Visualization of Solutions of Abstract Argumentation Frameworks

Report by: Rajwardhan Kumar

October 19, 2016

Abstract

Navigating through the solution space of an argumentation framework (AF) is a complex task as the solution space can be extremely large. A solution for a semantics of an AF is a set of extensions. Each extension is a subset of arguments following a certain criteria. This project aims at introducing different ways to visualize all extensions for a semantics of an AF on a directed graph. The project introduces an interactive tool for visualization of solutions called *solutionVizArg*.

1 Introduction

Argumentation frameworks (AFs) as introduced in the paper of Dung [6] can be represented as directed graphs. Each argument corresponds to a node and each attack (conflict) corresponds to an edge.

Dung [6] also introduced extension-based semantics of an argumentation framework, where each extension is a subset of the set of arguments following a certain criteria.

Exploring the solution space for a semantics of an argumentation framework is a difficult task. Visualizing each extension by iteratively navigating through the solution space and coloring the arguments present in the extension differently from other arguments in the AF is a very extensive process due to the size of the solution space.

The aim of the project is to introduce ways to visualize all extensions for a semantics of an argumentation framework, apart from having to iteratively navigate through each extension of the solution space and coloring arguments present in an extension differently from the other arguments not present in the extension.

The methods introduced for visualization of solutions are based on using a simple concurrent calculative method of MapReduce [5] on the solution space for a semantics of an argumentation framework, which counts the presence of arguments in all the extensions and keeps track of the extensions in which arguments have its presence. The visualization is done by changing the shape, color, size and layout of all arguments present in the argumentation framework.

The main methods of visualization introduced are by using credulous acceptance, skeptical acceptance [7] and by degree of acceptance of an argument in the solution space.

The project is also accompanied with an interactive argumentation framework tool *solutionVizArg* which includes the above mentioned visualization methods. The tool is also capable of combining the various methods of visualization and enhance its capability.

The paper is organized as follows: Section 2 consists of the basic definitions and properties of the argumentation framework, Section 3 discusses the methods used for the visualization of the solution space and a thorough analysis and comparison of the methods, Section 4 describes the layout algorithms used for the arrangement of an argumentation framework in the tool *solutionVizArg* and Section 5 discusses tools that are similar to *solutionVizArg*.

2 Preliminaries

This section introduces the formal definition of Dung's argumentation framework along with the extension-based semantics and its basic properties. The definition of Dung's argumentation framework is as follows:

Definition 1. *An argumentation framework [6] (AF) is a pair* $(\mathbb{A}, \mathbb{R})$ *where*

- $\mathbb{A}$ *is a set of arguments,*
- $\mathbb{R} \subseteq \mathbb{A} \times \mathbb{A}$ *is a relation representing the conflicts (i.e. attacks).*

If $(a, b) \in \mathbb{R}$ holds we say that a attacks b, or b is attacked by a in AF. $\mathbb{A}$ may be finite or infinite in general but for the purpose of this project, we will restrict the definition to the case of a finite set of arguments. An argumentation framework is represented as a directed graph, where nodes are arguments and edges are drawn from attacking to attacked arguments. A simple example of an argumentation framework is shown in Example 1 (Figure 1).

Example 1. *Let* $\mathbb{F} = (\mathbb{A}, \mathbb{R})$ *be an argumentation framework. The graphical representation of* $\mathbb{F}$ *is shown in Figure 1, where* $\mathbb{A}$ *and* $\mathbb{R}$ *is given as follows:*

- $\mathbb{A} = \{a1, a2, a3, a4, a5, a6, a7\}$ *being the set the arguments and*
- $\mathbb{R} = \{(a1, a2), (a2, a3), (a3, a1), (a1, a4), (a4, a4), (a5, a4), (a5, a6), (a6, a5), (a6, a7)\}$ *denotes the set of conflicts (attacks).*

2.1 Semantics

In this section we define the extension-based semantics. Semantics for argumentation frameworks are given via a function $\mathcal{E}_\sigma$ which assigns to each AF $\mathbb{F} = (\mathbb{A}, \mathbb{R})$ a set $\mathcal{E}_\sigma(\mathbb{F}) \subseteq 2^{\mathbb{A}}$ of extensions. We shall consider σ as cf, ad, st, co, pr and gr which stand for conflict-free, admissible, stable, complete, preferred and grounded semantics respectively. The basic definitions of the semantics for argumentation frameworks as presented in [6] are as follows.

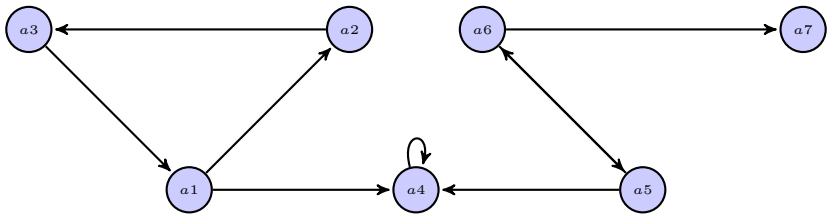

Figure 1: Example of an argumentation framework

Definition 2. *Given an AF* $\mathbb{F} = (\mathbb{A}, \mathbb{R})$*. A set* $E \subseteq \mathbb{A}$ *is conflict-free in* $\mathbb{F}$ *i.e.,* $E \in \mathcal{E}_{cf}(\mathbb{F})$ *if and only if for every* $a, b \in E$*, we have* $(a, b) \notin \mathbb{R}$*.*

A set E of arguments attacks b (or $(\mathrm{E}, \mathrm{b}) \in \mathbb{R}$) if b is attacked by an argument in E.

Definition 3. *Given an AF* $\mathbb{F} = (\mathbb{A}, \mathbb{R})$*. An argument* b *is defended by a set* $E \subseteq \mathbb{A}$ *iff for any argument* $a \in \mathbb{A}$*, if* a *attacks* b *then* $(E, a) \in \mathbb{R}$*.*

Definition 4. *Given an AF* $\mathbb{F} = (\mathbb{A}, \mathbb{R})$ *and* $E \subseteq \mathbb{A}$*, E is an admissible extension* $(E \in \mathcal{E}_{ad}(\mathbb{F}))$ *iff E is conflict-free and each* $a \in E$ *is defended by E in* $\mathbb{F}$*.*

Definition 5. *Given an AF* $\mathbb{F} = (\mathbb{A}, \mathbb{R})$ *and* $E \subseteq \mathbb{A}$*, E is a stable extension* $(E \in \mathcal{E}_{st}(\mathbb{F}))$ *iff E is conflict-free and for each* $a \in \mathbb{A} \setminus E$*,* $(E, a) \in \mathbb{R}$ *holds.*

Definition 6. *Given an AF* $\mathbb{F} = (\mathbb{A}, \mathbb{R})$ *and* $E \subseteq \mathbb{A}$*, E is a preferred extension* $(E \in \mathcal{E}_{pr}(\mathbb{F}))$ *iff* $E \in \mathcal{E}_{ad}(\mathbb{F})$ *and for each* $E^{'} \in \mathcal{E}_{ad}(\mathbb{F})$*,* $E \not\subset E^{'}$ *holds.*

Definition 7. *Given an AF* $\mathbb{F} = (\mathbb{A}, \mathbb{R})$ *and* $E \subseteq \mathbb{A}$*, E is a complete extension* $(E \in \mathcal{E}_{co}(\mathbb{F}))$ *iff* $E \in \mathcal{E}_{ad}(\mathbb{F})$ *and for any* $a \in \mathbb{A}$ *defended by E in* $\mathbb{F}$*,* $a \in E$ *holds.*

Definition 8. *Given an AF* $\mathbb{F} = (\mathbb{A}, \mathbb{R})$ *and* $E \subseteq \mathbb{A}$*, E is a grounded extension* $(E \in \mathcal{E}_{gr}(\mathbb{F}))$ *iff* $E \in \mathcal{E}_{co}(\mathbb{F})$ *and for each* $E^{'} \in \mathcal{E}_{co}(\mathbb{F})$*,* $E^{'} \not\subset E$ *holds.*

Consider the AF presented by Figure 1 and Example 1. Table 1 represents columns as conflict-free set (cf), admissible set (ad), stable extension (st), preferred extension (pr), complete extension (co) and grounded extension (gr). The column cf contains subsets of arguments from the argumentation framework of Example 1. The "$\times$" represents the conflict-free set for which the extension is true.

2.2 Notion of Acceptance State

In this section we discuss the acceptance state of an argument $a \in \mathbb{A}$ that can be conceived in terms of its extension membership. A basic classification encompasses only two possible states for an argument, namely accepted or not accepted. In this respect, two alternative types of acceptance, namely credulous

Solution					
cf	ad	st	pr	co	gr
$\{\}$	$\times$			$\times$	$\times$
$\{a5\}$	$\times$				
$\{a6\}$	$\times$		$\times$	$\times$	
$\{a5, a7\}$	$\times$		$\times$	$\times$	
$\{a7\}$					
$\{a1, a7\}$					
$\{a1, a5\}$					
$\{a1, a5, a7\}$					
$\{a3\}$					
$\{a3, a7\}$					
$\{a3, a5\}$					
$\{a3, a5, a7\}$					
$\{a2\}$					
$\{a1\}$					
$\{a1, a6\}$					
$\{a2, a6\}$					
$\{a3, a6\}$					
$\{a2, a5, a7\}$					
$\{a2, a7\}$					
$\{a2, a5\}$					

Table 1: Semantics Example

and skeptical can be considered. For a set $\mathrm{E} \subseteq \mathbb{A}$ and semantics σ introduced above, we define the following acceptance states.

Definition 9. *Credulous Acceptance $Cred_\sigma$: Given an AF $\mathbb{F} = (\mathbb{A}, \mathbb{R})$ and an argument $a \in \mathbb{A}$. Is a contained in some $E \in \mathcal{E}_\sigma(\mathbb{F})$?*

As one can observe in the above Table 1 of Example 1 that arguments $a5, a6$ and $a7$ are credulously accepted for preferred and complete semantics.

Definition 10. *Skeptical Acceptance $Skep_\sigma$: Given an AF $\mathbb{F} = (\mathbb{A}, \mathbb{R})$ and an argument $a \in \mathbb{A}$. Is a contained in each $E \in \mathcal{E}_\sigma(\mathbb{F})$?*

If no extension exists then all arguments are skeptically accepted and no argument is credulously accepted, which is only valid for stable semantics. As one can observe in the above Table 1 of Example 1 for stable semantics.

3 Visualization of Solutions

Given a set of n arguments in the argumentation framework $\mathbb{F} = (\mathbb{A}, \mathbb{R})$, where $\mathbb{A} = \{a_1, a_2, \ldots, a_n\}$. A set of extensions $\mathbb{S} = \{\mathbb{A}_0, \mathbb{A}_1, \mathbb{A}_2, \ldots, \mathbb{A}_m\}$ for some semantics, where each set $\mathbb{A}_i \subseteq \mathbb{A}$ and $\mathbb{A}_i \neq \mathbb{A}_j$ $(0 \leq i, j \leq m$ and $i \neq j)$ and $0 \leq m \leq 2^n$.

As seen above if there are n number of arguments in the argumentation framework, there can be a maximum of 2^n number of extensions for a semantics. An argumentation framework with no attacks and n number of arguments has only one extension, the extension consists of all n arguments for any semantics. The number of extensions for a semantics in an argumentation framework varies from 0 to 2^n. The number of extensions for grounded semantics is always one.

We use the method of MapReduce [5] on the solution space to calculate information about arguments and their acceptance state that can be conceived in terms of its extension membership. The complexity [1] of MapReduce on the solution space is $\mathcal{O}(2^n)$.

Let $\mathbb{S} = \{\mathbb{A}_0, \mathbb{A}_1, \mathbb{A}_2, \ldots, \mathbb{A}_k\}$ be the solution set and $\mathbb{A} = \{a_1, a_2, \ldots, a_n\}$ be the arguments in the argumentation framework. In this method we first collect all the sets into a map as separate sets $\mathbb{S} = \{\mathbb{A}_0, \mathbb{A}_1, \mathbb{A}_2, \ldots, \mathbb{A}_k\}$. We then associate the arguments $\mathbb{A} = \{a_1, a_2, \ldots, a_n\}$ with the sets that it is present in. The association can be reduced to $a_i \in \mathbb{A}_p, \ldots, \mathbb{A}_q$ where $0 \leq p, q \leq k$ and $k \leq 2^n$.

Let us see this in an example where $\mathbb{A} = \{a_1, a_2, a_3\}$ and $\mathbb{S} = \{\mathbb{A}_1, \mathbb{A}_2, \mathbb{A}_3\}$. The set of extensions $\mathbb{A}_1, \mathbb{A}_2, \mathbb{A}_3$ where,

- $\mathbb{A}_1 = \{a_1\}$
- $\mathbb{A}_2 = \{a_1, a_2\}$
- $\mathbb{A}_3 = \{a_1, a_2, a_3\}$

is mapped to

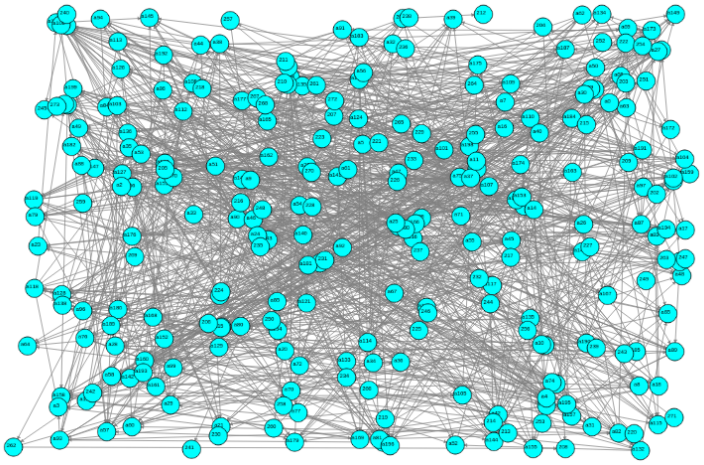

Figure 2: Argumentation Framework: Initial

- $[a_1, \mathbb{A}_1]$
- $[a_1, \mathbb{A}_2]$
- $[a_1, \mathbb{A}_3]$
- $[a_2, \mathbb{A}_2]$
- $[a_2, \mathbb{A}_3]$
- $[a_3, \mathbb{A}_3]$

and is reduced to

- $[a_1, [\mathbb{A}_1, \mathbb{A}_2, \mathbb{A}_3]]$
- $[a_2, [\mathbb{A}_2, \mathbb{A}_3]]$
- $[a_3, [\mathbb{A}_3]]$.

In this section we use the data gathered from the above method of MapReduce to count the number of extensions for each argument along with the extension details and use the data for the visualization of all extensions. We will use an initial argumentation framework (Example 2) Figure 2 to see the variations in the different kinds of visualization of solutions. The initial argumentation framework (Figure 2) consists of 273 arguments and 1066 attacks. The solution for stable semantics consists of 38 extensions, preferred semantics consists of 39 extensions, complete semantics consists of 491 extensions and grounded semantics always consists of 1 extension.

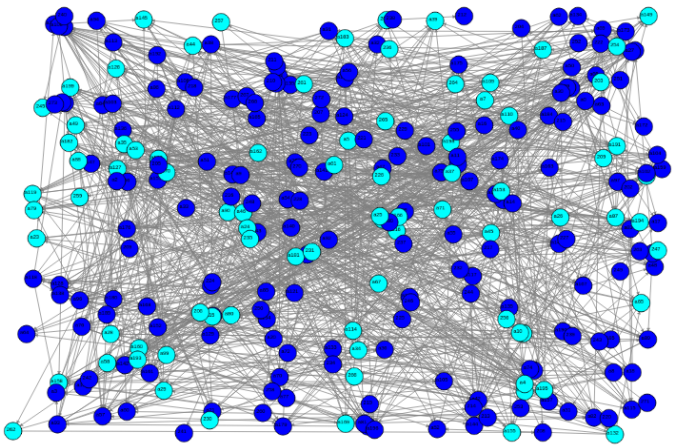

Figure 3: Credulous Acceptance: Color of Arguments in Solution for Complete Semantics

3.1 Visualizing Credulous Acceptance

Credulous acceptance is defined above in Section 2. The arguments satisfying credulous acceptance with respect to a semantics are visualized either by changing the shape or by changing the color of the respective arguments.

The shape of circular arguments can be changed to square arguments or the color can be changed from cyan to dark blue. This way we are able to distinguish arguments that are present in the solution space from the ones that are not present in the solution space.

Both changing of color and changing of shape of arguments have the same visualizing capability under credulous acceptance.

An example of visualizing credulous acceptance w.r.t. complete semantics for the argumentation framework of Figure 2 can be seen on Figure 3 and Figure 4.

3.2 Visualizing Skeptical Acceptance

Skeptical acceptance is defined above in Section 2. The arguments satisfying skeptical acceptance with respect to a semantics can be visualized either by changing the shape or by changing the color of the respective arguments.

The shape of circular arguments can be changed to square arguments or the color can be changed from cyan to dark blue. This way we are able to distinguish arguments that are skeptically accepted from the ones that are not skeptically accepted. If no extension exists with respect to stable semantics then all arguments are skeptically accepted and can be visualized in a similar manner.

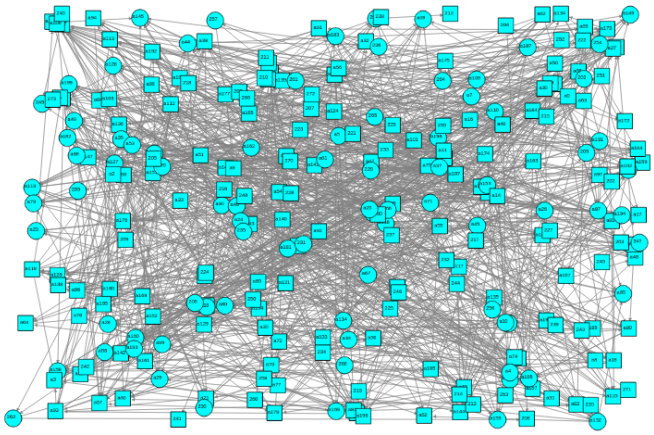

Figure 4: Credulous Acceptance: Shape of Arguments in Solution for Complete Semantics

An example of visualizing skeptical acceptance w.r.t. complete semantics for the argumentation framework of Figure 2 can be seen on Figure 5.

3.3 Visualizing the Degree of Acceptance

Visualizing the degree of acceptance for an argument is the arrangement of arguments according to presence of arguments in extensions i.e., the degree of acceptance of an argument with respect to a semantics is equal to the number of occurrences of that argument in the solution space.

An argument can be present in 0 to 2^n number of extensions in the solution space, where n is the number of arguments in the argumentation framework.

3.3.1 Degree of Acceptance using Size

Visualizing the degree of acceptance by changing size is based on increasing the size of arguments. The size factor of each argument is solely dependent on the degree of an argument in the solution space w.r.t. a semantics. The greater the size of an argument the more its degree for a semantics of an argumentation framework.

The calculation of the size of an argument is done in the following manner:

1. size = (factor / 10) + initial size

where, factor is the degree of an argument w.r.t. a semantics and initial size is the size of the argument initially. The more we increase the value of divisor for the quantity factor the lesser the difference in size with the initial size. The size denotes the radius in circular shapes and length of diagonals in square shapes.

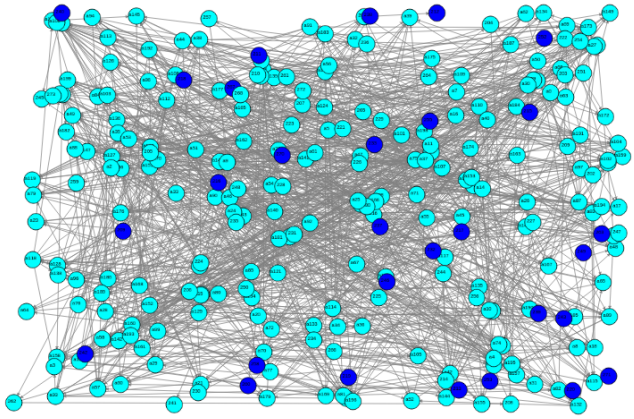

Figure 5: Skeptical Acceptance: Color of Arguments in Solution for Complete Semantics

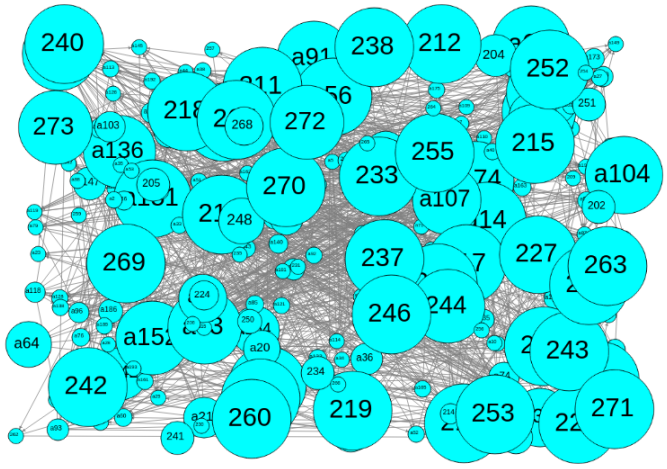

Figure 6: Degree of Acceptance: Visualizing using Size for Complete Semantics

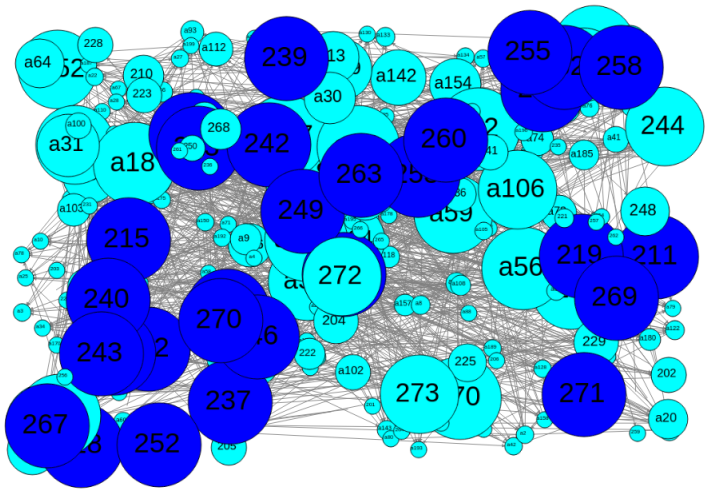

Figure 7: Degree of Acceptance and Skeptical Acceptance for Complete Semantics

The disadvantage of this method is that the size of some arguments are not distinguishable with naked eyes.

Several attempts were made for calculating the size, they are discussed here. Formula 2 adds the number of arguments in the AF to Formula 1, which in turn increases the size of all arguments by the same value. Formula 2 is similar to Formula 1 except for the size of arguments which are more due to the addition of total number of arguments. In Formula 3, if the total number of arguments in the AF is larger than the degree of an argument w.r.t. a semantics, the size of all our arguments is almost the same as initial size.

2. size = (factor / 10) + initial size + total number of args

3. size = (factor / total number of args) + initial size

An example of visualizing the degree of acceptance w.r.t. complete semantics for the argumentation framework of Figure 2 can be seen on Figure 6. Figure 7 is a combination of visualizing the skeptical acceptance and degree of acceptance using size. The difference in size of arguments that are skeptically accepted and a few other arguments are visibly unrecognizable.

3.3.2 Degree of Acceptance using Linear Layout

Visualizing the degree of acceptance using linear layout is the arrangement of arguments for a semantics of an argumentation framework according to the degree of arguments in the solution space i.e., Arguments from the argumentation framework that are present in same number of extensions of the solution space are arranged on the same y-coordinate value of the linear layout.

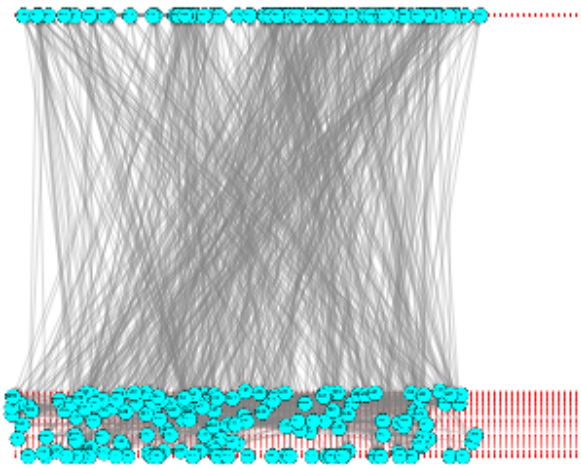

Figure 8: Degree of Acceptance Linear layout for Stable Semantics

The arguments are arranged using Formula 1. The x-coordinates of the arguments arrange themselves randomly. The linear y-coordinate value of the arguments increase with the increase in the degree of an argument in the solution space. where y_i is the initial y-coordinate.

1. y = y_i * 10.0 + degree of argument * 3.

An example of the linear layout w.r.t. stable semantics for the argumentation framework of Figure 2 can be seen on Figure 8. The arguments with highest degree in the solution space lay at the bottom and arguments at the extreme top are not present in the solution space.

The disadvantage of this method of visualizing is the extensive overlapping of arguments on the same y-coordinate value even though the x-coordinate value of the arguments are randomized.

Other notable disadvantage is the size of the y-coordinate value due to increasing number of extensions (491 extensions for complete semantics) as in example Figure 9. The increase in y-coordinate value makes the static image extremely large.

Linear layout of arguments cannot be used to answer skeptical acceptance. If the solution for a semantics of an argumentation framework contains an empty set as an extension along with other set of extensions, it implies that there are no skeptically accepted arguments. Such cases make it difficult to recognize the arguments on extreme y-coordinates as skeptically accepted.

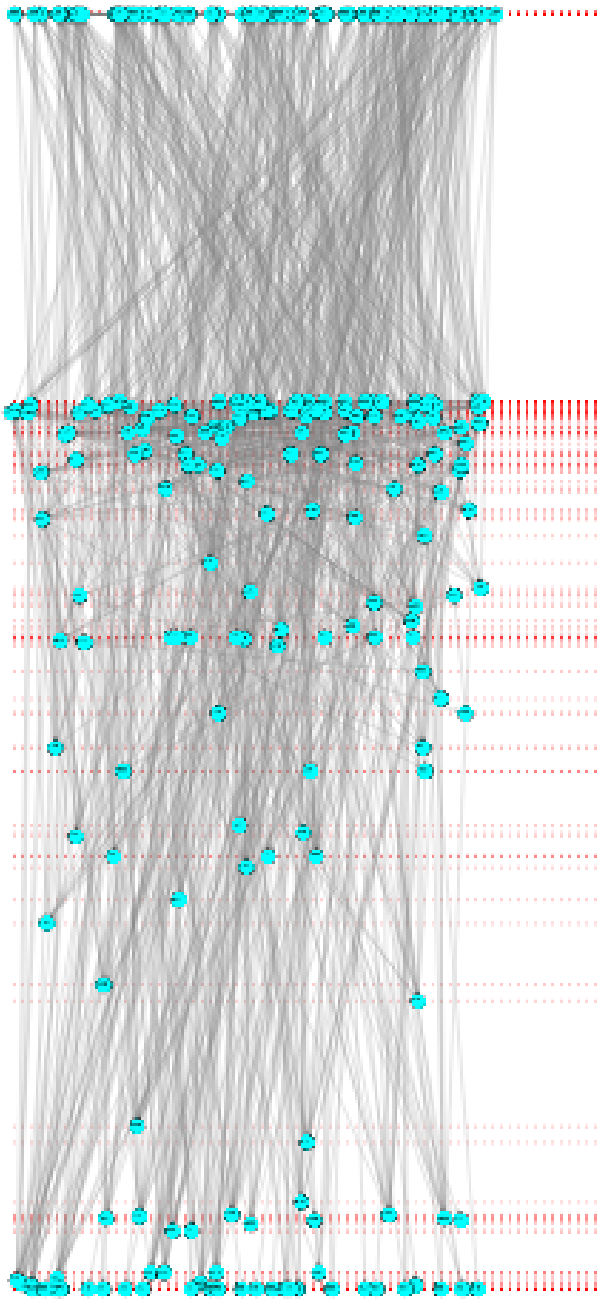

Figure 9: Degree of Acceptance Linear layout for Complete Semantics

3.3.3 Degree of Acceptance using Circular Layout

Visualizing the degree of acceptance using a circular layout is based on arrangement of arguments according to the degree of arguments in the solution space i.e., Arguments from the argumentation framework for a semantics that are present in same number of extensions of the solution space are arranged in a circle of the same radius and arguments that have occurrences in more number of extensions for a semantics of an argumentation framework are arranged in a larger circular radius.

An example of the circular layout w.r.t. stable semantics for the argumentation framework of Figure 2 can be seen on Figure 10. The arguments with highest degree in the solution space lay at the outer most circle and arguments at the inner most circle are not contained in the solution space.

The radius of each circle is calculated by taking parameters maximum radius of the canvas, degree of an argument in the solution space and the total number of arguments. The radius for the circle is as follows:

$$\text{radius} = (\text{ maximum radius -2} * \text{total number of arguments}) + (4 * \text{number of extensions for an argument})$$

The farther an argument is from the center of a circle the more the degree of the argument in the solution space. Another example of the circular layout w.r.t. complete semantics for the argumentation framework of Figure 2 can be seen on Figure 11.

Circular layout by degree of acceptance overcomes the visualizing problems faced by the previous two methods namely degree of acceptance using size and linear layout. Each circle clearly distinguishes arguments present in different number of extensions in the solution space. Circles occupy more space on the canvas than lines in linear layout, hence overlapping of arguments is no more a problem.

Circular layout of arguments also can not be used to answer skeptical acceptance. If the solution for a semantics of an argumentation framework contains an empty set as an extension along with other set of extensions, it implies that there are no skeptically accepted arguments. Such cases make it difficult to recognize the arguments on outer most circle as skeptically accepted.

3.4 Combination of Methods

Combination of the above discussed methods of visualizing by credulous acceptance, skeptical acceptance and degree of acceptance can give us a more intuitive visualization about the arguments present in the solution space for a semantics of an argumentation framework.

We know from above that it is difficult to answer skeptical acceptance when visualizing the degree of acceptance using linear or circular layout. We can thus

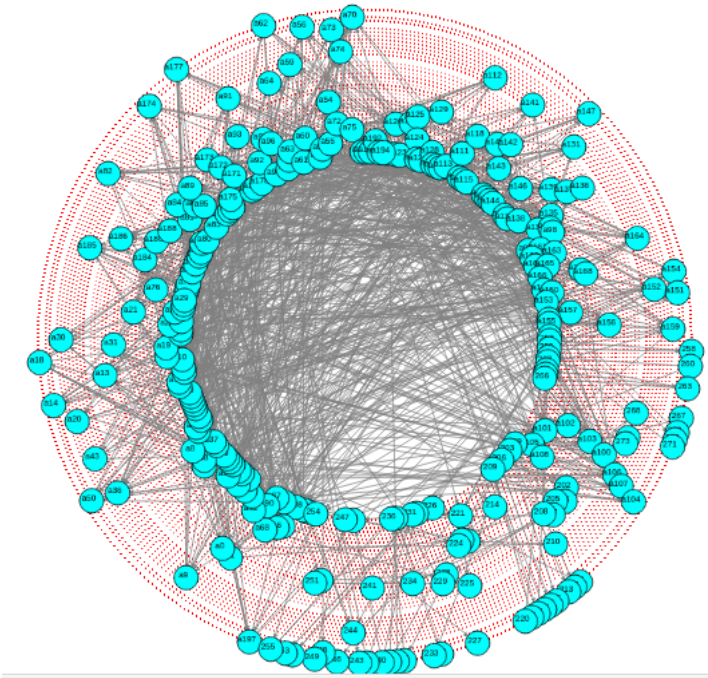

Figure 10: Degree of Acceptance using Circular layout for Stable Semantics

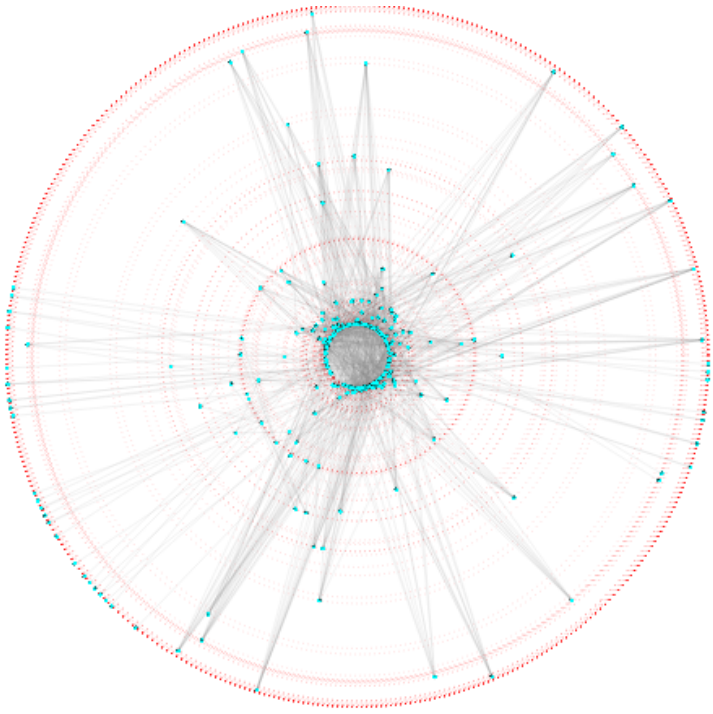

Figure 11: Degree of Acceptance using Circular layout for Complete Semantics

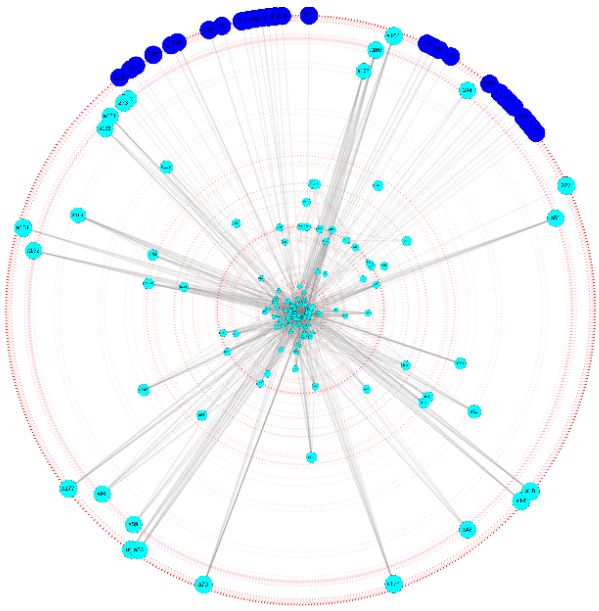

Figure 12: Degree of Acceptance using Circular layout and Skeptical Acceptance for Complete Semantics

combine the method of coloring arguments for skeptical acceptance along with a layout method.

An example of a combination for skeptical as well as circular layout or linear layout w.r.t. complete semantics for the argumentation framework of Figure 2 can be seen on Figure 12 and Figure 13 respectively. The arguments colored in dark blue are the skeptically accepted arguments. The degree of acceptance is clearly visible for every argument in the solution space using the method of circular layout or linear layout.

4 Visualization Techniques in *solutionVizArg*

The directed graph layout [11] of the argumentation framework represented in the tool *solutionVizArg* consists of spring layout, random layout, random circular layout, scaling of the canvas and rotation of the canvas. Random layout and random circular layout methods are randomly implemented on each argument. Scaling of the canvas helps in zooming in and out of the screen. Rotation of the canvas can be done by 360 degree. Spring Layout is implemented as follows:

The spring layout algorithms [8, 10] used are also known as force-directed layout. Force-directed methods use analogies from physics to compute the positions of a graph's vertices. The layout algorithm's goal is to find a configuration of those bodies where the sum of all forces is minimized, i.e. it assigns every

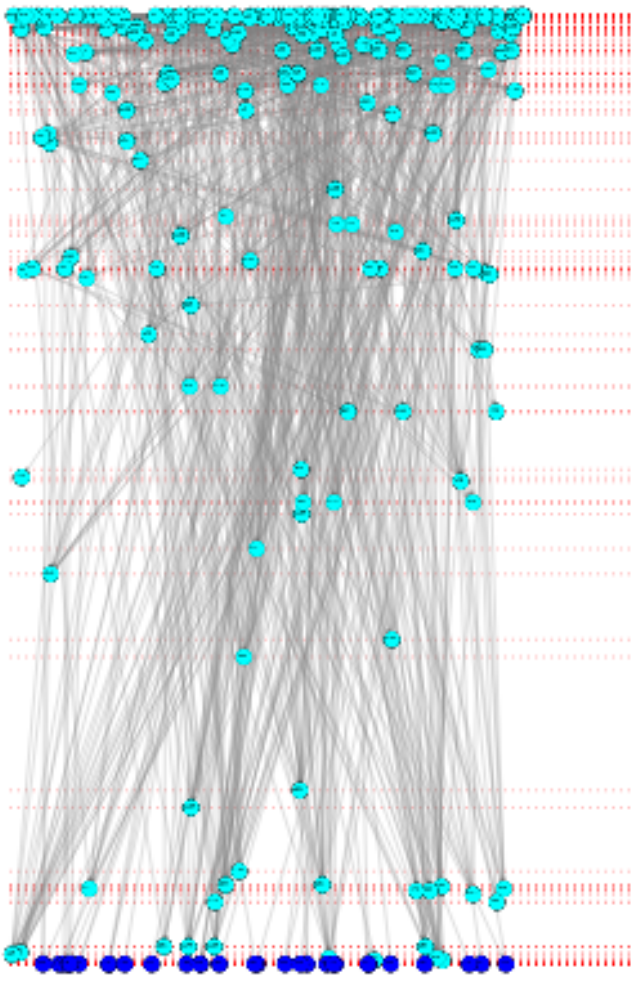

Figure 13: Degree of Acceptance using Linear Layout and Skeptical Acceptance for Complete Semantics

node to a position such that the sum of forces which influence this vertex is 0. Each argument (vertex) is associated with two kinds of forces namely attractive forces and repulsive forces, denoted by f_a and f_r respectively.

$$f_a = k_a \log(d)$$

$$f_r = - \frac{k_r}{d^2}$$

Where k_a and k_r are experimental constants and d is the euclidean distance between two vertices or arguments. The layout algorithm has a good performance if the number of arguments are less than 100.

5 Related Work

Since we are dealing with visualization related to argumentation framework, it is worth to mention the tools ConArg [2], Aspartix [9], grafix [3] and ArgTeach [4].

ConArg is a web based tool. Aspartix, ArgTeach and grafix are both desktop as well as a web based tool. ConArg and Aspartix are computing and visualizing tools with more number of semantics other than the classical Dung semantics, such as semi-stable, ideal and stage.

ConArg is a web based tool with interactive features. Visualization of solutions in ConArg includes list viewing of the solution sets and iterative visualization of extensions.

Aspartix is a web based tool which performs neatly with less number of arguments. The tool is good for giving solutions for a semantics. The visualization of solution can only be done by the method of iterative visualization of extensions.

grafix is a graphical tool for handling argumentation frameworks, that can be represented by weighted directed graphs. grafix can basically handle 3 different kinds of argumentation frameworks, namely classical argumentation frameworks, abstract bipolar argumentation frameworks and abstract partial argumentation frameworks.

ArgTeach is an interactive intelligent tutor that facilitates the learning of different labeling semantics in argumentation framework.

ConArg and Aspartix can visualize solutions for a semantics of an argumentation framework but are limited to iterative and list visualization.

solutionVizArg can visualize solutions in a number of ways such as iterative visualization, visualization by degree of acceptance, visualization by credulous and skeptical acceptance.

6 Conclusion

In this paper, we contribute methods for visualization of solutions for a semantics of an argumentation framework. We first use MapReduce on the solution

space to compute the data required for the visualization. The data is then used to visualize the solutions for a semantics of an argumentation framework. The methods used are visualizing the credulous acceptance, skeptical acceptance and degree of acceptance using size, linear layout and circular layout. Visualizing the Degree of acceptance is a newly introduced visualizing method.

Among the above mentioned methods visualization by degree of acceptance is the most promising and in particular degree of acceptance using circular layout. We have shown the advantages of visualization by degree of acceptance using circular layout. The combination of above mentioned methods gives us a better intuitive information about the solution space and answers several decision problems by using one static image of the argumentation framework.

An interactive tool *solutionVizArg* has been introduced. The tool incorporates all of the mentioned methods for visualization of solutions for a semantics of an argumentation framework.The performance of *solutionVizArg* depends on the number of arguments in the AF and the size of the solution space. The tool can handle more than 2000 arguments in the argumentation framework. The tools performance starts deteriorating after the threshold limit of 1500 arguments. The tools performance is excellent when number of arguments in the AF is less than 500.

Future work can focus on 3D visualization and representation of the solutions. Further work may also focus on using machine learning techniques on immensely large solution space and visualizing the results.

References

[1] F. Benjamin et al. "On the Computational Complexity of MapReduce". In: *Proceedings of DISC 2015*. Ed. by Yoram Moses and Matthieu Roy. Vol. LNCS 9363. 29th International Symposium on Distributed Computing. Toshimitsu Masuzawa and Koichi Wada. Tokyo, Japan: Springer-Verlag Berlin Heidelberg, Oct. 2015. DOI: 10.1007/978-3-662-48653-5_1.

[2] S. Bistarelli and F. Santini. "Conarg: A constraint-based computational framework for argumentation systems". In: *Proceedings of IEEE 23rd International Conference on Tools with Artificial Intelligence* (2011), 605–612. DOI: doi.ieeecomputersociety.org/10.1109/ICTAI.2011.96.

[3] C. Cayrol, S. Doutre, and MC Lagasquie-Schiex. "GRAFIX: a Tool for Abstract Argumentation". In: *Proceedings of COMMA 2014-Computational Models of Argument , Atholl Palace Hotel, Scottish Highlands, UK, September 9-12, 2014*. Ed. by Simon Parsons et al. Vol. 266. Frontiers in Artificial Intelligence and Applications. IOS Press, 2014, pp. 453–454. ISBN: 978-1-61499-436-7. DOI: http://dx.doi.org/10.3233/978-1-61499-436-7-453.

[4] J. Dauphin and C. Schulz. “Arg Teach - A Learning Tool for Argumentation Theory”. In: *Proceedings of 2014 IEEE 26th International Conference on Tools with Artificial Intelligence*. 2014, pp. 776–783. DOI: 10.1109/ICTAI.2014.120.

[5] J. Dean and S. Ghemawat. “MapReduce: Simplified Data Processing on Large Clusters”. In: *Communications of the ACM* 51 (Jan. 2008), pp. 107–113. ISSN: 0001-0782. DOI: 10.1145/1327452.1327492.

[6] P. M. Dung. “On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games”. In: *Artificial Intelligence* 77.2 (1995), pp. 321 –357. ISSN: 0004-3702. DOI: http://dx.doi.org/10.1016/0004-3702(94)00041-X.

[7] Paul E. Dunne. “Computational Properties of Argument Systems Satisfying Graph-theoretic Constraints”. In: *Artificial Intelligence* 171 (July 2007), pp. 701–729. ISSN: 0004-3702. DOI: 10.1016/j.artint.2007.03.006.

[8] P. EADES. “A Heuristics for Graph Drawing”. In: *Congressus Numerantium* 42 (1984), pp. 146–160.

[9] U. Egly, S. A. Gaggl, and S. Woltran. “Answer-set programming encodings for argumentation frameworks”. In: *Argument & Computation* 1.2 (2010), pp. 147–177. DOI: 10.1080/19462166.2010.486479.

[10] T. M. J. Fruchterman and E. M. Reingold. “Graph Drawing by Force-directed Placement”. In: *Software Pract. Exper.* 21 (Nov. 1991), pp. 1129–1164. ISSN: 0038-0644. DOI: 10.1002/spe.4380211102.

[11] R. M. Tarawaneh, P. Keller, and A. Ebert. “A General Introduction To Graph Visualization Techniques”. In: *Proceedings of IRTG 1131 Workshop 2011 - Visualization of Large and Unstructured Data Sets: Applications in Geospatial Planning, Modeling and Engineering*. Ed. by Christoph Garth, Ariane Middel, and Hans Hagen. Vol. 27. OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2012, pp. 151–164. ISBN: 978-3-939897-46-0. DOI: http://dx.doi.org/10.4230/OASIcs.VLUDS.2011.151.

Appendices

solutionVizArg is an interactive tool for the visualization and analysis of solutions for an argumentation framework. Solutions (all extensions) for semantics of an argumentation framework are gathered from Aspartix tool and visualized by *solutionVizArg*. The tool *solutionVizArg* is created using Qt 5.3 and GCC compiler 5.3.1 on Ubuntu 16.04.

System		
Platform	Compilers	Note
OS X 10.8, 10.9, 10.10, 10.11	Clang as provided by Apple	
Ubuntu 14.04 (64-bit)	GCC 4.6.3	
Ubuntu 16.04 (64-bit)	GCC 5.3.1	

Table 2: System Requirements

A Platform Requirements, Installation and Execution

A.1 Requirements

The tool *solutionVizArg* has been successfully executed in the mentioned platforms and compilers of Table 2. For more platform compatibility, visit http://doc.qt.io/qt-5/supported-platforms-and-configurations.html.

A.2 Installing Qt on Mac/Ubuntu for the Source Code

Download Qt from https://www.qt.io/download-open-source/ and Install version Qt 5.3 (tool is made in Qt 5.3 but supports all versions above 5.0).

After Installing Qt, Open "Qt Creator" and browse/open to the source code "solutionVizArg.pro" file after unzip of solutionVizArg.zip. You can make executable for your system by Qt creator. After opening the source code from the Build menu execute "Build" the project and then "run qmake". solutionVizArg is ready for use, then follow the further steps from sub-section Initial First Run. You can execute "run" to run the project for Initial First Run.

In the same directory a build folder is created, the Build folder contains an executable solutionVizArg app.

Note: If the ".pro" file does not work use ".pro.user" file.

Error: Could not resolve SDK path for 'macosx10.x'

- Navigate to where you installed Qt (*default /Users/your username/Qt*) using finder
- Go to the subdirectory *5.3/clang_64/mkspecs* directory
- Open the file called qdevice.pri with a text editor
- Change the line !host_build:QMAKE_MAC_SDK = macosx10.8 to:
  - !host_build:QMAKE_MAC_SDK = macosx10.9 if you are on OS X 10.9 (Mavericks), or

  - !host_build:QMAKE_MAC_SDK = macosx if you are on OS X 10.10 (Yosemite)

- Save the file and restart Qt Creator.

A.3 Installation and Running the app

Apart from executing via the Qt/C++ source code an executable named solutionVizArg.app is also provided for MacOS, we need to run the app. The app can be found after unzip of file solutionVizArg.app.zip.

If you want to run the app in debug mode, Please enter the path ".../solutionVizArg.app/Contents/MacOS/" and run the executable "solutionVizArg" through your command line. The debug mode is fully active when you "check" the "Debug Messages" button on the "solutionVizArg" tool in the "Options − > View" Menu or simply "F9."

A.4 Initial First Run

The initial first run of the system is for configuring the aspartix tool. The tool solutionVizArg asks for the creation of a folder named Argument-data in /home directory or /Users/UserName for Mac system users. Please accept the creation of this folder and unpack the Aspartix tool in this folder. For ubuntu users the aspartix script file aspartix.sh will be in path "/home/Aspartix-data/ASPARTIX-D/" and for Mac users aspartix.sh will be present in the path "/Users / UserName/ Aspartix-data / ASPARTIX-D/".

B User Interface

SolutionVizArg has a simple Graphical User Interface (GUI) composed of:

- menu bar
- toolbar
- dock or toolbox
- a canvas for visualization
- status bar.

B.1 The Menu

The top of the window consists of the menu bar, composed of 6 menus:

- File: Options to load, save the apx file, print the graph in pdf format, new window and close, exit the window.

- Edit: Options to create, delete, select, deselect and find arguments, create, delete attacks, change color of background and change color of all arguments.
- Graph Layout: Layout options for the argumentation framework.
- Options: Antiliasing, Debug and other View options.
- Visualize: Options to visualize the Solutions by credulous acceptance, skeptical acceptance, degree of acceptance, iteration and searching an argument.

B.2 The toolbar

Below the menu the tool bar provides some very important features:

The "select semantics" consists of a combo box comprising of "ST", "PR", "CO" and "GR" namely stable, preferred, complete and grounded respectively. The "Aspartix" button calls the external aspartix tool stored at ".../Aspartix-data/ASPARTIX-D" for the solution.

The button "Visualize" is to be used after using the "Aspartix" button.

The button "ReloadAF" can be used to view the initial argumentation framework text.

The button "Initial Position" sets the argumentation framework back to its starting initial position.

The button "Load AF from text" can be used to create the argumentation framework by writing text into the text space.

The "Rotate" tool is used to rotate the argumentation framework in a circular manner on the canvas.

B.3 The toolbox

The toolbox on the left consists of the area for displaying the text loaded or created. The toolbox also consists of 3 tabs Layout, statistics and number of extensions tab.

B.4 The canvas

The canvas displays the argumentation framework as directed graphs.

B.5 Status Bar

The status bar at the bottom displays the status of the processes executed.

C Argumentation Framework Creation

To start working on the visualization tool, we need to create the argumentation framework on the canvas. We can create an argumentation framework in 3 different ways:

- create/delete arguments and attacks from the menu bar
- create/delete arguments and attacks by canvas interaction
- create arguments and attacks by writing in the text box on the left
- create argumentation framework by uploading an apx file. Ctrl+O is the shortcut for uploading.

C.1 Create/Delete Arguments

To create a new argument, you can mouse double left click on the canvas or click on the "Add Argument" button. The keyboard shortcut is Ctrl+X. By typing "arg(argument name)." in the text space and pressing "Load AF from text" we can create arguments.

We can find an argument in large argumentation frameworks on the canvas by clicking "Edit $->$ Argument $->$ Find Argument" or by key shortcut Ctrl+F. Press Ctrl+A to select all arguments on the canvas. Press Ctrl+Shift+A to deselect all arguments.

To delete an an argument we can select the argument and by mouse right click to obtain the menu to delete that particular argument or by Ctrl+backspace and enter the argument name that you need to remove or you can also delete by going to menu "Edit $->$ Argument".
Note: Ctrl+backspace may not work on Mac systems.

C.2 Create/Delete Attacks

To create a new attack, you can mouse middle click on the source argument and again middle click on the target argument or click on the "Add Attack" button to enter the source argument name and target argument name. The keyboard shortcut is Ctrl+E. By typing "att(argument name1, argument name2)." in the text space and pressing "Load AF from text" we can create attacks.

To delete an an attack we can select the attack and by mouse right click to obtain the menu to delete that particular attack or by Ctrl+Z and enter the source argument name and target argument name that you need to remove or also delete by going to menu "Edit $->$ Attack".

D Layout of Argumentation framework

Graph layout for argumentation framework:

- Random

- Random circle
- Force Layout
  - Spring Layout (EADES)
  - Fruchterman and Reingold

To remove the guide circles created from random circles click "Graph Layout $->$ Remove Layout Guidelines". The shortcut key is Ctrl+5.

E Displaying Solution as Text

The procedure for getting the solution, after creating or loading the apx file:

- Click Save or Ctrl+S (If AF not uploaded or altered)
- Select semantics from combo (In order to add more semantics add more semantics to the StringList in the class MainWindow .cpp file add semantics to line 702 of the code)
- Click Aspartix.

The argumentation framework can be edited until Aspartix is clicked. After creating the argumentation framework we can save our text as a temporary file by pressing "Save" on the toolbar or menu File. The shortcut key is Ctrl+S. After clicking Aspartix we get the solution in the text box.

Now that we have the solution, which is visible in the text box. We can still rollback to the argumentation framework by clicking "reloadAF" and continue editing the argumentation framework.
Note: Skip first step i.e. Save when argumentation framework uploaded by Ctrl+O, but Save if some changes have been made to the argumentation framework.

F Visualizing and Analyzing

After the Solution sets are visible in the text box on clicking the "Aspartix" button. The procedure for visualizing is as follows:

- Click on "Visualize" button,
- Visualize by Credulous Acceptance
  - By Color
  - By Shape
- Visualize by Skeptical Acceptance by Color
- Visualize by Degree of Acceptance

  - By Size
  - By Circular Layout
  - By Linear Layout

- Iterative Visualizing by Coloring the Arguments.
- Visualizing by Searching an Argument.

Visualizing by searching an argument can be done by right clicking on any argument, clicking the "Get extensions" button, entering the argument name and selecting an extension to visualize.

"Visualize" button uses the data present in the text box i.e. The data retrieved from aspartix.

G Others

To zoom in or zoom out use the "Ctrl+Minus" keys and "Ctrl+Shift+Plus" keys respectively or use the mouse scroll bar. To rotate enter the amount in the rotate combo or simply press up arrow or down arrow to rotate , the canvas rotates in a circular manner. To view the number of extensions an argument is present right click on an argument.