Carneades User Manual

Tom Gordon

October 25, 2012

Contents

1 Getting Started

The Carneades argumentation system is open source software, freely available for downloading at http://carneades.github.com.

Carneades provides web-based, collaborative software tools for:

- summarizing the arguments of a debate in an *argument graph*
- visualizing, browsing and navigating argument graphs
- critically evaluating arguments
- forming opinions, participating in polls and ranking stakeholders by the degree to which they share your views
- obtaining clear explanations, using argument graphs, of the different effects of alternative policies in particular cases

In this manual, we distinguish three kinds of users:

Public Users Users interested in understanding arguments and policies, whom we assume have had *no specialist training* in argumentation or policy modeling.

Analysts Users who have had *specialist training* in argumentation or rule-based policy modeling.

System Administrators Users with sufficient computer skills to install, configure and administer the Carneades software.

These are only roles. Some people may be able to use the system in more than one of these roles. In particular, we aim to make the software simple to install and administer, so that ultimately anyone with basic computer skills will be able to serve as "system administrator".

The rest of this manual is organized in the following chapters:

- The Home Page
- Argument Graphs

- Browsing, Visualizing and Evaluating Arguments
- Forming, Polling and Comparing Opinions
- Analyzing and Comparing Policies
- Editing Argument Graphs
- Modeling Policies and Argumentation Schemes
- System Administration
- Application Scenarios
- Built-in Argumentation Schemes

The first two of these chapters, about the Carneades home page and argument graphs, are recommended to be read first by all users. The home page chapter explains how to access and navigate among the various tools provided by the system. The argument graph chapter provides a concise overview of the underlying data model used by all the tools. The next three chapters are for public users and explain how to use the tools for argument browsing, opinion polls and policy analysis, respectively. The chapters on editing argument graphs and modeling policies and argumentation schemes are for analysts with some prior knowledge of argument reconstruction methods and rule-based systems. The system administration chapter is recommended reading for anyone wishing to install, configure and administer the Carneades software. The chapter on application scenarios may be of interest to managers and public administrators looking for ways to improve the quality of decision-making processes in their organizations. The final chapter presents the initial set of argumentation schemes distributed with the system, which you can modify to suit your needs.

2 The Home Page

The home page provides an overview of the features and tools of the system along with the main menu for accessing these tools. The main menu is always available, on every page of the Carneades system, to make it easy to navigate directly from one tool to another at any time.

The items in the main menu are:

- Home
- Argument Graph
- Poll
- Policy Analysis

- System Administration

The "Home" menu item takes you back the home page, described here.

The "Argument Graph" menu item takes you a description of the debate and a hypertext outline of the argument graph model of the debate. The argument graph page includes a submenu of commands for generating argument maps, evaluating arguments, viewing argumentation schemes, exporting argument maps to XML and, if you are logged in as an analyst, for editing the argument graph.

The "Poll" menu item takes you to the opinion formation, polling and comparison tool, where you can moderate a virtual debate of the issues. In return, you will receive a guided tour of the issues and arguments, to help you to formulate your own opinion. The tool also provides you with a way anonymously share your opinion with other users. The polling results are aggregated by the system and used to help you to evaluate the acceptability of claims. Finally, the tool compares your opinions with the opinions expressed in the published documents used to create the argument graph, for example the opinions of political parties, and ranks the authors by the degree to which their opinions agree with yours.

The "Policy Analysis" menu item takes you to a tool for analyzing and comparing the effects of various policy proposals on example cases. In a dialog with the system, you will be asked a serious of questions to gather information about a real or hypothetical case. An argument graph will be generated, by applying the rules of the policy proposals to the facts of the case, showing how the various policies would work in the given case. The tool can help you to find the policies which best serve your interests. Here too your privacy is protected. Care is taken to not request or store information that could be used to reveal your identity.

Finally, the "System Adminstration" menu item takes you to a page for managing and configuring the Carneades server. You will need to log in with a user name and password. Here you can create, edit, archive and delete projects, where each project consists of an argument graph, a set of argumentation schemes, models of proposed policies, the test cases and possibly local copies of source documents.

The current project, the one being accessed and used from the user interface, is identified by a parameter in the Web address (URL) used to access the system. Only system administrators have access to the list of all projects served by a particular installation of the Carneades system.

3 Argument Graphs

Argument graphs model relationships among arguments, statements and source documents. To build the model, source documents need to be interpreted to

find the arguments, in a process called "argument reconstruction".

Consider the following simple legal argument: Johnny violated the law by riding his skateboard in the park.

This same argument can be expressed in many different ways. Here are some examples:

1. Because Johnny rode his skateboard in the park he violated the law.
2. Vehicles are prohibited in the park. Someone who does something which is prohibited violates the law. Johnny rode his skateboard in the park. A skateboard is a vehicle. Therefore Johnny violated the law.
3. Johnny hat gegen das Gesetz verstoßen, weil er sein Skateboard im Park benutzt hat.

The first example just reordered the premise and the conclusion of the argument, putting the premise first. The second example reveals some implicit premises of the original formulation of the argument. The third example is a German translation of the original formulation of the argument.

All four of these texts, including the original formulation, express the *same* argument, but in different ways. In a large-scale debate, for example about European policy issues, the same argument might be expressed in *thousands* of different ways in many different languages. (The European Union has 23 official languages.)

One of the purposes of argument graphs is to provide a way to summarize complex debates with exactly one node in the graph for each argument put forward in the debate. A single argument graph is used to represent all the arguments put forward in a debate, from all participants. The nodes can quote one or more source documents, and include links to these source documents, so no information need be lost and all voices can still be heard, inclusively. Grouping the different formulations of an argument together into a single node in the graph, abstracting away details, makes it possible to quickly obtain an overview of the arguments and to obtain a clearer picture of relationships among arguments. A hypertext or map of the source documents directly, without an argument graph, would make it difficult to "see the forest for the trees".

3.1 Data Model

The entity-relationship diagram above shows the elements of argument graphs and their connections. (The figure does not visualize a particular argument graph, but rather relationships between the elements of argument graphs in general.)

The two main elements of argument graphs are statements and arguments. Statements represent propositions, claims and assertions. Arguments represent simple inferences from one or more premises to a single conclusion. Again, there should be only one statement or argument in the graph for each statement and argument in the source documents, no matter how many different ways the statement or argument has been expressed in source documents. Some or all formulations of the statement or argument can be quoted or referenced in the metadata of the statement or argument node. See the discussion of metadata below for further information.

As can be seen in the entity-relationship diagram, arguments are linked to statements in two ways in argument graphs. Each argument has exactly one conclusion, which is a statement, and zero or more premises, where each premise has exactly one statement node. A statement may the conclusion or premise of more than one argument.

A statement may be both a conclusion and a premise, resulting in complex argument graphs, representing chains or trees of reasoning. Argument graphs may contain cycles. A simple cycle would result if a statement and a premise of the same argument. There are methods for resolving these cycles when evaluating argument graphs.

A statement in an argument graph represents both a proposition and the negation of the proposition. To continue with our example, the sentences "Johnny rode his skateboard in the park" and "Johnny did not ride his skateboard in the park" would be represented by a single statement in an argument graph. Conclusions and premises of argument can negated using con arguments and negative premises, respectively. That is, there are two kinds of arguments, pro and con. An argument is pro if its conclusion claims the statement is true and con if it claims the statement is false. Similarly, there are two kinds of premises, positive and negative. A positive premise holds if its statement is presumably true ("in"). Conversely, a negative premise holds only if its statement is presumably false ("out").

Prior models of argument graphs do not distinguish pro and con arguments or positive and negative premises. Rather, in these prior approaches all argument nodes are pro and all premises are positive. Our approach has the advantage of reducing the number of statements up to 50%, resulting in more compact summaries of the arguments.

3.2 Statement Properties

id A Uniform Resource Name (URN) serving as a unique identifier for the statement, world-wide.

text A concise formulation of the statement, written by the analyst who reconstructed the arguments from the source documents. Paraphrases the

various formulations of the statement in the sources. Translations of the text in several languages may be included in the model. Compare with the "description" property of the metadata of the statement, which can be used to quote some or all of the formulations of the statement in the sources and provide translations in several languages.

weight A real number in range of 0.0-1.0 representing the degree to which the statement is accepted as true by the users, where 0.0 means the statement is *rejected* (believed to be false by the users) and 1.0 means the statement is *accepted* (believed to be true). This information is collected via polls.

proof standard The method used to combine pro and con arguments. Several proof standards are supported by the system. For most purposes, the "preponderance of the evidence" standard should suffice. See the Editing Argument Graphs Chapter for details.

value A real number in the range 0.0-1.0, storing the output of the argument evaluation process, where 0.0 means the statement is *out* (presumably false), 1.0 means the statement is *in* (presumably true) and all other values mean the arguments are insufficient for making any presumptions about the truth or falsity of the statement.

atom An optional formal representation of the statement in predicate logic. (This feature is for analysts and need not interest public users.)

main A Boolean value (true or false) used to indicate whether the statement is one of the main issues of the debate modeled by the argument graph.

3.3 Argument Properties

id A Uniform Resource Name (URN) serving as a unique identifier for the argument, world-wide.

direction Pro or con.

strict A Boolean value (true or false) expressing whether the conclusion of the argument is necessarily true when its premises are true (strict arguments) or only presumably true. Nonstrict arguments are called "defeasible" arguments.

scheme The name of the argumentation scheme applied, if any. Optional. Examples: "argument from credible source", "argument from practical reasoning".

weight A real number in range of 0.0-1.0, representing the relative weight of the argument, compared to other arguments pro and con the conclusion of the argument. This information is collected via polls.

value A real number in in the range 0.0-1.0, used to record the output of the argument evaluation process, where 0.0 means the argument is *out* (not acceptable), 1.0 means the statement is *in* (acceptable) and all other values mean the arguments in the graph, taken together, are insufficient for determining the acceptability of this argument.

3.4 Premise Properties

polarity Positive or negative.

role The role of the premise in the argumentation scheme applied. Examples: "minor", "major".

implicit A Boolean value (true or false). Can be used to note that the premise was not explicit in the source documents from which the argument node was reconstructed.

3.5 Metadata

The argument graph as as whole, as well as each of its statements and arguments, can be annotated with metadata, using the Dublin Core. There are 15 elements in the Dublin Core. Each element may zero or more values. Here is a list of the Dublin Core elements:

1. Title
2. Creator
3. Subject
4. Description
5. Publisher
6. Contributor
7. Date
8. Type
9. Format
10. Identifier
11. Source
12. Language
13. Relation

14. Coverage
15. Rights

See Dublin Core for a detailed description and usage guidelines for each element. The Dublin Core is intended to be useful for describing a wide range of "resources" on the World-Wide Web. Not all of the elements may be applicable for argument graphs.

In addition, Carneades allows each metadata record to be assigned an optional "key", a string which can be used as a label to refer to the metadata record, such as "BenchCapon:2008", similar to the way citation keys are used in bibliographic databases such as BibTeX.[1] At most one key should be provided.

Carneades provides special support for providing description elements of the Dublin Core in multiple languages (English, German, French, ...) and for formatting these descriptions using the Markdown language. This feature can be used to include quotations from and links to source documents in the descriptions of both statements and arguments.

4 Browsing, Visualizing and Evaluating Arguments

This chapter of the Carneades user manual explains how to:

- Access the argument graph page for viewing and editing argument graphs on the World-Wide Web.
- Use hypertext in web pages to browse an argument graph.
- Visualizing argument graphs in diagrams, called "argument maps", and using these maps to navigate to more detailed views of statements and arguments.
- Evaluate arguments to reveal missing premises, check the form of arguments, ask critical questions and assess the acceptability of statements.
- Export an argument graph to XML, to archive the graph or process it using other software.
- Generate outlines of the arguments in a graph, for further editing using text editors or word processors.

[1]

4.1 The Argument Graph Page

The user interface of Carneades is a web application. You access the pages and views of the user interface with web addresses, called Uniform Resource Locators (URL), just like you access any resource on the World-Wide Web. Most of the time you will access the application by clicking on a link embedded in some page on the Web, for example in a news article, blog entry or e-participation web site. If you are using Carneades as a stand-alone, desktop application, these URLs will be local addresses, from the "localhost" domain, pointing to web pages served by the application on your personal computer.

Figure 1: An Argument Graph Page

An argument graph page consists of the following parts:

- The *title* of the argument graph. This title usually includes the topic of the discussion or debate.

- A *menu bar* of commands. The commands shown depend on the role of the user. Public users, who need not login to the system, are shown the commands "export" and "map", for exporting the argument graph to XML and viewing an argument map visualization of the graph, respectively. Analysts, who must login to the system with a password, are also shown "new statement" and "new argument" commands. Only analysts may modify the argument graph.

- A *description* of the topic of the discussion modeled in the argument graph. The description may be available in several languages. The user interface provides a way to select and change your preferred language during the session (*Note: not yet implemented*). The description can be arbitrarily long and include multiple sections, paragraphs, images, hyperlinks, lists and other content.

- A list of the *main issues* of the discussion. Each item in the list is linked to a page providing detailed information about the statement in the argument graph at issue.

- An *outline* of the top five levels of the arguments in the argument graph.The first level of the outline lists the main issues (again). The second level lists the arguments pro and con each issue. The third level lists the premises of each of these arguments. The fourth level lists the argument pro and con each premise. Finally, the fifth level lists the premises of these arguments. Deeper levels of the argument graph can be navigated to by first clicking on a statement or argument in the outline and then following the links on the next page. Since argument graphs may contain cycles and are not restricted to trees, some items may appear multiple times in the outline.

- A list of *references* to the source documents used to construct the argument graph. For documents available on the Web, the reference will include a hyperlink to the source document.

4.2 Using Hypertext to Browse an Argument Graph

There is a web page for each statement and argument in the argument graph providing detailed information about the element along with links to related statements and arguments in the graph. You can use these pages to navigate from node to node in the argument graph, by simply clicking on the links in the usual way. To go back to previous pages, use the back button of your web browser.

Figure 2: A Statement Page

4.2.1 Statement Pages

The top of the statement page displays the properties of the statement: its id, atom, whether or not it is a main issue, its proof standard, usually "pe" (preponderance of the evidence), its weight and value. The other proof standards available are "dv" (dialectical validity), "cce" (clear and convincing evidence), and "brd" (beyond reasonable doubt). See the section on Evaluating Arguments for further details about proof standards.

The next section displays the *text* of the statement. This formulation of the statement is written by the analyst or analysts who reconstructed the arguments to build the argument graph.

If metadata had been provided for the statement, it would be displayed next. Descriptions may be entered, by analysts, in multiple languages. The description, if available, will be displayed using the language chosen by the user. If no description has been entered manually by analysts for the selected language of the user but a description is available in some other language, a translation service will be used to generate a description in the selected language (*Note: not yet implemented*).

Finally, the statement pages lists pro and con arguments about the statement, i.e. arguments having this statement, or its negation, as a conclusion, as well as arguments which have this statement, or its negation, as a premise. The premises of the pro and con arguments are also listed. This makes it possible to navigate to nearby arguments and statements in the argument graph, by simply clicking on the links in these lists. Use the back button of your web browser to return to this statement page.

4.2.2 Argument Pages

Argument pages are quite similar to statement pages. The top of an argument page displays the properties of the argument: its id, the argumentation scheme applied (if any), whether it is a strict or defeasible argument, its weight and value. The argumentation scheme contains a hyperlink (*Note: not yet implemented*). Click on the link to view a description of the scheme.

If metadata had been provided for the argument, it would be displayed next. Descriptions can include quotations of one or more source texts expressing the argument, along with hyperlinks to the sources on the Web. The description, if available, will be displayed using the language chosen by the user. If no description has been entered manually by analysts for the selected language of the user but a description is available in some other language, a translation service will be used to generate a description in the selected language (*Note: not yet implemented*).

Next, the premises of the argument are listed. If available, the role of each premise in the argumentation scheme applied is shown (e.g. "major" or

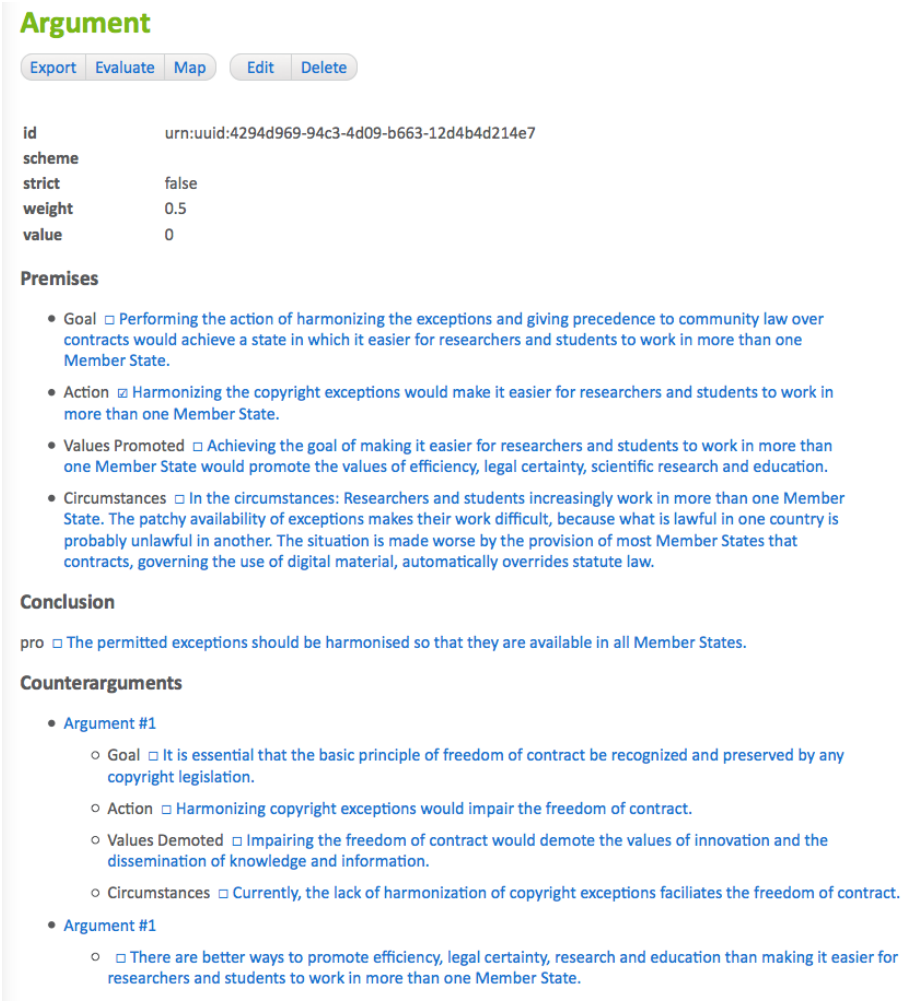

Figure 3: An Argument Page

"minor"). The check boxes to the left of each premise are used to indicate whether the statement is current *in* (checked box, meaning presumably true), *out* (crossed out box, meaning presumably false) or neither (empty box, not enough information to presume either truth or falsity), given the arguments in the graph and the opinions of users from polls about the acceptability of statements and relative weights of pro and con arguments.

After the premises, the conclusion of the argument is shown, preceded by "pro" or "con", showing the direction of the argument, and a check box showing the acceptability of the conclusion, as for the premises.

Finally, a list of counterarguments is shown.[2] The premises of the counterarguments arguments are also listed. This makes it possible to navigate to nearby arguments and statements in the argument graph, by simply clicking on the links in these lists. You can use the back button of your web browser to return to this argument page.

4.3 Visualizing Argument Graphs in Argument Maps

The menus of the argument graph page, statement pages and argument maps include a "map" button. Clicking on the "map" button generates a diagram, called an "argument map", which visualizes the argument graph as a network (directed graph) of statement nodes and argument nodes connected by links. Statement nodes are shown as boxes; argument nodes with circles and boxes with rounded corners.

For statement nodes, the text of the statement is shown inside the box, possibly truncated if the text is too long. In argument nodes, the circle is filled with a plus sign, if the argument is a pro argument, or a minus sign, for con arguments. The edges (links) between argument nodes and statement nodes show the premises and conclusion of the argument. The conclusion of the argument is the statement node pointed to by the edge with the normal arrowhead. The other statement nodes linked to the argument, without arrowheads, are its premises. Negative premises are displayed with a circular (dot) arrowhead on the statement side of the edge.

The statement and argument nodes in argument maps contain hyperlinks. Clicking on a statement or argument node displays the details of the node in a statement or argument page, respectively.

In argument maps, argument nodes whose conclusion is another argument node, rather than a statement node, visualize "undercutting" arguments. These are

[2]By counterarguments here we mean *rebuttals* (arguments with the opposite conclusion) and *undercutters* (arguments which deny the applicability of this argument). Arguments which attack a premise of this argument ("undermining" arguments), are not listed. To navigate to undermining arguments, click on the premise of the argument of interest. The undermining arguments will be listed on its statement page.

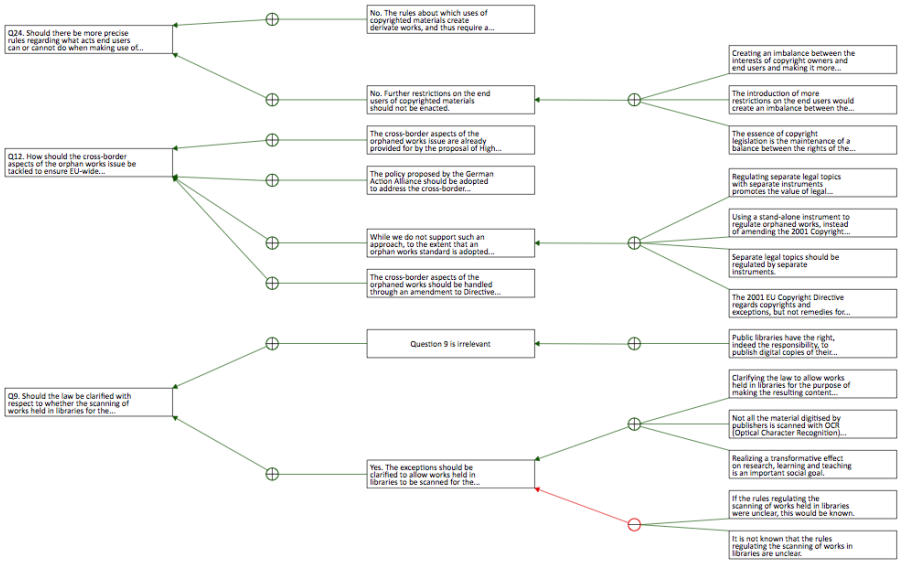

Figure 4: An Argument Map

arguments which question the applicability of another argument. This is the only case where two nodes of the same time are directly connected in the map.

Argument maps are represented using structured vector graphics (SVG) not bitmaps. You can zoom the map in or out, to any scale, without loss of resolution. How this zooming is done depends on your device and web browser.

When argument graphs have been evaluated, the status of the argument and statement nodes is visualized in argument maps using both color and icons. Nodes which are "in" are filled with a green background and contain a checked box. Nodes which are "out" are shown with a red background and contain a crossed box (a box filled with an X). Nodes which are neither in nor out are filled with white background color and contain an empty checkbox. The colors are redundant to accommodate black and white printing and color-blind users.

Argument graphs may contain cycles. However, currently the algorithm used to layout the argument and statement nodes in the map is not able to handle cycles. (*Note: This limitation will be removed in a later version of the system.*)

Argument graphs can be very large. Currently the *entire* argument graph is displayed in the argument maps. In the future only a partial view of the argument graph in maps will be shown, at least for larger graphs. The part of the graph shown will depend on the context. The map generated from the argument graph page of the argument graph will show the arguments and statements near the main issues of the map. The maps generated from statement and argument pages will show the part of the argument graph near the selected statement or

argument. A method for scrolling the maps, to bring other parts of the graph into view, will be provided.

4.4 Searching for Arguments

The argument graph page will provide access to a command for searching for arguments.

(*Note: Not yet implemented*)

4.5 Evaluating Arguments

By argument "evaluation" we mean the process of critically assessing arguments by

1. revealing implicit premises
2. validating whether the arguments are formally correct, by instantiating accepted argumentation schemes
3. asking appropriate critical questions, depending on the schemes applied
4. and determining which claims are acceptable, taking into consideration the assumptions of the users and their collective assessment of the relative weights of conflicting pro and con arguments.

The first three of these tasks can be accomplished by comparing the argument with its argumentation scheme. On the argument page, click on the argumentation scheme to view a description of the scheme (*Note: not yet implemented*). Most web browsers provide some way to open the link in a new tab, so that you can easily switch back and forth between the argument page and the description of the scheme. Now you can check whether any of the premises listed in the scheme are missing from the argument. The argument is formally valid if all the premises of the scheme are explicitly provided by matching premises of the argument and the conclusion of the argument matches the conclusion of the scheme.

Argumentation schemes define exceptions and assumptions which can be used to ask critical questions. The exceptions provide reasons for not applying the argument, undercutting it. If an exception is true, this doesn't mean that the conclusion of the argument is false, but only that the argument does not provide a good reason to presume the conclusion to be true. The assumptions of the scheme are implicit premises which need to be proven only if they are called into question. So, if you think an assumption does not hold, you should consider making an issue of it using whatever channels are available for you for

participating in the discussion. (The Carneades system does not provide this service. Its function is to provide a tool for summarising and understanding arguments, wherever they take place.)

The assumptions of the users and their collective assessment of the relative weights of conflicting pro and con arguments are collected and aggregated using the opinion polling tool described in the opinion polling chapter of this manual. At any time, an analyst can execute the "evaluate" command, shown in the menu bars of the argument graph page, and the statement and argument pages, to compute the acceptability of the arguments and statements in the argument graph, on the basis of the information gathered from users via the polls. A statement is considered acceptable if it is *presumably true* given only the arguments modelled in the argument graph and their own assumptions and assessment of the relative weights of these arguments.[3]

If you do not agree with the result of the evaluation, there are at least three reasons why you may be right and the system's evaluation wrong:

1. Not all relevant arguments have been included in the model, or put forward in the debate. If you are aware of some missing argument, consider contributing it to the discussion yourself.

2. The collective, averaged opinion of the users who participated in the poll may be incorrect. Minority views can be correct. If you haven't yet participated in the poll, you may want to do so now.

3. The formal model of argument we are we are using to compute acceptability, based on the state of the art of the field of computational models of argument, may be incorrect. Of course, specialist knowledge is required to assess the correctness of the model. If you, like most people, do not have this knowledge, then we recommend a skeptical but respectful attitude. If you agree with the results of the model, then the model gives you a reason to have more confidence in your opinion. If you do not agree with the results of the model, then you may want to take pause to reconsider your views, even if in the end you do not change your mind.

[3] A state-of-the-art computational model of argument is used to compute the acceptability of statements. This model maps argument graphs to a so-called "Dung abstract argumentation framework" (Dung 1995), using a method based on the ASPIC+ model of structured argument (Prakken 2010), but adapted to preserve the features of the Carneades model of argument graphs (Gordon, Prakken, and Walton 2007). This ASPIC+ version of Carneades can handle cyclic argument graphs, removing the main limitation of the original version of Carneades. Several semantics are available for Dung Abtract Argumentation Frameworks. Carneades uses grounded semantics, which is the semantics most cautious (skeptical) semantics about accepting claims.

4.6 Exporting Argument Graphs to XML

The menu bar of the argument graph page and the statement and argument pages includes an "export" button. Click on this button to generate an XML file containing all the data and metadata in the argument graph, including quotations of and links to source documents.

The XML files use a schema called the "Carneades Argument Format" (CAF), documented elsewhere. (*Note: write the CAF documentation*)

These XML files can be used to transfer argument graphs from one installation of the Carneades system to another, to merge argument graphs from several sources, to archive argument graphs, or to translate argument graphs into other formats, such as the Argument Interchange Format (AIF), or the generate reports or other kinds of visualisations..

4.7 Generating Outlines

(*Note: Not yet implemented*)

5 Formulating, Polling and Comparing Opinions

The opinion formation and polling tool of the Carneades system serves two purposes:

- It guides you step by step through the arguments on all sides of a complex policy debate, in a kind of simulated debate, providing you with an overview of the issues, positions and arguments in a systematic way. The tool can help you to form your own opinion, if you don't yet have one, or to critically evaluate and reconsider your current opinion, if you do. The tool also enables you to compare your responses with the published positions of some stakeholders, such political parties, helping you to find persons and organizations which best represent or share your views and interests.
- At the same time the tool conducts a poll to collect your views and opinions on the issues, taking care to protect your privacy. You will not be asked or required to enter any personal data, such as your name or email address, which could be used to identify you or associate your answers to poll questions. The anonymous and aggregated results of the survey can provide valuable feedback to you as well as policy makers, going beyond the information provided by traditional surveys. It enables users to discover not only how much support policies enjoy, but also to learn precisely

why particular aspects of the policies, or their underlying assumptions, are supported or not.

5.1 Accessing the Opinion Formation and Polling Tool

Every page of the Carneades system includes the main menu. Click on the "poll" button of the main menu to access the opinion formation and polling tool.

The first page of the opinion formation and polling tool provides an overview of the features of the tool and explains the following procedure for using the tool:

1. Log in using a pseudonym to protect your privacy. You can login again using this name to change your responses at any time. Choose a cryptic name, like a secure password, to make it difficult for others to access or change your answers. (The system will generate a username for you, which you can use or change, as you prefer.)
2. Read an introduction to the topic of the debate and select an issue of interest.
3. Answer a series of survey multiple-choice questions about the selected issue, asking you whether you agree or disagree with claims made in arguments. To help you to formulate or reconsider your opinion, you have the option to first view quotations of formulations of the arguments from the source documents. No information about the authors of the quotations will be shown during this phase, to allow the arguments to speak for themselves and avoid prejudicing your answers. (The authors will be revealed at the end, when comparing your opinions to theirs.) The system will inform you about your progress and estimate the remaining number of questions. The precise number of questions may vary, as the order and number of questions depends on your answers to prior questions. You decide how deep to delve into the issues and can control how much time to spend answering poll questions.
4. View a summary of the questions and your responses. You are provided with an opportunity to change your answers.
5. Compare your opinions with those of the authors of the source documents used to create the argument graph. Rank the authors of the source documents, such as political parties, by how much they agree or disagree with your opinions. You can click on the titles of the source documents to download and view their full text and check for yourself the extend to which your agree or disagree with the claims made.

This procedure is flexible and you are in control. You are the moderator of the virtual debate. You can stop at any time, and continue later if you'd like, or jump backwards or forwards to any step in the procedure.

5.2 Question Types

Three types of questions are asked during the poll. The questions are generated automatically by traversing, in a systematic way[4], the nodes of the argument graph. The first time you are asked for your opinion about some statement, you will be asked whether you

- agree with the statement
- disagree with the statement
- want to first see the arguments pro and con the statement before expressing an opinion, or finally
- want to skip this issue and go on the next.

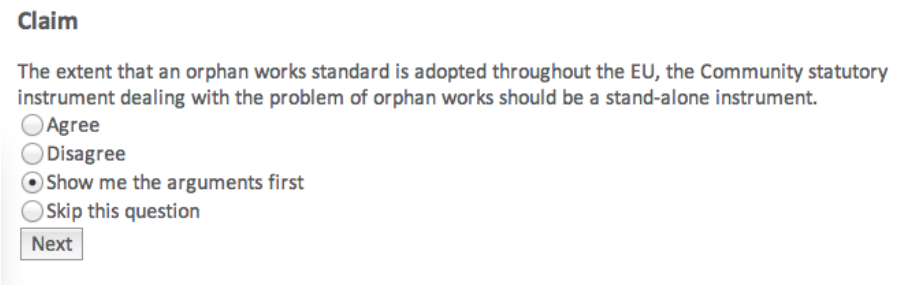

Figure 5: First Time Question About a Claim

If you first want to see the arguments before answering, and thus choose the third alternative, then the question will be put aside and you will be shown questions about the arguments pro an con this statement. This second type of question shows you the argument, quoting the formulations of the argument in the source texts, and asks you whether you agree or disagree with its premsies.

After you have seen the arguments, to the depth and level of detail you choose, you will be asked again for your opinion of the statement. This second time, however, the question is formulated somewhat differently. You will first be asked to weigh the arguments pro and con the statement which you have agreed are good arguments, that is arguments whose premises, in your opinion, are true. You can easily adjust the relative weights of these arguments, using sliders, as shown in the figure.

After you have weighed the arguments, you will be asked, at the bottom of the same page, whether you

[4]Depth-first.

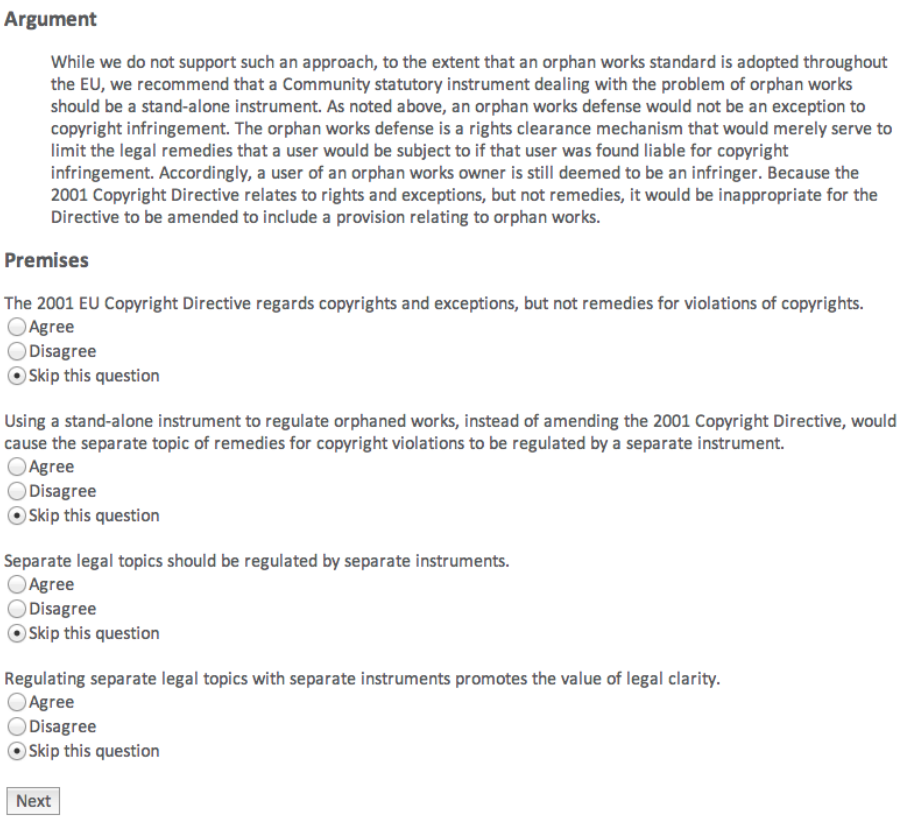

Figure 6: Questions About the Premises of an Argument

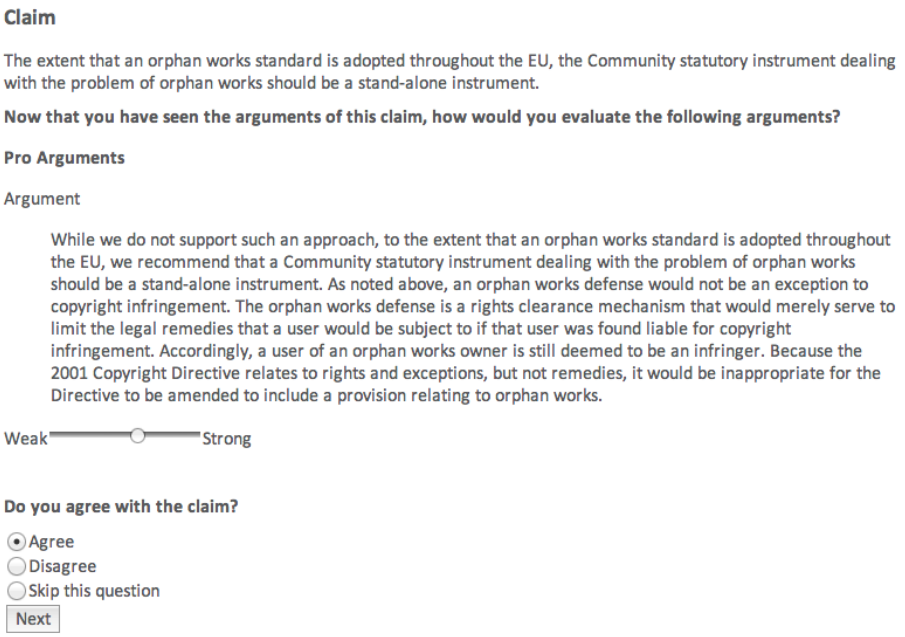

Figure 7: Second Time Question About a Claim

- agree with the statement
- disagree with the statement, or
- want to skip this issue and not express and opinion

Please be careful to take your own assessment of the relative weights of the pro and arguments into consideration when answering the question, remembering that these are all arguments you agree with, even if they happen to conflict. Weigh the arguments to resolve the conflict and justify or explain your opinion.

5.3 Checking and Changing Your Answers

To check or change your answers go to the "summary" page, by clicking on the "summary" menu item of the menu bar.

The summary page lists all the statements with which you have agreed or disagreed, showing your opinion.[5] To change your opinion about some statement, click on the "change" button next to the statement. This will reveal, inline on

[5]The statements you skipped, without expressing an opinion are not listed. (*Note: Not yet implemented: some way for the user to review and answer the skipped questions, including taking a tour of the arguments about the skipped questions.*).

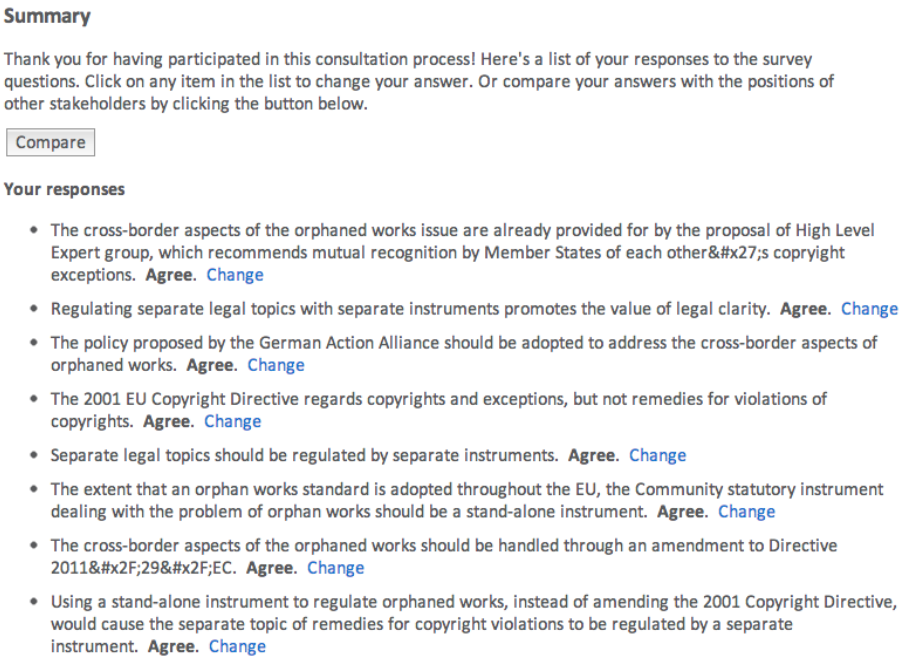

Figure 8: A Summary Page

the same page, a summary of the pro and con arguments you agree with about the statement and give you an opportunity to adjust the weights of these arguments and change your opinion of the statement. Try to take care to make sure that your new opinion remains consistent with the relative weights you assign to the pro and con arguments.

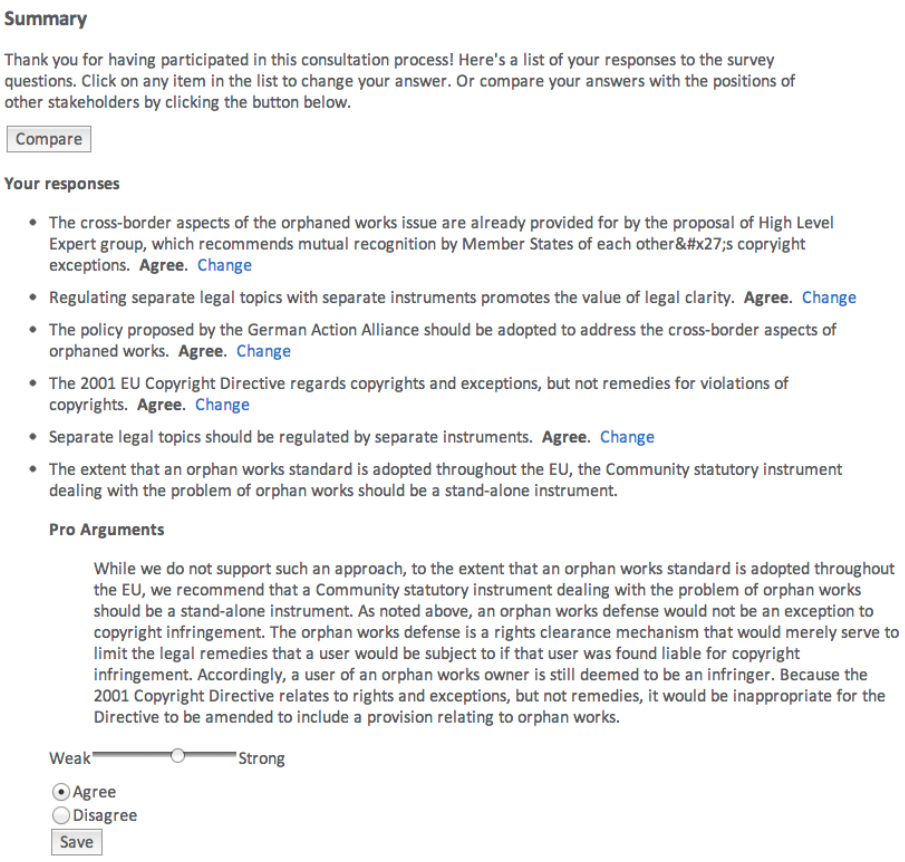

Figure 9: Changing Your Answer

After you have changed your opinion, or cancelled the dialog, the arguments will be folded away, out of view, and you can continue checking your other answers where you left off.

5.4 Comparing Your Opinions with Others

To compare your opinions with those expressed in the source documents used by the analysts to build the argument graph, click on the "compare" button in the menu bar of the opinion formation and polling tool. This will take you to a

page showing the source documents grouped into several categories, ordered by how much the opinions expressed in the documents have in common with your opinions, based on your answers to the poll questions.

Figure 10: Opinion Comparison Page

In each category, full references to the documents are provided (author, title, etc). The title includes a hyperlink to the source of the document on the Web. You can click on the title to download and read the original document, to judge for yourself how much you agree or disagree with the opinions expressed in the document.

The rest of this section explains briefly how the comparison is computed.

All of the arguments modeled in the argument graph are tagged with the keys of source documents in which the argument has been made, from the corpus of source documents used by the analysts to construct the graph.[6] These documents do not merely cite or quote the argument. They express agreement with the argument, by claiming that the premises and the conclusion of the argument are true. Since the arguments are linked to their conclusion and premises in the argument graph, it is easy to compute from the source metadata of arguments the set of claims, i.e. statements claimed to be true or false, in each source document. These claims are then compared to your opinions, based on your answers to the poll questions.[7]

[6] These keys are stored in the "source" property of the metadata of arguments. The keys are the identifiers of the metadata records for documents in the references section of the argument graph.

[7] The similarity of opinions is measured by the percentage of claims in the document with which you have expressed agreement. We may replace this with some other metric, such as "Euclidean distance" (Segaran 2007, 9–15), in a future version of the system.

6 Analysing and Comparing Policies

The Carneades argumentation system provides a tool to analyze the legal effects of specific policies on the facts of cases. The tool is useful for evaluating and comparing policies which formulate rules at a level of detail comparable to legislation or regulations.

To access the tool, click on the "policy analysis" item in the main menu of the system, shown on every page. This will take you to introduction page of the tool, which briefly explains the purpose of the tool and the following steps for using it.

1. Select an issue of interest.
2. Answer a series of questions to enter the facts of a case.
3. View an argument graph automatically constructed by the system by applying the alternative policy models to the facts of the case.
4. Select a policy to evaluate. This will take you back to the argument graph, with the effects of the selected policy highlighted.
5. Repeat the last two steps to try another policy.

After reading the instructions on the first page, click on the "start" button at the bottom of the page to begin. This will take you to the "issues" page.

6.1 Selecting an Issue to Analyse

In the figure, only a single issue is shown but in general they may be several issues to choose form. Select an issue, by clicking on a radio button, and then click on the "submit" button to continue. This will take you to the "facts" page for entering the facts of a case.

6.2 Entering Case Facts

The "facts" page asks a series of questions to gather the facts of the case. The process of asking questions is driven, in a goal-directed way, by the process of applying the rules of the model of the various policies to try to construct pro and con arguments about the chosen issue, using an inference engine. Only relevant questions will be asked, depending on the rules and your prior answers. Related questions will be grouped together, to faciliate a more coherent dialogue.

Be careful not to enter any information which would enable you or or another person to be identified. Do not use real names, but rather pseudonyms or artificial identiers, such a P1, for persons and organizations.

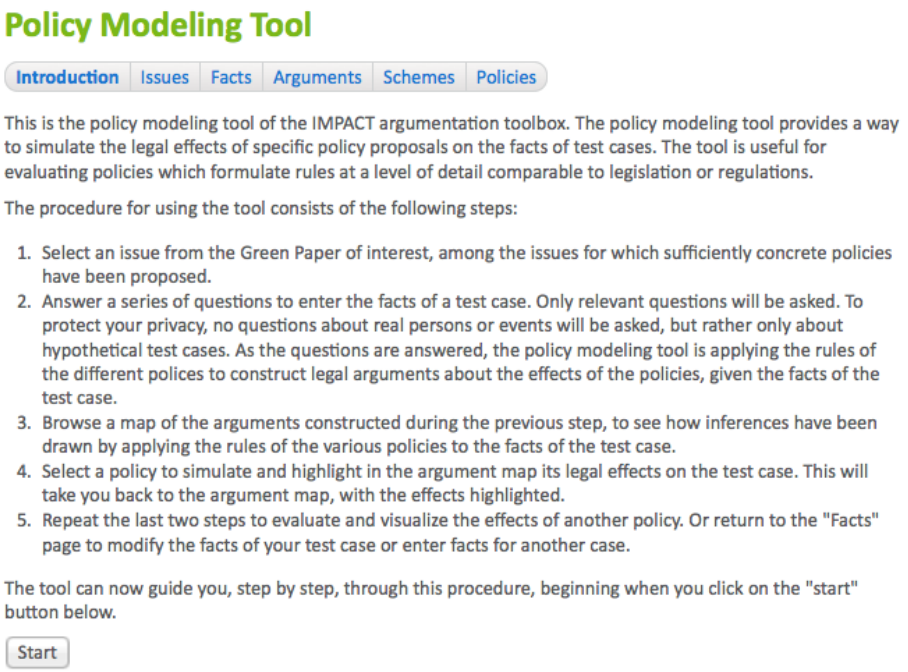

Figure 11: Policy Analysis Introduction

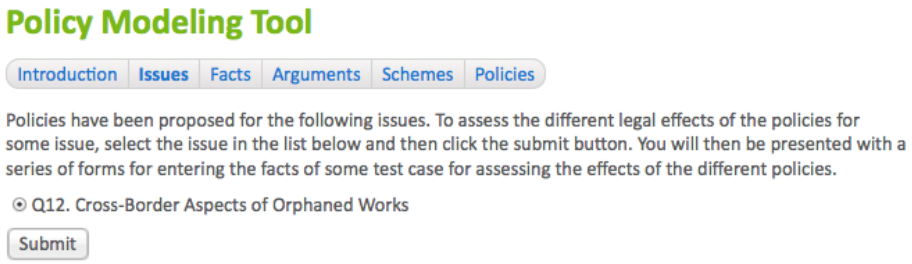

Figure 12: Selecting Issues to Analyze

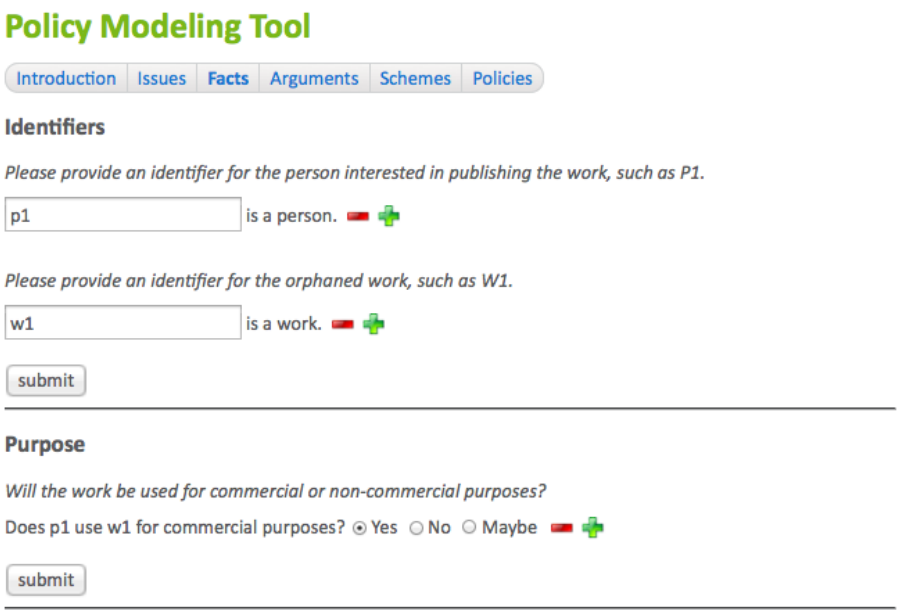

Figure 13: Entering Case Facts

After sufficient facts have been entered to apply the rules of the policies, you will be taken to the "analysis" page of the tool. (You can return to the facts page at any time to check or modify the facts of the case. (*Note: Not yet implemented.*) The analysis page displays the argument graph constructed when applying the rules of the policies to the facts entered for the case. The user interface is exactly the same as the one described in the chapter entitled "Browsing, Visualizing and Evaluating Arguments". The only difference is the content of the argument graph being displayed. Here we are viewing arguments constructed by an inference engine when applying policy models to the facts of a case, rather than manual reconstructions by analysts of arguments in source documents.

6.3 Viewing the Arguments Constructed from the Policies

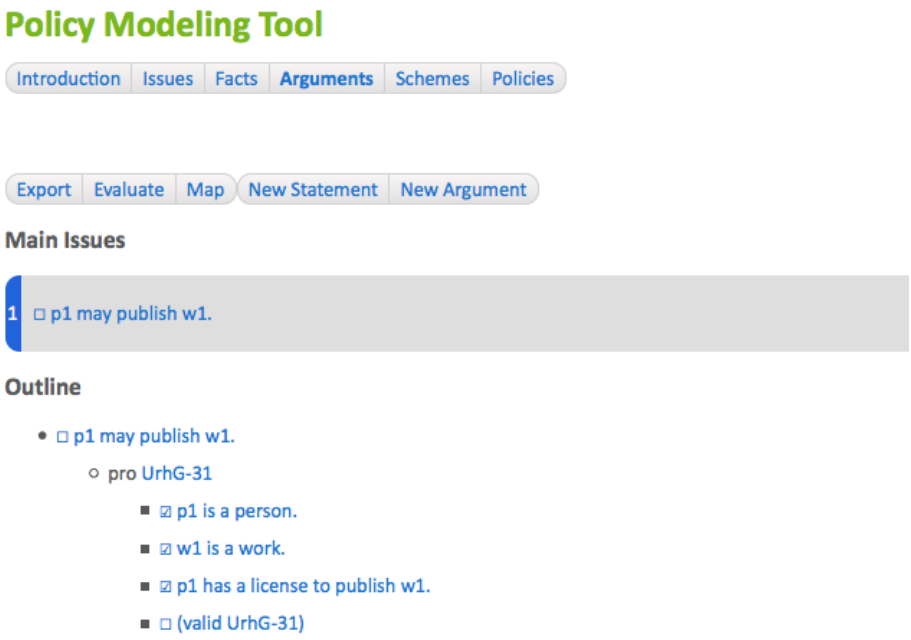

Figure 14: Arguments Constructed from the Policies

The argument graph can be viewed using both a hypertext interface as well as graphically, in an argument map. Click on the "map" item in the menu bar of the analysis page to generate and view the argument map.

The next step is to select one or more policies to evaluate. Click on the "Policies" menu item in the menu bar of the policy analysis tool to view the available policies.

6.4 Viewing the Policy Models

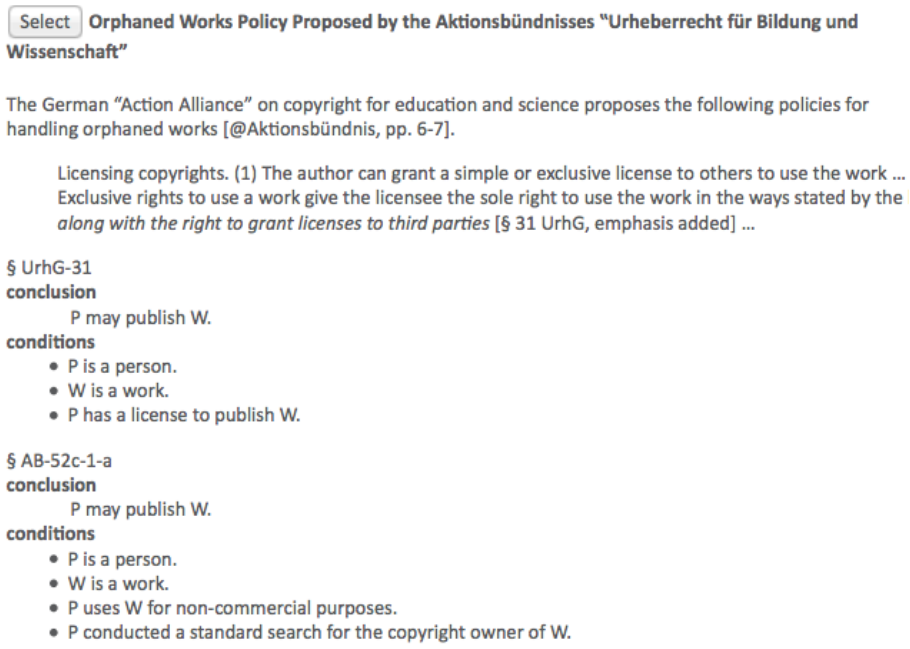

Figure 15: Viewing the Policy Models

All of the proposed policies which have been modeled are displayed. These can be alternative policies under consideration in a policy deliberation process. Alongside the model of each policy, using a rule language, there is a natural language description of the policy, possibly quoting and linking to the text of some document in which the policy was proposed. The descriptions will be shown in your preferred language, which can be selected on the home page of the Carneades system (*Note: not yet implemented*). The rules of the model are shown in an easy-to-read format, generated automatically from the source code of the model, not the source code itself.

6.5 Evaluating and Comparing Policies

To evaluate a policy to see how it would work given the facts of the case you have entered, click on the "select" button to the left of the title of the policy. This will take you back to the "analysis" page, again showing the argument graph, but this time visualizing what the legal consequences of the selected policy would be in the given case, if the policy were enacted and put in force. You can also view the results of the chosen policy in an argument map, by clicking on the "map" button in the menu bar.

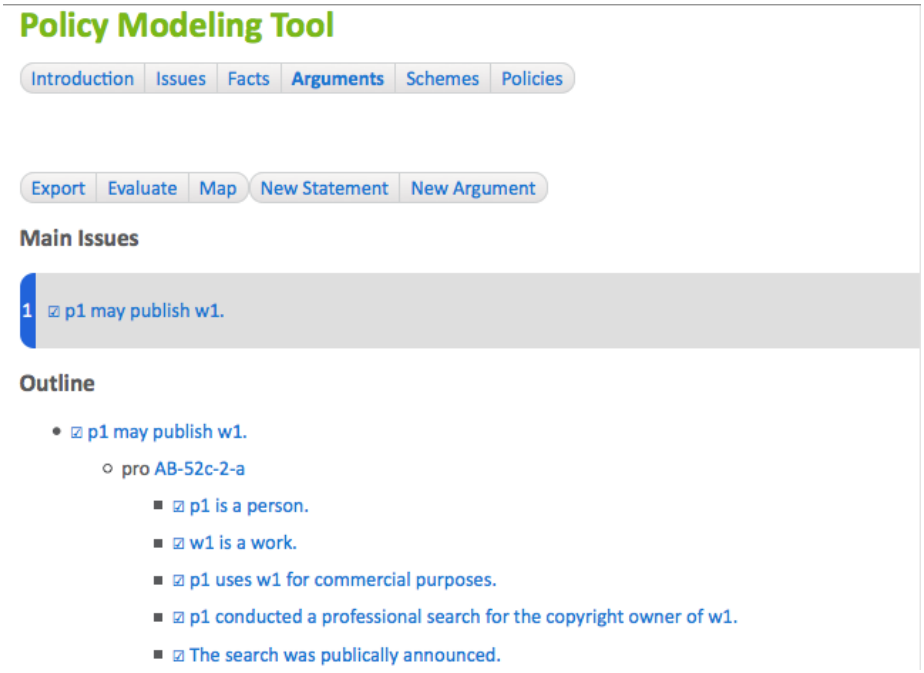

Figure 16: Viewing the Effects of a Selected Policy

To analyze the effects of other policies, go back to the policies page, by clicking on the "policies" button in the menu bar, and select another policy. Repeat this process to compare several policies.

6.6 Finding Policies with Desired Effects

(*Note: Not yet implemented.*)

Rather than manually trying out a number of policies, one at a time, it is possible to automatically search for the policies with the desired result in your case, as follows:

1. In the menu bar of the "policies" page, click on the "find" button.

2. You will be asked whether the main issue of the case should be "in" (true), "out" (false) or "undecided". Select your preferred alternative.

3. The system will then present a list of the policies, if any, which produce the preferred result in this case. This list has the same form and user interface as the policies page, but showing only the subset of the policies

which achieve the desired result. You can now select from these policies, as before, to view the argument graph evaluated using the selected policy, to understand just why the policy leads to the desired result in this case.

6.7 Sharing Cases

You can share the cases you have entered with others, to allow anyone to see how the various policies work in this case and make it unnecessary for others to have to reenter the facts of the case.

Every case is stored in its own database on the server and assigned web address (URL) by the Carneades system. This URL can be used, by anyone, to view and browse the argument graph constructed for the case and to compare the effects of the different policies on the case.

The URL to use for sharing cases is the URL of the argument graph view of the case. To copy the URL:

1. Click on the "analysis" button in the menu bar of the policy analysis tool.
2. Copy the URL shown in the address bar of your Web browser, where you type in URLs to visit web sites.

To share the case, paste this URL into an email, a blog or discussion forum article, or indeed anywhere else text can be entered.

(*Note: Not yet implemented:*) The case will be read-only, not modifiable, by other users. (The case is editable only during the session in which it was created.) However, an editable copy of the case, with its own URL, can be created, as described in the next session.

6.8 Making Editable Copies of Shared Cases

(*Note: not yet implemented*)

To make it easier to create variations of test cases, without having to reenter the facts they have in common, it is possible to creating a copy and then edit the facts of this copy. The copy will have its own URL, which can be shared with others on the Web, just like the original. The facts of the copy can be modified during the session in which the copy was created.

To copy a shared case:

1. Enter the URL of the case in the address bar of your web browser, or just click on some link with this URL. This will take you to the analysis page for viewing the argument graph of the case.

2. Click on the "copy" button in the menu bar of the analysis page. This will create the copy and then take you to the analysis page for viewing the argument graph of this copy. When you first visit this page, it will display a clear warning, so that you can be sure that the copy is being displayed, and not the original case.

You can now edit the facts of this copy during this session, and also share this copy with others in the usual way, as described in the Sharing Cases section of this manual.

6.9 Policy Opinion Polls

(*Note: Not yet implemented.*)

Opinions can differ of course about what the correct or best result is for a particular case. The policy analysis tool provides a way to poll users to gather their opinions about the claims of the case (its main issues in the argument graph). Any user can take part in the poll, not just the user who entered the facts of the case.

To take part in the poll, click on the "vote" button in the menu bar of the policy analysis tool. The main issues of the case will be displayed, for example "The work may be copied.", along with the following question for each claim:

> Given the facts of this case, should claim "The work may be copied." be: 1) accepted, 2) rejected, or 3) undecided?

It is not necessary to login to the system or provide a username to take part in the poll.

After answering the questions, the current tally of the poll results will be displayed in a table, like this:

Claim	Accepted	Rejected	Undecided
The work may be copied.	55%	25%	20%

Table 1: Results of the Case Poll

Notice that users are not asked to directly express their preferences regarding the polices under consideration, just what they think the correct result should be in particular cases. See the section of this manual entitled Generating Policy Reports for further information.

6.10 Generating Policy Reports

(*Note: Not yet implemented.*)

The policy analysis tool can generate a report ranking the policy proposals by the degree to which they achieve the results preferred by the majority of the users who took part in the poll.

To generate the report, click on the "report" button in the menu bar of the policy analysis tool. This will cause the system to apply every modeled policy to all the cases and then compare the results with those preferred by the majority of the users who participated in the poll.

The report will be presented in the form of a table, similar to the following:

Policy	Agreement
Org 1 Proposal	55%
Org 2 Proposal	35%
Org 3 Proposal	15%

The policy names, such as "Org 1 Proposal" in the example, will be linked to the descriptions and models of the policies. The policies are ranked by the percentage of issues in all the cases (there may be more than one issue per case) which have the result preferred by the majority of poll participants. In the example report, the "Org 1 Proposal" obtains the results preferred by the majority for 55% of the issues.

7 Editing Argument Graphs

Normally, argument graphs can be edited only after logging into the system with a password. (*Note: the login form has not yet been implemented. Currently all users are logged in automatically, sidestepping this protection.*) Interpreting source texts to reconstruct arguments using argumentation schemes requires some training. The editing functions of the system are intended for use by "analysts" with the required skills, not "public users". See the Getting Started chapter for a discussion of the different user roles supported by the system. System administrators can make the editing functions available to the public, if this is desired for some application. See the System Administration chapter for further information.

To edit an argument graph, first go to the argument graph page, as described in the Chapter entitled Browsing, Visualizing and Evaluating Arguments.

You can add nodes to the graph by clicking on the "New Statement" or "New Argument" buttons in the menu bar. In both cases you will be presented with a

Figure 17: An Argument Graph Page

form to enter the required information. The form will be inserted and displayed at the top of the current argument graph page, so that you can scroll down to view information about the argument graph, without having to toggle back and forth between two tabs or pages in your web browser. See the Entering New Statements or Entering New Arguments sections for further information.

To add a new argument pro or con some existing statement, go to the statement page and click on the "New Argument" button in its menu bar. The conclusion of the new argument will be set to the existing statement. Then complete the rest of the form as described in the Entering New Arguments section.

To edit or delete existing statements or arguments, first go to the page of the statement or argument and then click on the "Edit" or "Delete" button of its menu bar. Editing statements and arguments is done using the same forms used to create new statements and arguments. Deleting a statement will also delete the arguments pro or con the statement. Deleting an argument does not delete the conclusion or premises of the argument. This can leave some statements in the argument graph being unused in any argument.

Warning: There is no undo function, so all editing and delete operations are permanent. However, you will be asked to confirm all delete operations and have the option of cancelling or saving editing operations.

7.1 Reconstructing Arguments

To reconstruct a *new* argument in some source text, one which is not already in the argument graph, follow the procedure below. If instead the source text is *another formulation* of an argument already in the argument graph, you can modify the description of the existing argument to also quote this source.

1. Optional: If the source text is not already in the list of references of the argument graph, you can add it by following the instructions in the Editing the Metadata and Reference List of an Argument Graph section of this manual. When doing so you will assign the source document a "key" (label). Remember or copy this key.

2. Click on the "New Argument" button in menu bar of the argument graph page.

3. Copy the text of the argument from the source document and paste it into the description field of the new argument form. You can quote the text of the argument, using Markdown, by preceding each line with a ">" symbol. Be careful not the reveal the authors of the quoted source texts if you are planning to use the opinion formation and polling tool, to avoid prejudicing the opinions of the users taking part in the poll. At the end of the quotation, cite the source document, using the key you assigned it when

adding it to the list of references, using the Pandoc extension of Markdown for citations. For example, if the key is "BenchCapon:2011" the cite would have the form `[@BenchCapon:2011, pg. 5]`. The page number or numbers are optional. See the Pandoc documentation for details.

4. Optional: You can choose an argumentation scheme to apply, from the pull down list of schemes in the argument editor. If you choose a scheme, its documentation will be shown and the form will be customized, with fields intialized for each of the premises, exceptions and assumptions of the scheme. You can modify the argument however you want, unconstrained by the chosen scheme, for example by deleting or adding premises, or renaming premise roles. The schemes are there to help you, not constrain you. You can however check whether the argument complies with the given scheme. See the Validating Arguments section for details. See the Argumentation Schemes chapter for documentation of the schemes included in the Carneades. These schemes may be modified or extended, or replaced entirely, as described in the Modeling Policies and Argumentation Schemes chapter.

5. Enter the conclusion of the argument, either by choosing a statement already in the graph with the same meaning as the conclusion of the argument in the source text or by creating a new statement, by clicking on the plus (+) button to the right of the "select a statement" message. (Move the move over the "select a statement" message to see the plus sign, which is otherwise hidden.) If you create a new statement, you can either quote the conclusion of the argument in the "text" field of this new statement, or reformulate the conclusion in your own words.

6. Similarly, add the premises of the argument, either by choosing existing statements in the argument graph or by adding new statements.

7. At the bottom of the form, click the "Save Argument" or "Cancel" button. **Warning:** Any changes you make using the form will not be saved to the argument graph until you execute the "Save Argument" command by clicking on this button.

7.2 Editing the Metadata and Reference List of an Argument Graph

(*Note: Not yet implemented.*)

7.3 Entering New Statements

The form for entering new statements is shown when you click on the "New Statement" button in the menu bar of the argument graph page or on a plus sign

(+) next to the conclusion or premise pull-down selection boxes for statements. (The plus sign allows you to enter a new statement if the statement you need is not listed in the pull-down menu.)

A new statement should not be entered into the argument graph if the negation of the statement is already in the argument graph. That is, for every proposition P, a single statement node should be entered into the graph to represent *both* P and $\neg P$. It doesn't matter whether the positive or negative form of the statement is included explictly in the argument graph. Typically the positive form is used, but in some cases you may prefer the negative form. For example, you may prefer to include the sentence "The payment was illegal" explicitly in the argument graph and then represent arguments pro the legality of the payment as argument con the claim the payment was illegal. The best choice may depend on the procedural context of the debate.

Only one form of the statement is required, since arguments pro the negation of the statement are equivalent to arguments con the statement. And premises of arguments can be explicitly negated in argument graphs, and in the forms for entering and editing arguments. This approach has the advantage of reducing the number of statements in the graph up to 50%.

7.3.1 Statement Description

You can use the *description* field to provide whatever background information you want about the statement. Unlike the mandatory "text" field (see below), this information is optional. You can structure and format the description, including headers, lists, quotations, hypertext links and other elements, using the Markdown wiki language. The form includes a Markdown editor to make this easier for you. You can enter translations of the description. There is a tab of the description form for each available language. (The set of languages can be configured by system administrators when installing the system. *Note: not yet implemented.*) The system is preconfigured for the following languages:

En English

De German

Fr French

It Italian

Nl Dutch

Sp Spanish

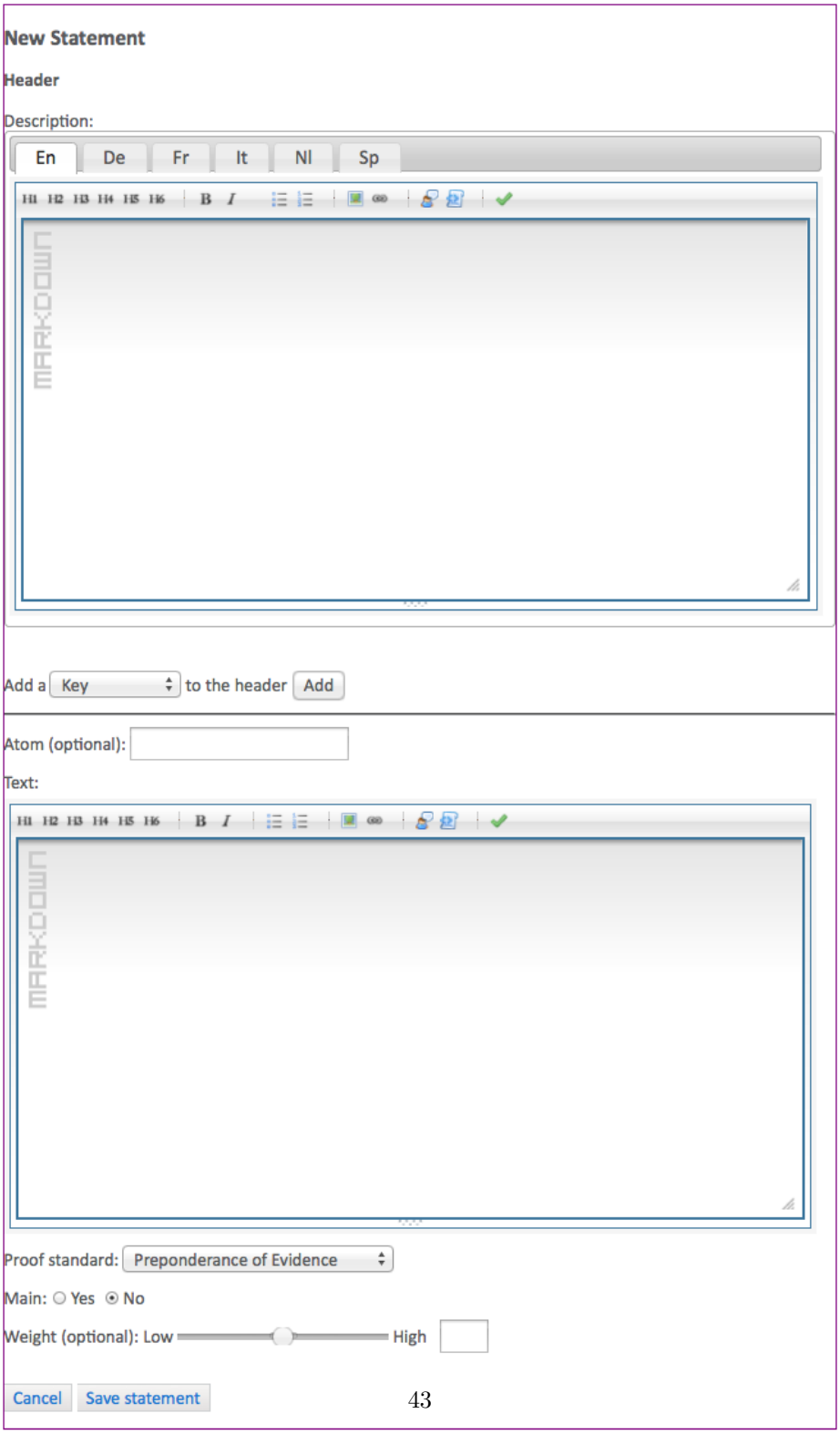

Figure 18: New Statement Form

7.3.2 Statement Metadata

Statements can be annotated with metadata properties, using the Dublin Core metadata elements. The Dublin Core elements are briefly summarized in the Metadata section of this manual.

To add a metadata element to a statement, select the element from the pull-down menu and click on the "Add" button. A text box for entering the value of the element will be added to the form. Multiple values of an element can be added by clicking on the "Add" button as often as needed. A value can be deleted by clicking on the "x" icon to the right of the value.

7.3.3 Atoms: Formalizing Statements using Predicate Logic

Optionally, statements can be formalized in predicate logic, by providing a value for the "atom" property in the form. This is an advanced feature that may be needed only for more specialized application scenarios.

Atoms are formalized in Carneades using the prefix notation of s-expressions ("symbolic expressions"), as in the Lisp family of programming languages.

For example, Socrates is a man could be formalized as `(man Socrates)`. Here, `man` is a unary predicate symbol. The fact that Socrates died in 399 B.C. could be represented using a binary predicate, for example as `(died Socrates -399)`.

The language for Atoms is quite expressive:

- The arity of atoms can be 0 or more, as in predicate logic, not restricted to unary and binary predicates, as in description logic, RDF and OWL. Examples: `(good)`, `(between 0 1 2)`.
- Compound terms may be used. For example `(= 2 (+ 1 1))`. The is, the atom language is not restricted to Datalog.
- Atoms can be higher-order. Example: `(believes Gerald (and (made-of Moon green-cheese) (really-exists Yeti)))`. Notice that compound propositions can be reified as terms, as in this example.

If you choose an argumentation schemes to use as a template when editing arguments, the statements of the argument will be initialized with atom schemas. These are atoms with schema variables, represented by strings starting with a question mark, such as `?x`. If your application does not make use of atoms, you can simply ignore the atoms, or delete them, as you prefer. If your application does use atoms, you need to substitute the variables in the schemas with constants from your application domain, where constants are represented by symbols not beginning with a question mark, such as `work1`, or numbers.

7.3.4 Statement Text

The "text" property of a statement node of an argument graph is for expressing the statement concisely in natural language. You should always provide such a text. As for descriptions, translations of the text in several languages may be included in the model and the statment may be structured using Markdown. (*Note: the tabs for entering the text is more than one language is not yet implemented. Update the screenshots when this has been fixed.*) This is the text that will appear in statement boxes in argument maps and in hypertext views of the argument graph. Whereas descriptions are optional and part of the *metadata* about the statement, this text is the *data* representing the statement itself.

7.3.5 Proof Standard

The proof standard of a statement determines how much proof is required for the statement to be deemed acceptable (presumably true). The proof standard is used by the computational model to argument to compute the acceptability of the statement. Several proof standards are available:

Preponderance of Evidence (pe) This standard is meet if at least one pro argument is *in* that weighs more than any *in* con argument. (Only arguments pro or con this statement are compared.)

Clear and Convincing Evidence (ce) This standard is satisfied if the preponderance of evidence standard is meet and, in additional, the difference between the strongest *in* pro argument and the strongest *in* con argument is above a certain threshold.

Beyond Reasonable Doubt (brd) This standard is meet if the clear and convincing evidence standard is meet and, in addition, the weight of the weakest *in* con argument is below a certain threshold.

Dialectical Validity (dv) This standard is the only one that does not make use of arugment weights. It is satisfied if at least one pro argument is *in* and no con argument is *in*.

The default proof standard is preponderance of the evidence, and for most applications this proof standard should be sufficient. Note that the preponderance of evidence standard is meet whenever the dialectical validity standard is met. If arguments are not weighed, the dialectical validity and preponderance of evidence standards will give the same results. The preponderance of evidence, clear and convincing evidence and beyond reasonable doubt standards are ordered by the amount of proof required, with beyond reasonable doubt requiring the most proof. Whenever one of these standards is meet, any weaker standards are also meet.

7.3.6 Main Issue

An argument graph should have at least one main issue. These are the issues which are central to the debate. All the other issues are subsidiary issues only important to the extent that they are relevant for resolving one of the main issues.

The main issues are offered as entry points on the argument graph page and by the "guided tours" of the opinion formation and polling tool.

The make the statement a main issue, click on the "Yes" radio button next to the "Main" label in the statement editor.

7.3.7 Statement Weight

The weight of a statement encodes the extent to which users agree or disagree with the statement, on a scale of 0.0 to 1.0, where 0.0 represents disagree (reject), 0.5 means no opinion and to 1.0 is denotes agree (accept).

The weight can be entered directly in the statement editor, using a slider. This can be useful for exploring interactions among the arguments given various assumptions. Alternatively the public users can be polled for their opinions and the results of the poll can be averaged to compute the weights. See the Using Polls to Weight Statements and Arguments section of this chapter for further information.

7.4 Editing Statements

To edit an existing statement in an argument graph, first navigate to the statement, using either the hypertext view of the graph, beginning on the argument graph page, or the argument map. In the map view, click on the box containing the statement you want to edit.

Either way you will now be viewing the statement page. Click of the "Edit" button in the menu bar of the page. This will reveal, on the same page, a form for modifying the statement. This is the same form used to enter new statements, described in the Entering New Statements section of this chapter, but with the fields of the form filled in with the current values of the properties of the statement. After making changes, click on the "Save Statement" button at the bottom of the form, to have the changes stored in the argument graph database, or the "Cancel" button to abort the editing process and retain the prior values of the properties of the statement.

Warning: The changes cannot be undone after saving them.

7.5 Deleting Statements

To delete a statement from an argument graph, first navigate to the statement, using either the hypertext view of the graph, beginning on the argument graph page, or the argument map. In the map view, click on the box containing the statement you want to delete.

Either way you will now be viewing the statement page. Click of the "Delete" button in the menu bar of the page. You will be asked to confirm or cancel the deletion.

Warning: The deletion cannot be undone after it has been confirmed. Deleting a statement also deletes all arguments pro or con this statement, i.e. with this statement as the conclusion of the argument. The statements for the premises of these arguments are not deleted.

7.6 Entering New Arguments

The form for entering new arguments is shown when you click on the "New Argument" button in the menu bar of the argument graph page or a statement page. When used on a statement page, the form for entering the new argument will be initialized with the statement as the conclusion of the argument.

See the section on Reconstructing Arguments for tips about how to proceed when interpreting and modeling arguments in source texts.

7.6.1 Argument Description

You can use the *description* field of the argument to provide whatever background information you want about the argument, including quotations of formulations of the argument in source texts, along with hyperlinks to the sources.

You have the same possibilities for structuring the text of the description, using the Markdown wiki language, and entering translations of the description as you have for statement descriptions. See the Statement Description section of this chapter for further information.

If you plan to use the argument graph with the opinion formation and polling tool, be careful to formulate the descriptions in a way which does not risk prejudicing the answers by revealing the authors of quotations. Reference the source of the quotation using using a citation key. See the section on Reconstructing Arguments for further information.

7.6.2 Argument Metadata

Arguments can be annotated with metadata properties, using the Dublin Core metadata elements, in the same way as statements. See the Statement Metadata

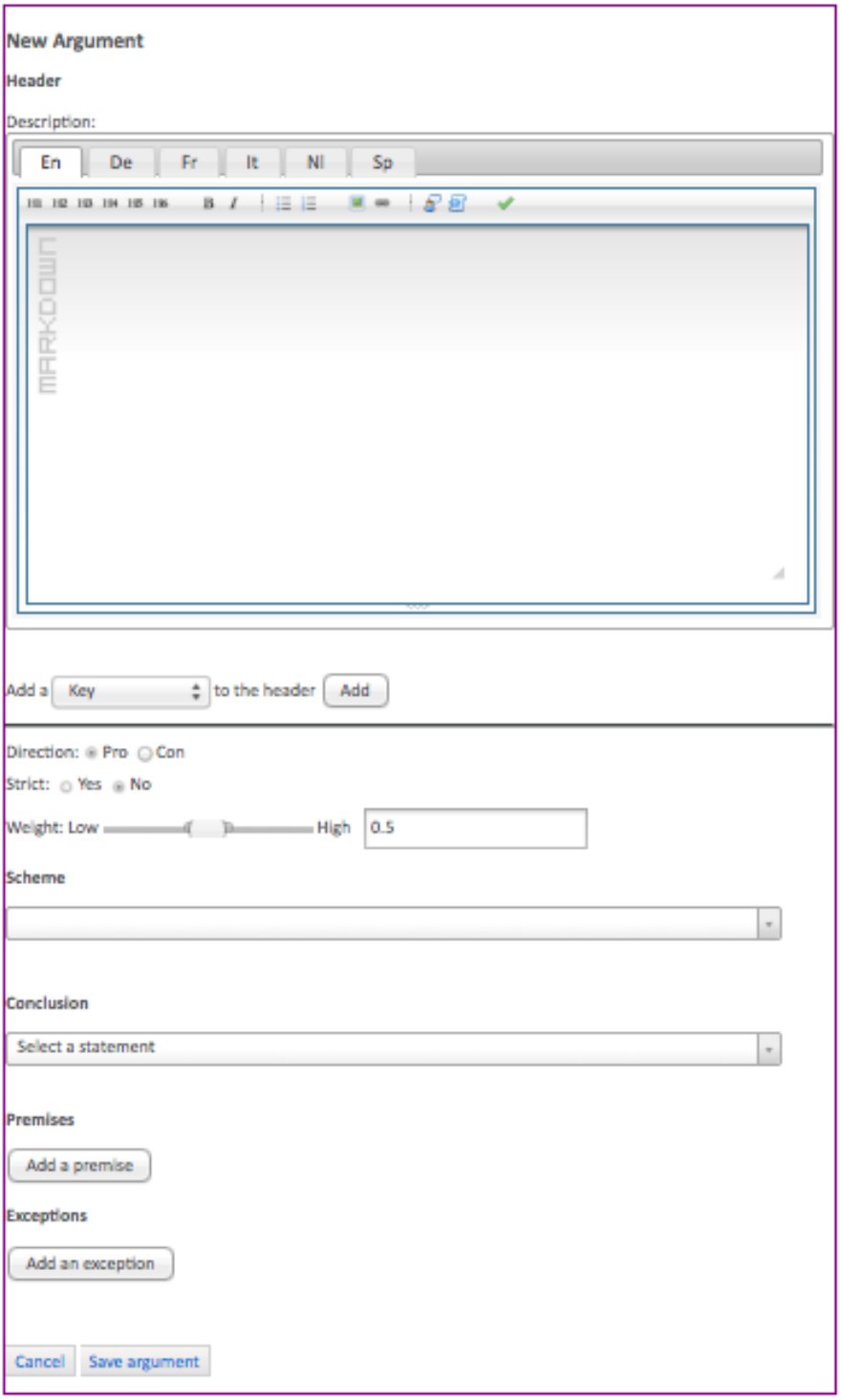

Figure 19: New Argument Form

section for further instructions.

7.6.3 Argument Direction

Arguments can be *pro* or *con* their conclusion. An argument con a statement is semantically equivalent to an argument pro the negation of the statement. Select the direction of the argument in the form using the radio buttons.

7.6.4 Strict or Defeasible Arguments

An argument may be *strict* or *defeasible* (not strict). An argument is strict if and only if its conclusion must be true, with no exceptions, if its premises are true, whether or not its premises are in fact true. For example the following arguments are strict:

- The moon is made of green cheese since the moon is made of green cheese and unicorns have horns.
- The figure is a triangle since it has three sides.

The first example argument is of course cyclic, not to mention fanciful. But it is strict nonetheless.

Arguments are defeasible if they are not strict. With defeasible arguments, the conclusion is only *presumably* true when the premises are true. The argument gives us a reason to accept the conclusion, but there may be exceptions or other reasons leading to the opposite conclusion. Here are a couple of famous examples from the computational models of argument literature:

- The object is red since it looks red.
- Tweety flies sinces Tweety is a bird.

The object may look red because, for example, if is being illuminated by a red light. And Tweety the bird may be a penguin, or have a broken wing, and so on.

The conclusion of a defeasible argument need not be *probably* true, in some statistical sense. It is not necessary to have good empirical data allowing us to draw conclusions about what is probably the case in order to make a defeasible argument. The argument only needs to give us some good reason to believe the conclusion, at least until we have heard arguments giving us reasons to the contrary.

Select whether the argument is strict in the form using the radio buttons. By default, arguments are defeasible.

7.6.5 Argument Weight

Argument weights only become important when two or more arguments with contradictory conclusions, that is arguments pro and con some conclusion, come into play. Such arguments are called "rebuttals". The conflict between the rebuttals is resolved by weighing the arguments and applying proof standards. See the Proof Standard of this chapter for further information.

The weight can be entered directly in the argument editor, using a slider. This can be useful for exploring interactions among the arguments and experimenting with the effects of different proof standards. Alternatively the public users can be polled for their opinions and the results of the poll can be averaged to compute the weights. See the Using Polls to Weight Statements and Arguments section of this chapter for further information.

7.6.6 Using Argumentation Schemes

Arguments can be entered into the system, without using argumentation schemes. Their use is entirely optional. That said, argumentation schemes can be helpful for interpreting source texts when trying to reconstruct arguments. They serve as templates which can guide you in your task of understanding the source text. For example, if the argument looks like it might be an argument from expert opinion, using the expert opinion scheme will help you to remember the conventional form of this kind of argument. Can the text reasonably be understood as including all the premises of the expert witness scheme? Were any missing premises left implicit by the author, or would this be reading too much between the lines? Does the argument perhaps better fit some other scheme?

See the section on Reconstructing Arguments for tips about how to proceed when interpreting and modeling arguments in source texts.

You can view a list of the available argumentation schemes using the pull-down menu. Type in a string, such as "expert", to display the schemes containing the string in the title. Then select the desired scheme from the list. This will display a description of the scheme, perhaps with example arguments, along with references to publications about the scheme. You can go back and select another scheme at any time.

Whenever a scheme is selected, the form will customized to include premises and exceptions fields for the chosen scheme. The roles of the premises and exceptions will be modified to match the selected scheme. The premise roles can be edited. You are not constrained by the scheme.

If you change your mind and select another scheme, any statements you have selected for the conclusion, premises and exceptions of the argument will remain selected, but you will need to check whether they are still appropriate and select or add other statements if necessary.

Again, the argumentation schemes are meant to be helpful, not get in your way. They do not constrain you to enter arguments matching the schemes. You can, however, check whether arguments correctly instantiate the schemes they have been assigned. See the Validating Arguments section for further information.

7.6.7 Entering Premises and Exceptions

In the premise and exception fields of the form, you can choose an existing statement in the argument graph or click on the + icon shown on the right-hand side of the field, when you move the mouse cursor over the field, to open up a form for entering a new statement. When you add a new statement, its "atom" property will be initialized with a predicate logic formula. If you are interested in formalizing the statements, you can use this formula as a template, replacing its schema variables with constants, or edit the formula to meet your own requirements. If you don't need to formalize statements, you can either delete the atoms or just ignore them.

You can additional premises or exception by clicking on the "Add a premise" and "Add an exception" buttons. Premises and exceptions can be deleted by clicking on the X icon shown on the right-hand side of the premise or exception field, when you move the mouse cursor over the field.

7.7 Editing Arguments

To edit an existing argument in an argument graph, first navigate to the argument, using either the hypertext view of the graph, beginning on the argument graph page, or the argument map. In the map view, click on the circle or rounded box containing the argument you want to edit.

Either way you will now be viewing the argument page. Click of the "Edit" button in the menu bar of the page. This will reveal, on the same page, a form for modifying the argument. This is the same form used to enter new arguments, described in the Entering New Arguments section of this chapter, but with the fields of the form filled in with the current values of the properties of the argument. After making changes, click on the "Save Argument" button at the bottom of the form, to have the changes stored in the argument graph database, or the "Cancel" button to abort the editing process and retain the prior values of the properties of the statement.

Warning: The changes cannot be undone after saving them.

7.8 Validating Arguments

(*Note: not yet implemented*)

By argument "validation" here we mean checking whether an argument correctly instantiates the argumentation scheme it has been assigned, if any. (The validation process does *not* check whether the premises or conclusion of the argument are true or acceptable.) The argument passes this validity test if it matches its scheme. If it does not match the scheme, a report is displayed showing how the argument fails to match the scheme.

Arguments can be validated at your choice of different levels of granularity. From the least to the most fine-grained, these levels are:

1. The argument matches the scheme if and only if it has the same number of premises as the scheme, and these premises also have the same "role" labels.
2. In addition to the above test, the text of the statements in the conclusion and premises of the argument match the statements in the conclusion and premises of the scheme, for each natural language representation of the text in the argument. The natural language templates defined in the scheme are used to test the matches, by trying to replace variables in the templates with substrings in the text of the statements in the argument.
3. In addition to the above test, the atoms (predicate logic formulas) of the conclusion and premises of the argument match the atoms of the scheme.

Exceptions in argumentation schemes are represented as undercutting arguments in argument graphs and are validated a bit differently. Each exception of an argumentation scheme is used to construct a separate undercutter argument, with a single premise matching the exception in the scheme. Thus, an undercutter is valid if its premise matches one of the exceptions of the scheme.

Undercutters are not validating automatically when validating the argument they undercut.

To validate an argument, go to the argument page and click on the "Validate" button in the menu bar. You will be asked to select the level of granularity to check. The command will then generate and display a validation report either confirming that the argument is valid or listing all the mismatches so that you can repair them using the argument editing tool.

You can also validate all of the arguments in the argument graph, in one step, by going to the argument graph page and clicking on the "validate" button of its menu bar.

7.9 Deleting Arguments

To delete an argument from an argument graph, first navigate to the argument, using either the hypertext view of the graph, beginning on the argument graph

page, or the argument map. In the map view, click on the circle or rounded box containing the argument you want to delete.

Either way you will now be viewing the argument page. Click of the "Delete" button in the menu bar of the page. You will be asked to confirm or cancel the deletion.

Deleting an the argument does not delete the statements used in its conclusion, premises or exceptions.

Any arguments undercutting this argument will also be deleted, since the conclusion of such an undercutter refers to this argument specifically and can serve no function once the argument has been deleted.

Warning: The deletion cannot be undone after it has been confirmed.

7.10 Using Polls to Weigh Statements and Arguments

Public users can be polled for their opinions and the results of the poll can be averaged to compute the weights of statements and arguments. See the Formulating, Polling and Comparing Opinions chapter for further information about polls.

To compute the weights of statements and arguments from the poll results, go to the argument graph page and click on the "Weigh" button in the menu bar. (*Note: not yet implemented.*) Afterwards you may also want to re-evaluate the argument graph to update the acceptability labels of the statement and argument nodes, as explained in the Evaluating Argument Graphs section.

7.11 Evaluating Argument Graphs

The term "argument evaluation" has two different but related meanings in this manual. The broader meaning, described in the Evaluating Arguments section of this manual, concerns the process of critically assessing the validity of arguments, by

1. revealing implicit premises
2. validating whether the arguments are formally correct, by instantiating accepted argumentation schemes
3. asking appropriate critical questions, depending on the schemes applied
4. and determining which claims are acceptable, taking into consideration the assumptions of the users and their collective assessment of the relative weights of conflicting pro and con arguments.

The final step of this process, determining which claims are acceptable, is what we mean by argument evaluation in this section. Perhaps a better term for this narrower conception of argument evaluation, to avoid confusion, would be "argument labeling", since the result of this kind of argument evaluation is to label the statement and argument nodes of an argument graph as being "in", "out" or "undecided", where:

in means the statement is *acceptable* (presumably true) or the argument is *applicable*, because all of its premises are in and it has not been undercut by an in argument.

out means the negation of the statement is acceptable (i.e. the statement itself is presumably false) or the argument is not applicable.

undecided means the statement or argument is neither in nor out.

The latest verison of the Carneades computational model of argument labels statement and argument nodes in an argument graph by mapping the argument graph to a Dung abstract argumentation framework (Dung 1995), using the ASPIC+ model of structured argument (Prakken 2010) in a way which preserves the features of the original version of Carneades, in particular its support for proof standards (Gordon, Prakken, and Walton 2007). Carneades uses grounded semantics for the resulting Dung argumentation framework, which is the most cautious (skeptical) semantics with regard to the claims it finds acceptable.

To have all the statements and arguments in the argument graph automatically labelled, using the Carneades computational model of argument, go to the argument graph page and click on the "Evaluate" button in the menu bar. This will cause the argument graph to be labelled using the weights currently assigned to the statement and argument nodes.

The labels are not updated automatically when changes are made to the argument graph. So you will need to manually re-evaluate the argument graph whenever any changes are made which might affect the acceptability of some statement. This includes not only changes to the weights of the statements and arguments, but also changes to the assigned proof standards and, the addition or deletion of any arguments. (Additional arguments, not only deletions, can cause the labels to change, since acceptability is a *nonmonotonic* inference relation.)

To evaluate the arguments using the (latest) poll results to weight the statements and arguments, first update the weights by clicking on the "Weigh" button in the menu bar of the argument graph page, before executing the "Evaluate" command. See the Using Polls to Weigh Statements and Arguments section for further information.

7.12 Importing XML

(*Note: XML import, unlike XML export, is not yet implemented.*)

The Carneades Argument Format (CAF) is an XML schema for storing argument graphs in files, in a structured but plain text form which can be read or edited by humans, using any text editor, that is also suitable for machine processing.

To facilitate moving statements and arguments between argument graphs, across installations of the Carneades system, every statement and argument is assigned an Uniform Resource Name (URN). URNs are identifiers which are extremely likely to be unique, world-wide.

(*Note: Write a reference manual describing the CAF format.*)

For information about how to *export* argument graphs to XML, see the Exporting Argument Graphs to XML section of this user manual.

To import a CAF file, navigate to the argument graph page and click on the "Import" button in the menu bar. You be asked to select a file from your file system. The selected filed will be validated to make sure that it is a valid CAF file. It if it is valid, it will be used to update the argument graph, as follows.

1. First, you will be asked if you want to *merge* or *overwrite* the data in the argument graph in the database with the data in the CAF file.
2. If you choose to *merge* the data:
   - The properties of statements or arguments in the argument graph will be *replaced* by the values of the same properties in the CAF file but the values of properties undefined in the CAF file will retain their current values in the database. (**Note**: all values of a multivalued metadata property will be *replaced* by the values of the property in the CAF file. The values of individual properties are not merged.)
   - Any new statements or arguments in the CAF file will be added to the database, without deleting statements or arguments in the database which are not in the CAF file.
   - The metadata of the argument graph as a whole will be updated using the metadata in the CAF file. Properties of the argument graph will be replaced by the values of the same properties in the CAF file. (**Note**: If a property has multipled values, they will all be replaced, not merged.) Properties not in the CAF file will retain their current values in the database.
   - References of the argument graph in the database will be replaced by references with the same keys in the CAF file, but references in the database which are not in the CAF file will be retained, not deleted.

3. If instead you choose to *overwrite* the data:

   - The properties of statements or arguments in the argument graph will be *replaced* by the values of the same properties in the CAF file and properties undefined in the CAF file will be deleted in the database. (**Note**: all values of a multivalued metadata property will be *replaced* by the values of the property in the CAF file. The values of individual properties are not merged.)
   - Any new statements or arguments in the CAF file will be added to the database, without deleting statements or arguments in the database which are not in the CAF file.
   - The metadata of the argument graph as a whole will be updated using the metadata in the CAF file. Properties of the argument graph will be replaced by the values of the same properties in the CAF file. (**Note**: If a property has multipled values, they will all be replaced, not merged.) Properties not in the CAF file will be deleted in the database.
   - References of the argument graph in the database will be replaced by references with the same keys in the CAF file and references in the database which are not in the CAF file will be deleted.

(*Note: Look for a simpler solution.*)

8 Modeling Policies and Argumentation Schemes

Argumentation schemes generalize the notion of an inference rule to cover defeasible as well as strict reasoning patterns. Carneades provides a high-level programming language for representing argumentation schemes and "deep conceptual" or "semantic" models of domains. This language is based-on a higher-order predicate logic expressive enough to represent axiomatizations (i.e. axioms and inferences rules) of theories of any domain, including policies.

The Carneades scheme language is semantically similar to Prolog, with some extensions required for representing argumentation schemes (i.e. strict and defeasible inference rules), along with meta-data and documentation, in multiple languages, and for organizing schemes in an hierarchical outline, to facilitate "isomorphic modeling" of source documents, such as legislation.

The language for argumentation schemes is an executable knowledge-representation language, not an "interchange format". No database is used for representing or storing argumentation schemes. (Carneades does provide a relational database for storing the arguments which instantiate these schemes.) Rather, the argumentation schemes are represented in text files, using the scheme language. The Carneades system provides an inference engine for this

language which is able to fully automatically construct arguments from schemes and "deep conceptual" or "semantic" models represented in this language. Using a high-level declarative rule language for representing argumentation schemes and semantics models, and generating arguments from these schemes and models, makes it easy for domain experts to extend the model to support further argumentation schemes, without requiring knowledge about relational database schemes or the implementation of middleware.

In this section the language is illustrated by reconstructing the argumentation schemes presented in Version 0.5 of Deliverable D5.2.[^2] Other argumentation schemes, or alternative formulations of these schemes, can be represented in a similar manner.

The semantics of argumentation schemes in this language is formally defined in Section "Generating Arguments from Semantic Models". Briefly, instantiations of argumentation schemes are mapped to argument graphs. Argument graphs, in turn, are mapped to Dung abstract argumentation frameworks. Thus, argumentation schemes, in this formal semantics, are abstractions of Dung argumentation frameworks.

A domain theory is represented by first defining a *language* (dictionary) of symbols, denoting predicates and terms, and then a set of inference rules, called *schemes*, using this language.

The language is represented as a map from symbols to predicates and individuals, in the Clojure programming language. Each symbol of the language is mapped to a structure with fields for the symbol of the predicate (redundantly), the arity of the predicate (i.e. the number of columns in a tabular representation of the relation denoted by the predicate) and an optional number of forms for expressing statements and questions about this predicate in one or more natural languages.

To illustrate, below are the definition of some of the predicates of the language used in the D5.2 version of the schemes for practical reasoning.

```
(def L
  {'circumstances
    (make-predicate
     :symbol 'circumstances
     :arity 1
     :forms
     {:en (make-form
           :positive "The circumstances are %s."
           :negative "The circumstances are not %s."
           :question "Are the circumstances %s?")})

   'should-do
   (make-predicate
```

```
 :symbol 'should-do
 :arity 2
 :forms
 {:en (make-form
       :positive "In cirumstances %s, we should do %s."
       :negative "In circumstances %s, we should not do %s."
       :question "In circumstances %s, should we do %s?")}),

'results-in
(make-predicate
 :symbol 'results-in
 :arity 3
 :forms
 {:en (make-form
       :positive "In circumstances %s doing %s would
          result in circumstances %s."
       :negative "In circumstances %s doing %s would
                  not result in circumstances %s."
       :question "In circumstances %s would doing %s
                  result in circumstances %s?")})

'avoids
(make-predicate
 :symbol 'avoids
 :arity 3
 :forms
 {:en (make-form
       :positive "In circumstances %s not doing %s
                  would avoid circumstances %s."
       :negative "In circumstances %s not doing %s
                  would not avoid circumstances %s."
       :question "In circumstances %s would not doing %s
                  avoid circumstances %s?")})

'realises
(make-predicate
 :symbol 'realises
 :arity 2
 :forms
 {:en (make-form
       :positive "Circumstances %s would realize goal %s."
       :negative "Circumstances %s would not realize goal %s."
       :question "Would circumstances %s realize goal %s?")})

'promotes
(make-predicate
```

```
 :symbol 'promotes
 :arity 2
 :forms
 {:en (make-form
       :positive "Achieving %s would promote the value %s."
       :negative "Achieving %s would not promote the value %s."
       :question "Would achieving %s promote the value %s?")}),

'demotes
(make-predicate
 :symbol 'demotes
 :arity 2
 :forms
 {:en (make-form
       :positive "Achieving goal %s would demote the
                  value %s."
       :negative "Achieving  goal %s would not demote the
                  value %s."
       :question "Would achieving goal %s demote the
                  value %s?")})

})
```

An argumentation scheme is represented as a structure with six fields: 1. id, 2. header, 3. conclusion 4. premises, 5. exceptions, and 6. assumptions. The id is a term in the language used to reify argumentation schemes and represent statements about argumentation schemes in domain models. The header enables metadata about the scheme to be represented (e.g. title, description). Descriptions can be represented in multiple natural languages. The conclusion is a formula schema, which may contain schema variables. Schema variables are represented by symbols beginning with a question mark, e.g `?Ag`, `?A`, and `?G`, and range over both terms and propositions. Thus, the conclusion of an argumentation scheme can be a schema variable. This feature is needed for representing schemes, such as arguments from expert witness testimony, whose conclusion may be any proposition whatsoever.

The premises, exceptions and assumptions fields of schemes are vectors of premise structures, where each premise has the following properties:

role A string naming the role of the premise in the argumentation scheme, e.g. "major", "minor", "circumstances", "goal".

positive Boolean. False if the premise is negated. Default: true.

statement A formula schema.

implicit True if the premise was not explicitly stated in the natural language text of the source document or documents. (Arguments with implicit premises are called "enthymemes".)

Next, using this language we formally define the positive and negative versions of the scheme for practical reasoning from the draft of D5.2. Schemes may be organized in an hierarchy of *sections*, each section with its own metadata. Since there are only four schemes in this example, sections are not illustrated here.

```
(def S
  [(make-scheme
    :id 'pras1
    :header (make-metadata :title "Practical Reasoning Scheme")
    :conclusion '(should-do ?S1 ?A)
    :premises
    [(make-premise :role "circumstances"
                   :statement '(circumstances ?S1))
     (make-premise :role "action"
                   :statement '(results-in ?S1 ?A ?S2))
     (make-premise :role "goal" :statement '(realizes ?S2 ?G))
     (make-premise :role "value" :statement '(promotes ?G ?V))])

   (make-scheme
    :id 'pras2
    :header
    (make-metadata :title "Negative Practical Reasoning Scheme")
    :conclusion '(not (should-do ?S1 ?A))
    :premises
    [(make-premise :role "circumstances"
                   :statement '(circumstances ?S1))
     (make-premise :role "action" :statement '(avoids ?S1 ?A ?S2))
     (make-premise :role "goal" :statement '(realizes ?S2 ?G))
     (make-premise :role "value" :statement '(demotes ?G ?V))])
  ])
```

Now, let's us package the language and schemes together in a theory. Conceptually, a theory is a set of propositions. Since the set may be infinite, theories are represented intensionally, as a set of axioms and strict and defeasible inference rules, called argumentation schemes.

A theory is modelled as a structure with the following fields:

header A header with metadata (e.g. title, description) about the theory. Descriptions can be in multiple languages and can be arbitrarily long, structured texts, represented using the Markdown wiki language.

language The formal language of the theory; a dictionary mapping symbols to terms and predicates.

schemes Strict and defeasible inference rules.

sections A sequence of sections, which in turn consist of a header, schemes and (sub)sections, enabling theories to be organized hierarchically, similar to the hierarchical structure of books and articles.

references A sequence of metadata structures, for providing bibliographic information about source documents.

Next, we complete this section, and our illustration of how to implement argumentation schemes, by defining `liverpool-schemes` to be the following theory:

```
(def liverpool-schemes
  (make-theory
   :header
   (make-metadata
    :title "Reconstruction of Liverpool Argumentation Schemes"
    :creator "Tom Gordon"
    :publisher "Fraunhofer FOKUS"
    :date "2012"
    :description {:en "This is a reconstruction of the version of
                the Liverpool schemes in [@Atkison2012a]."})

   :language L
   :schemes S
   :references
   {"Atkison2012a"
    (make-metadata
     :creator "Katie Atkinson, Adam Wyner and Trevor Bench-Capon"
     :title "Report No. D5.2 -- Report on Prototype 1 of
         Structured Consultation Tool"
     :publisher "IMPACT"
     :date "2012")}))
```

9 System Administration

9.1 Downloading Carneades

The Carneades Argumentation System is open source software available on Github at http://carneades.github.com/.

You can download Carneades as a ready-to-use web "binary" application from the downloads page. To use the version of Carneades described in this manual, download the latest file with a name of the form `carneades-webapp.x.x.x.zip`, where the x's are the version number.

The source code can be downloaded using Github from the Carneades GitHub project page.

9.2 License

The source code of the Carneades system is licensed using the European Public License (EUPL), Version 1.1. An English version of the license is distributed with the software, in the `/licenses` directory. The EUPL license is certified by the Open Source Initiative (OSI). Like the Gnu General Public License (GPL), the EUPL is a recipriocal license. Derivative works of Carneades must also be licensed using either the EUPL *or a compatible license*. Several OSI-certified Open Source licenses are compatible with the EUPL. See the EUPL license for details.

9.3 Prerequisites

- Version 7 or better of a Java Runtime Environment (JRE 7).

9.4 Binary Installation

1. Prerequisites

- Version 7 or better of a Java Runtime Environment (JRE 7).

1. Download the latest version of the file with a name of the form `carneades-webapp.x.x.x.zip`, from the downloads page.
2. Unzip the `carneades-webapp.x.x.x.zip` Zip archive file using some Zip tool. This will create a directory (folder) with the following hierarchical structure

    - carneades
        - doc
        - bin
        - projects
        - config
        - schemes

You can move this directory to some other location on your file system, at any time.

This creates a standard installation, with the default configuration.

To start the Carneades web application server double click on the `carneades-webapp.jar` file in your file system browser, for example the "Finder" on Mac OS X or the "Windows Explorer" on Windows PCs.

9.5 Using the Web Application Locally

To start the server to from a command line, for local use, type

```
$ java -jar bin/carneades-webapp.jar
```

Either way, after the server starts it will open up the "home page" of the Carneades web application in your default web browser.

Depending on your operating system and how you started the server, the Carneades web application can be shut down by either quitting the Carneades application or, if you started the server from a command line, using a terminal application, by ending this process , typically by typing `control-c` in the terminal.

(*Note: the Carneades process is currently named "main". This needs to be fixed.*)

9.6 Using the Web Application with Java Application Servers

(*Note: to be written*)

9.7 Source Code Installation

1. Prerequisites

- The Java Runtime Environment (JRE) 7. (You may also use
- Leiningen 2 `http://github.com/technomancy/leiningen`

2. Download the sources using the Git source code management system:

$ git clone git://github.com/carneades/carneades.git

3. Change directory to the src subdirectory of the carneades project.

$ cd carneades/src

4. Build the Carneades system:

$ chmod +x build.sh ; ./build.sh –dev

5. Configure Carneades

- Copy the file config/carneades.properties into your $HOME directory and rename it to .carneades.properties (with a dot in front). Edit it and adapt the values to your configuration.

6. Starting the Carneades web application server

$ cd PolicyModellingTool $ lein ring server 8080

(*Note: Change the name of the PolicyModellingTool package to CarneadesWebApp. *)

1. Using the Web client with your Web browser

- Point your browser to http://localhost:8080/policymodellingtool/#/introduction

(*Note: Change "policymodellingtool" to "carneades" in these URLs.*)

9.8 System Configuration

You can change the configuration of the system globally, for all users with accounts on your server. Alternatively, each user can have there own configuration.

To modify the global configuration, edit the `config/carneades.properties` file in the installation directory. To create a personal configuration for your own user account, copy the `config/carneades.properties` file to a file named `.carneades.properties` in your home (user) directory and then edit this copy.

If a required property is not defined in the user's `.carneades.properties` file, then the value of the property defined in the `config/carneades.properties` file in the installation directory will be used.

The configuration files files the format of a Java ".properties" file.

The following properties can be modified in the `config/carneades.properties` file or overriden in `.carneades.properties` in your home directory. **Warning:** Be careful not to modify or delete any of the other propertie in the `config/carneades.properties` file. The properties which should not be modified are clearly marked.

`argumentation-schemes-file` The full path name of the file containing the argumentation schemes to be used in projects by default. The default may be overriden on a project by project basis. The argumentation schemes are defined using the scheme language described in Modeling Policies and Argumentation Schemes chapter of this manual.

Example:

```
argumentation-schemes-file=/usr/local/carneades/schemes/walton-schemes.clj
```

`projects-directory` The full path name of the directory used to store Carneades projects. The default directory is the `projects` directory of the installation directory.

Example:

```
projects-directory=/usr/local/carneades/projects
```

9.9 Managing Projects

9.10 Project File Structure

Carneades projects are stored in a directory with the following structure:

```
project.properties
cases/
db/
policies/
schemes/
documents/
```

The `project.properties` file defines the attributes of the project, such as its title. It has the format of a Java ".properties" file.

The `cases/` directory stores the database files of the argument graph of the cases created by users when simulating the effects of the policies, when using the opinion formation and polling tool.

The `db/` directory contains the database files of the main argument graph of the project, containing the reconstructions of the argumetns in the source documents.

The `schemes/` directory contains models of argumentation schemes. The file with the model of the schemes currently used by the project is specified by the `argumentation-schemes-file` property in the `project.properties` file. The value of the property may be a relative pathname and will resolved relative to the `schemes` directory of the project. For example, suppose the `project.properties` file contains the line:

```
argumentation-schemes-file=walton.clj
```

This will be resolved to the `schemes/walton.clj` file in the project directory.

If no `argumentation-schemes-file` property is defined for the project the value of the property in the user's `.carneades.properties` file, in his or her home directory will be used, if it is defined there. Otherwise the value of the property in the `config/carneades.properties` in the installation directory will be used.

This general pattern applies to all properties. The value in the `project.properties` file overrides the value in the user's Carneades configuration file, which in turn overrides the default values in the `config` directory of the installation directory.

Providing a directory for schemes allows multiple sets of argumentation schemes to be stored together with the project. But only set of schemes is active at any time.

Finally, the `documents/` directory can be used to store copies of source documents used when reconstructing arguments, or any other projects files. These files can be referenced and linked to in the descriptions of statements and nodes using relative URLs. For example, the URL "src/foo.pdf" would be resolved to the file `documents/src/foo.pdf` in the `documents` directory of the project.

9.11 Listing Projects

9.12 Creating a New Project

9.13 Renaming Projects and Editing Project Metadata

9.14 Archiving Projects

9.15 Deleting Projects

10 Applications Scenarios

This chapter presents a number of application scenarios illustrating *some* ways the Carneades argumentation system can be used in practice to:

- improve the quality, efficiency, inclusiveness and transparency of democratic public policy deliberations,
- improve the efficiency and correctness of claims processing procedures both in public administration and the private sector, for example with regard to social benefits or insurance claims,
- help companies to comply with regulations,

- better manage complex legal cases in law firms, by creating maps of relationships between claims, legal arguments, case and statute citations, and testimonial and documentary evidence, and
- facilitate the learning of critical thinking and analysis skills, particularly in law, philosophy, religion and other humanities fields, where competing theories are constructed by interpreting texts.

10.1 Public Policy Deliberations

For some years now there has been considerable discussion and research on "e-democracy" or "e-participation", about ways to use the Internet and, in particular, the World-Wide Web to revitalize democracy, overcome political apathy and improve the quality, inclusiveness and transparency of governance and policy-making. The first generation of e-participation systems were customized versions of various kinds of collaborative software ("groupware"), such as discussion forums, polls, and wikis. More recently there have been attempts to leverage existing social networks, such as Facebook, Google+ and Twitter for e-participation purposes. While these approaches provide ways for citizens to voice their concerns and provide input to the policy-making process, they do not scale well as the level of participation increases. Imagine a EU-wide policy discussion, with 27 states, 23 official languages, and over 500 million citizens.

Carneades complements existing e-participation and other social media platforms by helping citizens, and government officials, to obtain a concise overview of the substance of the policy debates taking place on these platforms. Carneades can make it easier to identify and understand the issues, proposed policies, the pros and cons of these policies and the collective assessment of stakeholders about the truth of claims and the relative importance or weight of conflicting considerations.

Imagine the following application scenario, which uses Carneades to support wide-scale deliberations, such as the 2008 deliberations of the European Green Paper on Copyright in the Knowledge Economy:

1. The EU publishes the Green Paper, in PDF, on its web site and invites stakeholders, such as libraries and publishers, but also "ordinary" citizens, to submit comments by uploading a PDF file to the site.
2. The EU publishes the comments on their web site, as they are received.
3. In parallel, the issues raised in the Green Paper are discussed informally across the Internet, in discussion forums, on social networks, in online newspaper or blog articles, and in comments on these articles.
4. As the discussion proceeds, trained analysts employed by the EU, either as regular staff or as free-lancers, use Carneades to reconstruct the arguments

in a representive sample of the source documents, including both officially submitted comments and articles found elsewhere on the Internet, to build an argument graph.

5. In parallel, data-driven journalists, hired by media companies and NGOs, can use Carneades to construct their own, competing argument graphs of the debate to be offered as value-added content on the Web.

6. The argument graphs constructed by the analysts and data-journalists are published on the Web, providing citizens with high-level overviews of the ongoing-debate along with references (links) to source documents. The references enable anyone to check the accuracy and completeness of the argument graphs and provide easy access to further, more detailed information.

7. For policy proposals which are detailed enough to be modeled as rules, the EU analysts can use Carneades to create and publish policy models enabling stakeholders to evaluate the effects of each policy on the cases which interest them. Users could contribute, via the web site, benchmark cases they think require special consideration and share these cases with others on the Web.

8. After sufficient time has passed to allow for an in-depth discussion of the issues and the reconstruction of the arguments in an argument graph, the opinion formation and polling tool of Carneades is use to obtain informed feedback from citizens about their opinions of the claims and arguments. The questions asked in the poll are generated automatically from the argument graph. Several polls can be conducted, at regular intervals, depending on the duration of the offical EU procedure for the public deliberation of the Green Paper.

9. At the end of the discussion period, the EU analysts use Carneades to generate an outline of the results of the deliberation which can be processed further using word processing software.

10.2 Claims Processing

Every day, large public agencies and companies, for example in the banking and insurance industries, process thousands of transactions requiring the correct application of laws, regulations, court decisions, and internal policies to decide claims and justify these decisions.

The bureaucracy costs of processing these claims is enormous. When decisions are incorrect or insufficiently transparent, further unnecessary and potentially high costs can be incurred due to long appeal processes and possibly court proceedings.

START HERE

10.3 Regulatory Compliance

10.4 Legal Case Management

10.5 Humanities Education

11 Argumentation Schemes

This chapter presents the argumentation schemes included with the distribution of the Carneades system. The system is pre-configured to use these schemes, but you can configure the system to use other schemes, or modify these schemes to meet your requirements.

The argumentation schemes are shown here in pseudocode for readabililty. See the Modeling Policies and Argumentation Schemes chapter for a description of the syntax used to formally define the schemes.

The schemes can be viewed online, using Carneades, by clicking on the "Schemes" button in the menu bar of the Carneades home page.

Most of the schemes here are derived the book "Argumentation Schemes" (Walton, Reed, and Macagno 2008). The schemes for arguments from credible source and practical reasoning are based on (Wyner, Atkinson, and Bench-Capon 2012) and (Atkinson and Bench-Capon 2007), respectively.

The schemes from these sources been modified to fit the Carneades computational model of argument. For example, generic critical questions which undermine premises, undercut the argument or rebut its conclusion, have been omitted, since these critical questions apply to all defeasible arguments in Carneades.

All the argumentation schemes presented here are defeasible *unless* they have been explicitly declared to be strict.

11.1 Argument from Position to Know

```
id: position-to-know

conclusion: S

premises:
    major: W is in a position to know about things in a certain
        subject domain D.
    minor: W asserts that S is true.
    domain: S is in domain D.

exceptions:
    CQ1: W is dishonest.
```

11.2 Argument from Credible Source

```
id:  credible-source

conclusion: S

premises:
    source: W is a credible source about domain D.
    assertion: W asserts S.
    domain: S is in domain D.

exceptions:
    CQ1: W is biased.
    CQ2: W is dishonest.
    CQ3: Other credible sources disagree with S.
```

11.3 Argument from Witness Testimony

```
id: witness-testimony

conclusion: A

premises:
    position to know: W is in a position to know about things in a
        certain subject domain A.
    truth-telling: Witness W believes A to be true.
    minor: W asserts that A is true.

assumptions:
    CQ1: A is internally consistent.

exceptions:
    CQ2: A is inconsistent with the facts.
    CQ3: A is inconsistent with the testimony of other witnesses.
    CQ4: W is biased.
    CQ5: A is implausible.
```

11.4 Argument from Expert Opinion

```
id: expert-opinion

conclusion: A

premises:
```

```
   major: Source E is an expert in subject domain S.
   domain: A is in domain S.
   minor: E asserts that A is true.

exceptions:
    CQ1: E is untrustworthy.
    CQ2: A is inconsistent with the testimony of other witnesses.
    CQ3: A is based on evidence.
```

11.5 Argument from Analogy

```
id: analogy

conclusion: S

premises:
    major: Case C1 is similar to the current case.
   case: S is true in case C1.
   minor: E asserts that A is true.

exceptions:
    CQ1: There are relevant differences between case C1 and the
        current case.
    CQ2: S is false in case C1, which is more on point
        than case   C2.
```

11.6 Argument from Precedent

```
id: precedent

conclusion: S

premises:
    major: Case C1 is similar to the current case.
    ratio: Rule R is the ratio decidendi of case C1.
    conclusion: Rule R has conclusion S.

exceptions:
    CQ1: There are relevant differences between case C1 and the
        current case.
    CQ2: Rule R is inapplicable in this case.
```

11.7 Argument from Verbal Classification

```
id: definition-to-verbal-classification

strict: true

conclusion: O is an instance of class G.

premises:
    individual: O satisfies definition D.
    classification: Objects which satisfy definition D are
        classified as instances of class G.
```

11.8 Argument from Definition to Verbal Classification

```
id: definition-to-verbal-classification

strict: true

conclusion: O is an instance of class G.

premises:
   individual: O satisfies definition D.
   classification: Objects which satisfy definition D are
        classified as instances of class G.
```

11.9 Defeasible Modus Ponens

```
id: defeasible-modus-ponens

conclusion: B

premises:
    major: If A is true then presumably B is also true.
    minor: A
```

11.10 Argument from an Established Rule

```
id: established-rule

conclusion: C

premises:
```

```
    major: Rule R has conclusion C.
    minor: Rule R is applicable.

assumptions:
    CQ1: Rule R is valid.
```

11.11 Argument from Positive Consequences

```
id: positive-consequences

conclusion: Action A should be performed.

premises:
    major: Performing action A would have positive consequences.
```

11.12 Argument from Negative Consequences

```
id: negative-consequences

conclusion: Action A should not be performed.

premises:
    major: Performing action A would have negative consequences.
```

11.13 Argument from Practical Reasoning

```
id: practical-reasoning

conclusion: A1 should be performed.

premises:
    circumstances: S1 is currently the case.
    action: Performing A1 in S1 will bring about S2.
    goal: G would be realized in S2.
    value: Achieving G would promote V.

assumptions:
    CQ1: V is indeed a legitimate value.
    CQ2: Realizing G is a goal.
    CQ3: Action A1 is possible.

exceptions:
    CQ4: There exists an action that, when performed in S1, would
```

```
        bring about S2 more effectively than A1.
    CQ5: There exists an action that, when performed in S1,  would
        realize G more effectively than A1.
    CQ6: There exists an action that, when performed in S1, would
        promote V more effectively than A1.
    CQ7: Performing A1 in S1 would have side-effects
        which demote V or some other value.
```

11.14 Argument from Cause to Effect.

```
id: cause-to-effect

conclusion: Event E2 will occur.

premises:
    minor: An event E1 has occurred.
    major: Event E1 causes event E2.

exceptions:
    CQ1: An event E3 interferred with E1.
```

11.15 Argument from Correlation to Cause

```
id: correlation-to-cause

conclusion: Event E1 causes event E2.

premises:
    major: Events E1 and E2 are correlated.

assumptions:
    CQ1: There exists a theory explaining the correlation
        between E1 and E2.

exceptions:
    CQ2: E3 causes E1 and E2.
```

11.16 Argument from Sunk Costs

```
id: sunk-costs

conclusion: Action A should be performed.
```

```
premises:
    costs: The costs incurred performing A thus far are C.
    waste: The sunk costs of C are too high to waste.

assumptions:
    CQ1: Action A is feasible.
```

11.17 Argument from Appearance

```
id: appearance

conclusion: O is an instance of class C.

premises:
    minor: O looks like a C.
```

11.18 Argument from Ignorance

```
id: ignorance

conclusion: S

premises:
    major: S would be known if it were true.
    minor: S is known to be true.

exceptions:
    CQ1: The truth of S has not been investigated.
```

11.19 Argument from Abduction

```
id: abduction

conclusion: H

premises:
    observation: observed ?S
    explanation: Theory T1 explains S.
    hypothesis: T1 contains H as a member.

exceptions:
    CQ1: T2 is a more coherent explanation than T1 of S.
```

11.20 Ethotic Argument

```
id: ethotic

conclusion: S

premises:
    assertion: P asserts that S is true.
    trustworthiness: P is trustworthy.
```

11.21 Slippery Slope Argument

This version of the slippery slope scheme is intended to be used together with the argument from negative consequences schema, to derive the conclusion that the action should not be performed.

Notice that the scheme is represented by *two* Carneades schemes, one for the base case and one for the inductive step.

11.21.1 Base Case

```
id: slippery-slope-base-case

conclusion: Performing action A would have negative consequences

premises:
    realization: Performing A would realize event E.
    horrible costs: Event E would have horrible costs.
```

11.21.2 Inductive Step

```
id: slippery-slope-inductive-step

conclusion: Event E1 would have horrible consequences

premises:
    causation: Event E1 causes E2.
    consequences: Event E2 would haves horrible consequences.
```

12 Credits

- Conception and Design

  - Tom Gordon
  - Douglas Walton

- Programming
  - Pierre Allix
  - Stefan Ballnat
  - Tom Gordon
  - Matthias Grabmair

- Funding (EU Projects)
  - Standardized Transparent Representations in order to Extend Legal Accessibility (Estrella, FP6-IST-027655), 2006-2008.
  - Quality Platform for Open Source Software (Qualipso, FP6-IST-034763), 2006-2010.
  - Integrated Method for Policy making using Argument modelling and Computer assisted Text analysis (IMPACT, FP7-IST-247228), 2010-2012.
  - The MARKet for Open Source – An Intelligent Virtual Open Source Marketplace (MARKOS, FP7-ICT-317743), 2012-2015.

13 References

Atkinson, Katie, and Trevor J. M. Bench-Capon. 2007. "Practical reasoning as presumptive argumentation using action based alternating transition systems." *Artificial Intelligence* 171: 855–874.

Dung, Phan Minh. 1995. "On The Acceptability Of Arguments And Its Fundamental Role In Nonmonotonic Reasoning , Logic Programming And N-Persons Games." *Artificial Intelligence* 77: 321–357.

Gordon, Thomas F., Henry Prakken, and Douglas Walton. 2007. "The Carneades Model of Argument and Burden of Proof." *Artificial Intelligence* 171: 875–896.

Prakken, Henry. 2010. "An abstract framework for argumentation with structured arguments." *Argument & Computation* 1: 93–124.

Segaran, Toby. 2007. *Programming Collective Intelligence*. O'Reilly.

Walton, Douglas, Chris Reed, and Fabrizio Macagno. 2008. *Argumentation Schemes*. Cambridge University Press.

Wyner, Adam, Katie Atkinson, and Trevor Bench-Capon. 2012. "A Functional Perspective on Argumentation Schemes." In *Proceedings of the 9th International Workshop on Argumentation in Multi-Agent Systems (ArgMAS 2012)*, ed. Peter McBurney, Simon Parsons, and Iyad Rahwan, 203–222.