ArguMed - a template-based argument mediation system for lawyers

Bart Verheij

Department of Metajuridica
Universiteit Maastricht
P.O. Box 616
6200 MD Maastricht
The Netherlands

+31 43 3883048
bart.verheij@metajur.unimaas.nl
http://www.metajur.unimaas.nl/~bart/

This is the final manuscript of the following paper that has been presented at the JURIX '98 conference:

Verheij, Bart (1998). ArguMed - A Template-Based Argument Mediation System for Lawyers. *Legal Knowledge Based Systems. JURIX: The Eleventh Conference* (eds. J.C. Hage, T.J.M. Bench-Capon, A.W. Koers, C.N.J. de Vey Mestdagh and C.A.F.M. Grütters), pp. 113-130. Gerard Noodt Instituut, Nijmegen.

ArguMed - a template-based argument mediation system for lawyers

Bart Verheij

Department of Metajuridica, Universiteit Maastricht
P.O. Box 616, 6200 MD Maastricht, The Netherlands
bart.verheij@metajur.unimaas.nl, http://www.metajur.unimaas.nl/~bart/

Abstract

This paper introduces the ArguMed-system. It is an example of a system for computer-mediated defeasible argumentation, a new trend in the field of defeasible argumentation. In this research, computer systems are developed that mediate the process of argumentation by one or more users. Argument-mediation systems should be contrasted with systems for automated reasoning: the latter perform reasoning tasks for users, while the former play the more passive role of a mediator. E.g., mediation systems keep track of the arguments raised and of the justification status of statements.

The argumentation theory of the ArguMed-system is an adaptation of Verheij's CumulA-model, a procedural model of argumentation with arguments and counterarguments. In the CumulA-model, the defeat of arguments is determined by the structure of arguments and the attack relation between arguments. It is completely independent of the underlying language. The process-model is free, in the sense that it allows not only inference (i.e., 'forward' argumentation, drawing conclusions form premises), but also justification (i.e., 'backward' argumentation, adducing reasons for issues).

The ArguMed-system has been designed in an attempt to enhance the familiarity of the interface and the transparency of the underlying argumentation theory of its precursor, the Argue!-system. The ArguMed-system's user interface is template-based, as is currently common in window-style user interfaces. The user gradually constructs arguments, by filling in templates that correspond to common argument patterns. An innovation of the ArguMed-system is that it uses dedicated templates for different types of argument moves. Whereas existing mediation systems are issue-based (in the style of Rittel's well-known Issue-Based Information System), the ArguMed-system allows free argumentation, as in the CumulA-model. In contrast with the CumulA-model, which has a very general notion of defeat, defeat in the ArguMed-system is only of Pollock's undercutter-type. The system allows three types of argument moves, viz. making a statement, adding a reason and its conclusion, and providing an (undercutter-type) exception blocking the connection between a reason and a conclusion.

To put the ArguMed-system in context, it is compared with selected existing systems for argument mediation. The differences between the underlying argumentation theories and user interfaces are striking, which is suggested to be a symptom of the early stages of development of argument mediation systems. Given the lack of system evaluation by users in the field, the paper concludes with a discussion of the relevance of current research on computer-mediated defeasible argumentation. It is claimed that the shift of argument mediation systems from theoretical to practical tools is feasible, but can as yet not be made by system developers alone: a strong input from the research community is required.

1 The logic of law

Most lawyers have some awareness of logic, although the awareness is often limited. The logical connectives '… and …' and ' … or …' are known, and maybe even the ambiguous interpretation of a composite sentence of the form 'a and b or c' is familiar. Some might regard the connective 'if …, then …' as the abstract form of a legal rule and the rule of inference Modus Ponens as the general template of legal reasoning.

Why do lawyers pay so little attention to logic? The main problem is that logic in its classical appearances (such as propositional or predicate logic) is not sufficiently satisfying as a model of legal argument: it is too far apart from the argument forms that lawyers use in practice. In recent years, there has been a large amount of research on the development of logical tools for legal argument (see, e.g., the work of Gordon [1993, 1995], Hage [1997], Lodder [1998], Prakken [1993, 1997] and Verheij [1996]). Argument forms that have been studied include arguments concerning exceptions to rules, conflicts of reasons and rule applicability.

The logical tools that have recently been developed can be categorized under three headings: defeasibility, integration of logical levels, and the process character of argument [Verheij *et al*., 1997].

Defeasibility is a characteristic of arguments and, in a derived sense, of conclusions. A conclusion is defeasible if it is the conclusion of a defeasible argument. Defeat occurs if a conclusion is no longer justified by an argument because of new information. For instance, the conclusion that a thief should be punished is no longer justified if it turns out that there was a legal justification for the theft, such as an authorized command.

The *integration of logical levels* is for instance required if reasons are weighed. If arguments lead to incompatible conclusions, weighing of reasons is necessary to determine which conclusion follows. Additional information is necessary to determine the outcome of the weighing process. In some views, this information is on a higher logical level than the facts of cases, and the rules of law. However, since there can also be arguments about the weighing of reasons, the integration of levels is required.

The *process character of argument* also led to the development of new logical tools. For instance, the defeasibility of arguments cannot be separated from the process of taking new information into account. During the process of argumentation conclusions are drawn, reasons are adduced, counterarguments are raised, and new premises are introduced. In traditional models, only the end products of the process are modeled.

2 Argument mediation for lawyers

In the research on the logic of law, the focus has been primarily on the technical development of the logical tools, and only in the second place on their practical adequacy for modeling legal argument. Presently a convergence of opinions on the necessary logical tools takes shape, and a systematic practical assessment of the logical tools becomes essential.

In the research reported on in this paper, a step towards the practical assessment is made by the development of an experimental computer system for *argument mediation* for lawyers, the ArguMed-system. The development of computer-mediated defeasible argumentation is a relatively recent trend in the field of defeasible argumentation. In this research, computer systems are developed that can mediate the process of argumentation by one or more users. The systems can mediate the process in which arguments are drafted and generated by the users, e.g., by

- administering and supervising the argument process,
- keeping track of the issues that are raised and the assumptions that are made,
- keeping track of the reasons adduced and the conclusions drawn,
- keeping track of the counterarguments that have been adduced,
- evaluating the justification status of the statements made, and
- checking whether the users of the system obey the pertaining rules of argument.

It is crucial to note the difference between argument mediation systems and systems for automated reasoning. The latter automatically perform reasoning on the basis of the information in their 'knowledge base'. In this way, an automated reasoning system can do (often complex) reasoning tasks *for* the user. Argument mediation systems do not reason themselves; the goal of mediation systems is not to *replace* the user's reasoning, but to *mediate* the reasoning process of the user. As a result, argument mediation systems are more passive than automated reasoning systems. At the same time, in the development of argument mediation systems, the notorious difficulties of inherent complexities of the law (such as its open and dynamic nature) are less intense than for automated reasoning systems, since they can be left to the user.

Several experimental systems for the mediation of defeasible argumentation have been developed (e.g., IACAS by Vreeswijk [1995], Room 5 by Loui *et al.* [1997], Zeno by Gordon and Karacapilidis [1997], and DiaLaw by Lodder [1998]).[1] The systems differ in their goals, the underlying argumentation theories, and the user interfaces.

In this paper, an experimental system for legal argument mediation is described, the ArguMed-system. It has been developed as the successor of an earlier experimental argument mediation system, the Argue!-system[2]. The latter system has a graphical interface, in which the argument data structures are 'drawn' by the user. During the development of the Argue!-system, the idea emerged that the graphical interface would be too unfamiliar for the intended users, and that its underlying argumentation theory was not sufficiently transparent. A sample screen of a session with the Argue!-system is shown in Figure 1. The graphical interface and the underlying argumentation theory are hard to grasp immediately.

[1] This is only a selection, guided by two criteria: 1. the system must be meant for argument mediation, and 2. argumentation must be defeasible. Not mentioned are, for instance, Nute's [1988] d-Prolog, Pollock's [1995] OSCAR, Tarski's World by Barwise and Etchemendy (see http://csli-www.stanford.edu/hp/), and HUGIN (http://www.hugin.dk/). For an overview of argument models in law, see Bench-Capon [1995].

[2] Verheij [1998] most extensively describes the Argue!-system. Lodder and Verheij [1998] and Verheij and Lodder [1998] present Lodder's DiaLaw and Verheij's Argue!-system as examples of two approaches to argument mediation, viz. the verbal and the visual approach, respectively.

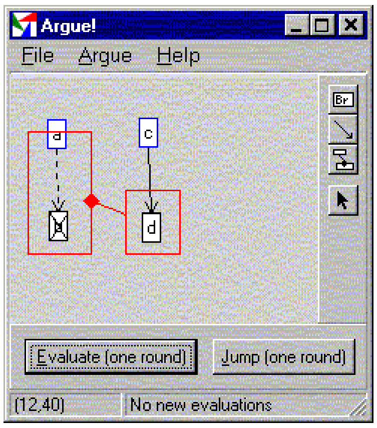


Figure 1: a sample screen of the Argue!-system, ArguMed's precursor

As a result, in the successor of the Argue!-system, both the familiarity of the user interface and the transparency of the argumentation theory would have to be enhanced.

In order to achieve this, the newly developed ArguMed-system presented here, has a *template-based* user interface: the user gradually constructs arguments, by filling in templates. It is expected that in this way the user interface becomes more familiar since filling in templates is common in present-day, window-style interfaces, and that the argumentation theory underlying the system becomes more transparent since the possible argument moves are restricted to a small number of common argument patterns, each accessible by a different, dedicated template.

Section 3 contains an introduction to CumulA, the procedural model of argumentation with arguments and counterarguments, which underlies the system. In section 4, the functionality and user interface of the ArguMed-system are described. Section 5 contains a sample session with the ArguMed-system. In section 6, the system is compared with selected related systems, especially with regard to their underlying argumentation theories and user interfaces. Section 7 discusses the relevance of computer-mediated defeasible argumentation. In section 8, a shift of argument mediation systems from theoretical to practical tools is suggested.

3 CumulA: a model of defeasible argumentation in stages

The argumentation theory underlying the ArguMed-system is an adapted version of Verheij's [1996] CumulA-model. Below, CumulA is briefly discussed. For an extensive discussion, both informally and formally, the reader is referred to Verheij [1996].

CumulA is a procedural model of argumentation with arguments and counterarguments. It is based on two main assumptions. The first assumption is that argumentation is a *process* during which arguments are constructed and counterarguments are adduced. The second assumption is that the arguments used in argumentation are *defeasible*, in the sense that whether they justify their conclusion depends on the counterarguments available at a *stage* of the argumentation process. If an argument no longer justifies its conclusion it is said to be defeated. The defeat of an argument is caused by a counterargument (that is itself undefeated).

For instance, if Peter has broken Ellen's window, a *prima facie* conclusion would be that Peter has the duty to repair the damages suffered by Ellen. The conclusion can be rationally justified, by giving *support* for it. E.g., the following *argument* could be given:

Peter has broken Ellen's window.
So, Peter has violated a property right of Ellen.
So, Peter has committed a tort against Ellen.
So, Peter has the duty to repair the damages suffered by Ellen.

An argument as above is a reconstruction of how a conclusion can be supported. It is assumed that an argument that supports its conclusion does not always justify it. For instance, if in the example it turns out that there is a ground of justification for Peter's act, e.g., in case he acted in *force majeure*, the conclusion that Peter has the duty to repair the damages suffered by Ellen, would no longer be justified. The argument has become *defeated*. For instance, the argument (in fact a statement) could be given:

Peter acted in *force majeure*.
So, there is a ground of justification for Peter's act.

In this case the argument that Peter has the duty to repair the damages suffered by Ellen, is defeated by the *counterargument* that there is a ground of justification for Peter's act. It is a counterargument of *undercutter-type*, as distinguished by Pollock [1987]: it blocks the connection between the reason that Peter has violated a property right of Ellen, and the conclusion that Peter has committed a tort against Ellen. An argument attacking the conclusion of another argument is said to be of *rebutter-type*.

As a result of the counterargument, the argument that Peter has committed a tort against Ellen, is *defeated*, i.e., does no longer justify its conclusion, and, as a result, also the argument that Peter has the duty to repair the damages suffered by Ellen, is defeated.

CumulA is a procedural model of argumentation with arguments and counterarguments, in which the defeat status of an argument, either undefeated or defeated, depends on:

(1) the structure of the argument;
(2) the attacks by counterarguments;
(3) the argumentation stage.

Each is briefly discussed below. The model especially builds on the work of Pollock [1987, 1995], Vreeswijk [1993, 1997] and Dung [1995] in philosophy and artificial intelligence, and was developed to complement the work on the model of rules and reasons Reason-Based Logic (see, e.g., Hage [1996, 1997] and Verheij [1996]).

In the CumulA-model, the structure of an argument is represented as in the argumentation theory of Van Eemeren and Grootendorst [1981, 1987]. Both the subordination and the coordination of arguments are possible. It is explored how the structure of arguments can lead to their defeat. For instance, the intuitions that it is easier to defeat an argument if it contains a longer chain of defeasible steps ('sequential weakening'), and that it is harder to defeat an argument if it contains more reasons to support its conclusion ('parallel strengthening'), are investigated.

In the model, *counterarguments* against other arguments are taken to be a primitive notion [cf. Dung, 1995]. For the example above, this means that the fact that the argument that there is a ground of justification for Peter's act, is a counterargument to the argument that Peter has committed a tort against Ellen, is explicitly represented in the system. This is in contrast with Vreeswijk's [1993, 1997] model, in which *conflicts of arguments* (i.e., arguments with conflicting conclusions) are the primitive notion. As a result, Vreeswijk's model can be considered as a generalization of rebutter-type defeat, and CumulA as a generalization of undercutter-type defeat.[3]

In CumulA, so-called *defeaters* indicate which arguments are counterarguments to other arguments, i.e., which arguments can defeat other arguments. In this way, CumulA shows that the defeasibility of arguments can fully be modeled in terms of argument structure and the attack relation between arguments, independent of the underlying language. Moreover, it turns out that defeaters can be used to represent a wide range of types of defeat, as proposed in the literature, e.g., Pollock's [1987] undercutting and rebutting defeat. Also some new types of defeat can be distinguished, namely defeat by sequential weakening (related to the well-known sorites paradox) and defeat by parallel strengthening (related to the accrual of reasons).

In the CumulA-model, argumentation stages represent the arguments and the counterarguments currently taken into account, and the status of these arguments, either defeated or undefeated. The model's lines of argumentation, i.e., sequences of stages, give insight in the influence that the process of taking arguments into account has on the status of arguments. For instance, by means of argumentation diagrams, which give an overview of possible lines of argumentation, phenomena that are characteristic for argumentation with defeasible arguments, such as the reinstatement of arguments, are explicitly depicted.

It is shown how in a line of argumentation not only new conclusions are inferred ('forward argumentation', or inference), but also new reasons are adduced ('backward argumentation', or justification). In other words, CumulA's process-model is *free*, as opposed to *premise-based* systems (that focus on inference, i.e., drawing conclusions from fixed premises, as, e.g., in Vreeswijk's [1993, 1997] model) and *issue-based* systems (that focus on justification, i.e., adducing reasons for a fixed issue).

To summarize, CumulA shows

(1) how the subordination and coordination of arguments is related to argument defeat;

[3] In my dissertation [Verheij, 1996], I spoke of *counterargument-triggered* vs. *inconsistency-triggered* defeat, to contrast CumulA and Vreeswijk's model. I am of the opinion that (Dung-style) counterargument-triggered defeat is philosophically most attractive and innovative. One reason is that counterargument-triggered defeat can be modeled completely independent of the language, as in the CumulA-model. In contrast, Vreeswijk's model is *almost* independent of the language, since he needs an element denoting contradiction. Another reason is that one of the central types of defeat, viz. defeat by an undercutter, is not inconsistency-triggered. In this respect, it is noteworthy that Vreeswijk needs to adapt his formalism by adding a conditional to his language in order to model undercutters.

(2) how the defeat of arguments can be described in terms of their structure, counterarguments, and the stage of the argumentation process, and independent of the logical language;
(3) how both inference and justification can be formalized in one model.

CumulA has obvious limitations. First, its underlying language is completely unstructured. It contains for instance no logical connectives, no quantifiers, and no modal operators. This is certainly a limitation, but one of the research objectives was to show that argument defeat can be fruitfully studied independent of the language. Second, the role of rules in argumentation in CumulA is not clarified. This is in part due to the first limitation: the language of CumulA does not contain a conditional, which can express rules.[4]

4 The ArguMed-system

The ArguMed-system is a system for the mediation of argument with a template-based interface. The user gradually constructs arguments, by filling in templates that correspond to argument patterns. The system keeps track of the constructed arguments and of the justification status of the statements made.

There are three basic *argument moves*: making a statement, adding a reason and its conclusion, and providing an (undercutter-type) exception blocking the connection between a reason and its conclusion.

The ArguMed-system has three central *data structures*. The elementary data structure of the ArguMed-system is that of a *statement*. It consists of a *sentence* that represents the propositional content of the statement. Statements can be of two types, viz. of *issue-type* and of *assumption-type*. The former are the topic of the argumentation at hand: whether a statement of issue-type is justified is determined by the user's line of argumentation. Statements of assumption-type form the basis of the user's argumentation: their justification is taken for granted.

A second data structure of the ArguMed-system is that of an *argument*, which is simply a tree of statements. The child nodes of each statement node in the tree represent the reasons for the statement.

A third data structure combines *reasons* with their *conclusions*: it consists of two statements, one of which represents a reason, the other the conclusion supported by the reason. Exceptions (of undercutter-type) that can block the connection between the reason and the conclusion, are also contained in this data structure.

Given the data available, each statement has a *justification status*: it is justified, unjustified or neither. The justification status of an statement must obey the following constraints:[5]

1. If a statement is of assumption-type, it is justified.
2. If a statement is of issue-type, it is
   a. justified in case there is a justified reason for the statement, and there is no justified exception that blocks the connection between them.
   b. unjustified in case there is a justified reason for the statement, and there is a justified exception that blocks the connection between any such reason and the statement.
   c. neither justified nor unjustified in case there is no justified reason for it.

It can be shown that, under restrictions, the above give a well-defined definition of the justification status of statements.[6] After each argument move, the ArguMed-system computes the justification status of all statements.[7]

The opening screen of the ArguMed-system is shown in Figure 2.

---

4 A formal model in which rules play a central role is Reason-Based Logic (see Hage [1993, 1996, 1997], Verheij {1996], and Verheij *et al.* [1998]).

5 Pollock [1995] and several others have given related constraints.

6 E.g., the restriction that the number of 'attack sequences' is finite, suffices. Here an attack sequence is a sequence of arguments A_1, A_2, ..., A_n, such that each argument in it is attacked by its successor. An argument A is said to attack another argument A' in case the conclusion of A is an exception to one of the reason/conclusion-connections in the argument A'. The difficulties arising by 'attack loops' are excluded by this restriction. These are irrelevant for present purposes.

7 As said, the constraints do not always provide a well-defined definition of the justification status of statements. The ArguMed-system is not designed to deal with the deviant cases. If the constraints allow no status assignment, or more than one, the algorithm used by the ArguMed-system loops, or gives one of the possibilities.

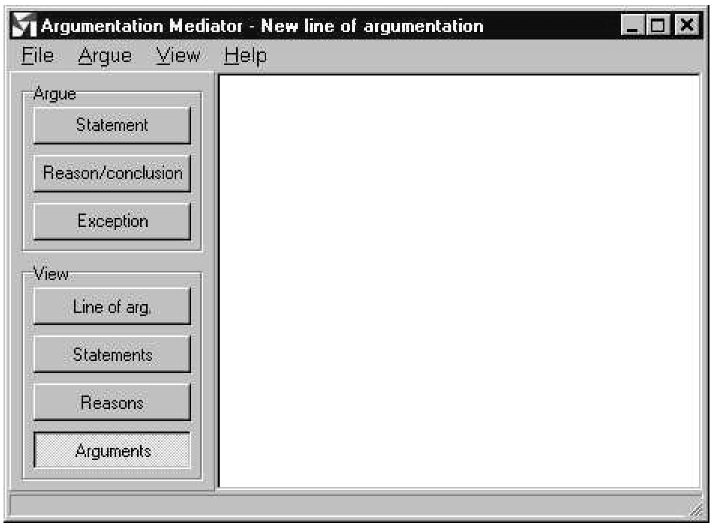


Figure 2: the opening screen of the ArguMed-system

Each of the three 'Argue'-buttons gives access to one of the three argument *templates*, provided by the ArguMed-system, each corresponding to one of the argument moves of the system. To perform an argument move, the user fills in a template. The first template is the *statement-template* (Figure 3). It allows the input of a statement: the user can type a sentence and choose the statement's type. For new statements, the issue-type is selected by default. The template can also be used to change the type of a statement added at a previous stage.

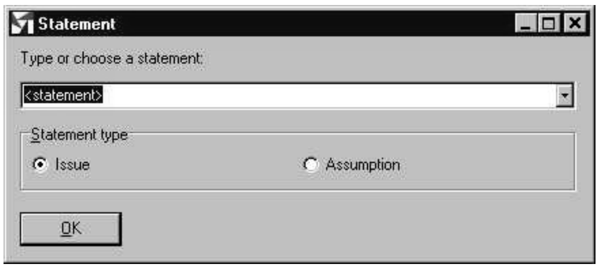
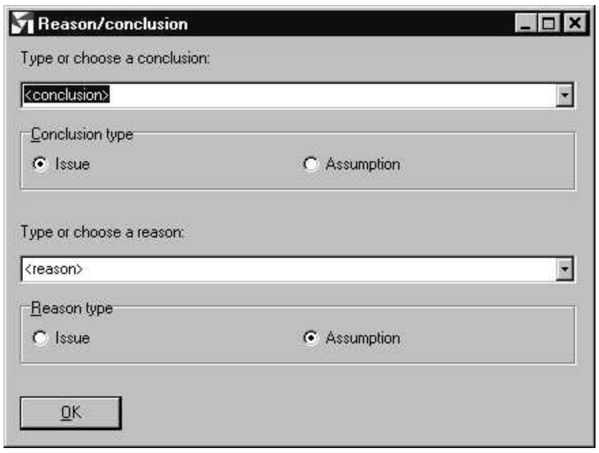


Figure 3: the statement-template

The second is the *reason/conclusion-template* (Figure 4). It allows the input of a reason, and a conclusion supported by the reason. Both the reason and the conclusion can be new statements, or can be selected from statements added at a previous stage.

For a new conclusion, the issue-type is selected by default, for a new reason, the assumption-type. The intuition behind the latter default choice is that a reason is normally given as the immediate justification of a conclusion, and only a justified reason, such as a reason of assumption-type, can provide such support. If a reason is itself of issue-type, it can only indirectly justify its conclusion, viz. if the reason is supported by another (justified, non-blocked) reason.


Reason/conclusion

Type or choose a conclusion: <conclusion>

Conclusion type: [✓ Issue] Assumption

Type or choose a reason: <reason>

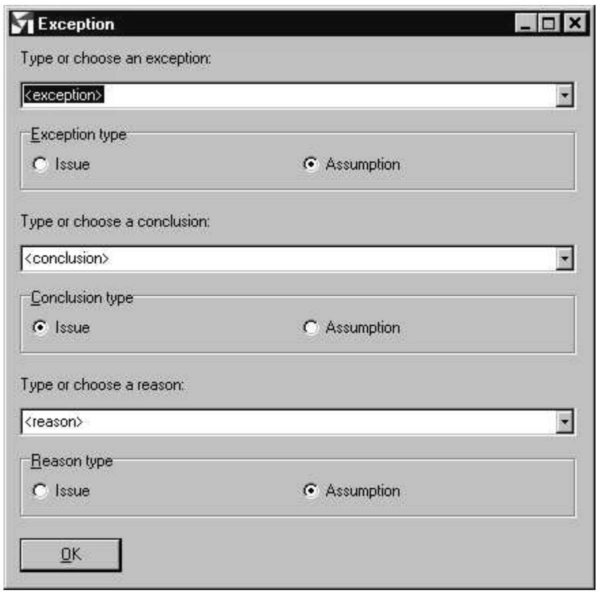
Reason type: Issue [✓ Assumption]

OK

Figure 4: the reason/conclusion-template

The third is the *exception-template* (Figure 5). It allows the input of an (undercutter-type) exception, and the reason and the conclusion, the connection of which is blocked by the exception. The user provides three statements, viz. the exception, the reason and the conclusion. Each can be new, or selected from the previously added statements.

For a new exception, the assumption-type is selected by default. The intuition behind this choice is that an exception normally is meant as an immediate block of the connection between the reason and the conclusion, and only a justified exception is such a block. If the exception is of issue-type, it only blocks the connection between the reason and the conclusion if it is itself supported by a justified, non-blocked reason. For a new conclusion and reason, the default types are the same as in the reason/conclusion-template.


Exception

Type or choose an exception: <exception>

Exception type: Issue [✓ Assumption]

Type or choose a conclusion: <conclusion>

Conclusion type: [✓ Issue] Assumption

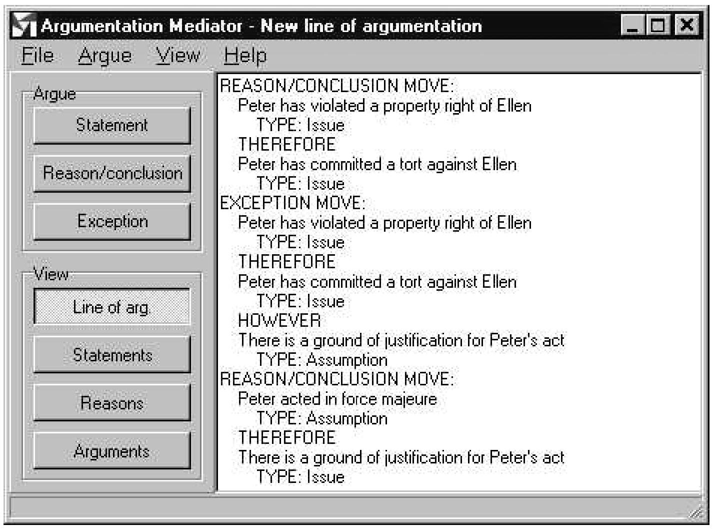
Type or choose a reason: <reason>

Reason type: Issue [✓ Assumption]

OK

Figure 5: the exception-template

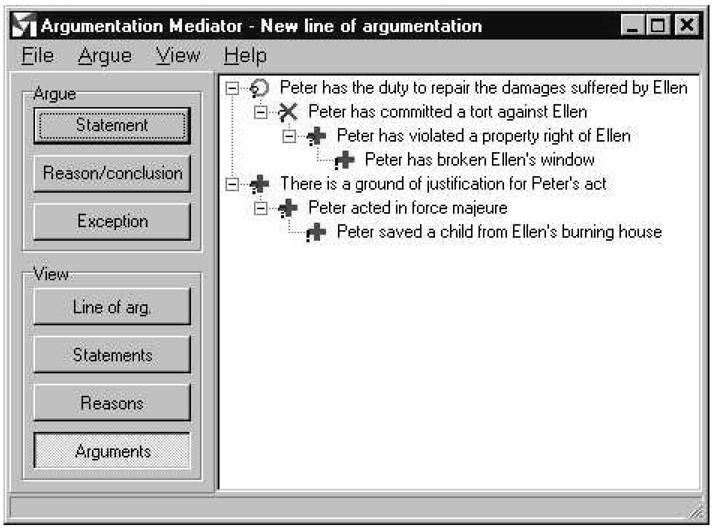
The ArguMed-system provides four *views*, providing information about the current argumentation session. Each view is accessible by one of the four 'View'-buttons (see Figure 2). In the 'line of argumentation'-view, the argument moves as performed by the user are listed (Figure 6).



```
REASON/CONCLUSION MOVE:
  Peter has violated a property right of Ellen
    TYPE: Issue
  THEREFORE
  Peter has committed a tort against Ellen
    TYPE: Issue
EXCEPTION MOVE:
  Peter has violated a property right of Ellen
    TYPE: Issue
  THEREFORE
  Peter has committed a tort against Ellen
    TYPE: Issue
  HOWEVER
  There is a ground of justification for Peter's act
    TYPE: Assumption
REASON/CONCLUSION MOVE:
  Peter acted in force majeure
    TYPE: Assumption
  THEREFORE
  There is a ground of justification for Peter's act
    TYPE: Issue
```

Figure 6: the 'line of argumentation'-view

In the 'statements'-view, all statements made by the user are presented. In the 'reasons'-view, the reasons, as currently added by the user, are shown with their conclusions. Also the exceptions blocking the connection between the reason and the conclusion are shown in this view. In the 'arguments'-view, the arguments that can be constructed on the basis of the current user input, are shown (Figure 7). The reasons supporting a conclusion are shown as 'children' of the conclusion. The type of the statements is visualized as follows: a question mark indicates a statement of issue-type, an exclamation-mark a statement of assumption-type. The justification status is also shown: a (green) plus indicates a statement that is justified, a (red) cross a statement that is unjustified, and a (gray) circle a statement that is neither justified nor unjustified.



```
Peter has the duty to repair the damages suffered by Ellen
  Peter has committed a tort against Ellen
    Peter has violated a property right of Ellen
      Peter has broken Ellen's window
There is a ground of justification for Peter's act
  Peter acted in force majeure
    Peter saved a child from Ellen's burning house
```

Figure 7: the 'arguments'-view

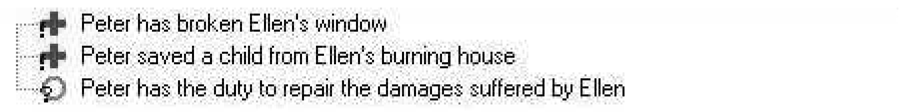
The argumentation theory underlying the ArguMed-system is an adaptation of the CumulA-model. There are three main differences. First, only defeaters representing undercutter-type exceptions are allowed. It would be in conflict with the choice to make the underlying argumentation theory of the system as transparent as possible, if the defeaters of CumulA would be allowed in their full generality. Some defeaters of CumulA have too abstract applications (e.g., defeat by sequential weakening is applied to the sorites paradox, which is mostly of technical-philosophical interest). Defeaters with more concrete applications could be added in the future. For instance, defeaters for rebutter-type exceptions and for the weighing of reasons are good candidates. Second, argument moves are central in the ArguMed-system, and not argument stages, as in CumulA. In the ArguMed-system, the argument stages of the lines of argumentation are computed by the system on the basis of argument moves.

Third, both the arguments and the defeaters can be added dynamically in the ArguMed-system, whereas in CumulA the defeaters are part of the initial data.[8]

5 A sample session with the ArguMed-system

In the following, a sample session, using an elementary example from the domain of Dutch tort law, illustrates the ArguMed-system. Assume that Peter has saved a child from Ellen's burning house, but that he has broken one of the house's windows in order to enter her house. The question is whether Peter has the duty to repair the damages suffered by Ellen.

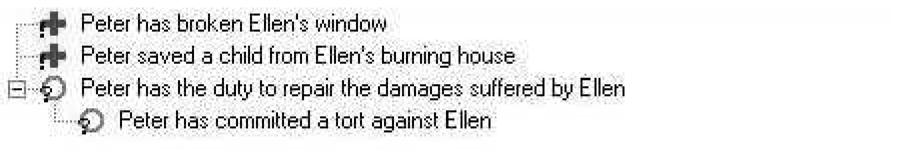
Initially, the user makes three statements using the statement-template. The case facts that Peter has broken Ellen's window and that he saved a child from Ellen's burning house, are added as assumptions, the legal consequence in question is added as an issue. The result is shown in the 'arguments'-view:



```
Peter has broken Ellen's window
Peter saved a child from Ellen's burning house
Peter has the duty to repair the damages suffered by Ellen
```

The two assumption-type statements are justified (indicated by the plus), the issue-type statement is neither justified nor unjustified (as indicated by the circle).

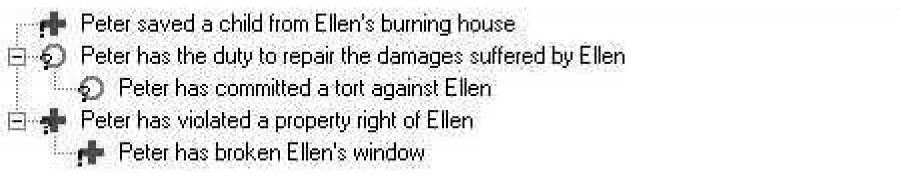
The user recalls that the duty to repair damages can be the result of a tort, and adds a reason using the reason/conclusion-template:



```
Peter has broken Ellen's window
Peter saved a child from Ellen's burning house
Peter has the duty to repair the damages suffered by Ellen
    Peter has committed a tort against Ellen
```

At this point, the user has left open whether Peter has committed a tort against Ellen. He has added it as an issue, and further justification is required. As a result, the reason that Peter has committed a tort against Ellen, does currently not justify the conclusion that Peter has the duty to repair the damages.

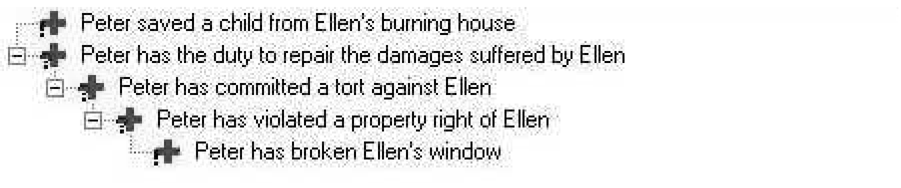
Next, the user adds a conclusion to the statement that Peter has broken Ellen's window, viz. that Peter has violated a property right of Ellen:



```
Peter saved a child from Ellen's burning house
Peter has the duty to repair the damages suffered by Ellen
    Peter has committed a tort against Ellen
Peter has violated a property right of Ellen
    Peter has broken Ellen's window
```

The conclusion that Peter has violated a property right of Ellen, is justified, since the statement that Peter has broken Ellen's window, is justified, and there are no exceptions.

Since in Dutch tort law, violations of property rights are (*prima facie*) torts, the user can close the gap between the conclusion that Peter has committed a tort against Ellen and the reason that he has violated a property right of Ellen, using the reason/conclusion-template:



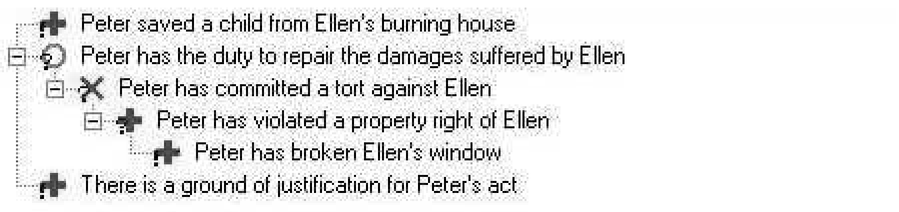
```
Peter saved a child from Ellen's burning house
Peter has the duty to repair the damages suffered by Ellen
    Peter has committed a tort against Ellen
        Peter has violated a property right of Ellen
            Peter has broken Ellen's window
```

The argument that Peter has violated a property right of Ellen, has been connected (by subordination) with the argument starting from the premise that Peter has committed a tort against Ellen.

At the present stage of the line of argumentation, the conclusion that Peter has the duty to repair the damages suffered by Ellen, is justified, since there is a justified support for it, and there are no exceptions.

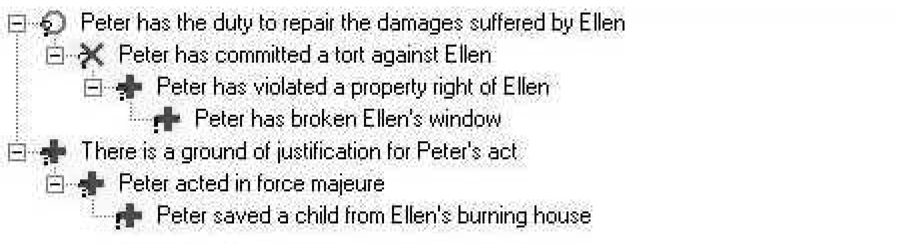
[8] Minor differences include slightly different terminology and the abstraction from the subreasons of reasons.

In this case, however, the user argues that there is a ground of justification for Peter's act, blocking the connection between the reason that Peter has violated a property right of Ellen and the conclusion that he has committed a tort against her. The user's reasoning is warranted by Dutch tort law, where violations of property rights are torts unless there is a ground of justification. The user adds the appropriate exception using the exception-template:


Peter saved a child from Ellen's burning house
Peter has the duty to repair the damages suffered by Ellen
Peter has committed a tort against Ellen
Peter has violated a property right of Ellen
Peter has broken Ellen's window
There is a ground of justification for Peter's act

The conclusion that Peter has committed a tort against Ellen, has become unjustified (indicated by the cross) since there is a justified reason for it, that is blocked by a (justified) exception. As a result, the conclusion that Peter has the duty to repair the damages suffered by Ellen, has become neither justified nor unjustified. It should be noted that in the present view (the 'arguments' view) it is not indicated which statement is the exception that makes the conclusion that Peter has committed a tort against Ellen, unjustified. This information is for instance shown in the 'reasons'-view.

The user finishes the line of argumentation by arguing that the ground of justification is supported by the case fact that Peter saved a child from Ellen's burning house, since Peter acted in *force majeure*:


Peter has the duty to repair the damages suffered by Ellen
Peter has committed a tort against Ellen
Peter has violated a property right of Ellen
Peter has broken Ellen's window
There is a ground of justification for Peter's act
Peter acted in force majeure
Peter saved a child from Ellen's burning house

6 A comparison of argument-mediation systems

In order to put the ArguMed-system in context, it is briefly compared to four other systems, namely IACAS by Vreeswijk [1995], Room 5 by Loui *et al.* [1997], Zeno by Gordon and Karacapilidis [1997], and DiaLaw by Lodder [1998].[9] First, the underlying argumentation theories are discussed; second, the user interfaces.

6.1 The underlying argumentation theories

In the underlying argumentation theories of all four systems argumentation is *dynamic*. Statements can be made, and reasons can be adduced. In IACAS, Room 5, Zeno and DiaLaw, argumentation is *issue-based* (as in Rittel's well-known Issue-Based Information System (IBIS) [Rittel and Webber's, 1973]). No new conclusions can be drawn, since these systems focus only on justification of an initial central issue. In the ArguMed-system, argumentation is *free*, in the sense that there is no central issue, and both inference (i.e., 'forward' argumentation, drawing conclusions from premises) and justification (i.e., 'backward' argumentation, adducing reasons for issues) are allowed. Also connecting previously made arguments (e.g., by subordination, as in the sample session) is only possible in the ArguMed-system.

All systems model a notion of *defeasibility* of argumentation. IACAS, Room 5, Zeno and DiaLaw have a notion of *reasons for and against conclusions*. In Zeno and DiaLaw, weighing the conflicting reasons determines which conclusions are justified. DiaLaw and the ArguMed-system have an *undercutter-type exception*.

IACAS and DiaLaw have a notion of the *rules* underlying argument steps. DiaLaw is based on the theory of rules and reasons Reason-Based Logic (see, e.g., Hage [1996, 1997] and Verheij [1996]).

In Room 5, Zeno and DiaLaw, argumentation is considered as a *game with participants*. In Room 5 and Zeno, the game character is left *implicit*, but obtained by the distributed access to the systems, on the World-Wide Web. In DiaLaw, the game character is made *explicit* in the form of a dialogue game with two parties. IACAS and the ArguMed-system have no explicit notion of participants, but can be considered as one-participant games.

[9] See note 1.

IACAS, Zeno and the ArguMed-system are *evaluative*: the status of statements and arguments can be determined by the system.

6.2 The user interfaces

IACAS and DiaLaw have a *command-line-based interface*; moves are typed at a command-prompt. Room 5, Zeno and the ArguMed-system have a *template-based interface*: users fill in forms to perform an argument move. The ArguMed-system innovates this type of interface by using different templates for different types of moves.

Room 5, Zeno and the ArguMed-system present arguments in a *visual* manner. Zeno and the ArguMed-system use a *tree*-like presentation. Room 5 uses a clever system of *boxes-in-boxes* in an attempt to avoid 'pointer-spaghetti' (as in ArguMed's precursor, the Argue!-system; see Figure 1). In IACAS and DiaLaw, argumentation is presented in a *verbal* manner, since they have a command-line interface.

In Room 5 and Zeno, counterarguments (formed by reasons against conclusions) are *grouped together* in the visual argument structure. In the ArguMed-system, counterarguments (the exceptions) are shown in a special view. In IACAS, the system can generate a list of arguments and counterarguments. In DiaLaw, counterarguments are not directly accessible.

In DiaLaw and the ArguMed-system, the dynamic aspect of argumentation is shown by a *view on the sequence of moves*. In IACAS, Room 5, Zeno, only a *view on the current stage* of the argumentation process is visible. In Room 5 and the ArguMed-system, it is possible to switch between different views showing different types of information.

6.3 Conclusions

The main conclusion that can be drawn from the above discussion is that the argumentation theories underlying the argument mediation systems and their user interfaces are strikingly different. As yet, it is unclear which choices should be preferred.

With respect to the underlying argumentation theory, an issue-based argument mediation system has the advantage that the process of argumentation has a focus, which can be useful, or even necessary (e.g., in a game-like situation). However, a system that is not issue-based (such as the system presented in this paper) is more general and adds flexibility, since it not only allows justification (i.e., adducing reasons for issues), but also inference (i.e., drawing conclusions from premises). Current argument mediation systems have very different notions of defeasibility. It seems necessary to strive for integration, or explicitly defend choices. It is remarkable that none of the discussed systems with a visual, window-style interface has a notion of rules.

With respect to the user interfaces, the argument mediation systems with visual, window-style interfaces seem to be significantly more user-friendly than those with command-line-based interfaces. In this paper, it is posited (but not supported by evaluation with users) that a template-based interface is enhanced if different templates are used for different types of argument moves. Supposedly, the choice of the available argument moves that are available to the user, will be crucial for user acceptance.

The need for evaluation by users is obvious, both of the underlying argumentation theories and of the interfaces of the argument mediation systems. Since such evaluation is lacking, some remarks are in place on the relevance of computer-mediated defeasible argumentation, given the current state of research in the field.

7 The relevance of computer-mediated defeasible argumentation

In my opinion, the following points make the development of implemented systems of computer-mediated defeasible argument worthwhile:

- Implemented systems for computer-mediated defeasible argumentation are *realizations* of formal models of defeasible argumentation. As a result, they provide an existence proof: they show that implementing the formal theory is feasible.
- Implemented systems for computer-mediated defeasible argumentation are *test beds* for formal models of defeasible argumentation. They are tools for experimentation, both technically and philosophically.
- Implemented systems for computer-mediated defeasible argumentation can be *showcases*, giving formal models more credibility. There is a proviso here: the system's 'look and feel' must be good.
- Implemented systems for computer-mediated defeasible argumentation can be *practical aids*. Applications in law, decision making, planning and education are among the ambitions, which are even already partially met. One can think of the drafting of court pleadings and motivations, and the training of argumentation skills.

I think these points make the development of computer-mediated defeasible argumentation currently relevant for researchers (and not only for system developers). This holds not only for the first two more theoretically oriented points, but also for the second two more practically oriented ones. As yet, system developers alone cannot achieve the development of showcases of defeasible argumentation and practical aids. In my opinion, the early stage of development of computer-mediated defeasible argumentation necessitates a strong input from the research community.

8 Argument mediation systems for lawyers: from theoretical to practical tools

The recent advances in the theory of legal argument, especially with respect to defeasibility, integration of logical levels, and the process character of argument require a practical assessment. One way towards such assessment is to build usable systems for argument mediation. In this paper, an experimental system, the ArguMed-system, has been presented, and briefly compared to selected related systems, namely IACAS, Room 5, Zeno, and DiaLaw. The differences between the underlying argumentation theories and user interfaces are striking, which is a symptom of the early stages of development of argument mediation systems.

On the one hand, current argument mediation systems seem not yet sufficiently mature to be used as *practical* tools by practicing lawyers. On the other hand, they already turn out useful as *theoretical* tools, and help to enhance argumentation theory. The move from theoretical to practical tools will take serious effort, both by researchers and by system developers, but seems manageable for the near future. When this move is made, the evaluation by users finally comes within reach.

Acknowledgments

The author gladly acknowledges the financial support by the Dutch National Programme Information Technology and Law (ITeR) for the research reported in this paper (project number 01437112). He also thanks Arno Lodder, Jaap Hage and Gerard Vreeswijk for comments and discussion.

References

Bench-Capon, T. (1995). Argument in Artificial Intelligence and Law. *Legal knowledge based systems. Telecommunication and AI & Law* (eds. J.C. Hage, T.J.M. Bench-Capon, M.J. Cohen and H.J. van den Herik), pp. 5-14. Koninklijke Vermande, Lelystad.

Dung, P.M. (1995). On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and *n*-person games. *Artificial Intelligence*, Vol. 77, pp. 321-357.

Eemeren, F.H. van, Grootendorst, R., and Kruiger, T. (1981). *Argumentatietheorie*. Uitgeverij Het Spectrum, Utrecht.

Eemeren, F.H. van, Grootendorst, R. and Kruiger, T. (1987). *Handbook of Argumentation Theory. A Critical Survey of Classical Backgrounds and Modern Studies*. Foris Publications, Dordrecht. Translation of van Eemeren *et al.* (1981).

Gordon, T.F. (1993). The Pleadings Game. *An Artificial Intelligence Model of Procedural Justice*. Dissertation.

Gordon, T.F. (1995). The Pleadings Game. *An Artificial Intelligence Model of Procedural Justice*. Kluwer Academic Publishers, Dordrecht.

Gordon, T.F., and Karacapilidis, N. (1997). The Zeno Argumentation Framework. *The Sixth International Conference on Artificial Intelligence and Law. Proceedings of the Conference*, pp. 10-18. ACM, New York (New York).

Hage, J. (1993). Monological reason based logic. A low level integration of rule-based reasoning and case-based reasoning. *The Fourth International Conference on Artificial Intelligence and Law. Proceedings of the Conference*, pp. 30-39. ACM, New York (New York). Also published as report SKBS/B3.A/93-08.

Hage, J. (1996). A Theory of Legal Reasoning and a Logic to Match. *Artificial Intelligence and Law*, Vol. 4, pp. 199-273.

Hage, J.C. (1997). *Reasoning with Rules. An Essay on Legal Reasoning and Its Underlying Logic*. Kluwer Academic Publishers, Dordrecht.

Lodder, A.R. (1998). *DiaLaw – on legal justification and dialog games*. Dissertation, Universiteit Maastricht.

Lodder, A.R., and Verheij, B. (1998). Opportunities of computer-mediated legal argument in education. *Proceedings of the BILETA-conference.* March 27-28, 1998, Dublin.

Loui, R.P., Norman, J., Altepeter, J., Pinkard, D., Craven, D., Lindsay, J., and Foltz, M. (1997). Progress on Room 5. A Testbed for Public Interactive Semi-Formal Legal Argumentation. *The Sixth International Conference on Artificial Intelligence and Law. Proceedings of the Conference*, pp. 207-214. ACM, New York (New York).

Nute, D. (1988). Defeasible reasoning: a philosophical analysis in Prolog. *Aspects of Artificial Intelligence* (ed. James H. Fetzer), pp. 251-288. Kluwer Academic Publishers, Dordrecht.

Pollock, J.L. (1987). Defeasible reasoning. *Cognitive Science*, Vol. 11, pp. 481-518.

Pollock, J.L. (1995). *Cognitive Carpentry: A Blueprint for How to Build a Person.* The MIT Press, Cambridge (Massachusetts).

Prakken, H. (1993). *Logical tools for modelling legal argument.* Doctoral thesis, Vrije Universiteit, Amsterdam.

Prakken, H. (1997). *Logical Tools for Modelling Legal Argument. A Study of Defeasible Reasoning in Law.* Kluwer Academic Publishers, Dordrecht.

Rittel, H.W.J., and Webber, M.M. (1973). Dilemmas in a general theory of planning. *Policy Sciences* 4.

Verheij, B. (1996). *Rules, Reasons, Arguments. Formal studies of argumentation and defeat*. Dissertation Universiteit Maastricht. A summary and table of contents are available on the World-Wide Web at http://www.metajur.unimaas.nl/~bart/proefschrift/.

Verheij, B. (1998). Argue! - an implemented system for computer-mediated defeasible argumentation. To be presented on the *Tenth Netherlands/Belgium Conference on Artificial Intelligence (NAIC '98)*, November 18-19, 1998, Amsterdam.

Verheij, B., Hage, J.C., and Lodder, A.R. (1997). Logical tools for legal argument: a practical assessment in the domain of tort. *The Sixth International Conference on Artificial Intelligence and Law. Proceedings of the Conference*, pp. 243-249. ACM, New York (New York). An abstract is available on the World-Wide Web at http://www.metajur.unimaas.nl/~bart/papers/icail97.htm.

Verheij, Bart, Hage, Jaap C., and Herik, H. Jaap van den (1998). An integrated view on rules and principles. *Artificial Intelligence and Law*, Vol. 6, No. 1, pp. 3-26.

Verheij, B., and Lodder, A.R. (1998). Computer-mediated legal argument: the verbal vs. the visual approach. *Proceedings of the 2nd French-American Conference on AI and Law*. June 11-12, 1998, Nice.

Vreeswijk, G.A.W. (1993). *Studies in defeasible argumentation.* Doctoral thesis, Vrije Universiteit, Amsterdam.

Vreeswijk, G. (1995). IACAS: an Implementation of Chisholm’s Principles of Knowledge. *Dutch/German Workshop on Nonmonotonic Reasoning. Proceedings of the Second Workshop*, pp. 225-234. Delft University of Technology, Universiteit Utrecht.

Vreeswijk, G.A.W. (1997). Abstract argumentation systems. *Artificial Intelligence*, Vol. 90, pp. 225-279.