** property is there to support the latter case, where it becomes the name of the logical proposition representing the I-node. This is not strictly necessary (one-up integer labels could be used instead for example), but does aid readability of the generated ASPIC+ statements. Similarly, it can also be used to generate blank-node ID's in AIF for the same purpose.

In summary:

GraphML	AIF
nodeType (I-node|RA-node|CA-node)	Node of specified type
Description	***claimText*** property
known	***premise*** property (extending AIF)
defeasible	***defeasible*** property (extending AIF)
URL	If set, becomes the node URI
id	Not used in AIF (supports GraphML -> ASPIC+)

Notes

1. The use of yEd as tool, and OVA-style argument maps as a representation, are arbitrary choices. The same general approach can apply to any XML representation of arguments where a mapping can be defined between glyphs in the argument map and the classes and properties of AIF. Notably, this includes SVG and XHTML.
2. Other transformations of the argument map can be imagined. For example, generating a SKOS vocabulary from the I-nodes.
3. It's possible to go the other way, and generate GraphML from AIF. Generating a graph with no visualization or layout is simple, but manual effort is needed to style and layout the

[2] Currently ***http://to.do/premise*** and ***http://to.do/defeasible*** - where the namespace is intended to indicate the current status of this piece of modelling.
[3] See the AIF Ontology at: *https://www.arg-tech.org/index.php/projects/contributing-to-the-argument-interchange-format*
[4] DOI: 10.1080/19462166.2013.869766

argument map once the GraphML is imported into yEd. There are likely to be options within yEd and associated tools to make this easy, but these have not yet been investigated.

4. This is strictly a proof-of-concept exercise and does not imply that yEd should be used a tool for argumentation. It does imply, however, that GraphML can be used as format for documenting and exchanging example arguments between researchers; and for authorities to communicate argumentation requirements to suppliers. Further, this approach (or something like it) is required to effectively and unambiguously define and disseminate argument schemes between parties.
5. Currently, this is an informal proof-of-concept. However, since it supports collaboration, further development is best done in an open and collaborative way.