Automated Argument Assistance 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

The ArguMed-system is presented, which is an example of an argument-assistance system. Its goal is to assist the user while making statements, adducing reasons, inferring conclusions and providing exceptions.

Automated argument assistance should be distinguished from automated reasoning: while automated reasoning systems *replace* the reasoning of the users, argument-assistance systems do not reason themselves, but are tools assisting the users' reasoning.

ArguMed is the successor of the Argue!-system. Argue!'s graphical interface was considered too unfamiliar for the intended users, and its underlying argumentation theory was not sufficiently transparent.

The system as described here is the second version of the ArguMed-system. Two drawbacks of the previous version have been solved. First, though the arguments were presented graphically, argument attack was not. It was graphically shown *that* an argument was defeated by an attacking argument, but not *by which* argument. Second, in the argumentation theory underlying ArguMed's first version, it was not possible to put at issue that a particular statement was a reason for another statement, or that a statement was an exception.

Solving the first of these two drawbacks has led to a new graphical representation of the arguments, in which argument attacks are shown, and to a change in the argumentation theory, viz. the introduction of a novel notion of an argument, viz. that of a *dialectical* argument. Briefly, a dialectical argument is an argument in which attacks (and counterattacks) are incorporated. Solving the second drawback has led to the introduction of *step warrants* and *undercutter warrants* into the argumentation theory. The resulting notion of a *warranted dialectical argument* is the analog for defeasible argumentation of the notion of a (Hilbert-style) proof of classical logic.

The present version of the ArguMed-system is put in context by a brief comparison with selected other systems, viz. Loui's Room 5, Gordon and Karacapilidis' Zeno, ArguMed's precursor Argue! and the previous version of ArguMed.

1. ASSISTING LEGAL ARGUMENT

1.1 Argument assistance systems

In recent research in legal information technology, a number of experimental *argument assistance systems* have been presented, i.e., systems that can assist the argumentation of one or several users. One can think of a lawyer who uses an argument-assistance system in order to draft his pleading in court. Such a system could be part of the lawyer's word processing package, and provide assistance, e.g., by helping the lawyer to structure his unpolished arguments, and by providing tools for analyzing the arguments. Argument-assistance systems can also serve in a context of more than one user: so-called *argument-mediation* systems can be used to keep track of the diverging positions with regards to a public issue, and assist in the evaluation of the opinions.[1]

More specifically, argument-assistance systems are aids to draft and generate arguments, 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, the conclusions drawn, and 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.

Marshall (1989) speaks in a similar vein of tools to support the formulation, organization and presentation of arguments.

Argument-assistance systems must be distinguished from the more common automated reasoning systems. 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-assistance systems do not (or not primarily) reason themselves; the goal of assistance systems is not to *replace* the user's reasoning, but to *assist* in the reasoning process of the user.

The different nature of argument-assistance systems and automated reasoning systems has two consequences. First, argument-assistance systems are more passive than automated reasoning systems. Several of their functions are implicitly available, or operate 'in the background'. For instance, the evaluation of argument data, such as the currently justified statements, can occur in the background, much like the spelling checks of recent word processing systems: after each action by the user, the argument-assistance system automatically updates previous evaluations.

Second, in the development of argument mediation systems, the notorious difficulties of the inherent complexities of the law (such as its open and dynamic nature) are less intense than for

ICAIL-99 Oslo, Norway Copyright (c) 1999 ACM 1-58113-165-8/99/6 $5.00

[1] Previously, e.g., [Verheij, 1998a], I used the term 'argument-mediation systems' as the generic term for both single and multi-user argument-assistance systems. Examples of the latter are Room 5 by Loui *et al.* [1997] and Zeno by Gordon and Karacapilidis [1997]. Since mediation suggests that several users are involved, I now prefer the term 'argument-assistance systems'. The name of the ArguMed-system, that I would no longer call an argument-mediation system, but a single-user argument-assistance system, was chosen before the change of terminology.

automated reasoning systems, since they can to a large extent be left to the user.

Several experimental systems for legal argument assistance have been developed (e.g., Room 5 by Loui *et al*. [1997], Zeno by Gordon and Karacapilidis [1997], and the Argue!-system by Verheij [1998a]).[2] The systems differ in their goals, the underlying argumentation theories, and the user interfaces.

It should be noted that, for now, the development of argument-assistance systems is still mainly in an experimental phase. A first difficulty is the lack of a canonical theory of legal argumentation. However, in the recent abundance of research on legal argumentation,[3] a slow convergence of opinions seems to arise. A second difficulty is that argument-assistance systems require the design of user interfaces of a new kind. There is little experience with the way arguments can be sensibly and clearly presented to the users, or with the way argument moves should be performed by the user. Difficulties such as these could be the cause of the striking differences between the argumentation theories and user interfaces of argument-assistance systems (cf. section 4).

1.2 The ArguMed-system

In this paper, an experimental system for legal argument assistance is described, the ArguMed-system. The ArguMed-system has been developed as the successor of an earlier experimental argument-assistance system, the Argue!-system[4]. The latter system has a graphical interface, in which the user 'draws' the argument data on the screen. A sample screen of a session with the Argue!-system is shown in Figure 1. The screen is intended to depict that the (*prima facie*) reason that Peter has violated a property right does not imply its conclusion that Peter has committed a tort, as a result of the exception[5] that there is a ground of justification for Peter's act.

During the development of the Argue!-system, it became clear that the graphical interface was too unfamiliar for the intended users, and that its underlying argumentation theory was not sufficiently transparent.

In order to enhance both the familiarity of the user interface and the transparency of the argumentation theory, the newly developed ArguMed-system presented here, has a *template-based*

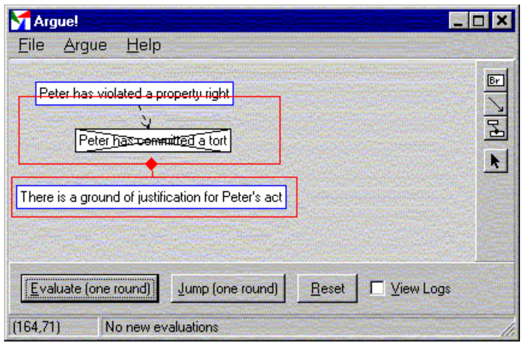

Argue!
File
Argue
Help
Peter has violated a property right
Peter has committed a tort
There is a ground of justification for Peter's act
Evaluate (one round)
Jump (one round)
Reset
View Logs
(164,71)
No new evaluations

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

user interface: the user gradually constructs arguments, by filling in templates, each corresponding to an argument move, such as making a statement. 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.

In fact, the system described in this paper is the second version of the ArguMed-system - the first version has recently been described by Verheij [1998b].

There are two significant differences between the first and second versions of the system. First, the present system uses a novel notion of an argument, viz. that of a *dialectical* argument. Briefly, a dialectical argument is an argument in which counterarguments (based on undercutting exceptions) are incorporated. Second, the present system allows *warrants*, both for argument steps, i.e., the reasons that support conclusions, and for undercutters, i.e., for the exceptions that block the connection between a reason and a conclusion. Step warrants express that a particular statement can be adduced as a reason for another statement. They are similar to Toulmin's [1958] warrants, and play a role that is analogous to that of the material implication in the classic rule of inference *Modus ponens*. Undercutter warrants express that a particular statement provides an exception that breaks the connection between a reason and a conclusion. They are a new notion, and are specific for dialectical arguments.

These two novelties in the argumentation theory have led to corresponding adaptations of the interface. First, a way had to be found to present dialectical arguments to the user. Second, the user had to be given the opportunity to make the warrants of an argument explicit (both for the steps and for the undercutters in the argument).

The novelties in the second version of ArguMed solve two drawbacks of the previous version. First, though the arguments were presented graphically, argument attack was not. It was graphically shown *that* an argument was defeated by an attacking argument, but not *by which* argument. Second, since the argumentation theory underlying ArguMed's first version did not have step and undercutter warrants, it was not possible to put at issue that a particular statement was a reason for another statement, or that a statement was an exception.

The three systems (Argue!, ArguMed 1.0 and ArguMed 2.0) can be downloaded at http://www.metajur.unimaas.nl/~bart/aaa/.

[2] This is only a selection, guided by three criteria: 1. the system must be meant for argument assistance (including argument mediation), 2. argumentation must be defeasible, and 3. the systems must be developed with a (semi-)legal context in mind. Not mentioned are, for instance, Nute's [1988] d-Prolog, Pollock's [1987, 1995] OSCAR, Gordon's [1993, 1995] Pleadings Game, IACAS by Vreeswijk [1995], DiaLaw by Lodder [1998], Tarski's World by Barwise and Etchemendy (see http://csli-www.stanford.edu/hp/), and HUGIN (http://www.hugin.dk/).

[3] For an overview of argument models in law, see, e.g., Bench-Capon [1995] and the special issue of *Artificial Intelligence and Law*, Vol. 4, Nos. 3/4, 1996, edited by Prakken and Sartor.

[4] Verheij [1998a] 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 the verbal and the visual approach to argument presentation, respectively.

[5] In this paper, exceptions are of Pollock's [1987] undercutter-type, i.e. they block the connection between a reason and its conclusion.

In section 2, ArguMed's underlying argumentation theory is described, and in section 3 its interface. Section 4 contains a brief comparison with selected other argument-assistance systems.

2. ARGUMENTATION IN ARGUMED

In this section, the argumentation theory underlying the ArguMed-system is informally explained. Connections and differences with related argumentation theories are discussed in section 2.5. The notion of dialectical arguments and warrants, and the distinction of assumptions and issues are new or occur in an innovative way. Example arguments based on Dutch tort law serve as illustrations.

2.1 Reasons, conclusions, exceptions

The simplest form of an argument (in ArguMed's argumentation theory) is a *statement*, e.g.:

> Peter has committed a tort.

In an argument, *reasons* can be given for other statements, e.g.:

> Peter has committed a tort, since he has violated a property right.

The converse of adducing reasons is inferring *conclusions*:

> Peter has committed a tort. Therefore he has the duty to pay for the damages.

In defeasible argumentation, it can be the case that a conclusion is not justified though there is a *prima facie* justifying reason for it. For instance, an *exception* (of Pollock's [1987] undercutter-type, cf. note 5) can break the 'connection' between a reason and a conclusion:

> Peter has violated a property right. As a result, at first sight, he has committed a tort. However, there is a ground of justification for Peter's act. As a result, on second thoughts, Peter's violation of a property right does not justify that he has committed a tort.

It is characteristic of an undercutter-type exception that the *prima facie* conclusion is not replaced by its opposite, viz. that Peter has *not* committed a tort: there could be a *another* reason justifying the conclusion that Peter has committed a tort, even though the reason that Peter has violated a property right, does not justify that conclusion.

In Figure 2, the reason/conclusion/exception-structure of an argument is graphically depicted. In the argument there are two reasons for the statement that Peter has committed a tort, viz. that he has violated a property right, and that he has acted against proper social conduct. Only the first of these reasons is blocked by an exception, viz. that there is a ground of justification.

Reason/conclusion/exception-structures, as in Figure 2, are (unwarranted) *dialectical arguments*. They can be thought of as structures of *argument steps*, i.e., the directed connections of a reason with its conclusion, and *argument undercutters*, i.e., a step with an exception breaking the connection between the step's reason and conclusion. The argument in Figure 2 consists of three steps, and one undercutter that encompasses one of the steps. A

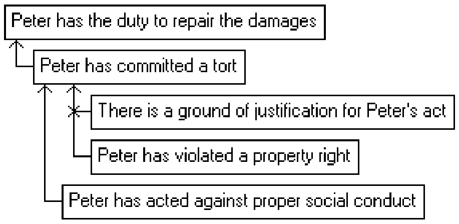

Peter has the duty to repair the damages
Peter has committed a tort
There is a ground of justification for Peter's act
Peter has violated a property right
Peter has acted against proper social conduct

Figure 2: A dialectical argument (without warrants)

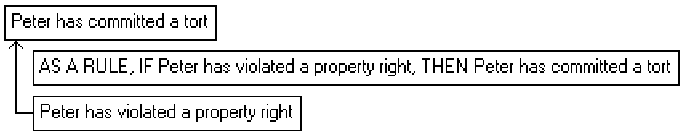

Peter has committed a tort
AS A RULE, IF Peter has violated a property right, THEN Peter has committed a tort
Peter has violated a property right

Figure 3: A warranted step

reason for a conclusion can itself be supported by a reason (*subordination*), a conclusion can be supported by more than one reason (*coordination*), a step can be undercut by more than one exception (*multiple attack*), and reasons for undercutters can themselves be undercut (*counterattack*).

2.2 Warrants

It is not the case that *any* statement is a reason for *any* other statement. If such a connection between a reason and a conclusion exists, the corresponding argument step is said to be *warranted*. That some statement implies another statement, in the sense that it can be adduced as a reason for the statement, is itself a statement, and can, e.g., be expressed as follows:

> As a rule, if Peter has violated a property right, then he has committed a tort.

Any step in an argument (i.e., any connection of a reason with a conclusion) has a corresponding *step warrant* (or *rule*) that can be attached to it. An example is shown in Figure 3.[6]

Step warrants play a role that is analogous to that of the material implication in the classical rule of inference *Modus ponens* (from P and $P \rightarrow Q$, infer Q).

Analogously, it is not the case that *any* statement is an exception, breaking the connection between *any* reason and conclusion. Just as steps, undercutters need to be warranted. *That* some statement is an undercutting exception, is itself a statement. Such 'excepting statements' provide the warrants of undercutters. An *undercutter warrant* can, e.g., be expressed as follows:

> That there is a ground of justification for Peter's act, is an exception to the rule that, if Peter has violated a property right, then Peter has committed a tort.

In Figure 4, an undercutter is shown with its warrant.

[6] The use of uppercase characters in the 'step warrant statement' in the figure are meant to suggest that AS A RULE, IF ..., THEN ... should be considered as a two-place logical connective. A more 'logical' notation expressing a step warrant would, e.g., be $p \rightsquigarrow q$.

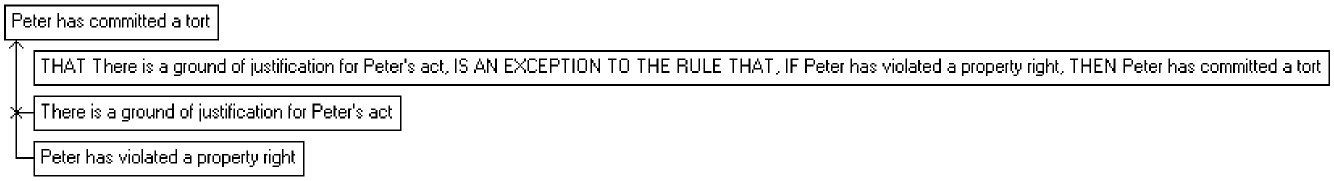

Peter has committed a tort
THAT There is a ground of justification for Peter's act, IS AN EXCEPTION TO THE RULE THAT, IF Peter has violated a property right, THEN Peter has committed a tort
There is a ground of justification for Peter's act
Peter has violated a property right

Figure 4: A warranted undercutter

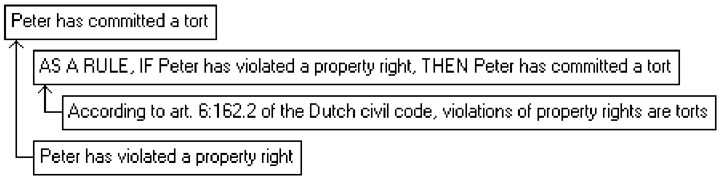

Peter has committed a tort
AS A RULE, IF Peter has violated a property right, THEN Peter has committed a tort
According to art. 6:162.2 of the Dutch civil code, violations of property rights are torts
Peter has violated a property right

Figure 5: A reason for a step warrant

Any undercutter in an argument (i.e., any exception 'crossing out' the connection between a reason and a conclusion) has a corresponding undercutter warrant that can be attached to it.[7]

The reason/conclusion/exception-structures with warrants attached to each step and each undercutter, as discussed above, are *warranted dialectical arguments*, or *arguments*, for short. They are recursively constructed as follows (for brevity, using the logic-style notation of notes 6 and 7):

1. A statement is an argument (containing one statement, no steps and no undercutters). Its conclusion and only premise are the statement itself.
2. Any argument containing a statement ψ can be extended with a step φ // ψ by adding statements $\varphi \rightsquigarrow \psi$ and φ to the argument. In the resulting argument, φ is a reason for ψ. The resulting argument's conclusion is that of the original argument; its premises are $\varphi \rightsquigarrow \psi$, φ and those of the original argument, minus φ.
3. Any argument containing a step φ // ψ can be extended with an undercutter by adding statements $\chi \rtimes (\varphi \rightsquigarrow \psi)$ and χ (for some χ) to the argument. In the resulting argument, χ is an exception to the step φ // ψ. The resulting argument's conclusion is that of the original argument; its premises are $\chi \rtimes (\varphi \rightsquigarrow \psi)$, χ and those of the original argument.

There can be more than one reason for a conclusion and more than one exception to a step.[8]

It may be thought that step warrants and undercutter warrants add little to an argument, and only make explicit what is already in the example steps and undercutters themselves. However, warrants can themselves be the *subject* of argumentation. An example is shown in Figure 5. In this argument, a *reason* is adduced for the step warrant that, as a rule, if Peter has violated a property right, then he has committed a tort. This reason is that, according to art. 6:162.2 of the Dutch civil code, violations of property rights are torts.

[7] Sentences expressing undercutter warrants are obtained by a combination of other sentences, using the three-place logical connective THAT ..., IS AN EXCEPTION TO THE RULE THAT, IF ..., THEN ... . A more 'logical' notation expressing an undercutter warrant would, e.g., be $e \rtimes (p \rightsquigarrow q)$. Cf. note 6.

[8] To avoid unnecessary technicalities, all arguments in this paper are assumed to be *finite*. Looping steps (as, e.g., in the argument '*P*. Therefore *Q*. Therefore *P*') and looping attacks are allowed.

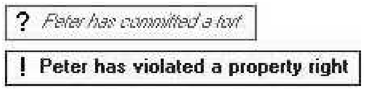

? Peter has committed a tort
! Peter has violated a property right

Figure 7: An issue and an assumption

Similarly, the argument in Figure 6 shows an argument containing support for an undercutter warrant.

Note that the arguments shown in Figures 5 and 6 are strictly speaking not warranted since not *all* steps and undercutters in the arguments are warranted. For instance, in the argument of Figure 5, there is a step without its corresponding warrant: the step from the reason that, according to art. 6:162.2 of the Dutch civil code, violations of property rights are torts, to the step-warrant statement ('As a rule, if ..., then ...'), is itself not warranted.

Any dialectical argument that is not or not completely warranted can easily be extended to a warranted dialectical argument, simply by attaching the appropriate warrant statement to any step and undercutter that does not yet have a warrant attached to it. (Note that the sentences expressing step warrants and undercutter warrants are the result of a formal combination of other sentences by an appropriate logical connective. Cf. notes 6 and 7.) In practice, it is convenient to leave all warrants implicit that are not themselves the subject of further argumentation.

2.3 Justification

Warranted dialectical arguments are the analog for defeasible argumentation of the proofs of classical logic: a warranted dialectical argument determines whether its conclusion is justified assuming its premises (i.e., the statements at the 'roots' of the argument 'tree'). In contrast with classical proofs, that are always justifying, a warranted dialectical argument is, as explicated below, either justifying, or not justifying, e.g., as a result of an exception occurring in the argument. Moreover, it will become clear that extending a warranted dialectical argument (e.g., by adding an exception) can change its justification status. Whether a dialectical argument justifies its conclusion depends on the structure of the argument, i.e., on the reasons, conclusions, exceptions and warrants that occur in it, and on the way they are related.

The justification status of an argument is here made dependent on given *assumptions*, that not necessarily include the premises of the argument. This allows that arguments have 'hanging' premises, i.e., premises that are not assumed, but an issue for further argumentation, which is convenient in practical argumentation (see below).

For dialectical arguments with the simplest structure, viz. statements, the justification status with respect to the given assumptions is trivial. If the statement is itself an assumption, the justification status of the statement (considered as an argument with trivial structure) is justifying. The 'conclusion' of the argument, i.e., the statement itself, is justified. If the statement is

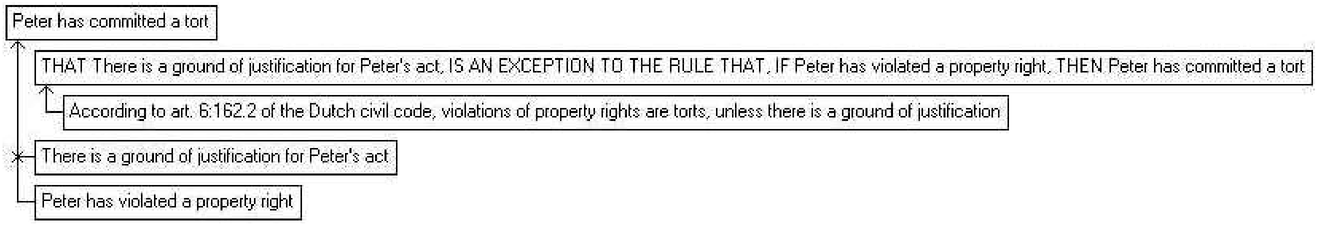

Peter has committed a tort
THAT There is a ground of justification for Peter's act, IS AN EXCEPTION TO THE RULE THAT, IF Peter has violated a property right, THEN Peter has committed a tort
According to art. 6:162.2 of the Dutch civil code, violations of property rights are torts, unless there is a ground of justification
There is a ground of justification for Peter's act
Peter has violated a property right

Figure 6: A reason for an undercutter warrant

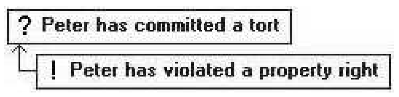

? Peter has committed a tort
! Peter has violated a property right

Figure 8: An issue justified by a justified reason

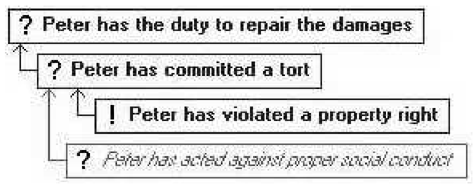

? Peter has the duty to repair the damages
? Peter has committed a tort
! Peter has violated a property right
? *Peter has acted against proper social conduct*

Figure 9: A hanging premise

not an assumption - in which case it is called an *issue* -, the statement is (as an argument) not justifying and (as a statement) not justified. In the figures, assumptions are marked with an exclamation mark, issues with a question mark. Statements that are justified, are shown in a bold font, statements that are not, in an italic font. For instance, in Figure 7, the statement that Peter has committed a tort, is an issue, while the statement that he has violated a property right, is an assumption.

In an argument with no exceptions and no explicit step warrants, an issue is justified if there is a justified reason for it. For instance, the dialectical argument shown in Figure 8 is justifying, and the issue that Peter has committed a tort, is justified.

A justifying dialectical argument with a slightly more complicated structure is shown in Figure 9. In this argument, the issue that Peter has the duty to repair the damages, is justified by the (justified) reason that he has committed a tort. The issue that Peter has committed a tort, is justified since one of the reasons is justified, viz. the assumption that Peter has violated a property right. The issue that Peter has acted against proper social conduct, has no effect on the status of the issue that Peter has committed a tort, since it is itself not justified, and therefore is not a justifying reason. It is an example of a *hanging premise* in an argument: a premise of an argument that is not assumed, but an issue for further argumentation. Hanging premises do not affect the justification status of the argument. In the dynamic practice of argumentation (cf. section 2.4), a hanging premise can become justifying once it is itself justified.

Exceptions have the effect that a reason does not justify its conclusion, even if it is itself justified. For instance, the argument

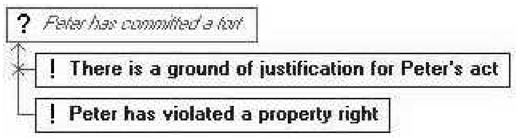

? *Peter has committed a tort*
! There is a ground of justification for Peter's act
! Peter has violated a property right

Figure 10: An exception making a reason non-justifying

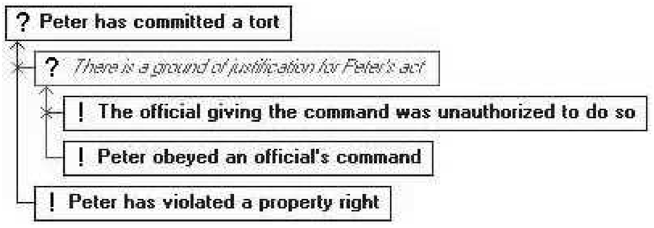

? Peter has committed a tort
? *There is a ground of justification for Peter's act*
! The official giving the command was unauthorized to do so
! Peter obeyed an official's command
! Peter has violated a property right

Figure 11: An exception that is not justified

in Figure 10 is not justifying, since the only reason for its conclusion is undercut by the exception that there is a ground of justification for Peter's act. The exception itself is trivially justified since it is an assumption.

If an exception is itself not justified, it has no undercutting effect. For instance, the argument in Figure 11 is justifying.

Until now, the warrants of the steps and undercutters have been left implicit (cf. the discussion at the end of section 2.2). As long as the warrants of a dialectical argument are not at issue (but are instead implicitly assumed), they do not influence the justification status of the argument. If the warrants are at issue, they have effect on the justification status, as follows. If a step warrant statement is not justified, the conclusion of the step is not justified by the step's reason. Similarly, if an undercutter warrant is not justified, the corresponding undercutter has no effect (i.e., the undercutter's exception does not block the connection between reason and conclusion).

For instance, in Figure 12 an argument is shown that is not justifying because one of its steps is not warranted (in the sense that one of its step warrants is not justified). The argument is based on the opinion in the literature on Dutch tort law that bare violations of property rights are not torts by themselves. The fact that the step is unwarranted, is visualized by the use of a dotted arrow.

As said, an exception has no undercutting effect if the corresponding undercutter warrant is not justified. For instance, in the argument shown in Figure 13, the statement that Peter is not

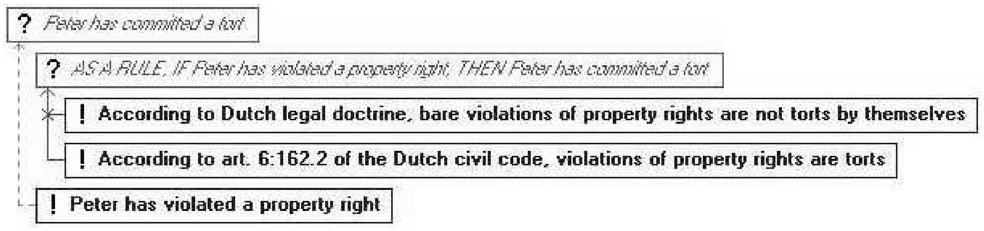

? *Peter has committed a tort*
? *AS A RULE, IF Peter has violated a property right, THEN Peter has committed a tort*
! According to Dutch legal doctrine, bare violations of property rights are not torts by themselves
! According to art. 6:162.2 of the Dutch civil code, violations of property rights are torts
! Peter has violated a property right

Figure 12: A step warrant that is not justified

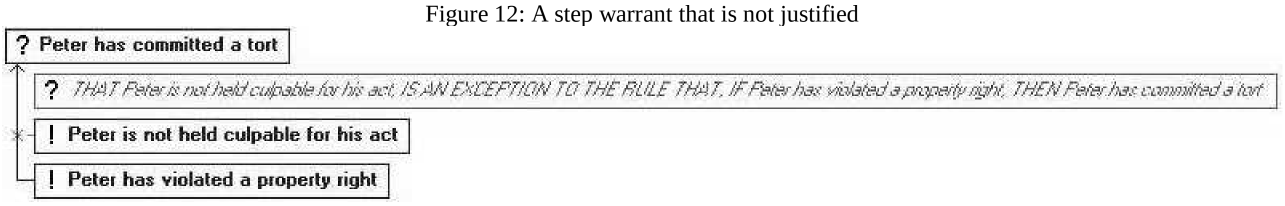

? Peter has committed a tort
? *THAT Peter is not held culpable for his act, IS AN EXCEPTION TO THE RULE THAT, IF Peter has violated a property right, THEN Peter has committed a tort*
! Peter is not held culpable for his act
! Peter has violated a property right

Figure 13: An undercutter warrant that is not justified

held culpable for his act, occurs as an exception to the argument that Peter has committed a tort because of his violation of a property right. However, it has no effect since the corresponding undercutter warrant is an (unjustified) issue. In fact, justifying the undercutter warrant in this argument would be in conflict with actual Dutch tort law: the lack of culpability does not exclude that one has committed a tort, but can have effect on the duty to repair the damages arising from the tort.

To summarize, whether a statement is justified, a reason is justifying, or an exception is undercutting, depends on the (recursive) structure of the argument in which they occur (cf. section 2.2), and on the assumptions, as follows:

A statement is *justified* if
1. the statement is of assumption-type, or
2. the statement is of issue-type, and there is a reason justifying the statement.

Otherwise, the statement is not justified.

A reason *justifies* a conclusion if
a. the reason is justified, and
b. the corresponding step warrant statement is justified (or not made explicit), and
c. there is no exception undercutting the corresponding argument step.

Otherwise, the reason does not justify its conclusion.

An exception *undercuts* an argument step if
a. the exception is justified, and
b. the corresponding undercutter warrant statement is justified (or not made explicit).

Otherwise, the exception does not undercut an argument step.

This definition suffices for finite arguments (recall note 8).

2.4 Argumentation as a process

Argumentation is a dynamic process. During argumentation, new reasons and exceptions are adduced and new conclusions are drawn. Step and undercutter warrants can be put at issue. As a result, a statement that is justified can become unjustified, and vice versa. Premises are not fixed during argumentation, since new statements can be adduced as reasons for the premise of an argument.

The arguments above provide examples. A line of argumentation can start with the two statements of Figure 7. Argumentation can continue by turning one of the statements into a reason for the other (Figure 8). The arguments in Figures 9, 10, 12 and 13 are each possible continuations of the argument in Figure 8. In Figure 9, an additional conclusion is drawn, viz. that Peter has the duty to repair the damages, and a new (unjustified) reason is adduced, viz. that he has acted against proper social conduct.

If the argument in Figure 8 is extended to that in Figure 10, a status change occurs because of a new undercutting exception: the issue that Peter has committed a tort, that at first was justified, becomes unjustified. If argumentation proceeds, resulting in the argument of Figure 11, the statement is *reinstated*: it again becomes justified. Note also that the statement that there is a ground of justification for Peter's act, is first an assumption (in Figure 10), but has been turned into an issue (in Figure 11) for which a (non-justifying) reason is adduced.

In Figures 12 and 13, respectively, a step and undercutter warrant of the argument in Figure 8 have been made explicit. In both cases, the warrant is not justified. In the case of the unjustified step warrant of Figure 12, the effect is that the statement that Peter has committed a tort, is no longer justified by the (justified) reason that Peter has violated a property right. In the case of the unjustified undercutter warrant, the (justified) exception that Peter is not held culpable for his act, does not actually undercut the argument that Peter has committed a tort.

In section 2.3, it was explained how hanging premises are convenient in dynamic argumentation.

2.5 Related argumentation theories

In this subsection, the informally presented argumentation theory underlying ArguMed, is briefly compared to selected other theories of defeasible argumentation. Knowledge of those theories is assumed.

As a start, Toulmin's [1958] argument scheme is discussed. Toulmin's notions of datum, conclusion, warrant and backing are respectively similar to the notions of reason, conclusion, step warrant and reason for a step warrant of the present paper. Toulmin's scheme contains rebuttals[9], that just as the exceptions in the present paper, make argumentation defeasible. The modal quantifier in Toulmin's scheme does not occur in the argumentation theory presented here. The present paper's undercutter warrants have no counterpart in Toulmin's scheme, since it is not possible to argue *that* some statement is a rebuttal. Toulmin does not give an explicit characterization of the justification status of statements. Toulmin's scheme is not put in a procedural context, and does not distinguish between assumptions and issues.

Next, Reiter's [1980] default logic deserves discussion, as a theory of defeasible argumentation *avant-la-lettre*. A difference between Reiter's default logic and the present argumentation theory is that the former uses a first-order language with variables and quantifiers, whereas the language of the latter only uses sentence connectives. The prerequisite α, the justification β and the consequent γ of a default $\alpha : \beta / \gamma$, correspond closely to a reason, the negation of an exception, and a conclusion, respectively. Step warrants and undercutter warrants are lacking in Reiter's default logic, resulting in the (for long recognized) drawback that defaults cannot be derived. Reiter's system definition of extensions can be interpreted as the definition of the sets of statements that are justified with respect to fixed assumptions.[10] Reiter's default logic is not put in a procedural context, and does not distinguish between issues and assumptions.

Pollock's [1987, 1995] theory of defeasible argumentation has already been mentioned. Pollock's logical system is richer than the one presented here, e.g., since Pollock models not only undercutting, but also rebutting exceptions, and adds numerical weights that measure the strengths of reasons. Pollock discusses expressions of the form 'P wouldn't be true unless Q were true', that are closely related to the step warrants of the present paper. Pollock characterizes undercutters as reasons for the negation of

[9] Toulmin [1958] does not yet make Pollock's [1987] distinction between undercutting and rebutting exceptions, that is by now standard.

[10] In this paper, Reiter's lacking or multiple extensions (that are the analog of the unevaluable or ambiguous justification statuses of statements in current argument-based formalisms) are excluded from the discussion. Technically, this is achieved by only considering finite dialectical arguments (cf. note 8).

these expressions. Apparently, there is no discussion of (an analog of) undercutter warrants in Pollock's work. Pollock's inference graphs (extended with his 'defeat links') are related to the dialectical arguments of the present paper, but are not considered as the analog of classical proofs of a conclusion. Pollock's central use of inference graphs is in the definition of justification. Formal differences are that Pollock considers the set of defeat links as a graph, whereas the undercutters in a dialectical argument are recursively ordered, as in a tree, and that Pollock's defeat links are a relation on sequents (a supposition-conclusion pair), while the present paper's undercutters in a dialectical argument connect a statement and an argument step. Pollock's notion of interests seems to be related to that of issues in the present paper.

In Vreeswijk's [1993, 1997] abstract argumentation systems, the tree-like reason-conclusion structure of arguments (but lacking the coordination of reasons) is studied in relation to defeat. Vreeswijk uses an (almost) unstructured language with one distinguished sentence that denotes contradiction, and therefore does not include step warrants in his main argumentation theory.[11] Vreeswijk considers *inconsistency-triggered defeat* (a term used by Verheij [1996]): an argument can only be defeated if there is an undefeated argument with conflicting conclusion. In Vreeswijk's argumentation theory, support and attack are considered separately, viz. in the definition of arguments, and in the definition of the 'in force' arguments, whereas in the argumentation theory of the present paper, support and attack occur side by side in dialectical arguments. Vreeswijk puts argumentation in a procedural context, but his argumentation sequences have fixed assumptions. Issues are not distinguished.

The arguments of Prakken and Sartor's [1996] argumentation theory are formed by chaining rules together. Prakken and Sartor's rules are the conditionals of logic programming, and cannot be nested. They are not comparable to step warrants since there can be no support for the rules themselves. There are no undercutter warrants. Support (by reasons) and attack (by exceptions) are treated separately, and not simultaneously as in the dialectical arguments here. Prakken and Sartor discuss a rebutting and an undercutting type of defeat, where it should be noted that the latter is unrelated to Pollock's [1987, 1995] standard distinction. A naming technique is used for argumentation about priorities. Argumentation is put in a procedural context by the definition of dialogues.

In CumulA [Verheij, 1996], arguments are tree-like structures of reasons and conclusions. An unstructured language is used, so that CumulA does not have notions of step warrant or undercutter warrant. Support and attack are separated. CumulA includes several types of defeat (including defeat by parallel strengthening and by sequential weakening), and not just the present paper's undercutting defeat. In CumulA, argumentation stages are chained in lines of argumentation, as a representation of the process of argumentation. Premises can change during a line of argumentation, and are comparable to the assumptions of the present paper. Issues are not distinguished.

Reason-Based Logic, as initiated by Hage, and further developed in cooperation with Verheij [Hage, 1996, 1997; Verheij, 1996], can be characterized as a theory of rules and reasons. It does not have a notion of an argument, but focuses on

[11] In an appendix, Vreeswijk [1997, p. 275ff.] uses a richer language, including defeasible conditionals, comparable to step warrants, in a brief discussion of Pollock's undercutters.

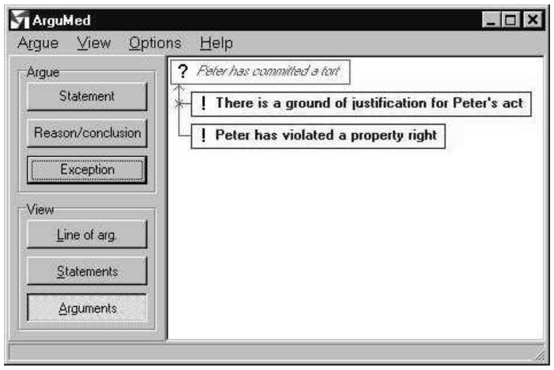

ArguMed
Argue View Options Help
Argue
Statement
Reason/conclusion
Exception
View
Line of arg.
Statements
Arguments
? Peter has committed a tort
! There is a ground of justification for Peter's act
! Peter has violated a property right

Figure 14: A sample screen

types of sentences related to rules and reasons, and on the states of affairs expressed by sentences of these types. It is of relevance here, since the argumentation theory presented in this paper, has resulted from my attempts to bridge the unsatisfactory gap between Reason-Based Logic and CumulA, as it occurred in my dissertation [Verheij, 1996]. Reason-Based Logic's sentences expressing the validity of a rule are comparable to the step warrant sentences of the present paper. Sentences expressing undercutter warrants do not occur in Reason-Based Logic, but are related to the validity of a rule with the exclusion of another rule as its conclusion. The definition of Reiter-style extensions in Reason-Based Logic can be regarded as a definition of the statements justified with respect to a set of assumptions. Issues are not distinguished.

Summarizing, the present paper's notions of dialectical arguments and of step and undercutter warrants are innovations. Limitations of the present paper's argumentation theory are the constrained expressiveness of the logical language, and the restriction of attack and defeat to undercutter-type exceptions.

3. ARGUMED'S INTERFACE

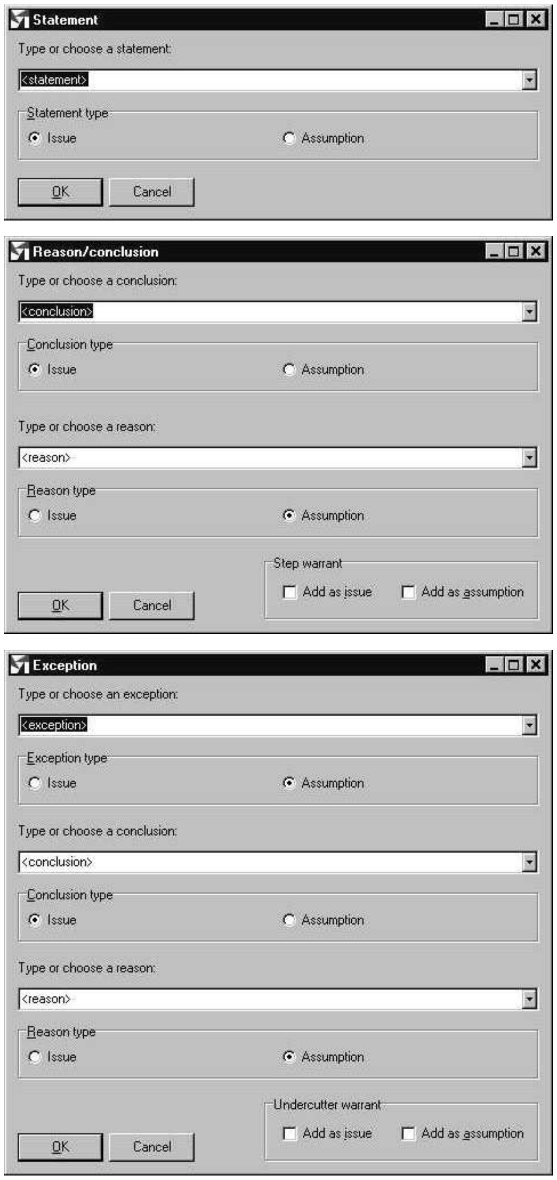
The ArguMed-system is an argument-assistance system with a template-based interface. The user gradually constructs arguments, by filling in templates that correspond to argument patterns. The system keeps track of the constructed arguments and of the justification status of the statements made. A sample screen of a session with ArguMed is shown in Figure 14.

3.1 Moves

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.

Each of the three 'Argue'-buttons (see Figure 14) 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 15). It allows the input of a statement: the user can type a sentence and choose the statement's type. Statements can be of two types, viz. of issue-type and of assumption-type, cf. the distinction between issues and assumptions, as discussed in section 2.3. 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.

The second is the *reason/conclusion template*. It allows the input of a reason, and a conclusion supported by the reason. Both


Statement

Type or choose a statement:

<statement>

Statement type: [✓ Issue] Assumption

OK Cancel

Reason/conclusion

Type or choose a conclusion:

<conclusion>

Conclusion type: [✓ Issue] Assumption

Type or choose a reason:

<reason>

Reason type: Issue [✓ Assumption]

Step warrant: ☐ Add as issue ☐ Add as assumption

OK Cancel

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]

Undercutter warrant: ☐ Add as issue ☐ Add as assumption

OK Cancel

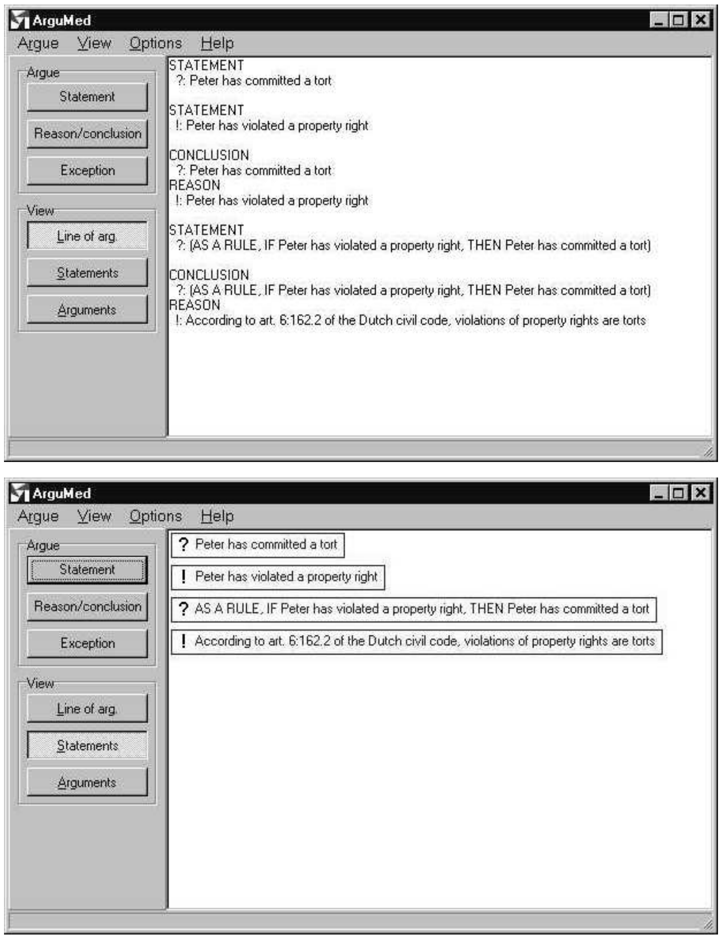
Figure 15: The three argument templates

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.

By default, the step warrant corresponding to the reason/conclusion-move is not made explicit. By selecting the appropriate box, the user can choose to add the step warrant as an issue or as an assumption.

The third is the *exception template*. 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



STATEMENT
?: Peter has committed a tort

STATEMENT
!: Peter has violated a property right

CONCLUSION
?: Peter has committed a tort
REASON
!: Peter has violated a property right

STATEMENT
?: (AS A RULE, IF Peter has violated a property right, THEN Peter has committed a tort)

CONCLUSION
?: (AS A RULE, IF Peter has violated a property right, THEN Peter has committed a tort)
REASON
!: According to art. 6:162.2 of the Dutch civil code, violations of property rights are torts

? Peter has committed a tort

! Peter has violated a property right

? AS A RULE, IF Peter has violated a property right, THEN Peter has committed a tort

! According to art. 6:162.2 of the Dutch civil code, violations of property rights are torts

Figure 16: The 'line of argumentation'-view and the 'statements'-view

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.

By default, the undercutter warrant corresponding to the exception-move is not made explicit. By selecting the appropriate box, the user can choose to add the undercutter warrant as an issue or as an assumption.

3.2 Views

The ArguMed-system provides three *views*, providing information about the current argumentation session. Each view is accessible by one of the three 'View'-buttons (see Figure 14). In the 'line of argumentation'-view, the argument moves as performed by the user are listed in the order in which they have been performed by the user (Figure 16).

In the 'statements'-view, all statements made by the user are presented. 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. Whether a statement is (currently) justified is shown by the use of colored boxes and arrows, and different fonts (bold/italic).

In the 'arguments'-view, the arguments that can be constructed on the basis of the current user input, are shown. The arguments are shown as in the figures of section 2.3. Optionally, only the structure of the arguments is shown, as in the figures of section 2.2.

3.3 Algorithms

The ArguMed-system has two basic algorithms. The first computes dialectical arguments, based on the argument moves performed by the user. The second computes which statements are justified, with respect to the computed dialectical arguments.

The algorithm computing dialectical arguments, straightforwardly constructs dialectical arguments using the statements, reasons, conclusions, exceptions and warrants, that are available by the user's moves. The recursive definition of arguments in section 2.2 is used.

Each computed dialectical argument makes maximal use of the available data; a restriction is that loops in (any branch of) a dialectical argument (as, e.g., in the argument '*P*. Therefore *Q*. Therefore *P*') are not further developed.[12] The algorithm depends on the order in which the moves have been performed: e.g., the order in which statements have been adduced has effects on the order in which they are shown on the screen.

The algorithm computing which statements are justified, follows the discussion in section 2.3.

4. A COMPARISON OF ARGUMENT-ASSISTANCE SYSTEMS

In order to put the ArguMed-system in context, it is briefly compared to other systems, viz., Room 5 by Loui *et al.* [1997] and Zeno by Gordon and Karacapilidis [1997]. Room 5 is called a testbed for public interactive semi-formal legal argumentation. Zeno is meant to create advanced support for complex multi-party/multi-goal decision-making. The ArguMed-system is also compared to its precursors, i.e., the Argue!-system [Verheij, 1998a] and the first version of the ArguMed-system [Verheij, 1998b].[13] The two versions of the ArguMed-system are referred to as ArguMed 1.0 and ArguMed 2.0, respectively. First, the underlying argumentation theories are discussed; second, the user interfaces.

4.1 The underlying argumentation theories

In the underlying argumentation theories of all systems argumentation is *dynamic*. Statements can be made, and reasons can be adduced. In Room 5 and Zeno, 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 on the justification of an initial central issue. In Argue! and both versions of ArguMed, 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 turning the conclusion of one argument into a reason for a premise of another argument) is only possible in these three systems.

[12] Also 'attack loops' (see, e.g., Verheij [1998a; 1996, pp. 146-151]) are not further developed.

[13] See note 2.

In all systems, reasons can be *chained* (subordination) and support a conclusion *in parallel* (coordination). In Room 5 and Zeno, a distinction is made between *reasons for and against* a conclusion. The arguments in ArguMed 2.0 incorporate counterarguments by means of *undercutting exceptions*. Only ArguMed 2.0 has a notion of the *warrants* underlying argument steps. It also adds undercutter warrants.

All systems model a notion of *defeasibility* of argumentation. In Zeno, *weighing* the conflicting reasons determines which conclusions are justified. In Argue! and both versions of ArguMed, *undercutter-type exceptions* can block the justification of a conclusion by a reason for it. Argue! has *composite-type defeat*, such as defeat by sequential weakening (terms used by Verheij [1996]).

In Room 5 and Zeno, 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. Argue! and the two versions of the ArguMed-system, all three designed as single-user systems, have no explicit notion of game participants, but can be considered as one-participant games.

Zeno, Argue! and the two versions of the ArguMed-system are *evaluative*: the status of statements and arguments can be determined by the system. In Zeno and the ArguMed-systems, evaluation occurs automatically *in the background*. In Argue!, the user *asks* the system to update the evaluation of the statements and arguments.

4.2 The user interfaces

All systems have a *window-style interface*. Room 5 and Zeno are web applications, Argue! and the two versions of ArguMed are PC applications (downloadable from the author's web site). Argue! has a *graphical interface*, in the sense that the user draws the argumentation data on the screen using a pointing device. Room 5, Zeno and the ArguMed-systems have a *template-based interface*: users fill in forms to perform an argument move. The ArguMed-systems innovates this type of interface by using different templates for different types of moves.

All systems present arguments in a *visual* manner. Zeno, Argue! and the ArguMed-systems use a *tree*-like presentation. Room 5 uses a clever system of *boxes-in-boxes* in an attempt to avoid 'pointer-spaghetti'.

In Room 5 and Zeno, counterarguments (based on reasons against conclusions) are *grouped together* in the visual argument structure. In Argue!, counterarguments are shown by a *special visual structure*. In ArguMed 1.0, counterarguments (based on undercutting exceptions) are not directly shown; the exceptions are presented with their corresponding argument steps in a *special viewing window*. In ArguMed 2.0, counterarguments are *incorporated* in the arguments themselves, which is possible by the new concept of dialectical arguments.

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

5. CONCLUDING REMARKS

In this paper, the notion of *automated argument assistance* has been introduced. It extends the notion of automated argument mediation. Whereas argument-mediation systems, such as Zeno and Room 5, aim to be used in a *public* context, e.g., in order to structure the public discussion concerning an issue, argument-assistance systems in general also have *private* uses, e.g., the drafting and testing of court pleadings.

Automated argument assistance should be distinguished from automated reasoning: while automated reasoning systems *replace* the reasoning of the users, argument-assistance systems do not reason themselves, but are tools assisting the users' reasoning.

Elsewhere [Verheij, 1998a, 1998b], I have argued that even in this experimental phase the development of argument-assistance systems is relevant. I distinguished four ways in which the development of argument-assistance systems is worthwhile: first, such systems can serve as *realizations* of (formal) argumentation theories, which is especially relevant because of the (well-recognized) technical difficulties of many theories; second, they are *test beds* for argumentation theories, technically, philosophically and in practice; third, argument-assistance systems can be *showcases*, giving the argumentation theories more credibility; and, finally, they can be *practical aids*, with applications in, e.g., legal decision making, planning and education. Currently developed systems are already worthwhile in the first two, more theoretically oriented ways, and are starting to become so in the second two, more practically oriented ways.

During the gradual maturing of argument-assistance systems, the growing value of these systems in legal practice will go hand in hand with a deeper understanding of legal argumentation.

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).

The author thanks Jaap Hage and Bram Roth 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.

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).

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).

Marshall, C.C. (1989). Representing the structure of a legal argument. *The Second International Conference on Artificial Intelligence and Law. Proceedings of the Conference*, pp. 121-127. 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).

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

Toulmin, S. E. (1958). *The uses of argument.* University Press, Cambridge.

Verheij, B. (1996). *Rules, Reasons, Arguments. Formal studies of argumentation and defeat*. Dissertation Universiteit Maastricht.

Verheij, B. (1998a). Argue! - an implemented system for computer-mediated defeasible argumentation. *NAIC '98. Proceedings of the Tenth Netherlands/Belgium Conference on Artificial Intelligence* (eds. Han La Poutré and Jaap van den Herik), pp. 57-66. CWI, Amsterdam

Verheij, B. (1998b). 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.

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).

Verheij, B., Hage, J.C., and Herik, H.J. 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. (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.