ARGUMENTATION DESIGNER

USER MANUAL

version 1.0 – jan 2026

A web-based tool for building, visualising, and analysing
abstract and quantitative ***argumentation frameworks***.

It combines an interactive graph editor, an APX-like textual interface,
and services to compute both extension-based and gradual semantics.

WEB APP:

argumentation.dimes.unical.it

UNIVERSITÀ DELLA CALABRIA

INDEX

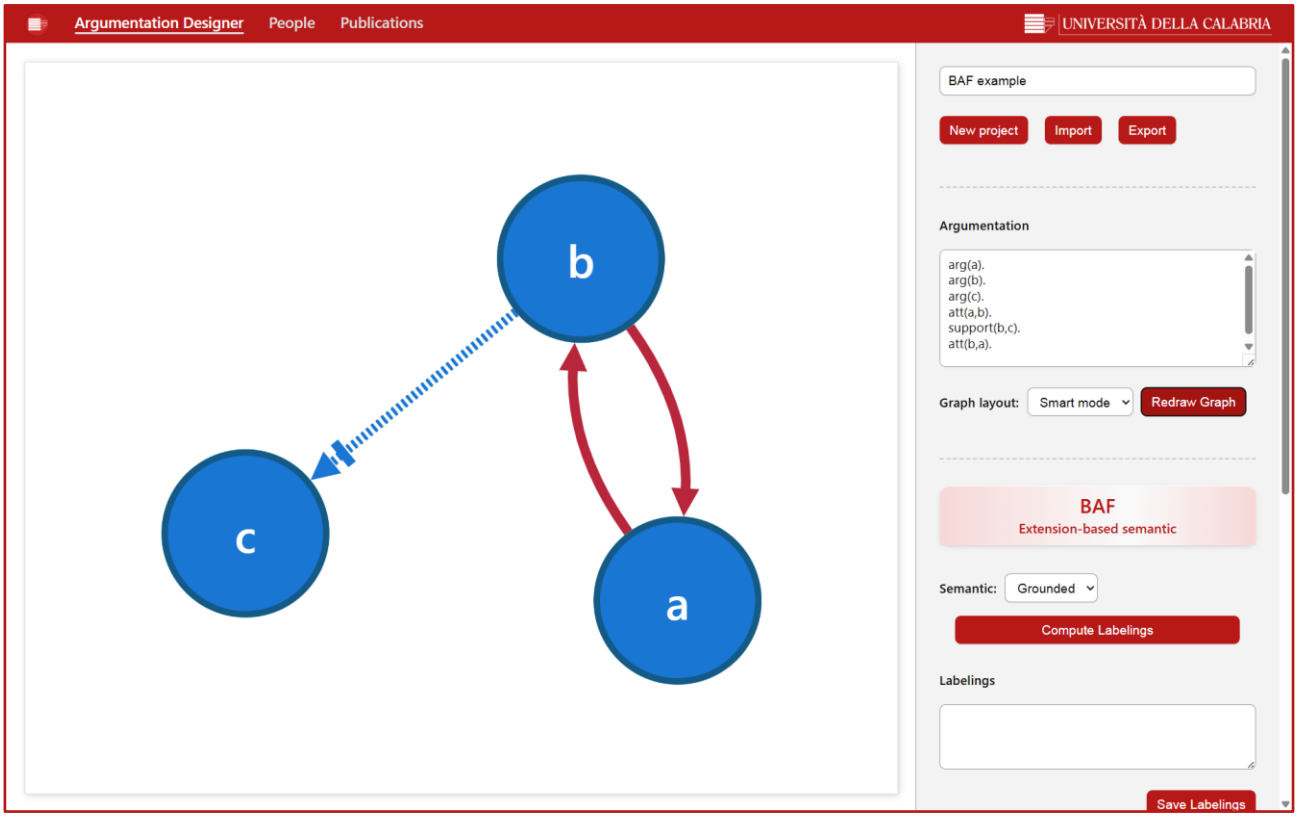
INTRODUCTION

The **Argumentation Designer** is a web-based tool for modeling and analysing *argumentation frameworks*. It allows users to create arguments and relationships visually as a graph, and to compute both extension-based and gradual semantics on top of that structure.

The application is designed for researchers, students, and practitioners who work with formal argumentation and need an interactive environment for experimentation, teaching, or demonstration. Typical use cases include building small example frameworks for lectures, exploring the effect of adding or removing arguments or attacks, and exporting results for inclusion in papers or presentations.

The **Argumentation Designer** runs entirely in the browser and requires only a modern web browser with *JavaScript enabled*. No installation is needed: the tool is accessed via the URL provided by the Università della Calabria and loads all required libraries directly from the web.

The main interface is divided into two parts: a central workspace showing the ***argumentation graph*** and a right sidebar with controls and textual descriptions. The workspace displays arguments as nodes and relationships as directed edges, while the sidebar provides access to project management, argumentation description in APX-like syntax, layout selection, semantic computation, and result visualization.



1. Main interface of the Argumentation Designer, showing the graph workspace on the left and the control sidebar on the right.

STARTING A NEW PROJECT AND MANAGING FILES

The **Argumentation Designer** organises the work around projects. A project is defined by a graph (arguments and relationships), a textual description, the chosen layout, and a set of computed results, all associated with a project name. The project name appears at the top of the sidebar and is used as a default file name when exporting data or images.

To start a new project, press the “New project” button. This command clears the current graph, removes any computed results, resets the description area, and restores default settings for layout and semantics.

After creating a new project, you should enter a meaningful title in the “Project name” field before exporting data or images.

Existing projects can be loaded from files using the “Import” button. The application accepts two formats: APX-like description files (.apx), which contain only the textual encoding of arguments and relationships, and JSON files (.json), which store full graph metadata including node positions, descriptions, layout choice, and project name. When importing, the interface is cleaned, the project name is automatically derived from the file name, and the graph is reconstructed in the workspace.

Projects can be saved using the “Export” button, which opens a file save dialog or a simplified prompt depending on browser support. You can choose among three export types: APX description (only the textual specification), JSON full data (for later re-import with graph layout and metadata), and PNG image (a high-resolution screenshot of the current graph).

MODELING THE ARGUMENTATION GRAPH

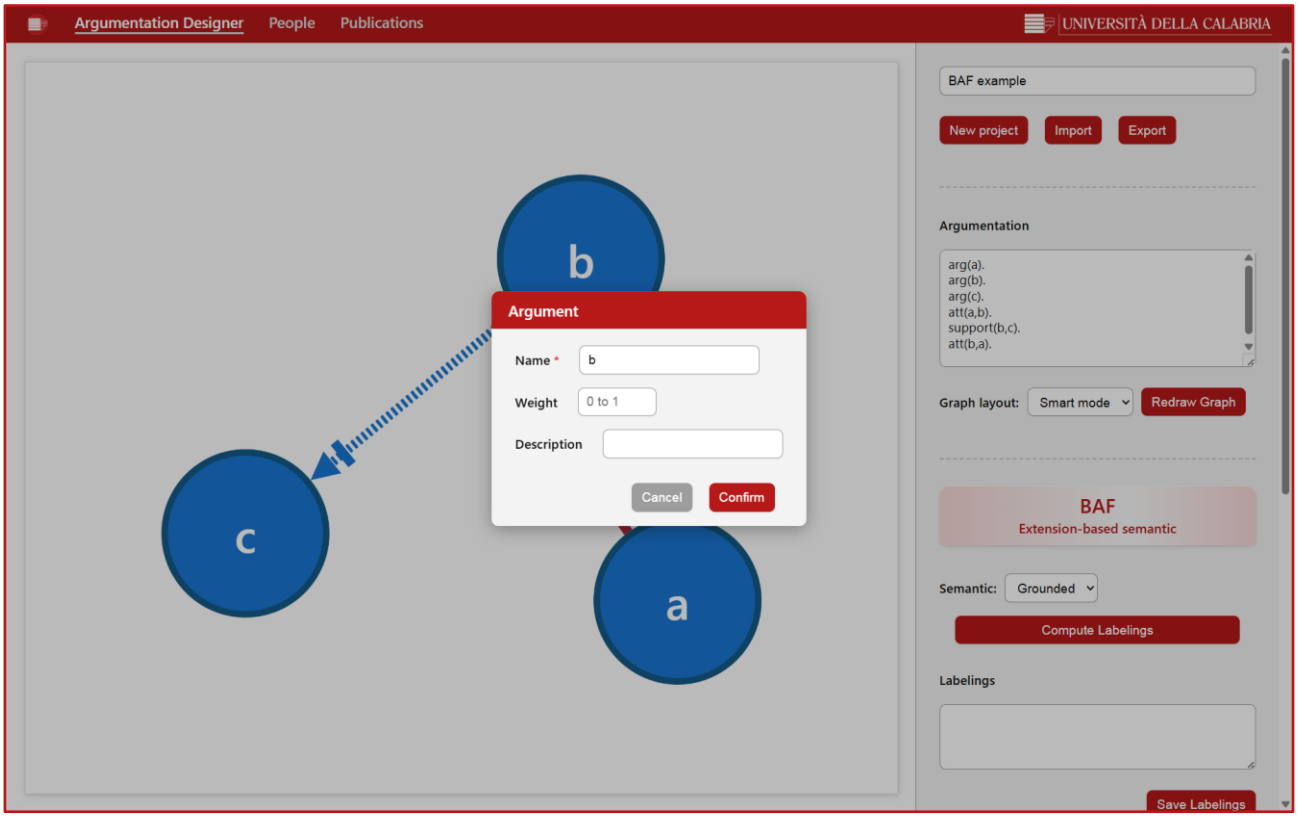
The central workspace displays the *argumentation framework* as a directed graph, where nodes represent arguments and edges represent relationships between them.

The graph can be freely panned and zoomed with standard mouse or trackpad gestures.

New arguments are created by clicking on an empty area of the workspace. This action opens the “Argument” dialog, where you can enter the *argument name*, an optional *weight* between 0 and 1, and an optional textual *description*. When you confirm the dialog, a new node is added at the clicked position and its label is drawn inside the node, with weight value shown in a compact layout.

Existing arguments can be edited or removed via the node context menu. By right-clicking on a node you open a small menu with “Edit” and “Delete” commands: “Edit” reopens the “Argument” dialog pre-filled with the current data, while “Delete” removes the node and all its incident edges.

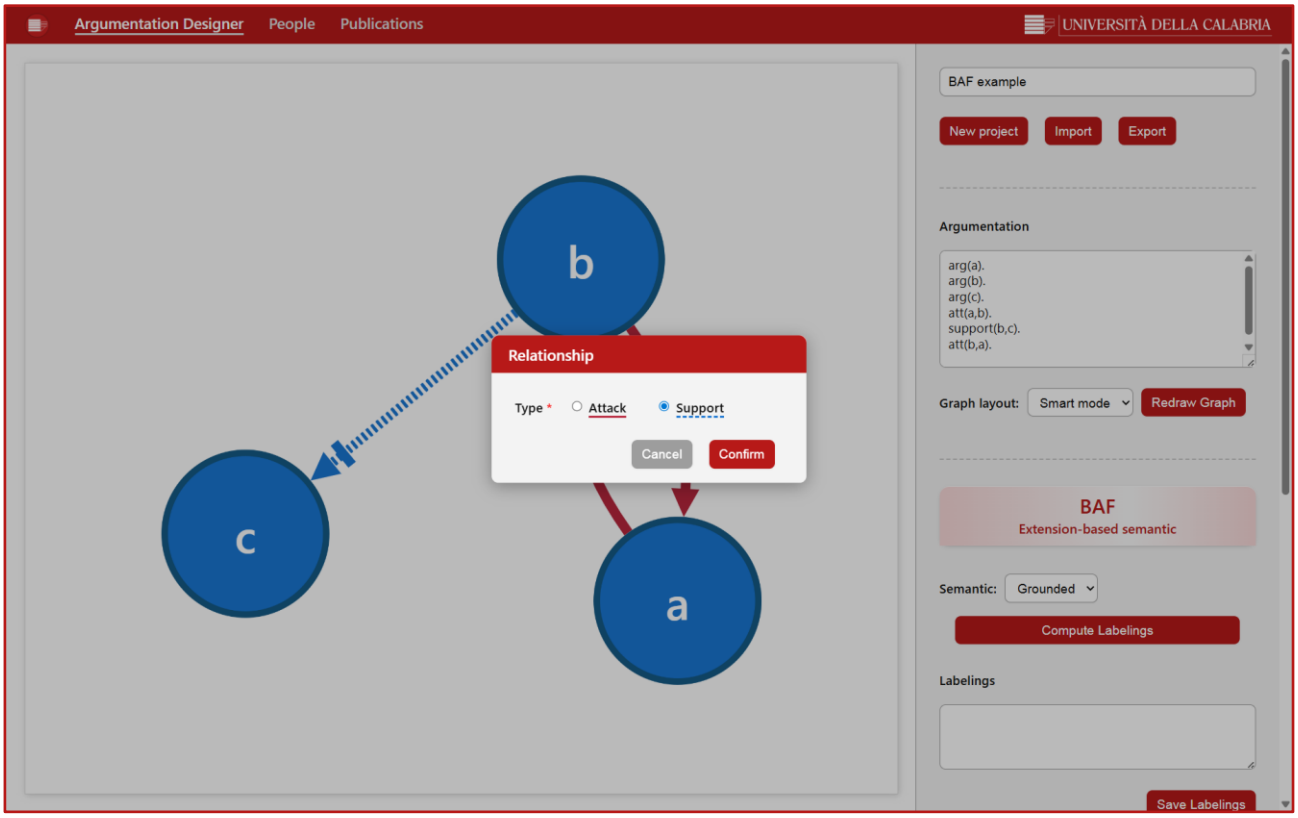
When renaming a node, all incoming and outgoing edges are automatically updated so that the structure of the framework is preserved.



2. "Argument" dialog used to create or edit a single argument, with fields for name, weight, and description.

Relationships between arguments are modeled as directed edges. To create a new edge, click first on the source node and then on the target node; during this two-step interaction a temporary arrow is displayed to preview the link. After selecting the target, the "Relationship" dialog appears and allows you to choose the *type* (Attack or Support). Confirming the dialog creates the edge and draws it with a solid line for attacks or a dashed line for supports.

Existing relationships can also be edited or deleted using the edge context menu. Right-clicking on an edge opens a small menu with "Edit" and "Delete": "Edit" shows the "Relationship" dialog with the current type, while "Delete" removes the edge from the graph.



3. "Relationship" dialog used to create or edit an attack or support between two arguments.

The workspace continuously updates node and edge shapes so that changes to names, types, or weights are immediately visible.

When you move the mouse over a node that has a *description*, a small tooltip appears near the cursor showing that description. **NOTE**: These descriptions are an additional annotation provided by the tool and are not part of the standard AF/APX format, so they are not included when exporting to APX files; they are saved only in JSON exports, which preserve all node metadata.

TEXTUAL ARGUMENTATION AND GRAPH SYNCHRONISATION

The sidebar contains an "Argumentation" textarea where the current framework is represented in an APX-like textual format. Each argument is written on a separate line using the form **arg(a).** or **arg(a,w).**, where **a** is the argument identifier and **w** is an optional weight between 0 and 1. Each relationship is written as **att(a,b).** or **att(a,b,w).** for attacks, and **support(a,b).** or **support(a,b,w).** for supports, where **a** is the source argument, **b** is the target, and **w** is an optional weight.

NOTE: edge weights are currently accepted and stored by the system but are not yet used in the semantic computations, and future releases will extend the tool to fully support *weighted argumentation frameworks*.

Whenever the graph changes in the workspace (for example when you add, edit, or delete nodes and edges), the textual description is automatically rebuilt. The tool iterates over the current set of arguments and relationships and writes one line per element using the syntax above, ensuring that the description in the textarea is always consistent with the graphical model. This makes it easy to inspect or copy the APX-like encoding of the framework at any time.

You can also edit the textual description directly and then press the “Redraw Graph” button to update the visual graph. The tool parses each line, validates the syntax and value ranges, checks that all edge endpoints correspond to declared arguments, and reports any errors in a dedicated message area. If the description is valid, the graph is rebuilt: obsolete nodes and edges are removed, new ones are added, existing ones are updated, and the selected layout is re-applied so that the workspace and the textual representation remain synchronised.

GRAPH LAYOUT

The “Graph layout” section in the sidebar lets you choose how the argumentation graph is arranged in the workspace. The default layout is “Smart mode”, which automatically selects a force-directed arrangement optimised for readability so that nodes and edges are distributed in a way that reduces overlaps and crossing lines.

In addition to Smart mode, several fixed layouts are available. “Circle” places all arguments on a circle, “Concentric” groups nodes in concentric rings according to their connectivity, “Grid” arranges them on a regular grid, “Breadthfirst” produces a layered layout that is useful for hierarchical structures, and “Random” scatters nodes uniformly at random.

Whenever you change the layout from the dropdown menu, the graph is recomputed and redrawn using the chosen algorithm. The selected layout is stored together with the project, so that importing a JSON file restores not only the nodes and edges but also the layout that was active when the project was saved.

FRAMEWORK TYPES AND SEMANTIC GROUPS

The Argumentation Designer automatically detects the type of argumentation framework represented by the current graph based on the presence of support relations and weights.

When only attacks are present and no weights are used, the system recognises a standard **Argumentation Framework (AF)**. When both attacks and supports are present but no weights are used, it recognises a **Bipolar Argumentation Framework (BAF)**.

When edge weights are used without support relations, the system interprets the graph as a **Weighted Argumentation Framework (WAF)**, where attacks carry numerical weights. When both support relations and edge weights are present, it corresponds to a **Weighted Bipolar Argumentation Framework (WBAF)**.

When node weights are used (with or without edge weights), the framework is treated as a **Quantitative Bipolar Argumentation Framework (QBAF)**, and in the presence of both support relations and edge weights this extends to a **Weighted Quantitative Bipolar Argumentation Framework (WQBAF)**.

The current framework type is shown in the semantic section title and determines which group of controls is visible.

For AF, BAF, WAF, and WBAF, the tool activates the **extension-based semantics** panel, while for QBAF and WQBAF it activates the **gradual semantics** panel.

Switching between these configurations happens automatically whenever you add or remove supports or weights in the graph.

EXTENSION-BASED SEMANTICS (AF, BAF, WAF, WBAF)

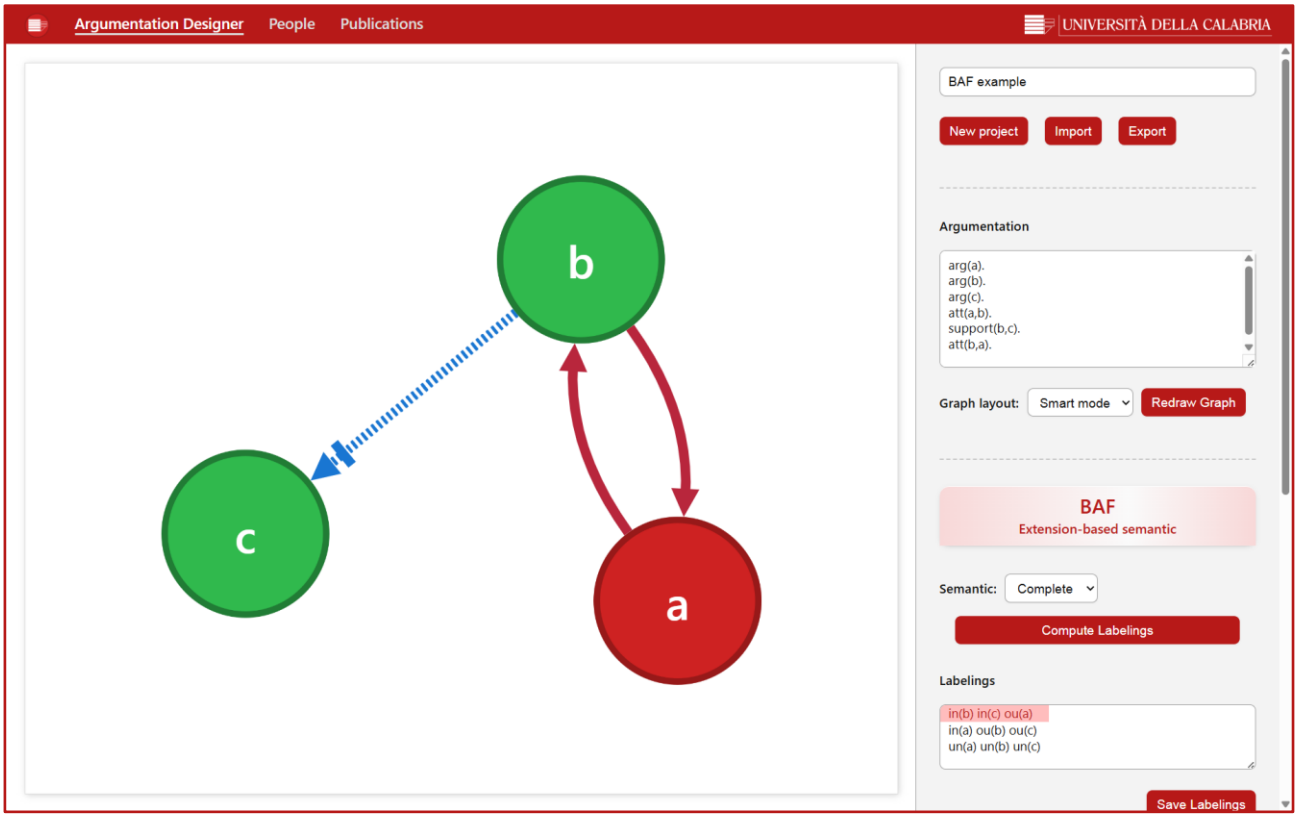
When the framework is of type AF, BAF, WAF, or WBAF, the “Extension-based semantic” panel becomes active.

In this panel you can select one of the standard Dung-style semantics (Grounded, Complete, Preferred, or Stable) from the dropdown menu and press the “Compute Labelings” button to request the corresponding labelings from the external reasoning service.

The results are shown as a list of labelings, each entry describing how arguments are classified (typically into sets such as “in”, “out”, and “undecided”).

You can select individual labelings from the list to inspect them, and use the “Save Labelings” button to export all computed labelings to a text file for later analysis or inclusion in other tools.

When you select a single labeling in the list, the graph is coloured accordingly: arguments labelled “in” are shown in green, arguments labelled “out” are shown in red, arguments labelled “undecided” are shown in yellow, so that the chosen extension is immediately visible in the workspace.



4. Extension-based semantics panel with a computed labeling selected and the graph coloured according to IN, OUT, and UN labels.

CONSTRAINTS, PREFERENCES, AND FILTERED LABELINGS

Below the main labelings list there is a “Constraints” area where you can specify simple conditions on the desired labelings, such as requiring that some arguments must be accepted or rejected.

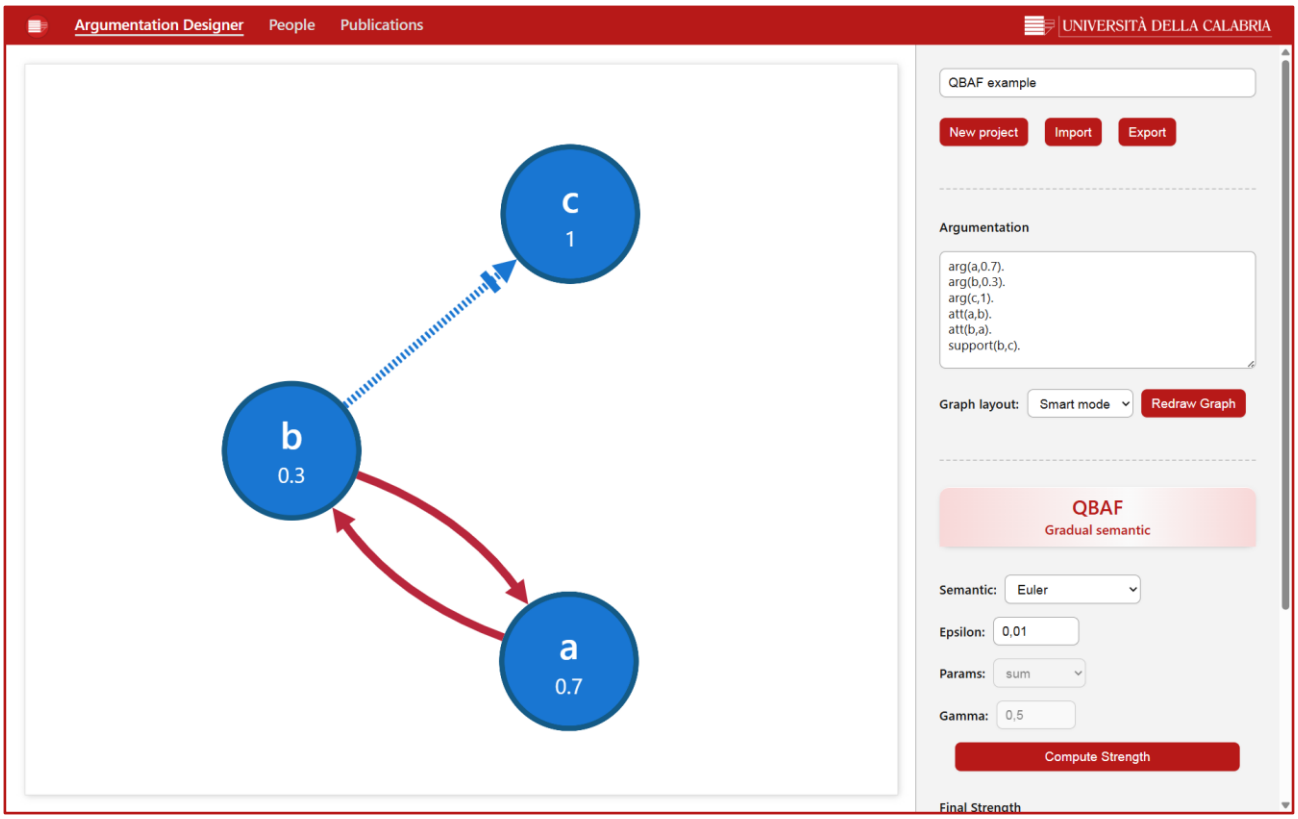
After writing the constraints, you can press the “Apply Constraints” button to ask the reasoning service to filter the previously computed labelings accordingly.

The filtered labelings are displayed in a separate list, which contains only those labelings that satisfy the specified constraints (and, in future versions, preferences).

You can then use the “Save Filtered Labelings” button to export this restricted set of labelings to a text file, which is useful when you want to focus on a subset of solutions that meet particular requirements.

GRADUAL SEMANTICS (QBAF, WQBAF)

When the framework uses node weights (with or without edge weights), the “Gradual semantic” panel becomes active and the system interprets the graph as a Quantitative Bipolar Argumentation Framework.



5. Gradual semantics panel for QBAF frameworks, showing the semantic selector.

In this panel you can choose one of several gradual semantics from the dropdown menu, including DReLU, DDReLU, Euler-based, DFQuAD, MLP-based, and Quadratic Energy, each providing a different way to aggregate supports and attacks into numerical strengths.

For gradual semantics you can configure additional parameters before running the computation. The “Epsilon” field controls the convergence or precision of some iterative methods, the “Params” dropdown selects how contributions (for example sum, max, deltasum, deltamax) are combined, and the “Gamma” field adjusts the sensitivity of the aggregation; some semantics ignore one or both of these parameters, and in that case the corresponding controls are automatically disabled.

After selecting the semantic and setting the parameters, you can press the “Compute Strength” button to send the current description and options to the reasoning service. The response is shown in the “Final Strength” area as a list of argument–value pairs and is also applied directly to the graph:

each node receives a numerical strength in the range from 0 to 1 and is coloured according to a predefined strength scale, allowing you to visually compare how strongly each argument is supported in the current framework.

ERROR MESSAGES AND TROUBLESHOOTING

The tool performs several validation checks when you edit the graph or the textual description. In the “Argument” and “Relationship” dialogs it verifies that argument names are present, within the allowed length, and use only permitted characters, and that any provided weights lie within the allowed numerical range; if a check fails, an error message is displayed under the dialog and the offending field is highlighted so that you can correct it before continuing. Similar checks are applied when you press “Redraw Graph” after editing the textual description: the parser reports unknown syntax, invalid weights, or edges that refer to non-existent arguments in a dedicated error area next to the description.

When interacting with the reasoning services, additional error messages may appear in pop-up alerts. If the backend returns an error, the tool shows the message received from the server (for example, when the description is inconsistent with the expected input format), while network problems or unreachable services are reported as API or network errors. In these cases, you should first check that the description is syntactically correct, then verify your internet connection, and, if the problem persists, contact the system administrator with the text of the error message.

PROJECT INFORMATION AND CREDITS

The **Argumentation Designer** is developed within the ***PROGETTO FAIR - SPOKE 9”, codice identificativo PE_00000013, CUP H23C22000860006, Piano Nazionale di Ripresa e Resilienza, Missione 4, Componente 2, Investimento 1.3, finanziato dall’Unione Europea, NextGenerationEU***.

The project is funded by the European Union under the NextGenerationEU programme.

For further information on the research team and related scientific publications, you can use the “People” and “Publications” links in the top navigation bar, which point to dedicated pages describing the contributors and the main results associated with the tool.