The Zeno Argumentation Framework

Thomas F. Gordon
German National Research Center for Information Technology
thomas.gordon@gmd.de

Nikos Karacapilidis
INRIA Sophia Antipolis, Action AID
Nikos.Karacapilidis@sophia.inria.fr

Abstract

The Zeno Argumentation Framework is a formal model of argumentation based on the informal models of Toulmin and Rittel. Its main feature is a labelling function using arguments to compute heuristic information about the relative quality of the alternative positions proposed as solutions for some practical issue. The Zeno Argumentation Framework was designed to be used in mediation systems, an advanced kind of electronic discussion forum with special support for argumentation, negotiation and other structured forms of group decision-making.

1 Problem Statement

The cities of Bonn and Sankt Augustin are planning a residential area and high-technology "park" in an area between the two cities. This area was zoned for agricultural purposes only, so a formal proposal was made to change the zoning ordinance. According to local law, changes of this kind are subject to a formal review procedure. This procedure requires that the plan be made available to the general public for their comments, criticisms and suggestions.

In the European GeoMed project (GeoMed stands for "Geographical Mediation"), our main task at GMD is to design and implement a *mediation system* for the World-Wide Web, which shall enable public review procedures in a way which is more efficient, transparent, fair and, it is hoped, democratic than current practice. The idea is to use the Web to enable interested citizens and representatives of public interest groups to more easily access and view the development plans and to take part in an electronic *discussion forum* to express and share their views and opinions.

A mediation system, as we conceive it, is a kind of computer-based discussion forum with particular support for argumentation. In addition to the generic functions for viewing, browsing and responding to messages, a mediation system uses a formal model of argumentation to facilitate retrieval, to show and manage dependencies between arguments, to provide heuristic information focusing the discussion on solutions which appear most promising, and to assist human mediators in providing advice about the rights and obligations of the participants in formally regulated decision making procedures.

The persons taking part in the discussion will typically come from a variety of professional backgrounds, with different sources of information and different preferred methods for processing this information. A city planner might model the information at his or her disposal using a geographical information system. A politician might rely on high-level briefs from his or her support staff. A social scientist might have constructed a statistical model. Last but not least, as our goal is to open up public policy and planning procedures to a wide public, most participants will rely on their common sense, without using some more professional methodology.

In summary, the intended application scenarios are characterized by the following features, which will lead to requirements for the mediation system:

- The participants have access to different sources of information, knowledge and models. They do not share a set of premises. The amount and quality of their background knowledge will vary widely.
- The participants use different methods to organize, structure process and evaluate information and knowledge. Most participants will not be proficient in formal methods from mathematics, logic, or computer science.
- The participants have competing viewpoints and interests. In many cases they will be adversaries, with only a limited willingness to cooperate. The social atmosphere is one of caution, suspicion and mistrust.
- The issues to be decided are practical, rather than theoretical. The central problem is to make a choice among alternatives; the issue is which position or claim to accept, or which course of action to take.
- Finally, time is of the essence. A decision must be reached with a fixed period of time. Other resources required to find an acceptable solution, such as money and information, are also limited.

The rest of this paper is organized as follows. The following section presents a software engineering analysis of the mediation system and includes an example discussion showing some of the kinds of argument we would like to support. The purpose of this first section is to make clear the requirements for our formal model of argumentation, presented next in Section 3. We finish up with a section on related work and some conclusions.

2 Software Engineering Analysis

This section presents a brief software engineering analysis of the problem, using the popular Object Modeling Technique (OMT)

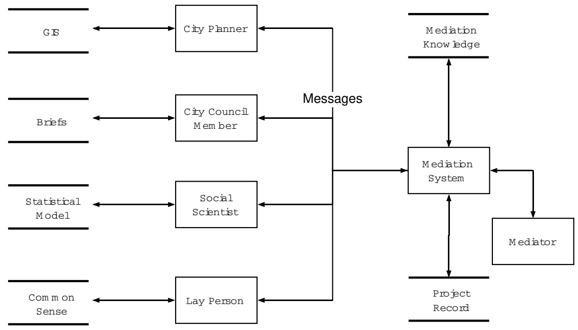

Figure 1: Functional Model of Mediation

[Rumbaugh, 1991]. OMT provides three basic diagramming techniques for designing and illustrating a system from three complementary perspectives: object diagrams show data structures and their relationships, functional diagrams show data flow between processes, and dynamic models show events, states and causal dependencies.

To start, Figure 1 shows a functional model for an example planning procedure with four participants with different professional backgrounds.

The participants communicate with each other through the mediation system. The *mediation knowledge* available to the system includes some general purpose model of argumentation and, in a more fully developed version, knowledge about the specific administrative procedures applicable, such as an environmental review procedure.

The *project record* stores the messages exchanged by the participants, as well as the evolving model of this particular discussion, using the concepts of the general purpose model of argumentation. This model also maintains a mapping to the original messages, to allow the model to be used as an index for finding messages, or parts of messages, of interest.

The *mediator* actor represents the human mediator or moderator of the discussion. The mediation system is conceived as an intelligent support system for human mediators. It is not our goal to fully automate this task.

Figure 2 is an OMT object model of the mediation system. Each box in this diagram represents a class, in the sense of object-oriented programming. The links between the boxes represent associations and inheritance relationships. The top part of the diagram, above the *project record* box, models the objects required for managing a communication connection with the user. It is an abstract view of a generic client-server architecture and not of particular interest to us here.

The *Project Record* and the classes below it are more relevant. They show the internal structure of the *Project Record* persistent store we saw above in the functional model. A project record consists of the discussion and a queue of incoming messages. (The mediator will have a chance to review incoming messages before they are made public.) A *discussion* object is associated with some (zero or more) messages and models of the discussion. Notice that, in principal, there can be several models of the discussion, to support different views or interpretations of the original messages.

A discussion model is constructed by creating a *marked message* for each message to be included in the model. A marked message is, if you will, a model of the message showing its argumentation structure. The marked messages are parsed to create and extend a relational model of the whole discussion. Each record in this relational model is called an *element*, as shown in the object model.

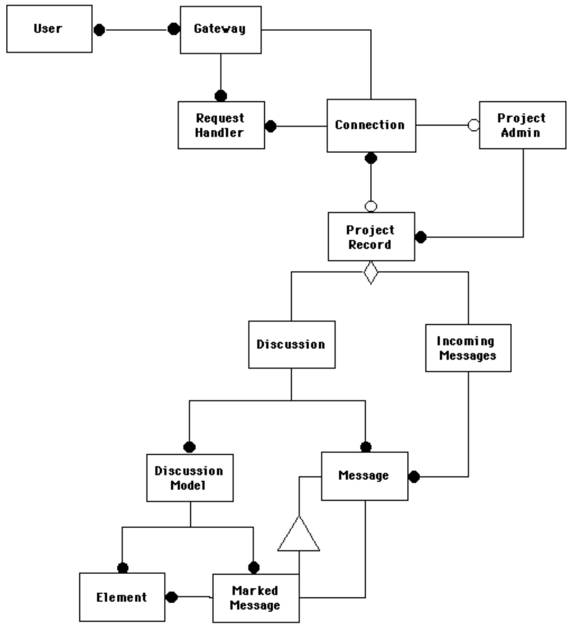

Figure 2: Object Model of Mediation

Elements are of several kinds, as shown in the Figure 3. This figure concisely displays the structure of the Zeno Argumentation Framework, which shall be the focus of our attention in this paper.

Our model is a formal variant of Horst Rittel's informal Issue-Based Information System (IBIS) model of argumentation [Kunz and Rittel, 1970, Rittel and Webber, 1973]. The basic elements of the IBIS model may be linked together in almost every imaginable way, to produce finite argumentation graphs. Using hypertext techniques, nice graphical user interfaces have been built for browsing IBIS graphs. The nodes of the graphs can contain arbitrary natural language expressions and other forms of media. Such systems can be quite useful for structuring and organizing information, despite their lack of formal semantics, and are particularly easy to build today using the protocols and document formats of the World-Wide Web.

Figure 4 provides another view of our argumentation model. It displays a *dialectical graph* for a discussion between a husband and wife about which car to buy. A dialectical graph shows a particular state of argumentation, at one moment in time. Although similar to a speech act graph, it emphasizes the role and function of the speech acts in argumentation, rather than their history. For example, a speech act which serves multiple argumentation functions will appear several times in the dialectical graph, but only once in a speech act graph.

One could well imagine that the following discussion took place:

Husband. Honey, we've been thinking about buying a new car. Do you have something particular in mind? (I1)

Wife. Well, yes. I think we should buy a Volvo station wagon. (P2)

Husband. But, that's such a family car. (P5) Let's buy a nice fast sports car. A Porsche would be great. (P1)

Wife. Isn't a Porsche pretty expensive? (P4) And besides, I think we should buy a safer car. Volvos are built like a tank. (P6)

Husband. What makes you think Volvos are so safe? (I2)

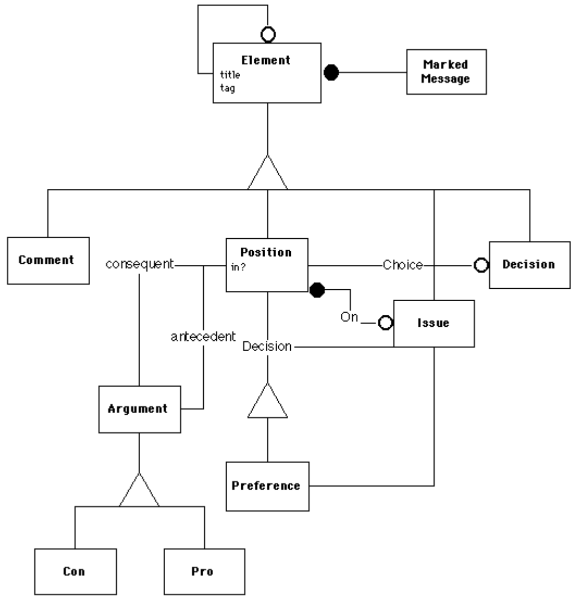

Figure 3: Object Model of Argumentation Elements

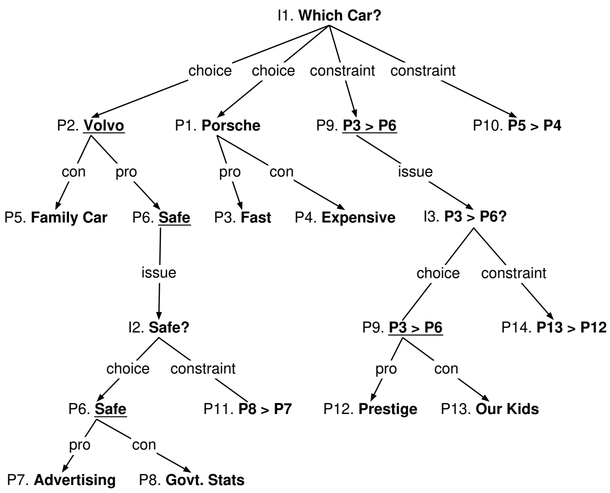

Figure 4: A Dialectical Graph

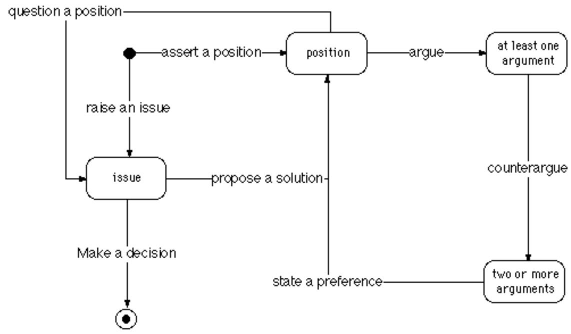

Figure 5: Dynamic Model of Argumentation

Wife. Don't you watch TV? Haven't you seen the advertisements? (P7)

Husband. Oh, come on Honey. I read a report in Auto Sports Today the other day which cited some government accident statistics. (P8) Do you know what? Volvos were said to be involved in more fatal accidents than almost any other brand. And besides, having a fast car is more important to me than having a safe car. (P9)

Wife. Why? (I3)

Husband. Look. I've been wanting a fast sports car ever since I finished law school. An attorney in this town has to have a dynamic image. (P12)

Wife. Yes, dear. But what about Betty and Susan. We have to think of the safety of our kids first. (P13, P14)

Husband. I guess you're right about that. But where does that leave us? I still think I rather pay a few thousand dollars more for a Porsche than drive such a boring family car. (P10)

Even though this is a short, if somewhat contrived example, it already may be becoming difficult to keep track of all of the arguments and their interrelationships. Which position has the most support at the moment? The Volvo or the Porsche? A good mediation system should make it easy to quickly obtain an overview of the state of the debate.

To complete our software engineering analysis, Figure 5 is an OMT dynamic model showing which kinds of speech acts are possible in various states of the project record. In this diagram, the rounded boxes describe states of the project record and the arcs represent speech acts which are possible in each state. The black dot is an initial state and the "bulls eye" is a terminating state for a single issue. The discussion may continue to resolve other open issues. The *argue* and *counterargue* speech acts should not be confused with *pro* and *con* arguments in the object model. Either kind of argument can be made using the argue speech act. The counterargue speech act asserts an argument of the complementary kind.

3 Dialectical Graph Labelling

Given the information in the model of a discussion, using the Zeno model of argumentation introduced in the previous section, it becomes possible to label the alternative positions of an issue, using a kind of *reason maintenance* procedure [Doyle, 1979]. This support for a kind of inference is an important advantage of our system over the informal Toulmin and IBIS models of argumentation.

This section presents a formal, algebraic reconstruction of our model of argumentation, ending with a definition of our labelling function. This definition will be a recursive, mathematical specification, not an efficient algorithm for computing labels. Although it could be implemented using a functional style of programming in a straightforward way, a real implementation should use techniques from the reason maintenance literature for caching labels and incrementally propagating changes through the dialectical graph as it is extended with new issues and positions.

Most of the examples below will be drawn from the Porsche example as displayed in the dialectical graph of Figure 4.

Regarding our notational conventions below: Upper case identifiers, such as `P1`, will be used to name object-level constructs in our formalism. Distinct letters will be used for each kind of object-level construct. Lower case identifiers, such as `p1`, will be used as meta-level variables ranging over objects of the type indicated by the letter.

Definition 1 (Positions) *Let* `P` *be a set of* positions, `P1, ..., Pn`.

Definition 2 (Terms) *Terms are defined inductively as follows:*

1. *Every position in* `P` *is a term.*
2. *If* `t1` *is term, then* `-t1` *is a term.*
3. *If* `t1` *and* `t2` *are terms, then* `t1+t2` *and* `t1-t2` *are terms.*
4. *Nothing else is a term.*

Example 1 `P1, P2, P3-P4` *and* `P5-P6` *are all terms.*

Definition 3 (Preference Expressions) *Let* `t1` *and* `t2` *be two terms. Then*

1. `t1 > t2` *and* `t1 = t2` *are preference expressions.*
2. *Nothing else is a preference expression.*

Example 2 `P3 > P6` *and* `P13 > P12` *are two preference expressions used in the Porsche example.* `P4+P5 > P10-P2` *and* `P16 = P6-P8+P13` *are also preference expressions.*

Definition 4 (Arguments) *Let* `p1` *and* `p2` *be two positions in* `P`. *There are two kinds of arguments:*

1. `pro(p1,p2)` *is an argument.*
2. `con(p1,p2)` *is an argument.*
3. *Nothing else is an argument.*

In both `pro(p1,p2)` *and* `con(p1,p2)`, `p1` *and* `p2` *are called the* antecedent *and* consequent *of the argument, respectively.*

Example 3 `pro(P6, P2)`, *and* `con(P5, P2)` *are arguments. In these examples, the positions* `P6` *and* `P5` *are antecedents; the position* `P2` *is the consequent of both arguments.*

Definition 5 (Constraints Table) *A* constraints table *is a finite mapping from positions to preference expressions. A position is a member of a constraints table if it is mapped in the table to some preference expression.*

Example 4 `({P9: P3 > P6, P10: P5 > P4, P11: P8 > P7, P14: P13 > P12}` *is a constraints table.*

Definition 6 (Issues) *Let* `C` *be a finite set of positions and* `K` *be a constraints table. An issue is a pair* `(C,K)`. *Intuitively* `C`, *the* choices, *represents the alternative solutions to the problem or question raised by the issue and* `K`, *the constraints table, contains the positions about the relative value of the arguments made for and against these choices.*

Example 5 `({P1, P2},{P9: P3 > P6, P10: P5 > P4})` *is an issue.*

Notice that the positions in a constraints table may also be members of the choices of some issue. This allows argumentation about constraints.

Example 6 `({P9}, {P14: P13 > P12})` *is an issue. (In the diagram, this is the issue about whether speed is more important than safety. The wife argued that it is not, pointing out that their children are more important than prestige.)*

Definition 7 (Dialectical Graphs) *Let* `P`, `A` *and* `I` *be finite sets of positions, arguments and issues, respectively, such that the antecedents and consequents of all arguments in* `A` *are members of* `P`. *Then* `(P,A)` *forms a directed, finite graph in which the positions are nodes and the arguments in* `A` *are edges. Arguments link the positions by their antecedent and consequent. The tuple* `(P,A,I)` *is called a* dialectical graph *if the positions of the choices and constraints of every issue in* `I` *are members of* `P`.

Example 7 *Figure 4 depicts a dialectical graph. Issues are also shown as nodes in this figure, using labelled edges to mark their sets of choices and constraints. If an issue is made out of an existing position, the position will be shown twice in this figure: once for its role in an argument or constraint, and once in its role as a choice for the issue.* `P6` *and* `P9` *are examples of this. Both instances of such a position are* identical, *not copies to be argued about separately.*

Notice that a position in the graph need not be a member of the choices of some issue. These positions are currently uncontested, or simply "not at issue".

Definition 8 (Well-Formedness of Dialectical Graphs) *A dialectical graph is well-formed if and only if:*

1. *Each antecedent and consequent of the arguments in* `A` *is some position in* `P`*;*
2. *It is acyclic;*
3. *No position is the antecedent of more than one argument;*
4. *At most one of the choice positions of an issue is the antecedent of an argument in* `A`.
5. *All of the positions used in the preference expressions of the constraints* `K` *of an issue are antecedents of arguments pro or contra the choices* `C` *of the same issue.*

The first requirement assures every argument links two positions in the graph.

The next four well-formedness conditions, taken together, assure that the issues in a dialectical graph form a forest of trees, in which the nodes are positions and the edges encode arguments about positions on these issues. The final condition also assures that the constraints of an issue all directly concern the pros and cons of its own choices, and not those of issues in other parts of the dialectical graph.

There is a somewhat philosophical argument we could make here to justify these well-formedness restrictions. The argument turns on the distinction between a proposition and a position. A proposition is a declarative statement; it is either true or false, independent of its use by a party in some particular discussion. A position, on the other hand, is defined by its role and use in a discussion. A position records a speech act and is embedded in a discussion.

There are many examples of this distinction in the Porsche discussion. The wife asserted that Volvos are safe. This seemingly unqualified statement must be understood in the context of this discussion about which car to buy. There is a common understanding between the husband and wife that they will buy some car. They are concerned only about the relative safety of various cars, not about the principal safety of cars. In a discussion about the relative virtues of various modes of transportation, if would be perfectly acceptable for the wife to claim that cars are not safe. Interpreting these claims as propositions, rather than positions, could easily lead to a contradiction:

```
all(x, if car(x) then not safe(x))
all(x, if Volvo(x) then car(x))
all(x, if Volvo(x) then safe(x))
Volvo(a)
```

This is not an example of defeasible reasoning or reasoning with exceptions. The intended interpretation is not that cars in general are unsafe, except for Volvos. Rather, the point here is that the acceptability of what we are calling positions depends on the problem solving context. Deciding whether to fly, drive or take a train somewhere is a different context from deciding which car to buy. Formalisations of deductive logic typically do not handle this task dependency within the formalisation. This is what we are trying to achieve here, and use a different term, position instead of proposition, to emphasize this distinction and help avoid confusion.

Another example in the Porsche discussion would be the preference claim made by the wife about the relative importance of their children's safety, compared to the prestige of her husband. She would be unlikely to make this particular argument in a discussion about flying with the children to another country or state to take on a new higher paid position. Again, the position must be understood in the context of the current problem, indeed the particular issue of the whole discussion. Seemingly equivalent positions may well have different meanings in different parts of the same discussion.

Definition 9 (Subissues) *Let* `(P,A,I)` *be a well-formed dialectical graph and* `i1` *and* `i2` *be two issues in* `I`. `i1` *is subissue of* `i2`, *denoted* `subissue(i1,i2)`, *if and only if:*

1. *There exists an argument* `pro(p1,p2)` *or* `con(p1,p2)` *in* `A` *such that* `p1` *is a member of the choices of* `i1` *and* `p2` *is a member of the choices of* `i2`, *or*
2. *There exists a position* `p1` *which is a member of the choices of* `i1` *and a member of the constraints of* `i2`.

Example 8 *In the Porsche example, the issue about the safety of Volvos is a subissue of the root issue, about which car to buy. This is a subissue of the first kind. Also in the Porsche example, the issue about whether or not being fast is more important than being safe is also a subissue of the root issue. This is a subissue of the second kind.*

Definition 10 (Issue Tree) *If* `(P,A,I)` *is a well-formed dialectical graph, then the subissue relation partitions the issues of* `I` *into a set of maximally connected issue trees,* {`(I1, subissue)`, `...`, `(In, subissue)`}. *Each issue in* `I` *is a member of the set of nodes of exactly one of these trees,* {`I1, ..., In`}.

Example 9 *The three issues of the Porsche example form an issue tree. The root issue is about which car to buy. The other two issues are subissues of this root issue, and are leaves of the issue tree.*

The definitions above completely specify the structure of dialectical graphs. What remains is to define the labelling function. We begin by defining the interpretation of terms and preference expressions.

Definition 11 (Interpretation of Terms) *Let* `v` *be a mapping from the set of positions,* `P`, *to the domain of integers,* `Z`. *The interpretation of a term, denoted* `I[t]` *for some term* `t`, *is defined as follows:*

1. `I[p] = v(p)`, *if* `p` *is a member of* `P`.
2. `I[-t1] =` $-$ `v(t1)`, *where* $-$ *is the unary negation operation on integers.*
3. `I[t1+t2] = I[t1]` $+$ `I[t2]`, *where* $+$ *is the addition operation on integers.*
4. `I[t1-t2] = I[t1]` $-$ `I[t2]`, *where* $-$ *is the substraction operation on integers.*

Definition 12 (Interpretation of Preference Expressions) *The = and > predicate symbols of preference expressions are defined to mean the equality and (strictly) greater than relations on the domain of integers, as follows:*

1. `t1 = t2` *is true if and only iff* `v(t1) = v(t2)`.
2. `t1 > t2` *is true if only iff* `v(t1) > v(t2)`.

The integer values of positions are not asserted by users in this system. Indeed, there is no syntax supporting this in the definitions for dialectical graphs above. Rather, users only assert *constraints* on the values of positions, using preference expressions. We are interested in qualitative, not quantitative reasoning here. The constraints are used in the labelling function to determine whether a position is acceptable or not, i.e. `in` or `out`.

One might object that this point that allowing the addition and subtraction of positions in preference expressions assumes that positions are independent, which may not always be the case. Although we accept this point, it is also true that two positions may in fact be independent, so we do not want to disallow combining them in preference expressions entirely. When two positions are independent, it should be possible to combine them to defeat some other position, which may be stronger than each of them separately, but not in combination [Hage et al., 1994]. Our *partial* solution to this problem is to allow the participants to make an issue out of this, i.e. to argue and decide for themselves whether or not to accept the combination of positions in a constraint. No additional machinery is required for this, since our framework already supports just this kind of meta-argumentation. This solution is incomplete, since several of the proof standards, discussed next, assume that positions used in arguments are independent. A complete solution would require some way to argue about the choice of proof standard.

Now we are just about ready to define the labelling function for the issues of a dialectical graph. Following the nomenclature of Doyle's "Truth Maintenance System", each choice position of an issue will be labelled `in` or `out` to indicate whether or not it meets the burden of proof standard selected for the issue. A large variety of proof standards are imaginable. For a while we tried to find *the* right standard, which would be universally applicable for all kinds of issues. But, we have come to believe that no standard is suitable for all purposes, drawing support from jurisprudence, with its variety of types of burden of proof for different kinds of proceedings and the work of [Farley and Freeman, 1995] in the field of AI and Law. We define some proof standards here, but do not claim that these exhaust all reasonable possibilities. Also, although the names we borrow from the law for some of these standards are intended to be suggestive and mnemonic, we do not claim that our formal definitions adequately capture the legal meanings of these concepts.

A few auxiliary functions will be useful several times below, to sum up the arguments pro and contra some choice. Precisely, let `(p1, ..., pn)` and `(c1, ..., cn)` be the antecedents labelled `in` of the arguments pro and con some choice `x` in a dialectical graph. Let `sum(pro,x)` denote the sum of these `in` pro antecedents, `(p1 + ... + pn)`, and `sum(con, x)` denote the sum of `in` con antecedents, `(c1 + ... + cn)`. Equations of the form `y = sum(pro,x) - sum(con,x)`, called *choice equations* below, constrain the choice itself, and are just syntactic sugar for a preference expression of the form `y = (p1 + p2 ... + pn) - (c1 + c2 ... + cn)`.

Definition 13 (Proof Standards) *Let* `(P,A,I)` *be a well-formed dialectical graph and* `(C,K)` *be an issue in* `I`.

Scintilla of Evidence. *A choice satisfies this standard if and only if there exists at least one* `in` *position which is the antecedent of a pro argument for it in* `(P,A,I)`.

Preponderance of the Evidence. *A choice* `x` *satisfies this standard if the* `in` *antecedents of the* `pro` *arguments supporting it outweigh the* `in` *antecedents of the* `con` *arguments against it. To be more precise, recall that the constraints* `K` *map positions to a set of preference expressions representing inequalities over these positions. The choice* `x` *satisfies the preponderance of the evidence standard if and only if the union of the preference expressions of the* `in` *positions in* `K` *and the choice equations entails* `sum(pro,x) > sum(con,x)`.

No Better Alternative. *A choice meets this standard if no other alternative currently has the better arguments. Formally, some position* `ci` *in the set of choices* `(c1, ..., ci, ..., cn)` *meets this standard if and only if for each* `cj` *in* `(c1, ..., ci-1, ci+1, ... cn)` *the union of the preference expressions of the* `in` *positions in the constraints* `K`*, the choice equations and* {`ci > cj`} *is consistent.*

Best Choice. *A choice meets this standard if it is currently better than all its alternatives. Formally, some position* `ci` *in the set of choices* `(c1, ..., ci, ..., cn)` *meets this standard if and only if for each* `cj` *in* `(c1, ..., ci-1, ci+1, ... cn)` *the union of the preference expressions of the* `in` *positions of the constraints in* `K` *and the choice equations entails* `ci > cj`.

Beyond a Reasonable Doubt. *A choice satisfies this standard if and only if the antecedent of every* `con` *argument against it is* `out` *and none of the antecedents of the* `pro` *arguments in favor of its alternatives is* `in`.

We leave open the question for future research about how best to decide which proof standard to apply for some issue. An obvious approach would be to allow the users to argue about this, ideally using the same argumentation framework. The problem is to find and justify a base case, so as to avoid an infinite regress.

Although we do not have a theory telling us which proof standard to apply, intuitively it seems clear that more than one such standard could be appropriate in our Porsche example. For the top level issue about which car to buy, the "no better alternative" or "best choice" standards seem to be appropriate, unless of course this is a brainstorming session, for which the less strict "scintilla of evidence" might be preferable. For the safety issue, about whether or not to accept that Volvos are safe cars, the "preponderance of the evidence" test is probably best. For such "yes or no" issues, where there is only one choice under discussion, the "best choice" or "no better alternative" standards are always trivially satisfied and therefore not particularly useful. The "preponderance of the evidence" test also appears appropriate for the final issue of the example, about whether or not speed is more important than safety.

Given a dialectical graph and a mapping from issues to proof standards, the labelling function is defined recursively using the tree structure of the issues in the graph. All positions which are not at issue are labelled `in`. Working up from the leaves of each issue tree, the proof standard of each issue is applied to determine the labels of its choices until we reach the root issue of each tree.

Definition 14 (Dialectical Graph Labelling) *Let* `(P,A,I)` *be a well-formed dialectical graph. The labelling function,* `label(T,p)`*, maps an issue tree* `T` *and a position* `p` *to one of* {`in, out`}*. Let* `(C,K)` *be the root issue of* `T` *and* `Ant` *be the union of the antecedent positions of all the* `pro` *and* `con` *arguments for the choices in* `C`*. Then* `label` *is defined as follows:*

1. *If the position* `p` *is a member of* `Ant` *or* `K` *:*
   (a) `label(T,p) = in`*, if* `p` *is not at issue;*
   (b) `label(T,p) = label(T1,p)`*, where* `T1` *is the issue tree rooted in the issue of which the position* `p` *is a choice, otherwise.*
2. *If the position* `p` *is in the set of* `C` *of choices:*
   (a) `label(T,p) = in` *if* `p` *satisfies the proof standard for its issue.*
   (b) `label(T,p) = out` *otherwise.*
3. *Otherwise,* `label(T,p)` *is undefined. (In this case,* `p` *is not a part of the dialectical graph for the issue tree* `T`*.)*

The careful reader will have noticed that the proof standards and the labelling function for dialectical graphs are mutually recursively defined. Although we have not formally proved that this circularity is harmless, intuitively the recursion ends in the leaf positions of a dialectical graph which are not at issue. Recall also that well-formed dialectical graphs, by definition, do not contain cycles.

Example 10 *The positions of Porsche example are shown labelled in Figure 4. The names of the* `out` *positions are shown underlined. Recall that all positions that are not at issue are* `in`*. Thus all the leaf positions which are not a member of some issue are* `in`*. In the example, these are all of the positions except* `P1`, `P2`, `P6` *and* `P9`*. Turning to the positions at issue, let us assume that the "preponderance of the evidence" standard is used for the two subissues. The wife's position that Volvos are safe (*`P6`*) fails to meet this standard, and is therefore* `out`*, because of the uncontested constraint,* `P11`*, claiming that government statistics are more to be trusted than advertising. Not only is it not entailed that the sum of the arguments in favor of safety are greater than those against, as required by he preponderance standard, worse still it would be inconsistent with the constraint to even presume this to be the case. Using the same kind of reasoning, the husband's position regarding speed being more important than safety,* `P9` *is also* `out`.

This leaves the top level issue, about which car to buy. Although the husband is behind in the speed vs. safety debate, his Porsche choice is still preferred to the Volvo. This is because the wife's position about Volvo's being safe is also currently `out`*, as just shown. The speed argument in favor of Porsche has some (unknown) positive value. This is reduced by some amount by the cost*

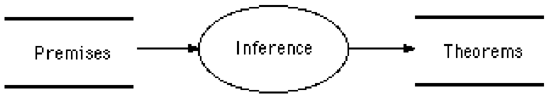

Figure 6: Functional Model of Deduction Process

argument, `P4`. *But however many points are lost because of this cost argument, the Volvo suffers even more, because of the "family car" argument and the husband's (still) uncontested constraint that the family car argument against Volvo is stronger than the cost argument against Porsche. Thus, for the moment, the Porsche meets both the "no better alternative" standard and even the stronger "best choice" standard, whereas the Volvo meets neither.*

This result would be somewhat different if the safety argument were still `in`. *In that case both cars would meet the "no better alternative" standard, but neither would meet the stronger "best choice" standard.*

4 Discussion and Related Work

Formal models of argumentation fall into two main categories: relational (aka "declarative") and dialectical (aka "procedural"). As its name suggests, the relational model views argumentation as a mathematical relation between some representation of information or knowledge (typically in the form of rules of various kinds, facts and cases) and the conclusions which are "warranted" or "justified" by this information.

The specification of a mathematical relation as such implies little about the *process* of reasoning, let alone argumentation. The same relation may be used in various ways: deductively to generate theorems from premises and abductively or inductively to generate sets of premises from goal "theorems". Figure 6 is a functional diagram for the deductive process.

This functional model should be compared with the much richer functional model for the mediation system presented in the "Software Engineering Analysis" section of this paper, Section 2.

Although the implementation of a theorem prover may well contain complex flows of data, this is beside the point here. Our concern is not with implementation details, but with the functional features of the *problem domain*.[1] These are, from a software engineering perspective, *analysis* issues, rather than system design or implementation issues.

Beginning with Dialogue Logic [Lorenz, 1961, Lorenzen, 1969], many authors have cloaked their proof theories in the style of argumentation, with proponents and opponents exchanging arguments to defend or defeat some claim. Whatever the advantages of this approach to proof theory, the resulting models of argumentation remain squarely in the relational camp. The only process modelled is the generation or testing of warranted conclusions from a static representation of undisputed, and within this framework undisputable, premises.

Computational Dialectics is concerned with modelling this larger process of argumentation, as a foundation for the design and implemention of computer systems suitable for supporting argumentation, negotiation and decision-making in groups.

Several abstract "frameworks" for computational dialectics research have been proposed. Our initial proposal [Brewka and Gordon, 1994, Gordon, 1996] consisted of three levels: 1) logic, 2) speech acts and 3) protocols. The logic layer is responsible for formalizing the notions of (necessary) consequence and contradiction. That is, its subject belongs to the field of mathematical logic, as it has become in this century. The speech act layer, which might better be called the "process" layer, defines the notions of state and action, but specialized on processes of disputation. This layer defines the space of possible actions. The protocol layer defines the rights and responsibilities of the participants, i.e. the norms regulating the procedure or "game".

Most dialectical or dialogical models of argumentation have not distinguished between the process and norm layers. Since these are formal games, arguably there is no need for a norm layer: all of the norms can, in principal, be collapsed into the definitions of the possible actions at the process level. But models which fail to make this distinction are not suitable as a foundation for computer systems which guide, or even regulate, human activity. As experience with "work flow" systems in the Computer-Supported Work (CSCW) community has shown, they lead to rigid, brittle systems which do not give their users the freedom they require for behaving effectively in a changing world.[2] This flexibility can only be obtained at the price of *enabling* users to violate the formal norms encoded in the system. Once violations become possible, a model of norms becomes useful, perhaps essential, as a way to specify how rights and responsibilities become modified in the event of violations [Jones and Sergot, 1992].

At the last ICAIL (International Conference on AI and Law), Henry Prakken introduced the notion of an *argumentation framework*, hereafter AF, which adds some useful structure to the process layer [Prakken, 1995]. In Prakken's view, an AF defines four elements:

1. The concept of an *argument*.
2. The notion of *conflict* between arguments.
3. *Preference relationships* between conflicting arguments.
4. The *status* of an argument, such as whether it has "won", "lost", or left the dispute undecided.

Prakken proposed a slightly different three layer model for dialectical systems, consisting of a logic layer, his idea of an argumentation framework, and finally a protocol layer. In this model, the process layer of Gordon's model is moved to within the protocol layer. The idea of a separate argumentation framework layer is attractive, but we would prefer to preserve the distinction between the process and protocol layer, for the reasons mentioned above.

These considerations lead us to propose splitting the process layer into an argumentation framework and action layers, which yields the following four layer abstract model of dialectical systems:

1. logic
2. argumentation framework
3. actions
4. protocol

If we think of argumentation in terms of language games, an AF can be viewed metaphorically as the *game board* or playing field. At any given time during the game, it is the structure which represents the state of play. The action layer defines the operations or moves which are possible on this structure, to change its state. The protocol layer, as before, defines the norms of the game.

To help make these distinctions clearer, consider the game of chess. The argumentation framework is like the playing board. The

[1] In the software engineering sense of "functional".

[2] See the Winograd/Suchman debate in the CSCW Journal [Suchman, 1994, Winograd, 1994] for further information.

action layer would define the possible ways of moving chess pieces on the board. Note that it is *possible* to move a pawn as if it were a queen.[3] The protocol layer would define the rules of the game, disallowing the movement of pawns in this way.

In the Pleadings Game, Gordon used the nonmonotonic logic of Conditional Entailment [Geffner and Pearl, 1992] as an AF. Like several other nonmonotonic logics [Pollock, 1988, Simari and Loui, 1992], the proof theory of Conditional Entailment is in the argumentation style of Lorenzon's Dialogue Logic. This approach suffers from two difficulties: conceptual and computational complexity. The conceptual complexity results from the embedding of an argumentation game for the AF within the larger dialectical game. Although from a formal point of view there is nothing objectionable about this, it is surely too confusing and unwieldly for practical mediation systems. The computational complexity is due to the warrant or entailment relation of these nonmonotonic logics, which are not even semi-decidable. Thus no algorithm can exist, let alone an efficient algorithm, for deciding whether a move is permitted at some stage of dialectical game, using this kind of logic as an argumentation framework.

Perhaps the computational complexity issue can be resolved by restricting the object language, at the cost of some expressiveness, or by restricting the entailment relation, by replacing the underlying monotonic consequence relation within something weaker, as was done in the Pleadings Game with the *known* relation. But these measures do not solve the conceptual complexity problem. On the contrary, they can make the whole system still more complex.

The use of Conditional Entailment as the AF in the Pleadings Game created an additional difficulty. All forms of preference relationships between competing arguments were encoded using specificity. This technique gave specificity higher priority than any other preference criteria, which is not universally correct. A newer law may take precedence over a conflicting more specific but older law, for example.

The Zeno Argumentation Framework was designed to overcome these difficulties. Rather than adapting a nonmonotonic logic for use as an AF in a dialectical model, the Zeno AF was designed from scratch for just this purpose. As the AF has a restricted function in the multi-tiered dialectical model, it can be much simpler, both conceptually and computationally, than a nonmonotonic logic.

Considered in isolation, the Zeno AF may look like a relational model of argumentation, since its labelling function defines a relation between dialectical graphs and `in` positions. But Zeno was designed to meet the requirements for an AF in the multi-layered procedural or dialectical model of argumentation. Proponents of the relational approach aim to completely model argumentation within a relational framework, without recourse to procedural notions at the system analysis level.

One limitation of the Zeno AF compared to AFs based on nonmonotonic logic is that our AF does not model the process of *generating* arguments. We do not consider this price to be very great, because in the application scenarios we are interested in the main problem participants face is not the generation of arguments from a common set of premises, but the generation of arguments from multiple, heterogeneous information sources and models, most of which will be completely informal. To use a formal logic to generate arguments, participants would first have to encode their premises in a common formalism. We do not consider this to be a realistic possibility in our intended application domains.

[3] Some have challenged us on this point, but the daughter of the first author, Caroline, does this all the time. She has to be watched like a hawk. Perhaps these critics play chess with a computer simulation of the game which fails to preserve this natural distinction between what is possible and what is permitted.

5 Conclusion

The Zeno Argumentation Framework is a formal version of Horst Rittel's Issue-Based Information System (IBIS) conceptual model of argumentation. In addition to a syntax for dialectical graphs, it includes a "semantic" labelling function supporting a kind of inference.

The concept of a position is modelled as a primitive, atomic element in this model, similar to the way atomic propositions are primitive, undifferentiated elements of propositional logic. But positions should not be confused with propositions. The intuition behind these concepts is quite different. A proposition is a context-independent declarative statement having a truth value. (That we speak of the logical consequences of a set of premises as being "true" only if all of the premises are "true" supports this point.) A position, on the other hand, records the performance of a speech act in a particular thread of discussion. Its meaning is defined by its *role* in the discussion and is dependent on its location in the argumentation graph. Things about the world which would have to be made explicit to turn the position into a context-independent proposition are left unsaid, because the participants in the discussion make assumptions about shared knowledge and only make explicit points which seem contentious or otherwise relevant for the particular issues being discussed.

For these reasons, we have imposed a strong restriction on the use of positions in arguments: they may only be used once. That is, only a single argumentation link may flow from one position to some other position. We also prohibit cycles. The resulting argumentation graph is thus a forest of trees. To reuse a position in another argument, one can "copy" the position. This makes it possible to argue about the implicit assumptions of the two copies of the position independently. Perhaps assumptions appropriate for one use are inappropriate for the other.

Prakken's concept of an Argumentation Framework requires some representation of the idea of *conflict* between arguments. Rittel's IBIS model includes pro and con arguments, but they have no effect on the status of positions within the model. Zeno extends IBIS with a means to express preferences and compute position labels, overcoming this limitation.

Preference expressions are a particular kind of position, with some internal structure. Arguments about these constraints are supported, in the same way as arguments about any other positions. This may involve arguing about preferences between the positions supporting and opposing the constraint at issue, and so on, to any "level".

Given a set of preference expressions, it becomes possible to make inferences about the relative quality of alternative proposed solutions of an issue. A set of *burden of proof* standards was defined for this purose. Given a dialectical graph and a mapping from issues to proof standards, the main task of the mediation system is to determine which of the positions of each issue satisfy its standard. The system is nonmonotonic. Further argumentation may cause some positions to now satisfy the selected test, and others to no longer make the grade. Several competing positions can satisfy the proof standard for an issue, even though the users of the system may ultimately have to choose one when deciding the issue.

This ability to qualify positions is the most significant advance over the informal IBIS model. It transforms IBIS from a lifeless method to organize and index information into a playing field for stimulating debate. The interested parties can see immediately whether their positions are currently "winning" or "losing", given the arguments which have been made so far, motivating them to marshall still better arguments in favor of their positions.

6 Acknowledgements

This work was partially funded by the European Commission for the GeoMed project, IE 2037 D13. We would like to thank Gerhard Brewka, Eckehard Gross, Ronald Loui, Henry Prakken, and Hans Voss for some interesting and helpful debates on argumentation in general, and on the pros and cons of the Zeno Argumentation Framework in particular.

References

[Brewka and Gordon, 1994] Brewka, G. and Gordon, T. F. (1994). How to buy a porsche, an approach to defeasible decision making. In *Working Notes of the AAAI-94 Workshop on Computational Dialectics*, pages 28–38, Seattle, Washington.

[Doyle, 1979] Doyle, J. (1979). A truth maintenance system. *Artificial Intelligence*, 12:231–272.

[Farley and Freeman, 1995] Farley, A. M. and Freeman, K. (1995). Burden of proof in legal argumentation. In *Proceedings of The Fifth International Conference on Artificial Intelligence and Law*, pages 156–164, Maryland.

[Geffner and Pearl, 1992] Geffner, H. and Pearl, J. (1992). Conditional entailment: Bridging two approaches to default reasoning. *Artificial Intelligence*, 53(2–3):209–244.

[Gordon, 1995] Gordon, T. F. (1995). *The Pleadings Game*. Kluwer.

[Gordon, 1996] Gordon, T. F. (1996). Computational dialectics. In Hoschka, P., editor, *Computers as Assistants – A New Generation of Support Systems*, pages 108–203. Lawrence Erlbaum Associates.

[Hage et al., 1994] Hage, J. C., Leenes, R., and Lodder, A. R. (1994). Hard cases: A procedural approach. *Artificial Intelligence and Law Journal*, 2:113–167.

[Jones and Sergot, 1992] Jones, A. J. I. and Sergot, M. (1992). Deontic logic in the representation of law: Towards a methodology. *Artificial Intelligence and Law*, 1(1):45–64.

[Kunz and Rittel, 1970] Kunz, W. and Rittel, H. W. (1970). Issues as elements of information systems. Technical Report 0131, Universität Stuttgart, Institut für Grundlagen der Planung.

[Lorenz, 1961] Lorenz, K. (1961). *Arithmetik und Logik als Spiele*. PhD thesis, Kiel.

[Lorenzen, 1969] Lorenzen, P. (1969). *Normative Logic and Ethics*. BI-Hochschultaschenbücher.

[Pollock, 1988] Pollock, J. (1988). Defeasible reasoning. *Cognitive Science*, pages 481–518.

[Prakken, 1995] Prakken, H. (1995). From logic to dialectic in legal argument. In *Proceedings of The Fifth International Conference on Artificial Intelligence and Law*, pages 165–174, Maryland.

[Rittel and Webber, 1973] Rittel, H. W. J. and Webber, M. M. (1973). Dilemmas in a general theory of planning. *Policy Sciences*, pages 155–169.

[Rumbaugh, 1991] Rumbaugh, J. (1991). *Object Oriented Modeling and Design*. Prentice-Hall.

[Simari and Loui, 1992] Simari, G. R. and Loui, R. P. (1992). A mathematical treatment of defeasible reasoning and its implementation. *Artificial Intelligence*, 53(2–3):125–157.

[Suchman, 1994] Suchman, L. (1994). Do categories have politics? the language/action perspective reconsidered. *Computer-Supported Cooperative Work*, pages 177–190.

[Winograd, 1994] Winograd, T. (1994). Categories, disciplines, and social coordination. *Computer-Supported Cooperative Work*, 2:191–197.