Artificial Intelligence and Law **8:** 205–231, 2000.
© 2000 *Kluwer Academic Publishers. Printed in the Netherlands.*

Representation of formal dispute with a standing order

GERARD A. W. VREESWIJK
University of Groningen, Faculty of PPSW, Grote Kruisstraat 2/1, 9712 TS, The Netherlands
E-mail: G.A.W.Vreeswijk@ppsw.rug.nl.

Abstract. Computational dialectics is concerned with the formal representation of argument and dispute. The field emerged from developments in philosophy, artificial intelligence and legal theory. Its goal is to suggest algorithms, procedures and protocols to investigate the tenability of logical claims, on the basis of information in the form of rules and cases. Currently, the field slowly converges to the opinion that dispute is the most fair and effective way to investigate claims. The basic assumption of this field is that dispute is the most fair and effective way to investigate claims. The definition of a formal dispute varies throughout the literature, but is considered not to vary within one and the same logical system. In this paper it is shown that parts of the definition of a dispute may change within one logical system. To this end, the notion of *partial protocol specification (PPS)* is introduced. A PPS is a part of the definition of the protocol. A modification to the protocol, in the form of a PPS, can be put forward, disputed, established and incorporated as an effective 'point of order'. The paper demonstrates the existence of self-undermining PPSs, it discusses the relevance of PPSs for dialectical models of legal argument and concludes with a description of how PPSs can be built into existing argumentation systems.

1. Introduction

Computational dialectics is a relatively new discipline. It tries to describe how knowledge can be inferred by using techniques from symbolic argumentation. The field emerged from developments in philosophy, artificial intelligence and legal theory. In particular, a number of researchers recently discovered that computational dialectics is one of the most appropriate techniques for reasoning with incomplete and uncertain information. (Cf. Loui, 1987: Page, 1991; Loui, 1990, 1994, 1998; Pollock, 1994; Brewka, 1994; Dung, 1995; Gordon, 1995; Loui and Norman, 1995; Prakken and Sartor, 1996, 1998; Verheij, 1995a,b; Lodder, 1997; and Vreeswijk, 1993, 1995b,g, 1997.) In Al & Law, this development was partly inspired by earlier dialectical models of case-based legal reasoning (e.g., Ashley and Rissland, 1988; Skalak and Rissland, 1992). Computational dialectics ties together the exactness of symbolic logic with the flexibility of informal argument. The result is a new logic, in which claims are investigated pro and con by an alternating inferential procedure.

The current state of the art in computational dialectics is characterized by a discussion on which protocol is actually 'the best'. Various protocols have been proposed. (Cf. Krabbe, 1985; Vreeswijk, 1993; Brewka, 1994; Loui, 1994; Vreeswijk, 1995g; Prakken and Sartor, 1996, 1998; Loui, 1998.) There are protocols for quick but shallow reasoning, and there are protocols for deep but slow reasoning. Some protocols let parties construct their arguments one step at a time, while other protocols allow parties to construct their arguments in one move. Some protocols allow parties to construct counterarguments ad libitum, while other protocols permit parties to rebut at most twice. Thus, it turns out that the rules of the game may vary. This variation is caused by the different circumstances under which a dispute may be organized. When there is not much time, for instance, it is likely that players use a quick but shallow protocol. Another circumstance is the amount of information available. If there are few rules and cases, it is likely that players are allowed to construct counterarguments without restriction. Conversely, if there are many rules and cases, there is an almost unlimited supply of counterarguments, so players are usually restricted in the number of arguments they are allowed to use. In conclusion, there are various interesting ways to organize a dispute, of which no one is necessarily 'the best'.

Researchers define protocols on the basis of the type of dispute they wish to invoke. But the desired type and order of dispute may change. The desired way of dispute may even change within the course of dispute itself. For example, time pressure may suggest a reduction of the number of rounds in which parties may respond to arguments. Or the importance of a claim that is scheduled at the end of an agenda, may suggest a reorganization of that agenda. In colloquial argument – in a meeting for instance – people know what to do in such cases. In meetings, participants raise so-called *points of order* to steer the protocol into a desired direction. This indicates that points of order have an important function. Since rules of order for meetings are a kind of (procedural) law, a formal investigation of this phenomenon is relevant for AI & Law. In particular, it will contribute to recent attempts to formally elaborate on the procedural aspects of juridical debate (see e.g., Gordon, 1995; Hage et al., 1994; Lodder, 1997).

A number of researchers have emphasized the use of points of order in a formal setting, most notably Loui (1998):

> Locutions pertain not only to the substance of dispute, but also to the protocol. Arguments can advance claims about how to define or alter the protocol. Arguments can seek to establish conditions relevant to protocol, such as defeat relations among arguments, adequacy of responses, and termination. These are rightly considered meta-arguments. The more effective the definitions of aspects of protocol, such as when an argument defeats another, when response is adequate, and when termination is appropriate, the less the need to ascend to meta-argument. Frequently, agreement over protocol is part of shared basis.

Thus, it must be possible to define parts of the protocol by way of argument. In personal communication, Loui indicated the difficulty of defining a formalism that is able to do this, because

> (. . .) we need languages to describe protocol first, or we can't utter sentences about them.

Until now, a simple formal protocol for dispute in which points of order can be raised has not been specified.

In this paper, we specify a protocol of which a part can be modified by rational claims. This result is achieved by representing protocol changes as partial protocol specifications (PPSs) that can be put forward and defended in debate. If a PPS is defended successfully, we assume that the parties involved agree to adopt it in the specification of the protocol. The modified specification causes the rules of dispute to change accordingly.

At the end of the paper we show that the concept of an amendable protocol is of consequence to technical as well as legal domains. In particular, we show that the ideas presented are of consequence to Al and law, and provides a formal handle on the procedural machinery of meta-juridical dispute.

2. Basic Concepts

The aim of this section is to present a simple calculus.[1] This calculus will be used to introduce the basic concepts of computational dialectics. It will then be used to formulate the main result.

It must be remarked that one always argues in a so-called *domain of discourse*, that is, a certain area of discussion. An appropriate example of a domain of discourse is the issue of allocating travel grants in a scientific department. Let us take this as an example. In most scientific departments, the annual travel budget is relatively small, while many researchers want to draw from the budget. A typical subject of discussion is therefore the assignment of a travel grant. The actual assignment of a grant depends on a number of factors, of which the most relevant include

E:	the applicant is entitled to submit a request
T:	the request has been submitted on-time
B:	there is room within the department's annual budget
S:	the budget will be shared by other donators (this is called matching)
X:	private contribution
C:	commercial funding
L1:	inexpensive location (L2: average, L3: expensive)
I:	part of international scientific research program/project
Y:	applicant is student

[1] Some would say 'formalism', or 'logic'.

F:	there was a former application, which was honoured
P:	a paper is being presented
V:	a visitation to a person or institute is on the program
R:	applicant promised to write a report afterwards

The identifiers in the left margin represent the decision-factors of an application, in the form of logical propositions. The application itself can then be represented as a collection of logical propositions or their negations. For instance, the set facts = {E,T,S,¬C} describes the facts of a situation in which the applicant is entitled to submit a request, the request has been submitted on-time, with a shared budget, and without commercial participation. Other factors, such as whether there is free room in the department's annual budget, i.e., B or ¬B, are not mentioned and might therefore be considered unknown. A set of facts that together describe a situation is called a CFS, which stands for *current fact situation*. (After Ashley et al., 1988, and Ashley, 1990.)

Ideally, departments use policies and precedents as a basis to decide whether employees receive money to go to a conference, symposium, or meeting. A precedent is an earlier decision that can be referred to in order to justify a similar decision. For instance, if an employee receives a grant for travelling to Hawaii without supplementary funding, the department sets a precedent for similar cases in the future. Thus if a colleague would apply for a grant on similar conditions in the future, i.e., if a colleague wants to visit Hawaii without supplementary funding as well, then the department is under pressure to acknowledge the request of this colleague with equal force, on the pain of holding an inconsistent travel policy.

Precedents are collected as cases in a so-called *case-base*. The following is an example of a ease base, consisting of 10 cases.

E,T,S,¬X,I,¬F,¬P,V ≻—G
E,T,¬S,¬I,P≻ ¬G
E,T,¬S,I,Y,P,R ≻G
E,S,X,I,¬F,V ≻G
E,T,B,S,C,L3,I,¬P,R ≻—G
E,T,S,Y,P,R ≻G
E,T,Y,P ≻ ¬G
E,T,¬S,¬X,I,R ≻ ¬G
E,T,C,L3,P ≻ ¬G
E,T,¬S,I,F,¬P,V ≻ ¬G

Each case describes a situation, plus a decision that was taken on the basis of that situation. For example, in the first case it was decided to assign a grant on the basis of E,T,S,¬X,I,¬F,¬P,V. In the second case, it was decided to refuse a grant on the basis of E,T,¬S,¬I,P. And so forth.

Further, there is the concept of a pair of complementary cases. Two cases are *complementary*, if the decision of one case is the negation of the decision of the

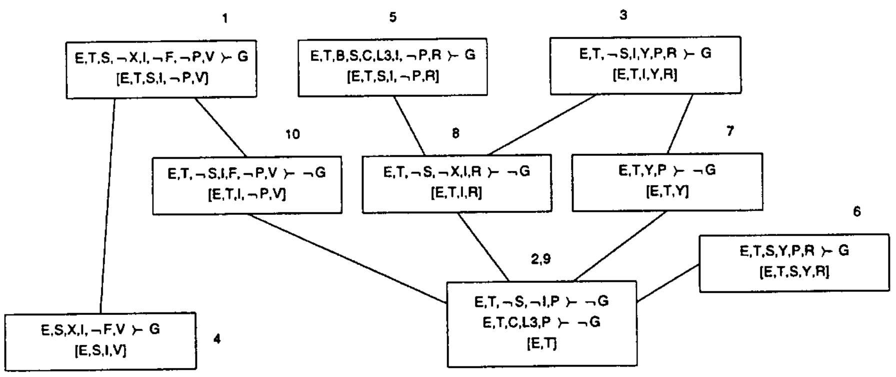

Figure 1. How facts = {E,T,S,*neg*C,I,Y,¬P,V,R} orders the case-base.

other. For example, the first and the second element of the above case-base are complementary. The above case-base is kept short for the sake of simplicity. Therefore, its elements are complementary with respect to only one issue, namely G. Realistic case-bases consist of elements that are complementary on different issues.

Let

facts = {E,T,S,¬C,I,Y,¬P,V,R}

be the CFS of a case we wish to consider. The problem is to decide G or ¬G on the basis of these facts, and to justify the decision thus formed with the help of precedents from the case-base.

To solve this problem, it is perhaps best to start with the observation that some elements of the case-base are more in line with the current case than others. For example, the facts of the 1st element of the case-base coincides with the facts of the current case, facts, on the points E,T,I,¬P,V, and S, while the facts of the 10th (= last) element of the case-base coincides with facts on the points E,T,I¬P, and V. Thus, we observe that case 1 of the case-base matches all facts that are matched by case 10, viz. E,T,I,¬P,V, plus one extra, viz. S. For this reason, we conclude that case 1 is more specific than case 10. Sometimes, it is also said that case 1 is more *on-point* than case 10. (Cf. Ashley & Rissland, 1988; Ashley, 1990.) We may draw similar conclusions for other pairs of cases. Case 3 is more specific than case 8, for example, and case 7 is more specific than case 2 (Figure 1). Specificity forms a partial order among cases. (Partial, because not every pair of cases is comparable with respect to specificity.) For example, case 1 is not more specific than case 3, and conversely. So case 1 and case 3 are incomparable. Finally, it must be noted that the order of specificity can reverse in the presence of other facts. For instance, if facts = {E,T,¬S,¬X,I,R}, then case 3 is no longer more specific than case 8. In this new situation, case 8 would be more specific than case 3. Thus, specificity also depends on the facts in the CFS.

2.1. DEFEAT AMONG ARGUMENTS

The first aim of research in computational dialectics is to find adequate criteria that describe which arguments apply best to the current situation. However, there is a 'clash of intuitions' on the validity and relevance of such criteria. Most researchers agree on the principle of specificity, that says that arguments that are most 'on point' are preferred. Poole introduced specificity in 1985, after which alternative definitions were proposed by Nute (1988), Loui (1989), and Prakken (1993). Other criteria to compare arguments include directness (Loui, 1989), pre-emption (Horty et al., 1988), combined defeat (Prakken, 1993), number of favorable factors with a Ceteris Paribus comparison (Ashley et al., 1988), and accumulation of numerical strength (Pollock, 1994). In the present situation our arguments are cases, which means that we deal with case-based reasoning. In the present academic discussion, it can safely be stated that there is little to no controversy on the criterion that says that more specific cases are more relevant. We adopt this criterion here as well.

DEFINITION 2.1. (Defeat among cases.) Let a current fact situation be given. A case D is said to *defeat* a complementary case C if D is strictly more specific than C, relative to the current fact situation. A case D is said to *interfere* with a complementary case C if neither case defeats the other, relative to the current fact situation.

Using a partial order notation $\leq$, we may define: $C < D$ if and only if D defeats C. Thus, it follows that D interferes with C if and only if neither $C \leq D$ nor $D \leq C$. Referring to Figure 1, we may for instance conclude that case 1 defeats case 10, case 10 defeats case 2, case 10 defeats case 9 case 5 defeats case 8, etc.; case 1 interferes with case 5 (and conversely) case 5 interferes with case 3 (and conversely) ease 10 interferes with case 7 (and conversely) case 1 interferes with case 7 (and conversely), etc.

2.2. DIALECTICS

The second aim of research in computational dialectics is to suggest algorithms, procedures and protocols that describe how rules and cases should be combined to arrive at a decision. Ashley et al. (1988) refers to this as dialectical information: information needed to find relevant cases and, having found them, how to use them effectively as examples for justifying positions in an argument. Dialectical information is often put in the form of a procedure. If the procedure is to be followed by more than one party, it is known as a *protocol*. There are many protocols for two parties. We use one of the most simple available:

DEFINITION 2.2. (Protocol.) A strongly alternating two-party immediate response dialectic with simple cases and the burden to establish – a *primitive dialectic*, henceforth – is a sequence of moves, in which player 1, denoted by PRO, uses odd-numbered moves to try to establish warrant for a claim, and player 2, denoted by CON, uses even-numbered moves to try to prevent player 1's success. A *move* is a citation of a case from the case-base, or the special word pass. It is further stipulated that all moves must interfere with preceding moves. For PRO, it is additionally stipulated that moves must not only interfere, but also defeat preceding moves, Repetition of moves is not allowed. Finally, the word pass is put forward if no further moves can be made with cases.

It follows that the game is merely a succession of analogies, where PRO is required to cite precedents that are more on-point, and CON need only provide nuisance analogies.

EXAMPLE 2.3. Consider the following case:

facts = {E,S,X,I,¬F,V}

This case is a perfect fit with the fourth element of the above case-base. It follows that PRO merely has to cite the analogy to settle a dispute on G in its favour:

1. PRO: E,S,X,I,¬F,V $\succ$G   [E,S,X,I,¬F,V]
2. CON: pass

CON cannot respond because there is no element in the case-base that interferes with PRO's case, i.e., there is no element in the case-base with a basis that uses a fact outside the grounds of PRO's case.

EXAMPLE 2.4. Consider the following case.

facts = {E,T,S,L3,P,R}

This case does not exactly match with one of the cases in the case-base. Nevertheless, PRO is able to present a case with five matches:

1. PRO: E,T,S,Y,P,R $\succ$ G   [E,T,S,P,R]
2. CON: E,T,C,L3,P $\succ$ ¬G [E,T,L3,P]
3. PRO: pass

PRO opens by citing the fourth case in the case-base. Then CON responds by citing a case that interferes with PRO's previous case, on point L3. PRO is unable to find a case that is more specific than CON's, and passes.

EXAMPLE 2.5. Consider the following case.

facts = {E,T,S,¬C,I,Y,¬P,V,R}

The facts of this case are the same facts that are used to order the case-base in Figure 1. This figure might therefore be used to identify the path 5 - 10 - 1 - 8 - 3 that both parties follow in the course of their argument.

1.	PRO: $E,T,B,S,C,L3,I,\neg P,R \succ G$	$[E,T,S,I,\neg P,R]$
2.	CON: $E,T,\neg S,I,F,\neg P,V \succ \neg G$	$[E,T,I,\neg P,V]$
3.	PRO: $E,T,S,\neg X,I\neg F,\neg P,V \succ G$	$[E,T,I,\neg P,V,S]$
4.	CON: $E,T,\neg S,\neg X,I,R \succ \neg G$	$[E,T,I,R]$
5.	PRO: $E,T,\neg S,I,Y,P,R \succ G$	$[E,T,I,Y,R]$
6.	CON: pass	

PRO begins by citing a case with six matches. At line 2, CON interfers by citing a case with a deviating fact, V. PRO can do better than this, by citing a case that is more specific. (Plus S.) At line 4, CON again interferes by citing a case with a deviating fact, R. At line 5, PRO is again able to be more specific, this time with Y. CON can't respond with $E,T,\neg S,I,F,\neg P,V \succ \neg G$ because that case was already cited at line 2, and successfully countered by PRO at line 3. Hence CON passes.

3. Warrant

A typical distinction in computational dialectics is the difference between being right, and being *proved* right. This is to say that the outcome of a dispute does not always come to the side that 'deserves' it. This distinction is manifest when both parties (agree to) follow a faulty protocol. For example, trivially, if both parties (agree to) follow a procedure in which the initiator always wins, then it will be clear that any party that opens a dispute with a false claim, nevertheless manages to establish that claim.

Another situation in which the difference between being right and being *proved* right is manifest, is where one party has a winning strategy, i.e., a successful order of citing cases, but fails to execute that strategy. It may then happen that the other party has the opportunity to take advantage of the fault of its opponent by choosing a strategy that turns the table and settles the dispute in its favour. Thus, it is possible to win a dispute, even when the rules are 'fair' and one is 'wrong'.

The official terminology for 'being right' is to say that the claim involved is *warranted*. If a claim is warranted, it is right in the Platonic sense of the word. Had unlimited time and memory been available, the claim would have proven to be true. Two papers in which warrant is formally defined are (Pollock, 1994) and (Vreeswijk, 1995b). There are various algorithms, procedures, and protocols to approximate warrant. Protocols have several desirable, or undesirable, properties.

1. A protocol is *fair* is any claim that can be established in a dispute with optimal opposition, is warranted. Fairness corresponds to the implications (3) $\Rightarrow$ (1) and (2) $\Rightarrow$ (6).

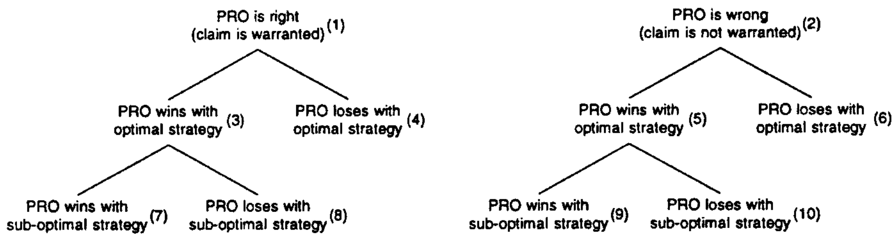
PRO is right (claim is warranted) (1)
PRO wins with optimal strategy (3)
PRO loses with optimal strategy (4)
PRO wins with sub-optimal strategy (7)
PRO loses with sub-optimal strategy (8)
PRO is wrong (claim is not warranted) (2)
PRO wins with optimal strategy (5)
PRO loses with optimal strategy (6)
PRO wins with sub-optimal strategy (9)
PRO loses with sub-optimal strategy (10)

Figure 2. Decision tree of outcome and strategy.

2. A protocol is *effective* if any claim that is warranted, can be established in dispute. Effectiveness corresponds to the implications (1) $\Rightarrow$ (3) and (6) $\Rightarrow$ (2).

Thus, a protocol is considered unfair if (2) but (5), i.e., if PRO is wrong on a claim, but nevertheless manages to find a winning strategy for it. A protocol is considered ineffective if (1) but (4), i.e., if PRO is right on a claim, but fails to establish it; not because PRO has not tried hard enough, but because there exists no winning strategy for that claim. Searching for an optimal strategy remains important, even if a protocol is effective. Thus, with an effective protocol, it may happen that (1) and (3) but (8). In that case, the optimality of the strategy is not only a sufficient, but also a necessary condition for establishment.

For simple representation languages, it is possible to define argumentation protocols that are fair and effective. In (Vreeswijk, 1995h), such a protocol was defined, and it was proven that it had the intended properties. In the present paper, we will deal with various kinds of protocols, including protocols that are unfair and ineffective. In Section 7 we will show that the application of unfair and ineffective protocols yields peculiar results.

4. Current Opinion

Continuing the example, assignment of a travel budget is merely one issue. A department has to deal with other issues as well. These include, for instance,

G:	assignment of a travel grant
A:	extension of secretarial support
J:	reimbursement of subscription to scientific journals
U:	funding of an external scholarship
Q:	modernizing computer-infrastructure

Every issue is disputed from time to time. When that is the case, we assume that the outcome of the dispute is saved in a set of logical propositions, called the *current opinion*. Sometimes, the current opinion is denoted by CO, to emphasize its similarity with the CFS. The similarity lies in the fact that both sets are responsible for describing a part of the current situation (CS).

An example of a current opinion is the set

opinion = {¬G,¬A,U,¬Q}.

This collection of propositions represents a situation in which recent discussions lead the department to an opinion that is against the assignment of a travel grant, against an extension of secretarial support, in favour of funding external scholarships, and against a modernization of computer-infrastructure. Further we see that neither J nor ¬J is represented as an opinion. This means that the department has not formed an opinion (yet) regarding the reimbursement of subscriptions to scientific journals. Thus, an opinion can be formed or it can be undetermined. An opinion may change in result of a dispute. For instance, if the dispute of Example 2.3 is conducted on the basis of facts = {E,S,X,I,¬F,V}, then the outcome G changes the opinion above into

opinion = {G,¬A,U,¬Q}.

The dispute in Example 2.4 on the basis of facts = {E,S,X,I,¬F,V}, would change the opinion back into the previous opinion, that included ¬G. Other elements of the current opinion, such as ¬A,U, and ¬Q may be changed in result of a dispute as well. In this way, the current opinion may change every time a dispute on a certain issue is completed, and thus forms a collection of the most recent outcomes of rational inquiries to topical issues in the domain of discourse.

The notion of a current opinion is needed to introduce a formal notion of standing order (cf. Section 6).

5. Different Protocols

We have seen how different issues can be disputed formally by following a well-defined procedure for argumentation, referred to as a protocol. Further, we have seen how an opinion may be formed as a result of a formal dispute.

That is not all there is to it. An important observation with respect to formal argumentation is that there are many ways to organize a dispute. In particular, the protocol described in Definition 2.2 is just one of many ways to organize an argument-exchange between two parties. The definition stipulates that PRO has the burden to establish, but with equal reason it could have been stipulated that CON has the burden to establish. It stipulates that moves may not be repeated, but with equal reason it could have been allowed for moves to be repeated. Definition 2.2 does not put restrictions on the number of rebuttals, but with equal reason it could have been stipulated that each party may respond at most twice to the same claim. Thus, there are various possibilities to organize a dispute.

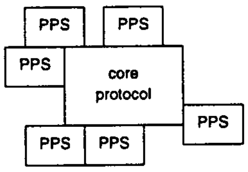

Figure 3. Protocol, completed with PPSs.

To identify the various possibilities, it is helpful to abbreviate some of them with the help of identifiers:

α:	burden to establish
β:	no repetition of moves
γ:	must defeat all previous claims (instead of preceding claim only)
δ:	reply to the same claim at most twice

The symbols α, β, γ, and δ are so-called *partial protocol specifications* (PPSs). A PPS is meant to specify a designated part of the protocol (Figure 3). This part can for instance be concerned with the burden to establish, the repetition of moves, the status of previous claims, adequacy of response, playing-time, or termination. Other 'rules of the game' may also be represented as PPSs.

The notation of PPSs can be differentiated amongst both parties with the help of accent-marks, as follows:

– an un-ornamented identifier means that the restriction applies to both parties;
– an accent means that the restriction applies only to PRO;
– a double accent means that the restriction applies only to CON.

Thus, α' means that PRO has the burden to establish; β'' means that CON cannot repeat moves; $\neg\gamma''$ means that CON does not necessarily have to defeat all previous claims of PRO. And so forth. Furthermore, we adopt the closed world assumption. This means that restrictions do not apply if they are not mentioned. When these conventions are put to work, the characteristic aspects of the procedure given by Definition 2.2, for instance, can be encoded by the combination α', β. This combination expresses that both parties follow a procedure in which PRO has the burden to establish, and in which repetition of moves is not allowed. The absence of γ means that both parties do not necessarily have to deal with all previous claims of their opponent; the absence of δ means that both parties can reply *ad libitum*. It will be clear how variations of the basic procedure can be given by other combinations of partial protocol specifications (PPSs). For example, the combination α', γ', δ represents a procedure in which PRO has the burden to establish, and must defeat all claims made by CON. Moreover, both parties are restricted to reply to the same claim at most twice. From the absence of β it can be concluded that repetition of moves are allowed. Thus, the presence (or absence) of PPSs help to specify the protocol.

The lion's share of an argumentation protocol is firm. Most PPSs are interpreted as immovable rules of interaction that must be followed by those that want to argue formally with (rules and) cases. This is fair, since a great deal of the protocol is universal and applies to all situations. For some PPSs, however, it is better that they lie open to dispute. For instance, the decision to put the burden to establish on PRO, is a PPS that would better be debatable. The reason is that this PPS might not be the most appropriate choice for all situations. Sometimes, it is better to put the burden to establish on PRO; at other times, it is better to put the burden to establish on CON. Still, in other situations it is best to put the burden to establish on both sides. (For example, the arguments required to provoke someone need not be as convincing as those needed to persuade someone; and arguments needed to persuade someone in one type of situation need not be as strong as those needed to persuade someone in another type of situation etc.) Thus, each situation requires another type of dispute, and it should ideally be possible to adapt the type of dispute to the situation at hand.

In the next section, it will be explained how a collection of PPSs can be altered by way of dispute.

6. Arguing about Different Protocols

There is no *à priori* reason why partial protocol specifications cannot form a part of the current opinion. Thus,

$$\text{opinion} = \{\neg G, \neg A, U, \neg Q\} \cup \{\alpha', \beta\}$$

is a perfectly normal expression, that represents a situation in which recent discussions lead the department to an opinion that is against the assignment of a travel grant ($\neg$G), against an extension of secretarial support ($\neg$A), and so forth, *plus* an order of procedure in which PRO has the burden to establish (α'), and repetition of moves is not allowed (β).

Once a part of the protocol specification belongs to the current opinion, it becomes possible (in principle) to alter that part by means of dispute. Factors that play a role in such disputes are, for instance,

D:	situation requires deep analysis
L:	large number of relevant cases
Z:	time pressure

Each protocol change has its own policies and precedents. Let us for example consider the partial protocol specification δ, according to which participants reply to the same claim at most twice. A (tacit) rule in meetings stipulates that a participant can only speak twice to a claim, but only after everyone who wants to speak for the first time does so. This rule keeps debate going and stops any member that is always popping up to talk after each member speaks. Let us assume that the

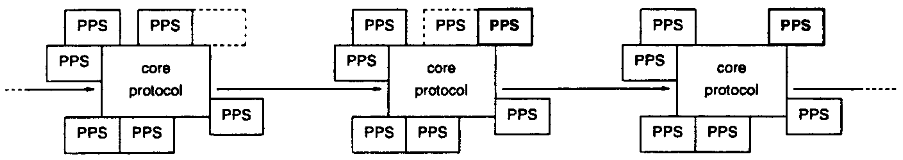

Figure 4. A PPS is established, at the expense of another.

case-base contains the following 9 guidelines on this issue, agreed upon by both parties:

$\mathsf{D} \succ \neg\delta$	$\mathsf{Z} \succ \delta$	$\mathsf{D,Z} \succ \delta$
$\neg\mathsf{L} \succ \delta$	$\mathsf{L,Z} \succ \delta$	$\mathsf{D,\neg T,\neg L,Z} \succ \delta$
$\mathsf{D,\neg L,Z} \succ \neg\delta$	$\neg\mathsf{D} \succ \delta$	$\mathsf{D,\neg L} \succ \neg\delta$

Thus, the first item suggests to allow for unlimited replies, in case a deep analysis is desired. The second item (in the second column) suggests to curtail search to two moves per party, whenever search must be conducted under time-pressure. The case-base also provides guidelines if conflicting interests are at stake. For example, if a deep analysis is desired under time-pressure. From the third item (in the third column) we conclude that it is recommended to let speed prevail over accuracy in such cases. (Which is not a universal truth. It just happens to be that the current case-base suggests this specific priority.) Other cases provide similar (procedural) guidelines.

Below we give two examples of establishing and modifying a PPS, respectively. In the first example, a PPS is established. In the second example, this PPS is modified by establishing another PPS that replaces it.

EXAMPLE 6.1. (Establishing a PPS.) Consider a situation in which an application is submitted beyond the deadline, which means that T does not hold. Let us further assume that the delayed reception is registered explicitly by ¬T. We also assume that the need for a profound analysis D is a fact. The protocol under which both parties will argue is the protocol described in Definition 2.2, and the same protocol that was used in the previous examples. With this protocol, PRO has the burden to establish (α'), and repetition of moves is not allowed for both parties (β). In addition, we assume that α' as well as β have the status of an opinion, which means that they are debatable:

$$\mathsf{facts} = \{\mathsf{E},\neg\mathsf{T},\mathsf{P},\mathsf{D}\}, \qquad \mathsf{opinion} = \{\alpha', \beta\}$$

Let us additionally assume that, for some reason, one of the two parties wishes to establish a potentially unlimited number of replies in dispute. That is, one of the two parties wishes to establish the PPS symbolized by $\neg\delta$. This dispute, on $\neg\delta$, typically precedes the dispute on the actual issue (for instance G).

1. PRO: $\mathsf{D} \succ \neg\delta$ [D]
2. CON: pass

New opinion: $\{\alpha', \beta, \neg\delta\}$.

What happens is the following. At line 1, PRO claims that the number of replies allowed would better be unlimited, since the need for a deep analysis (D) is a fact. CON passes at line 2, presumably because it failed in an attempt to find relevant material that interferes with PRO's claim. (CON could also have passed immediately, without searching for relevant counter-arguments first, but if CON's goal is to try to prevent PRO's success, that strategy would be highly unprofitable. It is in line with the spirit of the game to assume that CON has searched indeed.) Because CON passes, $\neg\delta$ is established by PRO.

In the preceding example, CON could not reply to PRO because none of the guidelines for δ were applicable. This, in turn, was caused by the fact that no 'δ-favourable indicators, such as $\neg$L and Z, were included in the current case.

EXAMPLE 6.2. (Modifying a PPS.) We now consider a case in which a 'δ-favourable indicator, namely Z, is included in the CFS. Together with the opinion just formed, this creates the following new situation:

$$\text{facts} = \{\text{E}, \neg\text{T,P,D,Z}\}, \qquad \text{opinion} = \{\alpha', \beta, \neg\delta\}$$

With the additional information, Z, the party that wishes to establish δ (which is now PRO) has more material at its disposal to argue for δ. This fact becomes apparent if a dispute on δ is initiated:

1. PRO: Z $\succ \delta$ [Z]
2. CON: D $\succ \neg\delta$ [D]
3. PRO: D, Z $\succ \delta$ [D, Z]
4. CON: pass

New opinion: $\{\alpha', \beta, \delta\}$.

At line 1, PRO claims that the number of replies allowed must be reduced to two, since the application must be handled under time-pressure (Z). Then CON replies at line 2 with the opposite claim (that the number of replies need *not* be reduced) based on the fact that worked in the first debate, namely, D. This time, however, CON's move is ineffective, since the party that favours δ (in this debate: PRO) is able to respond with a guideline that is more on-point than CON's at line 2. In fact, PRO's reply is based, not only on D, but also on Z. CON cannot find relevant material that interferes with PRO's move, and passes.

Example 6.1 and Example 6.2 show how a part of the protocol, in the form of a PPS, can be established in dispute. It is shown how an established PPS is incorporated in the current opinion, and thus forms a part of the standing order. It is also shown how the incorporation of a new PPS implies a modification of the protocol. However, neither δ nor $\neg\delta$ exercise an actual influence on the disputes. With both examples, both parties operate within the restrictions implied by the PPSs δ and $\neg\delta$. Thus, neither of the two PPSs were really effective. In Section 7, it is shown how PPSs actually control and direct the course of dispute.

Table I. Different types of propositions

	Domain of discourse	Procedure
Facts	Undebatable truths from the domain of discourse, e.g., E,T,¬S, . . . and E,T,S,¬X,I,¬F,¬P,V $\succ$ G E,T,¬S,¬I,P $\succ$ ¬G, . . .	Basic instructions that describe how to conduct a dispute
Opinion	Claims made in the domain of discourse, that are open to dispute, e.g., G, ¬A, U, . . .	Peripheral, situation-dependent, and changeable information about how a dispute should precisely be conducted, e.g., α', β, $\neg\gamma$, . . .

The use and meaning of the concepts introduced thus far – facts, protocol, opinion, and PPS – are summarized in the following table.

The table tells us that some of the propositions are facts, while other propositions are elements of the current opinion. Furthermore, some propositions pertain to issues in the domain of discourse, which other propositions pertain to the organization of dispute. This gives 2 × 2 = 4 different types of propositions.

Remark 1. Rules, cases, and CFSs are undebatable truths once a dispute is begun. Between disputes, they may be modified by the user at any time. Sometimes, the user analyzes different CFSs in the presence of an invariable case-base. Until the case-base is updated with a new precedent. Then, a new bunch of CFSs may be analyzed. But once a dispute is begun, it is assumed that the facts, rules and cases remain the same.

Remark 2. The basic procedure is not represented with the help of identifiers. This is because the basic procedure (alternatively referred to as the core protocol) is supposed to be a collection of low-level instructions to interpret facts, claims, and PPSs. (Cf. the LISP-code in Section 10.) This set of instructions does not change. Otherwise, the calculus would be interpreted by means of a self-modifying computer program. The latter is undesired, since self-modifying code may run incorrectly and may corrupt itself or other vital data.

7. The Strange Phenomenon of Self-Undermining PPSs

In computational dialectics, there is the idea that dispute is a proper instrument for test the tenability of a claim, without changing the status of that claim. The idea is that, if a claim is established, it can be established in subsequent disputes as well,

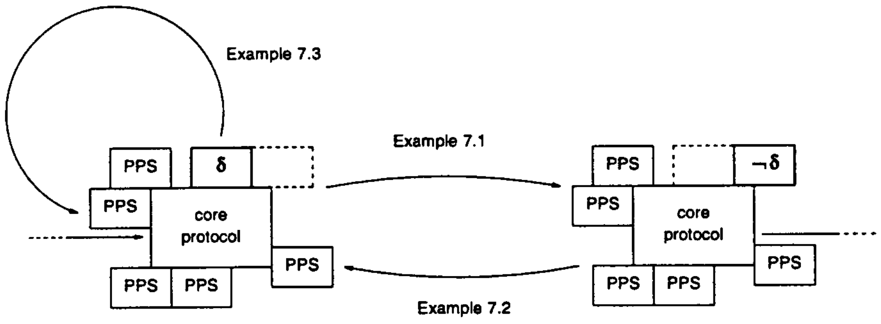

Figure 5. Alternating protocol, in the presence of fixed facts and cases.

provided that the underlying facts and cases remain untouched. Similarly, if a claim is denied, then it remains denied in subsequent disputes.

Surprisingly enough, it is not true that every dispute leaves the status of its main claim unaltered. In this section we show that there are situations in which the status of a claim can be altered by investigating it. Below, we show in detail how this may happen. We now introduce the phenomenon with an informal description:

1. The disputation protocol is defined, among others, with the help of a certain PPS, say δ.
2. For some reason, the opposite PPS, $\neg\delta$, is put forward as a claim in dispute. Thanks to the definition of the protocol and, in particular, thanks to the fact that δ in part of that definition, the PPS $\neg\delta$ can be established. Let us suppose that this indeed happens.
3. Since $\neg\delta$ is established in dispute, it is incorporated in the current opinion, at the expense of δ. Since the current opinion partially defines the protocol, the protocol changes.
4. Let us additionally assume that the protocol changes in such a way that δ can be established in dispute.
5. The claim δ is established in dispute, and reinstated as an effective PPS. We are back at point (1), where $\neg\delta$ can be established.

Because δ, as a PPS, undermines δ, as a claim, it is referred to as a so-called *self-undermining* PPS. Once δ is established, it changes the protocol in such a way that subsequent attempts to establish it fail. And once it is not part of the protocol, subsequent attempts to establish it, succeed. In the forthcoming examples we elaborate on self-undermining PPSs in further detail.

EXAMPLE 7.1. (Reinstating a PPS.) The previous dispute, in Example 6.2, has changed the protocol into a set of rules stipulating, among others, that each party is allowed to reply to the same claim at most twice (δ).

Let us assume that the CFS again grows by the introduction of a new fact, ¬L. This new fact informs both parties about the size of the case-base, which happens to be relatively small:

$$\text{facts} = \{\text{E},\neg\text{T},\text{P},\text{D},\neg\text{L},\text{Z}\}, \qquad \text{opinion} = \{\alpha', \beta, \delta\}$$

The introduction of a new fact, in this case ¬L, enables the proponent of the opposite PPS to try to reinstate $\neg\delta$. An actual attempt might result in the following dispute:

1.	PRO: D $\succ \neg\delta$	[D]
2.	CON: ¬ L $\succ \delta$	[¬L]
3.	PRO: D, ¬L $\succ \neg\delta$	[D,¬L]
4.	CON: Z $\succ \delta$	[Z]
5.	PRO: D, ¬ L,Z $\succ \neg\delta$	[D, ¬L,Z]

New opinion: $\{\alpha', \beta, \neg\delta\}$.

A consequence of the protocol under which the dispute is conducted, and a consequence of δ in particular, is that CON cannot move at δ. CON got stuck because it has replied already two times (at 2 and at 4), and 6 prevents CON from replying a third time. In particular, CON cannot reply with ¬T,D,¬L,Z $\succ \delta$, a case that would suffice to defend δ. Had this case been cited immediately at line 2, then CON could have repelled PRO's attack on δ with success. But CON has applied a bad strategy (at 2 and at 4), and therefore loses at 6; PRO wins at 5. Opinion shifts from δ to $\neg\delta$ (Figure 5).

EXAMPLE 7.2. (Reinstatement, part II.) The party that favours δ (which was CON but becomes PRO), can recover simply as follows:

1. PRO: ¬T,D,¬L,Z $\succ \delta$
2. CON: pass

New opinion: $\{\alpha', \beta, \delta\}$.

Current opinion is restored, without the introduction of new facts and cases. It follows that we are back in the situation at the beginning of Example 7.1, in which δ can be spoiled by a bad strategy of CON:

$$\text{facts} = \{\text{E},\neg\text{T},\text{P},\text{D},\neg\text{L},\text{Z}\}, \qquad \text{opinion} = \{\alpha', \beta, \delta\}$$

EXAMPLE 7.3. (Reinstatement, part III.) In the above situation, a better strategy for a defender of δ is to reply with the most specific case right from the start:

1.	PRO: D $\succ \neg\delta$	[D]
2.	CON: ¬T,D, ¬L,Z $\succ \delta$	[¬T,D,¬L,Z]
3.	PRO: pass	

The improved strategy makes PRO pass at line 3, before CON runs into the restriction implied by δ. Opinion is preserved, thanks to the application of a good strategy (Figure 5).

The effects of the various strategies are summarized by the state-transition diagram (STD) in Figure 5. In that diagram it is shown how different combinations of strategies and PPSs (δ or $\neg\delta$) lead to new PPSs.

The strange aspect of self-undermining PPSs, is that they cause protocol modifications against an otherwise stable background of fixed facts and cases. It is not strange that the definition of a protocol changes if new information becomes available. In such cases, modifications are even desirable. Examples of such protocol transitions were given in Example 6.1 and Example 6.2. These examples showed that new information causes a new CFS which, on its turn, produces a new protocol. Elaborating on these examples, one would expect that our primitive dialectic induces a function that maps every CFS onto a unique protocol, or onto a unique set of partial protocol specifications (PPSs). Thus, one would expect that every CFS has its 'own' unique (and probably 'best') protocol. However, Example 7.1 and Example 7.2 have shown that this is not true. There exist CFSs in which some claims, in particular PPSs, have an undefined dialectical status. These claims are neither 'established,' nor 'undecided,' nor 'denied'. Instead, the status of such claims remains unclear, and alternate between 'established' and 'denied', or between 'established' and 'undecided'. In the running example, the status of δ swings between 'established' and 'denied'.

Self-undermining PPSs occur in practice, in the form of self-undermining points of order. In a meeting, for example:

Consider a meeting of club members. Normally, the regulations prescribe that new proposals are adopted if and only if more than 50% of the present members favour that proposal in a voting. That is the rule in most meetings. Suppose that this particular meeting is concerned with a point of order, U, that proposes 100% consent as a condition for adoption of subsequent proposals. Thus if U is adopted, everybody has to vote in favour of any new proposal in order to let it pass. The presence, as well as the absence, of U (as an effective point of order) has particular effects. If U is in force, every decision of the meeting is mandated by every member, which is a fine property but, at the same time, delays the decision making process. With U, proposals pass if and only if every member agrees with it. If U is not in force, the opposite will hold. In that case. a reasonable speed of decision making is achieved at the expense of a reduced mandate. It is reasonable to assume that U is preferred by at least 50% of the members present, if it is absent as an effective point of order. This is so, because a considerable number of members feel themselves ignored by the outcome of recent votings, and will therefore vote for U to enforce participation in each voting. Thus, if U is absent, then at least 50% of the members will vote for U. Conversely, it is reasonable to assume that the removal of U is unanimously approved, if U is in force. This is so, because U holds up the progress of the meeting, and causes a backlog of proposals pending. Thus, if U is present, then 100% of the members will vote for the removal of U. U may be considered as a PPS, and if we do, we deal with a self-undermining PPS that is installed if it is absent, and removed if it is present.

We see that a practical instance of a self-undermining PPSs, U, undermines itself indirectly, via a causal chain of connected events. (Increased mandate implies an increased number of outvoted proposals, an increased number of outvoted proposals implies delay, delay implies reduced mandate, etc.) This is quite different from what we have seen with the abstract notion of self-undermining PPSs. The PPS δ in Example 7.1 and Example 7.2 undermines itself directly.

Before discussing the differences, it is probably best to consider another example:

Consider a small company, with decision parameter G. If G is established, the management decides to grow. (Attracting new personnel, expanding the office, etc.) If G is denied, the management decides to re-organize the company, by, among other things, reductions on personnel. In the beginning, the company is small, as it consists of the founders only. There are two possibilities: the company dies an early death and ceases to exist, or the company strikes root. Suppose that the company strikes root. In that case it has success, so that the founders wish to attract new man-power (G). Soon the ghost of bureaucracy emerges. People wear tailor-made suits, buy car-telephones, and have their own secretary. This new overhead causes unjustified expenses and reduces the overall success of the company. A reduction of the staff must be carried through (not G). Those that are ‘lucky to stay’ are frightened and shocked, but also happy that they have survived the re-organization. There is more commitment and reliability, and because of the shared sufferings the management perhaps notices an improved team-spirit. In consequence, the company manages to work itself out of the dip into a new period of success. Needless to say this revival involves the recruitment of new personnel (G).

Here, too, we see how a self-undermining point of order, G, controls itself through a number of intermediate events. From this example we may also conclude that self-undermining points of order, or self-undermining PPSs, are not necessarily ‘bad’. Instead of being unwanted anomalies, they help to govern complex social systems such as meetings, political institutions, bureaucratic offices, and other complex social organizations. Experiments in (Vreeswijk, 1995h) have shown how PPS-like protocol modifiers are used to govern a multi-agent system. (A multi-agent system is an artificial social organization of autonomous and intelligently operating software robots.) protocol modifiers then become ‘political measures,’ that exercise corrective control to the formation of social and/or political laws.

In this paper, however, a self-undermining PPS is nothing more than a logical curiosity that causes a protocol to alternate between two different states.

8. Direct Applications

The calculus that has been used to explain the use of PPSs is relatively simple. Here, arguments are cases, which are uniform and indivisible. Defeat among arguments is ruled by an elementary criterion of specificity, and every dispute simply alternates between two parties that must respond to previous claims immediately. Despite its simplicity, the calculus has proven to be adequate to explain the use of PPSs in computational dialectics.

A simple calculus permits us to get the main ideas across. But practical circumstances require a formalism that is more powerful and involves additional features, such as:

- compound arguments of different sorts [rules, cases, rules formulated from cases, arguments formulated from rules, compressed arguments, cases formulated from dispute]
- complex and refined notions of defeat among arguments [pre-emption, directness, shortness, preferred premises, accrual, floating conclusions, lex specialis, lex superior, lex posterior]
- different parties (>2) [plus adjudicator, plus mediator, group reasoning, a-synchronous group reasoning, groups with a changing set of participants, composition, participants with different competencies]
- Sophisticated protocol [multiple layers of amendment, resistant to various forms of deadlock and unfairness, reversibility-preserving]

Real disputation systems include one or more of these features, in order to establish a refined dialectical process: a process that is refined enough to meet the standards and requirements of practical application domains (Loui, 1998; Vreeswijk, 1995h).

An example of a practical application domain is the maintenance of a telecommunication network with high-level tools, such as SSCFI. SSCFI (Special Service Circuit Fault Isolation) is an expert system that is currently being deployed at all the GTE Telephone companies in the USA (Georgakopoulos et al., 1994). It functions as an expert test technician that reads and interprets trouble reports on special service circuits, decides to conduct tests and interprets the results of those tests using in the process a number of remote test and database systems, and finally routes the report to the appropriate repair group with the result of its analysis. Instances of SSCFI are autonomous processes. They operate in the background and have full control over the trouble report queues to which they are initially assigned. Only a minimal administrative interface is provided for administrators to monitor SSCFIs. SSCFIs are intelligent autonomous systems performing their tasks without direct human operational control. These tasks are not to represent information to people but to act within the context of a workflow system where field repair crews are at the receiving end.

The current evolution SSCFI points towards a distributed problem solving system involving SSCFIs throughout the GTE regions. SSCFIs need fast local access to all the database resources, test systems and test points involved in testing circuits. To maximize throughput, these resources are locally available on their LANs. SSCFIs can do simple load balancing by spawning children processes subjected to computing resources and locally sharing test load. The idea of autonomous protocol modification comes into the picture as soon as load sharing at a national level between all SSFIs is being considered to share computational resources. Some ideas under discussion are to use PPSs to propose other SSCFIs to accept responsibility of testing circuits along with remote control of the resources required to perform those test and to negotiate the conditions under which this is done. Even current SSCFIs respond to administrator commands in a delayed fashion and only when current testing is completed. This example illustrates a possible if not likely future evolution of an existing intelligent operating system involving a “retrofit” of distributed problem solving capabilities.

9. Legal Relevance

Currently, most of the work on formal justice follows a structural approach, in which the procedure is fixed, and arguments are analyzed by their content, composition, and mutual influence. This is appropriate, since the rules of procedure in court are laid down in the law, and cannot change during the time-span of a legal process.

In meta-juridical practice, e.g., in an environment where one *discusses* about the law, notably parliament and senate, procedural arguments play an important role. This is recognized by legal researchers, and, recently, attempts are made to formally elaborate on the procedural aspects of juridical debate. Most notably, this is done in Gordon’s pleadings game (1995), but comparable endeavours are (Loui, 1994), (Hage et al., 1994) and (Lodder, 1997). In these approaches, it is emphasized that legal procedure is as much part of the law as the more substantial rules, and that reasoning about the law can involve, and typically *does* involve, dissent from ruling, points of order, and other elements of a changing procedure. Peter Suber (1990) goes one step further, by studying what happens if amendments are made to the law that regulates the procedure in the environment where the law is formed (parliament). See also (Vreeswijk, 1995e, 1995h). But these are somewhat theoretical examples, and we do not need to go this far to obtain a fair estimate of the role of procedure in juridical practice. In fact, elements of procedure already pervade modern civil law. In civil procedure, parties often do not go at length to make a substantial argument during pleading. Instead, they assert or deny “essential” facts which are believed to entitle them to legal relief, such as monetary compensation for damages, or preemption by technicalities. Thus, the AI and Law research area acknowledges that a formal study of legal argumentation should not only be interested in the result. but that the process or procedure by means of which the result is achieved, is important too.

In this light, the value of a formal representation of the concept of a standing order is twofold:

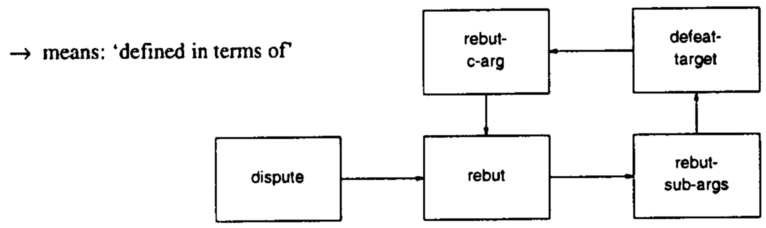

Figure 6. Dependency between essential LISP-functions in IACAS.

1. Firstly, formalization of procedural aspects of reasoning, and legal reasoning in particular, enables a new methodology for legal philosophy, and a formal vehicle on which cases in procedural justice can be tested and examined.
2. Secondly, formalizing procedural aspects such as exposed here, may provide key technology for new kinds of computer applications. The insight gained from AI models of procedural argument can be used in legal education to improve the quality of legal practice whether or not lawyers ever use computer systems in their daily work. Playing formal dialogue games in law school may also be instructive (Gordon, 1995 and Lodder's *DiaLaw*, 1997).

A formalization of the concept of a standing order thus uncovers essential aspects of procedural legel reasoning, and provides a formal handle on the procedural machinery of legal dispute.

10. Implementation

It is not difficult to incorporate partial protocol specifications into existing disputation systems, such as LMNOP (Loui, Olson, Normann, and Merrill, 1993), Nathan (Loui, 1993), the Pleadings Game (Gordon, 1989, 1995), and IACAS (Vreeswijk, 1994). In this section, we indicate how PPSs can be incorporated in IACAS.

IACAS is a program to play with the elementary ideas of computational dialectics. It allows the user to start a dispute, given a number of facts, rules and cases. The program is written in LISP and is originally meant to demonstrate the theory outlined in (Vreeswijk, 1993, 1997). IACAS stands for **I**nter**AC**tive **A**rgumentation **S**ystem. The user is allowed to interact with the system in many ways. He (or she) may set parameters, accommodate output, and tailor dispute records to personal taste. IACAS has several unique features. The foremost sophisticated feature of ICASAS is the possibility of *analyzing* the epistemic status of a proposition. This is done according to Chisholm's 'Theory of Knowledge' (1977). Chisholm has a theory in which propositions can, for instance, be 'certain', 'beyond reasonable doubt', or 'counterbalanced'. When IACAS is requested to analyze the epistemic status of a proposition, it initiates a debate on that proposition and its negation (the outcome of the one does not fix the outcome of the other) and synthesizes the outcomes into an epistemic modality.

An important drawback of IACAS is that it has a fixed protocol for dispute. the protocol is defined with the help of 5 functions. The function dispute is used to initiate the procedure, after which PRO and CON apply the other four functions recursively. Below is the code of rebut. the function computes whether an argument (denoted by arg in the function's argument list) can be rebutted with success. If so, the function rebut returns the value success. Else, the function returns either inconclusive or failure, depending on the existence of an adequate rebuttal:

```
(defun rebut (arg arg-nr arg-line-nr args-pending party level)

        ; arg =               argument for which a rebuttal must be sought
        ; arg-nr =            nr. of attempt to support claim
        ; arg-line-nr =       line nr. on which the argument appears
        ; args-pending =      arguments that lie open in a super-dispute
        ; party =             party that presented the argument in question
        ; level =             indicates depth of dispute

        ; if the argument in question is a fact, then it is impossible to find a rebuttal
        ; for it --- return 'failure:

        (when
        (factp arg)
        (return-from rebut 'failure))

; if an argument is strict, i.e., if it is a proof, then it is also impossible to find
; a rebuttal for it:

(when
        (strict-argumentp arg)
        (store-where-proven (conc-of arg) arg-nr arg-line-nr)
        (return-from rebut 'failure))

; if the argument in question has been defeated in an earlier stage of the dispute,
; then it is possible to find a rebuttal for it --- return 'success:

(let
        ((arg-previous-defeat (where 'failure arg)))
        (when
                (consp arg-previous-defeat)
                (return-from rebut 'success)))

; if the conclusion of the argument has been refuted by a strict argument, i.e., a
; proof, in an earlier stage of the dispute, then it is possible to find a rebuttal
; for it:
(let
        ((arg-proven-contrary (where-proven-contrary arg)))
        (when
                (consp arg-proven-contrary)
                (conc-of arg) arg-nr arg-proven-contrary party level)
                (return-from rebut 'success)))

; if the argument in question is pending, i.e., if it is put forward in an earlier
; stage of the debate, and that debate is not yet closed, then the rebuttal is
; simply equal to the remark that this argument may not yet be disputed --
; this remark is a successful rebuttal:
```

```
(let
        ((pending-at (where-submitted arg args-pending)))
        (when
                (consp pending-at)
                (return-from rebut 'success)))

; if the argument of the opponent is neither a fact nor a proof,
; if it is not defeated in an earlier stage of the dispute,
; if the opposite has not been proven, and if the argument is not
; pending, then try to defeat it on substantial grounds ---
; the latter is done by checking the tenability of the main argument
; and each of its (proper) sub-arguments

(rebutt-sub-args arg arg-nr arg-line-nr args-pending party level))
```

The other three functions are responsible for substantial rebuttal (on sub-arguments), finding specific counter-arguments for conclusions of sub-arguments (so-called *targets*), and finding rebuttals for the counter-arguments generated, respectively. They are defined likewise.

To build PPSs into IACAS, a notion of current opinion must be introduced. The theory in the previous sections indicates how this must be done: from Section 4 we know that the current opinion is defined as a collection of propositions, and from Section 6 we know that this collection may also contain PPSs. If this combination of propositions and PPSs is translated in LISP-syntax, we obtain the following:

```
(set   'OPINION
       (append

              '((not assignment-of-travel-grant)
                (not extension-of-secretarial-support)
                funding-of-external-scholarship
                (not modernizing-computer-infrastructure)
                . . .
                . . .)

              '((limited-rebuttal 3)
                (burden-of-proof PRO)
                . . .
                . . .)))
```

With the help of the definition of OPINION, one can begin to make slots for the various PPSs, in the function rebutt. A slot for the PPS limited-rebuttal, for instance, can made as follows:

```
(defun rebut (arg arg-nr arg-line-nr args-pending party level)

        (when
                (factp arg)
                (return-from rebut 'failure))

        (when
                (strict-argumentp arg)
                (store-where-proven (conc-of arg) arg-nr arg-line-nr)
                (return-from rebut 'failure))

>       (when                                                         < new
>               (memberp 'limited-rebuttal OPINION)                   < new
>               (when                                                 < new
>                       (< (assoc limited-rebuttal) arg-nr))          < new
>                       (return-from rebut 'success))                 < new
```

```
        ; et cetera
        ; et cetera
        ; et cetera

        ; if the argument of the opponent is neither a fact nor a proof,
        ; if it is not defeated in an earlier stage of the dispute,

>       ; if it has not exceeded a preconceived number of maximal rebuttals,          < new

        ; if the opposite has not been proven, and if the argument is not
        ; pending, then try to defeat it on substantial grounds ---
        ; the latter is done by checking the tenability of the main argument
        ; and each of its (proper) sub-arguments

        (rebutt-sub-args arg arg-nr arg-line-nr args-pending party level))
```

It will now be clear how slots for other PPSs can be inserted into the code of rebutt. Similar operations can also be performed at the other components of the core protocol, viz. rebutt-sub-args, defeat-target, and rebutt-c-arg. The final result will be a disputation protocol that can be modified on-the-fly by the parties that use it.

11. Conclusion

In this paper, it is shown how a disputation protocol may be modified with the help of partial protocol specifications (PPSs).

A PPS is a piece of protocol-related information in the form of a logical proposition that can be put forward as a claim in dispute. Claims can be established or denied. Once a PPS, as a claim, is put forward and established in dispute, it is saved as an effective ‘point of order’ in a set known as the current opinion. All PPSs in the current opinion help to control and direct the dispute. In this paper it is shown how PPSs are established, denied, and reinstated as effective regulators of the argumentation process. Furthermore, the existence of self-undermining PPSs is demonstrated. A self-undermining PPS is a piece of protocol that is established if and only if it is ineffective. (And denied if and only if it is effective.) The paradox of a self-undermining PPS is due to the fact that it may cause ‘unstable’ protocols, in the presence of fixed facts and cases. A self-undermining PPS may cause the protocol to alternate between to states ‘forever’.

The paper concludes with a description of how PPSs can be built into the code of existing disputation systems. As an example, we took IACAS. Once IACAS is adapted and extended, it is able to maintain an opinion, not only on issues in the domain of discourse, but also on issues that pertain to order and procedure. With a standing order, IACAS and other disputation systems are able to maintain a protocol that can be adapted to changing circumstances.

Acknowledgements

Thanks are due to Peter Braspenning, Henry Prakken and two anonymous referees for reading and commenting on my work.

References

Ashley, K. D. and Rissland, E. L. 1988. A case-based approach to modelling legal expertise. *IEEE Expert* **3**(3), 70–77.

Ashley, K. D. 1990. *Modeling Legal Argument: Reasoning with Cases and Hypotheticals*. MIT Press, Cambridge, MA.

Brewka, G. 1994. A reconstruction of Rescher's theory of formal disputation based on default logic. *Proceedings of the 11th European Conference on Artificial Intelligence*. John Wiley & Sons, Ltd., pp. 366–370.

Chisholm, R. 1977. *Theory of Knowledge.* 2nd edn, Foundations of Philosophy Series, Prentice-Hall, New Jersey.

Dung, P. M. 1995. On the acceptability of arguments and its fundamental role in non-monotonic reasoning, logic programming, and N-person games. *Artificial Intelligence* **77**(2), 321–357.

Georgakopoulos, D., Hornick, M., Krychniak, P., and Manola, F. 1994. Specification and management of extended transactions in a programmable transaction environment. *Proceedings of the 10th International Conference on Data Engineering.* Houston, Texas. Also published as TC-0207-02-93-165, GTE Laboratories Incorporated, February 1993.

Gordon, T. F. 1995. *The Pleadings Game: An Artificial Intelligence Model of Procedural Justice*. Kluwer, Dordrecht.

Hage, J. C., Leenes, R., and Lodder, A. R. 1994. Hard cases: A procedural approach. *Artificial Intelligence and Law*, Vol. 2. Kluwer Academic Publishers, pp. 113–167.

Krabbe, E. C. W. 1985. Formal systems of dialogue rules. *Synthese* **63**, 295–328.

Lodder, A. R. 1997. Procedural arguments. In *Proc. of the Tenth Jurix Conference.* Computer/Law Institute, Vrije Universiteit, Amsterdam, pp. 21–32.

Loui, R. P. 1987. Defeat among arguments: A system of defeasible inference. *Computational Intelligence* **3**(2), 100–106.

Loui, R. P. 1989. Defeat among arguments: II. Report No. WUCS-89-06, Department of Computer Science, Washington University, St. Louis, Missouri, Washington.

Loui, R. P. 1990. Ampliative inference, computation, and dialectic. In J. L. Pollock (ed.), *AI and Philosophy*. MIT Press.

Loui, R. P. 1994. Argument and arbitration games. In *Working Notes of the Workshop on Computational Dialectics, AAAI Conference*. Seattle, pp. 72–83.

Loui, R. P. and Norman, J. 1995. Rationales and argument moves. *Artificial Intelligence and Law* **3**, 159–189.

Loui, R. P. 1998. Process and policy: Resource-bounded non-demonstrative argument. *Computational Intelligence* **14**, 1–38.

Nute, D. 1988. Defeasible reasoning and decision support systems. *Decision Support Systems* **4**(1), 97–110.

Page, C. V. 1991. Principles for democratic control of bounded-rational, distributed, knowledge agents. *Proceedings of the European Simulation Multiconference*, 359–361.

Pollock, J. L. 1994. Justification and defeat. *Artificial Intelligence* **67**(2), 377–407.

Poole, D. L. 1985. On the comparison of theories: Preferring the most specific explanation. *Proceedings of the IJCAI*, pp. 144–147.

Prakken, H. 1993. *Logical Tools for Modeling Legal Argument*, doctoral dissertation, Vrije Universiteit, Amsterdam.

Prakken, H. and Sartor, G. 1996. A dialectical model of assessing conflicting arguments in legal reasoning. *Artificial Intelligence and Law* **4**, 331–368.

Prakken, H. and Sartor, G. 1998. Modelling reasoning with precedents in a formal dialogue game. To appear in *Artificial Intelligence and Law*.

Rissland, E. L. and Daniels, J. J. 1995. A hybrid CBR-IR approach to legal information retrieval. In *Proceedings of the Fifth Int. Conf. on Artificial Intelligence and Law (ICAIL-95)*. College Park, MD, pp. 52–61.

Rissland, E. L., Skalak, D. B., and Friedman, M. T. 1955. Evaluating a legal argument program: The bankXX experiments. CMPSCI Technical Report 95-30.

Suber, P. 1990. *The Paradox of Self-Amendment, a Study of Logic, Law, Omnipotence, and Change*. Peter Lang Publishing.

Verheij, H. B. 1995a. Accrual of arguments in defeasible argumentation. In *The Proceedings of the 2nd Dutch/German Workshop on Nonmonotonic Reasoning*. Utrecht, pp. 217–224.

Verheij, H. B. 1995b. Arguments and defeat in argument-based nonmonotonic reasoning. In *Progress in Artificial Intelligence: Proceedings of the 7th Portuguese Conference on artificial Intelligence (EPIA '95)*. Springer, Berlin, Madeira, Portugal, pp. 213–224. Also published as report SKBS/B3.A/95-04.

Vreeswijk, G. A. W. 1993. *Studies in Defeasible Argumentation*. Doctoral dissertation. Departments of Mathematics and Computer Science, Vrije Universiteit, Amsterdam.

Vreeswijk, G. A. W. 1993. Defeasible dialectics: A controversy-oriented approach towards defeasible argumentation. *The Journal of Logic and Computation* **3**(3), 3–27. Another version of this article is available via anonymous ftp from ftp.cs.rulimburg.nl :/pub/papers/vreeswyk.

Vreeswijk, G. A. W. 1994. IACAS user manual v1.0, Technical Report CS 95-03, Vakroep Informatica (FdAW), Rijksuniversiteit Limburg, Maastricht, The Netherlands. Available via anonymous ftp from ftp.cs.rulimburg.nl :/pub/papers/vreeswyk.

Vreeswijk, G. A. W. 1995b. Formalizing Nomic: working on a theory of communication with modifiable rules of procedure, Technical Report CS 95-02, Vakgroep Informatica (FdAW), Rijksuniversiteit Limburg, Maastricht, The Netherlands. Presented at the Fourth International Symposium on Cognitive Science, Donostia, San Sebastian, May 3–6, 1995.

Vreeswijk, G. A. W. 1995e. Several experiments in elementary self-modifying protocol games, such as Nomic, Technical Report CS 95-06, Vakgroep Informatica (FdAW), Rijksuniversiteit Limburg, Maastricht, The Netherlands. Submitted.

Vreeswijk, G. A. W. 1995f. IACAS: An implementation of Chisholm's principles of knowledge. In *The Proceedings of the 2nd Dutch/German Workshop on Nonmonotonic Reasoning*. Delft University of Technology, University of Utrecht, Utrecht, pp. 225–234.

Vreeswijk, G. A. W. 1995g. The computational value of debate in defeasible reasoning. *Argumentation: An International Journal* **9**(2), 305–342. Another version of this article is available via anonymous ftp from ftp.cs.rulimburg.nl :/pub/papers/vreeswyk.

Vreeswijk, G. A. W. 1995h. Emergent political structures in abstract forms of rule-making and legislation. Technical Report CS 95-08, Vakgroep Informatica (FdAW), Rijksuniversiteit Limburg, Maastricht, The Netherlands.

Vreeswijk, G. A. W. 1997. Abstract argumentation systems. *Artificial Intelligence* **90**, 225–279.