Precedent and Procedure: an argumentation-theoretic analysis

Adam Zachary Wyner
University of Liverpool
Department of Computer Science
Ashton Building
Liverpool, United Kingdom
azwyner@csc.liv.ac.uk

Trevor Bench-Capon
University of Liverpool
Department of Computer Science
Ashton Building
Liverpool, United Kingdom
tbc@csc.liv.ac.uk

ABSTRACT

In theoretical AI, much recent research on arguments treats them as entirely abstract, only related by an attack relation, which always succeeds unless the attacker can itself be defeated. However, this does not seem adequate for legal argumentation. Some proposals have suggested regulating attack relations using preferences or values. However, this does not explain how an audience can prefer or value an argument, yet be constrained by the procedure of debate not to accept it. Nor does it explain how certain types of attack may not be allowed in a particular context. For this reason, evaluation of the status of arguments within a given framework must be allowed to depend not only on the attack relations along with the intrinsic strength of arguments, but also on the nature of the attacks and the context in which they are made. In this paper we present a formal, functional decomposition style, description of arguments articulated into their component parts and contexts which allows us to represent and reason with types of attacks with respect to context. This machinery allows us to account for a number of factors currently considered to be beyond the remit of formal argumentation frameworks.

Keywords

argumentation, procedure, precedent

1. INTRODUCTION

In current AI research much of the theoretical work relating to argumentation is based on Dung [5], where arguments are entirely abstract, only related by an attack relation, which always succeeds unless the attacker can itself be defeated. This may work for mathematics and classical logic, but it seems inadequate for legal argumentation. Some people have enriched the structure of the theory such that attack succeed or fail depending on properties of the arguments involved as in preference-based (Amgoud and Cayrol [1]) or value-based (Bench-Capon [3]) analyses. This, however, does not explain several aspects of legal argumentation. In some contexts, while a court may be sympathetic to an argument, the court cannot accept it because the legal procedure prohibits it, as for example where a lower court is forced to follow the decision of a higher court whatever their sympathies. In other contexts, a type of attack is prohibited: hearsay evidence, persuasive in common debate, may be excluded in a legal proceeding. In addition, in many cases, the appeals process allows distinct courts in a legal hierarchy to decide a case differently, though the case is comprised of the same facts and legal issues. Given these problems, evaluation of the status of arguments within a given framework must be allowed to depend not only on the attack relations, nor only on these together with the intrinsic strength of arguments, but also on nature of attacks and the relative contexts of their component arguments. In this paper we present a formal, functional decomposition style, description of arguments which allows us to include the type of attacks and the context in which they are presented. With this, we can determine the status of arguments given a particular legal procedure. This machinery allows us to account for a number of aspects currently considered to be beyond the remit of formal argumentation frameworks.

This paper addresses the question of how can distinct legal decisions arise from the same facts and legal issues. We address this question with a formal argumentation-theoretic analysis of *relations of legal precedence between court contexts*. We discuss the formalism with respect to the *English Legal System* and particular examples. We draw concepts for our analysis from the formal argumentation theories of Dung [5], Bench-Capon [3], Gordon and Walton [6], and Atkinson [2], which we refer to and contrast on key points.

In a nutshell, the theory makes the following claims. Arguments are related to different sorts of legal *contexts*. We rank these contexts relative to each other. An argument in relation to one context may survive an attack by an attacking argument from a *weaker context*. Furthermore, we provide articulated arguments; this enables us to distinguish a variety of sorts of attacks, which may be ranked as well. Finally, procedural contexts are comprised of these elements – arguments with structure, legal contexts, and rankings on attacks and contexts.

The structure of the paper moves from examples to formalization of the examples to extensions of the basic theory. We outline judicial precedent and procedural contexts, how a case can be decided differently as it is passed upwards in a

ICAIL '07 Stanford, California USA

specified legal hierarchy. We also discuss how the elements of the legal hierarchy change. We point out some of what this paper does *not* address. The formal presentation has two main parts. In Sections 2 and 3 we focus on the use and change of procedural contexts. We define the basic elements, predicates, relations, and functions with respect to arguments and legal contexts, defining procedural contexts in terms of them. We relate our abstract formalization to the example case. We then extend the analysis in Section 4 along the lines of Gordon and Walton's Carneades framework [6]. In the final sections, we make some comparisons to other approaches to formalizing argumentation, outline future research, and draw some conclusions.

2. PROCEDURAL CONTEXTS

In this section we consider an appeal route in the English Legal System.[1] We describe a range of elements of the appeals process, not all of which we provide analyses of, as discussed below. We consider a sample case.

2.1 Crown Court

For our purposes, the lowest level of the legal hierarchy is the *Crown Court*, where trials on indictment come before a judge and jury. The evidence, legal arguments, and the decision are given according to the procedures specified for the Crown Court. In particular, the Crown Court is *bound* by precedents decided by courts higher in the legal hierarchy. The decisions on points of law made in a Crown Court are not binding on any higher level, nor are they binding on other judges in another Crown Court, though they are *persuasive*. We may refer to a *ratio decidendi* as the legal principle on which the decision is based. We do not here consider *obiter dictum*, which are counterfactual considerations which may have led the decision to be resolved differently.

The difference between *binding* and *persuasive* precedents is important. A binding precedent is a decided case which a given court *must* follow in making a decision on the case before it, though this depends on the similarities between the cases. A persuasive precedent is one which is not binding, but which can be applied should it not conflict with a binding precedent and the court which applies the precedent chooses to do so. For our purposes, we simply assert the status of the precedent. In this paper, we are particularly interested in the interactions between these levels of precedents and levels of courts.

2.2 Court of Appeal

We assume an appeal to a higher level *Court of Appeals*. Cases can be reconsidered on matters of evidence or of law; for matters of law, there is a claim that the law has been misapplied, the rule of law which was applied is no longer desireable, or some application of the law was inappropriately missed. In effect, the *ratio decidendi* of the prior decision is somehow faulty.

Judges do not retry the case, but hear the evidence and arguments. The Court of Appeals can overturn a decision of a Crown Court. While the decisions of a Court of Appeals are binding on Crown Courts, the decisions of a higher court are binding on Courts of Appeals. Moreover, a Court of Appeal is bound by the decision of another Court of Appeal, with a range of exceptions (cf. *Young v Bristol Aeroplane Co Ltd* [1944] KB 718).

2.3 House of Lords

We assume our case is appealed to the highest court – the *House of Lords*. The evidence and arguments are heard again. However, the Law Lords who judge the case are not bound by decisions made at either of the two lower courts. Following *Practice Statement [1966] 3 All ER 77*, the House of Lords is not even obligated to follow its own previous decisions.[2]

2.4 Example 1: Use of a Precedent

The courts follow the principle of *judicial precedent*, where judges follow the decisions of similar cases which have already been decided. Courts also abide by hierarchical relationships, where the inferior courts follow the legal principles and decisions of superior courts. The key point is that the *judicial context of the case under consideration and its relation to contexts of other cases which have been decided play a key role in the application of precedents and the decision of the case.*

Consider an abstract illustration of this point. Suppose there are three arguments, meaning that for each argument, evidence, claims, and a ratio decidendi have been given from which a conclusion follows. We indicate the arguments as a_1, a_2, and a_3. The argument a_1 is under consideration, meaning that no court has yet decided the matter. The arguments a_2 and a_3 have been decided in some court. Furthermore, we suppose that a_2 *attacks* a_1; that means that if we accept that the attack succeeds, then a_1 is defeated, and it is decided that the conclusion does not follow. On the other hand, supposing that a_3 *supports* a_1 (perhaps by attacking a_2), which means that if a_3, rather than the attacking argument a_2 succeeds, then a_1 is not defeated. Whereupon, it is decided that the conclusion holds.

One of the ways to determine which of a_2 or a_3 succeeds is to consider the relative contexts. For example, if the attacking argument a_2 has been decided by the House of Lords, the supporting argument a_3 has been decided by the Court of Appeals, and the argument a_1 is under discussion in the Crown Court, then it is very clear that a_1 should be defeated. The attacking argument a_2 has been decided by the House of Lords. Decisions by the House of Lords are binding on decisions of the Crown Court and trump decisions of the Court of Appeals. On the other hand, if the attacking argument a_2 has been decided by the Crown Court, the supporting argument a_3 has been decided by the Court of Appeals, and the argument a_1 is under discussion in the Court of Appeals, then the Court of Appeals can *choose* which decision to follow. This shows that the success or failure of attacks and supports are determined by the *relative hierarchical relationships between the context where the argument is under discussion and the contexts where the attacking or supporting arguments have been decided.*

We want our analysis to systematically represent these relationships and reasoning patterns in a *fixed* system of procedure.

2.5 Example 2: Change of Use of Precedent

[1]The following materials are based on www.lawteacher.com\english.htm.

[2]The House of Lords is the highest level in the English Legal System. In the European Legal System, above it is the European Court of Justice, which we do not consider.

In *Miliangos v George Frank (Textiles) Ltd [1976] AC 443*, the House of Lords overruled Re *United Railways [1961] AC 1007* and in favor of allowing damages to be awarded in a foreign currency. For 350 years, it had been take as settled in English Law that no English Court could award damages in a foreign currency. However, in 1975, on appeal after claims in the Court of Appeal, the House of Lords overruled the Court of Appeal and its own previous decision in favour of allowing payments in either English currency or a foreign currency equivalent (cf. Morris [8]). The change was related to a change in the English Legal System and new circumstances. Prior to 1966, the House of Lords was bound to follow all its prior decisions under the principle of *stare decisis*. Following the *Practice Statement [1966] 3 All ER 77*, the House of Lords granted itself the right to depart from its previous decisions where it seems right to do so. In addition, the new decision was tied to changes in the foreign exchange system in 1961; in particular, by 1976, exhange rates were floating rather than fixed, which resulted in the instability of English currency.

In the following, we provide a partial formalization of *change* of judicial procedural context.

2.6 Distinctions

Our topic is distinct from other work on *precedents*. In contrast to work on *case-based reasoning* such as Rissland and Ashley [10], in this paper, we are not concerned with the identification of precedents and their application in a particular case. Rather, we are interested in the *determination of the precedent in the first place*. Moreover, we are interested in what happens in the *change in decision* as a particular dispute passes from inferior to superior courts, not *what triggers the appeal*, which we presuppose. Nor do we attend to a range of variations, subtlies, and subissues of the English Legal System; our presentation above is deliberately simplified.

We focus on what we take to be fundamental issues: How do model the application of judicial precedence and hierarchy? How can we model a judicial procedure? How can we model changes in the use of precedence and procedure? Finally, the language we develop is formal and abstract, though we touch on examples; it is not designed to express some *particular* legal system, but instead is intended to be expressive enough to allow different legal systems to be expressed in it.

In the following, formally model aspects of the examples and judicial procedures we have discussed.

3. A FRAGMENT OF A LANGUAGE OF ARGUMENTS

The following language is a fragment, which illustrates some of the key elements of the analysis with respect to precedent and procedural notions outlined above. Some elements are given descriptively, then further specified in a subsequent section. In this way, we can more simply motivate the analysis. As an *illustration*, we propose specific definitions out of the logical space of possible definitions. As we point out, some of these variations are useful or interesting to our analysis. Other alternatives may find other purposes, but for brevity, we do not discuss them here.

3.1 Arguments and Relations

We assume atomic, abstract *argument* entities which we associate with a *type* `argument`.

***Definition** 1.* We assume a set of argument names a_1 ,..., a_n, which denote arguments a'_1 ,..., a'_n. Argument names are of type `argument`.

These arguments may be associated with the abstract arguments of Dung [5]. We can existentially quantify over them, predicate of them, and apply relations and functions to them.

We assume *contexts*, which essentially are just indices. We subdivide contexts into *legal* and *non-legal* contexts. Furthermore, we subdivide the legal contexts into *legal contexts of consideration* and *legal contexts of decision*.

***Definition** 2.* We assume a set of context names c_1 ,..., c_n, which denote contexts c'_1 ,..., c'_n. The predicate *context* denotes the set of contexts. Context names are of type `context`.

***Definition** 3.* We assume a set of legal contexts lc_1 ,..., lc_n and non-legal contexts nlc_1 ,..., nlc_n, and related predicates, which partition the contexts: $\forall x\,[context(x) \rightarrow [legalContext(x) \vee nonLegalContext(x)]] \wedge \neg\exists x\,[legalContext(x) \wedge nonLegalContext(x)]$

The *legal contexts of decision* are the contexts in which, given the evidence and the *ratio decidendi*, the court *has decided* that the conclusion of the argument holds. To streamline the presentation, we presume the evidence, ratio decidendi, and conclusion; we return to formalize these later. The *legal contexts of consideration* are those in which a decision about an argument has not been made.[3]

***Definition** 4.* We assume a set of legal contexts of consideration lcc_1 ,..., lcc_n and legal contexts of decision lcd_1 ,..., lcd_n, and related predicates, which partition the legal contexts: $\forall x\,[legalContext(x) \rightarrow [legalContextConsiderFun(x) \vee legalContextDecision(x)]] \wedge \neg\exists x\,[legalContextConsiderFun(x) \wedge legalContextDecision(x)]$

We can have variables of any of these types by subscripting the variable to the type, e.g. argument variables are x_{a_1} ,..., x_{a_n} of type `argument`.

Arguments are in relations to contexts.

***Definition** 5.* Suppose a relation *argumentContext* type `argument` $\times$ `context` $\rightarrow$ `boolean`.
$\square\ \forall x_{a_m}\ \exists y_{c_l}[argumentContextRel(x_{a_m}, y_{c_l})]$.

While arguments can be stated in a range of non-legal contexts or legal contexts of consideration, we assume that, for legal purpose, an argument is only *decided* in one context: we provide a function for this.

[3]Conceptually, the distinction between contexts of decision and contexts of consideration is related to Dynamic Logic (cf. Harel, et. al. [7]). In Dynamic Logic, the assignment of values to variables function may be partial and change systematically over time by updating. Analogously, the legal contexts where an argument is decided correlate to variables with a value, and where an argument is under consideration (i.e. not decided) correlates to variables with no determinate value. Making a legal decision on an argument correlates with updating the assignment function. We note this here, but leave it for future work.

***Definition** 6.* Suppose a function *argContextDecision*(a) = c of type `argument` → `context`, where *legalContextDecision*(c).

Furthermore, while arguments may be considered in many courts, we want to determine which court it is being considered in for a particular example.

***Definition** 7.* Suppose a function `argContextConsiderFun`(a) = c of type `argument` → `context`, where *legalContextConsider*(c).

Next, we consider how the legal contexts are ranked.

3.2 Ranking the Contexts

We are particularly interested to model the court hierarchy as well as the effect of *binding* and *persuasive* decisions on a legal argument. To do so, we assume a partial order relation on legal contexts of decisions.

***Definition** 8.* Suppose a relation partial order relation $\geq_{LegalCont}$ on the set of entites of type `legal context` such that $\geq_{LegalCont}(\mathrm{lc}_m, \mathrm{lc}_n)$ is true if and only if the legal context lc_m is superior or equal to lc_n in the hierarchy of courts.

Where need be, we can also have $>_{LegalCont}$ and $=_{LegalCont}$ as *strictly superior* and *equal to* legal contexts. The courts can be partitioned into *equivalence classes*, which are those courts that are equal with respect to this relation. For example, all the courts which are Crown Courts are equal to one another, all courts which are Courts of Appeal are strictly greater than all Crown Courts, but equal among other Courts of Appeal, and the House of Lords is strictly superior to all other courts, but equal to itself. We have one restriction, namely, that the non-legal contexts are strictly the lowest on the hierarchy.

***Definition** 9.* $\Box\ \forall \mathrm{x}_{lc}\ \forall \mathrm{x}_{nlc}\ [>_{LegalCont}(\mathrm{x}_{lc}, \mathrm{nlc})]$

We consider relations among arguments.

3.3 Attack/Support Relations and Defeat

Following Dung [5], arguments are in attack relations; that is, one argument can attack another argument. In Dungian analyses, the only way an argument survives an attack is if the attacker is itself attacked (and similarly for the survival of the attacker itself). Thus, one argument defeats another if the attacking argument is not itself successfully attacked. The general notion of a *supporting* argument in Dungian frameworks is that an argument A supports an argument B, if A is an member of an admissible set containing B, which would become inadmissible if A were removed. Later, we shall formalize the claim that arguments can be attributed a substructure, which then allows a range of articulated relations between arguments and so extends the system. However, while for the current discussion we need a range of attack relations, the particulars are not crucial at the moment, so we assume just that arguments are in relation to one another and informally describe the relationships. We will informally specify which argument in these relationships wins or loses.[4]

[4]In a general sense, our proposal is compatible with the *Argument Interchange Format* of Chesnevar et. al. [4].

***Definition** 10.* Suppose a_m and a_n are arguments, then `genArgRel`(a_m, a_n) is a boolean type and the relation is of type `argument` × `argument` → `boolean`.

This relation simply says that two arguments are in relation, leaving unspecified what that relation is. For example, we can interprete `genArgRel`(a_1, a_2) to mean that a_1 *attacks* a_2, say by denying the conclusion and one of the premises; we can interprete `genArgRel`(a_3, a_4) to mean that a_3 *supports* a_4, say by affirming the conclusion and the reasoning relation that draws the conclusion of a_3. It is not crucial to justify and elaborate these matters at present, for they are articulated in detail later. For clarity, let us create some subclasses of the argument relation, where $\texttt{genArgRel}_{attack}(\mathrm{a}_m, \mathrm{a}_n)$ is attacking and $\texttt{genArgRel}_{support}(\mathrm{a}_m, \mathrm{a}_n)$ is supporting.

3.4 Formal Use of a Precedent

We want to formalize the notions of precedence discussed in Section 2.4. In particular, to do so, we must *contextualize* these relations and rate their *effectiveness* in virtue of their relative contexts; in effect, different relations between arguments *have different forces depending on the contexts of the arguments*. More fundamentally, *properties (e.g. contexts) of the arguments determine properties of the attack relations*. This allows us a formal analysis of the example in Section 2.4. Suppose that there are three arguments, where a_1 is the argument under discussion. $\texttt{genArgRel}_{attack}(\mathrm{a}_2, \mathrm{a}_1)$ and $\texttt{genArgRel}_{support}(\mathrm{a}_3, \mathrm{a}_1)$. We see that a_1 is in two argument relations. How do we determine whether a_1 is defeated by a_2 or supported by a_3 (assuming that an argument can only be either defeated or supported)? That we can already distinguish between these two sorts of argument relations already makes our analysis distinct from the Dungian analyses. One way to determine the status of a_1 relative to a_2 and a_3 is to *rank* the relations themselves, say ranking the attack over the support. However, this does not represent the precedence relations in relative to the legal hierarchy discussed in Section 2.4.

To define this generally, we want to specify *how* an argument relation *succeeds*. For this, we suppose variables of argument relations: $\mathbf{A}_1, \ldots, \mathbf{A}_n$ are of the type in Definition 10. We define *binding* and *persuasive* relations in terms of the arguments relative to the court hierarchy, which we discuss further below.

***Definition** 11.* *bindingRel*($\mathbf{A}_m(\mathrm{a}_r, \mathrm{a}_t)$) if and only if $\mathrm{a}_r \neq \mathrm{a}_t \wedge \mathbf{A}_m(\mathrm{a}_r, \mathrm{a}_t) \wedge$ *argContextDecision*(x_{a_r}) > *argContextConsider*(x_{a_t})

***Definition** 12.* *persuasiveRel*($\mathbf{A}_m(\mathrm{a}_r, \mathrm{a}_t)$) if and only if $\mathrm{a}_r \neq \mathrm{a}_t \wedge \mathbf{A}_m(\mathrm{a}_r, \mathrm{a}_t) \wedge$ *argContextDecision*(x_{a_r}) ≤ *argContextConsider*(x_{a_t})

For a sample definition of success, we have several instances to consider. If one relation is a *binding* and the other is not, then the *binding* relation succeeds; if there is no binding relation, and there is a persuasive relation, then it succeeds; finally, if two relations are binding, then the argument with the higher legal context in the hierarchy succeeds. We consider the significance of alternatives below.

***Definition** 13.* *successArgRel*($\mathbf{A}_m(\mathrm{a}_r, \mathrm{a}_t)$, $\mathbf{A}_n(\mathrm{a}_s, \mathrm{a}_t)$) if and only if $\mathrm{a}_r \neq \mathrm{a}_s \neq \mathrm{a}_t$
$\wedge$ [[bindingRel($\mathbf{A}_m(\mathrm{a}_r, \mathrm{a}_t)$) $\wedge$ [$\neg$bindingRel($\mathbf{A}_m(\mathrm{a}_s, \mathrm{a}_t)$)]

$\vee$ [$\neg$bindingRel($\mathbf{A}_m$(a_r, a_t)) $\wedge$ persuasiveRel($\mathbf{A}_m$(a_r, a_t))]
$\vee$ [bindingRel($\mathbf{A}_m$(a_r, a_t)) $\wedge$ bindingRel($\mathbf{A}_m$(a_s, a_t)) $\wedge$
argContextDecision(a_r) > argContextDecision(a_s)]],
which we read as $\mathbf{A}_m$(a_r, a_t) succeeds over $\mathbf{A}_n$(a_r, a_t).

Let us assume that if both argument relations are persuasive, then one has a choice over which is successful.

To provide an example along the lines of Section 2.4, we suppose arguments a_1, a_2, a_3, where a_1 is the argument under consideration in lcc_1, a_2 and a_3 have been decided in lcd_2 and lcd_3, respectively. We also assume the ranking: $lcd_2 > lcd_3 > lcc_1$. Finally, we have $\texttt{genArgRel}_{attack}$($a_2$, a_1) and $\texttt{genArgRel}_{support}$($a_3$, a_1). Given the specifications above, $\texttt{genArgRel}_{attack}$($a_2$, a_1) succeeds over $\texttt{genArgRel}_{support}$($a_3$, a_1) since while both are binding, a_2 has been decided at a higher court level than a_3. Since we have interpreted $\texttt{genArgRel}_{attack}$($a_2$, a_1) to mean that a_2 implies the conclusion of a_1 is false, we can infer that a_1 does not hold in lcc_1 (though exactly how this is determined is provided in a later section). Unlike Dungian analyses, we have *another way* to infer what arguments win or lose by making the arguments relative to contexts and ranking contexts. Suppose we *change* the context of consideration of a_1 from lcc_1 to lcc_3, assuming that lcc_3 and $lc2_1$ are in an equivalence class of legal contexts (i.e. the same level of courts) and that we do not otherwise alter the contexts of decision of the other two arguments. We then can recalculate the winning argument. Neither argument relation is binding because the context of decision of a_2 and a_3 is not strictly higher than the context of consideration of a_1. By assumption, there is a choice about which argument relation is successful, so we may choose $\texttt{genArgRel}_{support}$($a_3$, a_1), which means that a_1 is supported and not defeated.

The definitions in (11) - (13) are only some of the possible ways to define the *bindingRel* and *persuasiveRel* relations between arguments and *successArgRel* on these relations. We have provided them to illustrate the issues clearly and simply for the moment. In particular, an argument is *binding* on another if it has been *decided* by a strictly higher court, and it is *persuasive* if it has been *decided* by a court at the same level or lower. Clearly this need not be so in a given Legal System. For instance, as discussed in Section 2.5, before the *Practice Statement*, the House of Lords was *bound* to follow any decision which it decide, but clearly, the House of Lords cannot outrank itself. Furthermore, it could be that decisions reached at one level of the court system are *binding* on other courts of the same level. Our notion of persuasive arguments could be refined with some further property that characterizes some arguments as persuasive and others not. In particular, we can use elements a *Value-based Argumentation Framework* (Bench-Capon [3]), where an argument is marked as *persuasive* if it is subjectively acceptable under a given value ordering. The force of an attack or support might then depend on such valuations on arguments. The advantage of the analysis at this point is that it can easily be modified by changing the ranking, further specification of properties of the contexts, or further specification of properties of the arguments.

We can tie our analysis into the Dungian analysis in that we use our *successArgRel* relation to identify *sets* of arguments, which may be in attack or support relations. We can then define the *admissible* sets of arguments as *preferred extensions*, *stable extensions*, and related Dungian argumentation-theoretic notions. Value-based Argumentation (cf. Bench-Capon [3]) allows us to remove attacks based on properties of the arguments relative to particular audiences: the theory proposed here allows attacks to be removed based on properties of the attacks themselves relative to the context. This complements Value-based Argumentation Frameworks by providing an additional degree of freedom when evaluating argumentation frameworks which enables us to capture additional phenomena, such as the effect of context as considered in this paper.

3.5 Procedural Change

We have, at this point, provided a formal mechanism to represent and reason with arguments in legal hierarchies. The last general point provides a suggestion of how to account for the case discussed in Section 2.5, where a previously decided case was overturned by a novel argument and a change in the House of Lords. For our purposes, the latter is key, for the House of Lords was no longer bound to decisions it had made on arguments. We can formalize the notion of binding relation on argument relations with respect to the House of Lords *before* the *Practice Statement*, where we assume "The House of Lords" is a legal context.

***Definition** 14. bindingHOLRel*($\mathbf{A}_m$(a_r, a_t)) if and only if [$a_r \neq a_t \wedge \mathbf{A}_m$($a_r$, a_t)] $\wedge$ [*argContextDecision*(x_{a_r}) > *argContextConsider*(x_{a_t}) $\vee$ *argContextDecision*(x_{a_r}) = "The House of Lords"].

Simply put, the effect of the *Practice Statement* was to drop the last disjunct and to have definition (11) instead, in which case since the House of Lords cannot outrank itself, there can be no binding decisions, but only persuasive decisions, in which case the House of Lords can choose which argument relation holds. We can define

procedural change, in part, as a function from one definition of binding relations to another.

4. EXTENSION OF ANALYSIS

We have discussed elements of an analysis which accounts for the use and change of precedent, some of which have been presented descriptively. In this section, we formalize these elements and extend the language to provide further sorts of articulated argument relations.

We assume that not only is an argument related to a context, but it has a mereological structure. While it may be debatable exactly what those parts are and how they are related, we will follow Gordon and Walton's Carneades framework [6], assuming an argument has *premises*, *presuppositions*, *exceptions*, which we refer to as the *assumptions* of the argument. From the assumptions, we draw a *conclusion* of the argument, which is based on the *reasoning relation* of the argument.[5] Our choice of the Carneades framework is for the purposes of illustration, concreteness, and comparison. However, our language is very expressive, so it allows us to define a range alternative analyses.

Because we can make reference to the mereological structure of an argument, we can provide a range of articulated notions of argument relations, varieties of attack and support, as well as a spectrum of procedural contexts, where we specify what sorts of relations hold between arguments and rank the criteria for winning per context.

[5]Gordon and Walton [6] refer to the *rule* of the argument. We discuss later why we prefer the somewhat awkward term *reasoning relation* instead.

4.1 Extended Definitions

We assume the definitions of Section 3, to which we add the following definitions, where we relate arguments to the component statements and reasoning relations.

***Definition** 15.* We assume set of statement names s_1 ,..., s_n, which denote atomic propositions. Statement names are of type `statement`. If s is a statement, then ¬s is a statement. In no model can s and ¬s both hold in any context.

Where one statement ¬s is the negation of the another statement s, we will say the statements are contrary. For our purposes, the only complex statement is the negation of a statement; we have no complex statements comprised of conjoint, disjoined, or conditional statements.

***Definition** 16.* If s is a statement and a is an argument, then *premise*(a,s) is a well-formed relation on arguments. It is read as *the statement s is a premise of argument a*. The premise relation is of type `argument` × `statement` → `boolean`.

We have similar forms of definitions for *presupposition*(a, s) and *exception*(a, s).

This is illustrated in Example 1 Premises are taken as true and not defeasible. Presuppositions are presumed background, but are defeasible. Exceptions can defeat presuppositions. If the exception does hold, then the presupposition is not defeated and we get Conclusion 1; if the exception does not hold (i.e. there is a secret passage), then the presupposition is defeated and we get Conclusion 2.

***Example** 1.* **Argument Elements**
Premise: John was seen to enter the house at 2pm.
Premise: John was seen to leave the house at 3pm.
Presupposition: John could not leave the house unobserved.
Exception: There is no secret passage out of the house.
Conclusion 1: Therefore, John was in the house at 2:30pm.
Conclusion 2: Therefore, John might not have been in the house at 2:30pm.

The relations *premise*, *presupposition*, and *exception* are many to many; that is, an argument may have many statements which are premises, or presuppositions, or exceptions; a statement which is a premise of one argument may also be a premise of another argument. Furthermore, the relations per argument are disjoint.

***Definition** 17.* □¬∃a ∃s [[premise(a,s) ∧ presupposition(a,s)] ∨ [premise(a,s) ∧ exception(a,s)] ∨ [presupposition(a,s) ∧ exception(a,s)]]

For convenient reference, we gather statements in the above relations together. If a statement is in either the premise, presupposition, or exception relation to an argument, it is an *assumption* of the argument[6]. If a is an argument, assumptionFun(a) is the set of all statements in either premise, presupposition, or exeption relations to that argument.

***Definition** 18.* assumptionFun(a) $=_{def}$ λx [premise(a, x) ∨ presupposition(a, x) ∨ exception(a, x)]. This is a set of statements, so of type `statement` → `boolean`.

[6]This is in contrast to Gordon and Walton and Walton, who use the term *premise* in two different ways, which can be confusing

***Definition** 19.* Nothing else is in the premise, presupposition, or exception relation to an argument.

Mereologically, an argument has assumptions (of several *sorts*), a conclusion, and something which systematically maps assumptions to a conclusion. This relation may be called *implication*, *inference*, Toulmin's [11] *warrant*, or *rule*. However, each of these may carry with it interpretations or background information which we do not intend.[7] Moreover, we are interested in a general notion which may be specified in a variety of ways. In order to avoid confusion and to avoid premature commitment to any particular approach, we refer to this relation as the *reasoning relation*.

***Definition** 20.* We assume a set of reasoning relation names r_1,..., r_n, which are of type `reasoning relation`. We discuss the semantics below.

***Definition** 21.* If a is an argument and r is a reasoning relation name, then reasongRelation(a, r) is a well-formed relation on arguments and reasoning relations. It is read as *the reasoning relation r is a reasoning relation of argument a*. The reasoning relation is of type `reasoning relation` × `argument` → `boolean`.

This is a many-to-one relation, as many arguments may have the same reasoning relation, but (by stipulation) no argument has many reasoning relations.

Given a set of assumptions and a reasoning relation, we define a conclusion of an argument in general. A conclusion is a statement which is *functionally* determined with respect to the assumptions and reasoning relation of the argument.

***Definition** 22.* If a is an argument name, A is a set of assumptions, r a reasoning relation name, s a statement, then conclusion(a, assumptionFun(a), r, s) is a well-formed relation. The function conclusion is of type `argument` × (`statement` → `boolean`) × `reasoning relation` × `statement` → `boolean`. It identifies the statements s which follow from assumptions A and reasoning relation r in argument a.

As a relation, an argument can have several statements which are conclusions; for example, any assumption can also be claimed to be a conclusion.

We must say something about the semantics of a reasoning relation such that the conclusion follows from assumptions in an argument. In Gordon and Walton [6], an argument is given a value *pro* or *con*. However, for clarity and simplicity, we use a truth-functional notion of the reasoning relation relation between an assumptions and a conclusion in an argument, which is but one of the possible definitions of the reasoning relation.

Suppose a reasoning relation r_1, where the specification of the relation is given in terms of the relation of the assumptions and conclusion. In definition (23), we say that a statement s_1 is the conclusion of an argument a_1, relative to a set of assumptions given by assumptionFun(a_1) and relative to a reasoning relation r_1. Each of the assumptions and

[7]For instance, do we mean *rule* as in expert systems, Prolog programs, or the 'rules' of logic such as modus ponens. We have also not used the term *scheme*, which can be understood along the lines of the schemes in Walton [12], where we have schemes such as *argument from sign*, *argument from hearsay*, *argument from expert testimony*, or *argument from witness testimony*.

the statement which is the conclusion must all be true for one to assert that a given statement s is a conclusion.[8]

***Definition** 23.* conclusion(a_n, assumptionFun(a_n), r_1, s_m) = 1 if and only if $\forall x$ [x $\in$ assumptionFun(a_n)], otherwise 0.

In this case, a rather strict condition on the conclusion relation is applied. There are a range of conceivable and useful alternatives, but for the purposes of this paper, we use this one.

With all the parts, we have the form of a sample argument which has premise s_1, presupposition s_2, and exception s_3 such that applying reasoning relation r_1 to the assumptions we conclude s_4.

***Example Spec.** 1.* $\exists$a [premise(a, s_1) $\wedge$ presupposition(a, s_2) $\wedge$ exception(a, s_3) $\wedge$
conclusion(a, assumptionFun(a), r_1, s_4)]

Our earlier Example 1 correlates with this, where the first two premises hold, the presupposition, and the conclusion.

4.2 Relations Among Arguments

We assume that two arguments a_1 and a_2 are identical when they have the same assumptions, conclusions, and reasoning relations. Furthermore, one argument a_1 is a subargument of another argument a_2 if the conclusions and reasoning relations of a_1 are the same as a_2, but the assumptions of a_1 is a proper subset of a_2. In addition, given two arguments with the same assumptions and reasoning relation, the same conclusion must follow.

To define these notions, we define two functions – one to get the reasoning relation from the argument *reasonRelFun* and another to get the conclusion from the argument *concludeFun*.

***Definition** 24.* reasonRelFun(a) = r if and only if reasonRel(a, r). The function reasonRelFun is of type `argument` $\rightarrow$ `reasoning relation`.

***Definition** 25.* concludeFun(a) = s if and only if conclusion(a, assumptionFun(a), r, s). The function concludeFun is of type `argument` $\rightarrow$ `statement`.

Given that *reasonRelFun* and *concludeFun* are *functions*, an argument can only have one reasoning relation and one conclusion.

To gather all the statements of a particular argument, we use both the assumption and conclusion functions.

***Definition** 26.* statementsFun(a) = λx [x $\in$ assumptionFun(a) $\vee$ x = concludeFun(a)]

With the reasoning relation and conclusion functions, we can specify argument identity, subargumenthood, and that from any two arguments with the same assumptions and reasoning relations, the same conclusion must hold. Any relation which is a relation between arguments has the following type `argument` $\times$ `argument` $\rightarrow$ `boolean`.

***Definition** 27.* argumentIdentity(a_1, a_2) if and only if $\Box$ $\forall a_1$ $\forall a_2$[[assumptionFun(a_1) = assumptionFun(a_2)] $\wedge$ [reasongRelationFun(a_1) = reasonRelFun(a_2)] $\wedge$ concludeFun(a_1) = concludeFun(a_2)] $\rightarrow$ a_1 = a_2]

[8]This definition is for illustration purposes, and it may be too strong. However, arguments only offer *presumptive justification* for the assertion of their conclusions.

***Definition** 28.* subargument(a_1,a_2), argument a_1 is a subargument of a_2, if and only if $\forall$x [x $\in$ assumptionFun(a_1) $\rightarrow$ x $\in$ assumptionFun(a_2)] $\wedge$ [reasonRelFun(a_1) = reasonRelFun(a_2)] $\wedge$ $\forall$x [x = concludeFun(a_1) $\rightarrow$ x = concludeFun(a_2)]

This definition stipulates that subarguments can be *monotonically* extended to the arguments of which they are a part because conclusions from a subargument must also hold of the superargument.

***Definition** 29.* conclusionIdentity(a_1, a_2) if and only if $\forall a_1$ $\forall a_2$[[assumptionFun(a_1) = assumptionFun(a_2) $\wedge$ reasonRelFun(a_1) = reasonRelFun(a_2)] $\rightarrow$ [ConcludeFun(a_1) = ConcludeFun(a_2)]]

An argument a_1 is *distinct* from an argument a_2 if a_1 is neither identical to nor a subargument of a_2. Argument individuals are, therefore, distinct in virtue of distinct parts. However, two distinct arguments may be the same in either their assumptions, or their reasoning relations, or their conclusions so long as the constraints above and below are satisfied.

***Definition** 30.* distinctArguments(a_1, a_2) if and only if [$\neg$argumentIdentity(a_1, a_2) $\wedge$ $\neg$subargument(a_1, a_2) $\wedge$ $\neg$subargument(a_2, a_1)]

Two distinct arguments may be the same in either their assumptions, or their reasoning relations, or their conclusions so long as the constraints above and below are satisfied. This gives us a *typology* of arguments. For brevity, we indicate this with *attacking* arguments, where we suppose these to be arguments with *contrary conclusions*. For later purposes, we give the denotation of the types of arguments by λ-abstracting over the arguments.

***Definition** 31.* **Argument Typology**

- Type1 $=_{def}$ λa_1 λa_2 [distinctArguments(a_1, a_2) $\wedge$ assumptionFun(a_1) $\neq$ assumptionFun(a_2) $\wedge$ reasonRelFun(a_1) = reasonRelFun(a_2) $\wedge$ concludeFun(a_1) = $\neg$ concludeFun(a_2)]
- Type2 $=_{def}$ λa_1 λa_2 [distinctArguments(a_1, a_2) $\wedge$ assumptionFun(a_1) $\neq$ assumptionFun(a_2) $\wedge$ reasonRelFun(a_1) $\neq$ reasonRelFun(a_2) $\wedge$ concludeFun(a_1) = $\neg$ concludeFun(a_2)]
- Type3 $=_{def}$ λa_1 λa_2 [distinctArguments(a_1, a_2) $\wedge$ assumptionFun(a_1) = assumptionFun(a_2) $\wedge$ reasonRelFun(a_1) $\neq$ reasonRelFun(a_2) $\wedge$ concludeFun(a_1) = $\neg$ concludeFun(a_2)]

We say that the *supporting arguments* TypeA-TypeC are the same, respectively, as Type1-Type3, except that the conclusions are equal; they are supporting arguments as they are different ways to argue for the same conclusion. Similarly, we say that *additional arguments* TypeD-TypeF are the same, respectively, as Type1-Type3, except that the conclusions are not equal; they are just other arguments we might find. These arguments might be *irrelevant* to one another in terms of a particular argument, but relevant in a larger network network of arguments.[9] The argument relations in Definition 31 correspond to an intuitive notion of

[9]Dung [5] has no analytic means to differentiate *relevant* from *irrelevant* attacks.

attack, where a successful attack implies that both arguments *cannot* both hold in one context.

***Definition** 32.* argumentAttack(a_1, a_2) if and only if distinctArguments(a_1, a_2) $\wedge$ concludeFun(a_1) = $\neg$ (concludeFun(a_2). We read this as *a_1 attacks a_2*.

Any argument relation that suits this definition correlates to the more familiar *rebuttal* attack. Notice that in this definition of argument attack, the relation is *symmetrical*; each argument denies the conclusion of the other. It makes *explicit* what may be otherwise *implicit*, namely, that any attack of one argument on another is at least an *attempt* to rebut.

The *argumentAttack* relation is the most general sort of attack; it just means that one argument attacks another in the most fundamental way, which is just the denial of the conclusion. We can make subsorts of attacks *keyed to the subproperties* of arguments. A similar point can be made for supporting arguments. For our purposes at the moment, these subproperties are: *premise*, *presupposition*, *exemption*, and *reasoning relation*. Clearly, we could define other sorts of sorts of attacks given other subproperties of arguments.[10] In effect, the subsort of attack expresses *why the conclusion is denied*. We have chosen

to present the notions analytically, first giving attacks on reasoning relations and assumptions, which are not incompatible attacks. Then, we provide subspecies of attacks on assumptions. We have also chosen to *atomize* the attacks where possible; that is, we prefer *not* to have attacks which are *both* attacks on an assumption and an attack on a reasoning relation. If one wants to attack both an assumption and a reasoning relation of one argument, one can, but with *different* attacks by two different attacking arguments.

An *attack on the reasoning relation* means that the two arguments attack one another and differ in terms of the reasoning relation. This correlates to the more familiar *undercutting* attack.

***Definition** 33.* reasonRelAttack(a_1, a_2) if and only if [distinctArguments(a_1, a_2) $\wedge$ reasonRelFun(a_1) $\neq$ reasonRelFun(a_2)

We can say here that the difference in the conclusion is *attributed to the differences in the reasoning relations that are applied*. In other words, the *reason* why we do not accept the conclusion is because we do not accept the reasoning relation which led to the conclusion.

An *attack on an assumption* means that the two arguments attack one another and that one argument contains a statement which is contrary to an assumption of the other argument. This correlates to the more familiar *premise defeat*. Notice that the contrary statement of the attacker could be either one of the assumptions or the conclusion. We break this into two definitions, one where an assumption of the attacking argument is the reason for the contrary conclusions, and another where the conclusion of the attacking argument on an assumption is the reason. We do not need to specify that conclusions attack conclusions since we have already specified that the attack relation only holds among such arguments.

[10] See Atkinson [2] for a detailed specification of the subproperties and the varieties of attack applicable to arguments following a particular scheme.

***Definition** 34.* assumptionAssumptionAttack(a_1, a_2) if and only if [distinctArguments(a_1, a_2) $\wedge$ $\neg$ reasonRelAttack(a_1, a_2) $\wedge$ $\exists$x [x $\in$ assumptionFun(a_1) $\wedge$ $\neg$x $\in$ assumptionFun(a_2)]]

This form of attack is *reciprocal* since each argument has an assumption which is the negation of an assumption in the other.

***Definition** 35.* conclusionAssumptionAttack(a_1, a_2) if and only if [distinctArguments(a_1, a_2) $\wedge$ $\neg$ reasonRelAttack(a_1, a_2) $\wedge$ $\exists$x [x = concludeFun(a_1) $\wedge$ $\neg$x $\in$ assumptionFun(a_2)]]

This form of attack is *asymettric* since it stipulates that the *conclusion* of an argument a_1 attacks an *assumption* of another argument a_2; however, it need not be the case that the conclusion of a_2 attacks an assumption of a_1.

We can have a spectrum of attack relations on parts or combinations of parts. The different relations may have different logical properties (symmetric, asymetric). Here we just give an example definition for an attack on a premise.

***Definition** 36.* premiseAttack(a_1, a_2) if and only if [assumptionAttack(a_1, a_2) $\wedge$ $\exists$x [x $\in$ statementFun(a_1) $\wedge$ premise(a_2, $\neg$x)]]

The definitions for presupposition and exception attack have similar forms. These three relations are asymetrical since we leave *underspecified* the property of the statement of the attacking argument.

To this point, we have just specified articulated attack relations between arguments. It is useful to introduce such relations explicitly because then we can define relations between attack relations. For example, we can rank one attack as stronger than another attack. This is conceptually similar to the valuation scheme for argumentation of Bench-Capon [3], though we extend the idea to the attack relations themselves to supplement valuations on arguments. Given such relations on attacks and criteria for winning and argument, we can define procedures.

Suppose we have a ranking by a *stronger than* relation over attack relations. We call these *ranking relations*.

***Example Spec.** 2.* strongerThanAttack (reasonRelAttack(a_1, a_2), premiseAttack(a_1, a_2)). We read this as *a reasoning relation attack is stronger than a premise attack*.

Assuming the *strongerThanAttack* relation is transitive, we could define the relative strength of all the attack relations. For example, we could specify the following relative strengths, where $>$ means the attack on the left is stronger than the attack on the right. We could have attacks which are equally ranked as well using $\geq$. We give an *example specification* of a strict ordering.

***Example Spec.** 3.* reasonRelAttack(a_1, a_2) $>$ premiseAttack(a_1, a_2) $>$ presuppositionAttack(a_1, a_2) $>$ exceptionAttack(a_1, a_2)

We call such orderings a Ranking Scheme (RS).

In the next section, we first define criteria for winning attacks only with respect to attack relations, then we add the criteria.

4.3 Winning Attacks

Having specified distinct attack relations relative to subproperties of arguments, we can specify winning attacks relative to attack relations. These specifications are then associated with specifications of procedural contexts. In Dung [5], where there is only one sort of attack, an attack is always successful unless the attacking argument is itself attacked. This measure of success stems from the assumption that there is but one sort of attack, for it is unclear what other measure of success could be introduced. The disadvantage of such an approach is that it conflates the notion of attack with winning. In contrast, where there are several sorts of attack relations, we make the success relative to the particular type of attack in the given context. This distinguishes the attack from the measure of success. For example, if a premise attack is stronger than an exception attack, then an argument might survive an exception attack, but not a premise attack. We can further differentiate contexts by the measure of success of attacks: in one context, the measure of success is such that an argument survives and exception attack but not a premise attack, while in another context, it would not survive either.

We can have a family of defeat relations. Closest to Dung's [5] attack, we have a relation where any attack on an argument defeats it so long as the attacking argument is not itself attacked. In Dung [5], attack and acceptability are with respect to *sets of arguments*, which we could define here, but do not for simplicity.

***Definition** 37.* defeatsGeneral(a_1, a_2) if and only if [argumentAttack(a_1, a_2) $\wedge$ $\neg\exists a_3$ [argumentAttack(a_3, a_2)]]
We read this as a_1 *generally defeats* a_2.

We also can express the property that an argument is defeated.

***Definition** 38.* defeatedGeneral(a_2) if and only if $\exists a_1$ [defeatsGeneral(a_1, a_2)]
We read this as a_2 is generally defeated.

Close to Dung's [5] acceptability of arguments, we have survival in general of an argument, which means there is no argument that defeats it.

***Definition** 39.* survivesDefeatGeneral(a_1) if and only if $\neg$[defeatedGeneral(a_1)]

In contrast to Dung [5], we can have attacks that are keyed to specific modes of attack – whether to the reasoning relation, premise, presupposition, or exception as well as to any combination. We could also specify defeat relative to supporting arguments; that is, an attacking argument defeats another argument not only if the attacking argument is not itself attacked, but also (or instead?) if the argument under attack does not have a *supporting* argument. Indeed, there are a myriad of possible definitions of defeat and corresponding survival properties, each defined by combinations of attack and support relations. We give just a small sample of the possible definitions.

***Definition** 40.* defeatsReasonRel(a_1, a_2) if and only if [reasonRelAttack(a_1, a_2)] $\wedge$ $\neg\exists a_3$ [argumentAttack(a_3, a_2)]].
We read this as a_1 *defeats* a_2 *by reasoning relation.*

***Definition** 41.* survivesReasonRelAttack(a_1) if and only if $\neg\exists a_2$ [defeatsReasonRelationGeneral(a_2, a_1)]]

If we allow second-order variables A_1 ,..., A_n over attack relations, then we could then specify that an argument survives any attack until a certain point, and otherwise it is defeated, as in the following example specification.

***Example Spec.** 4.* survivesUpToPremise(a_1) if and only if $\neg\exists A_1$ [A_1(a_1, a_2) $\wedge$ A_1(a_1, a_2) > premiseAttack(a_1, a_2)], otherwise defeated(a_1).

In this case, a reasonRelAttack would defeat the argument, while a premise attack would not. The clause *otherwise defeated* essentially allows us to suppose that any other attack defeats the argument; if the ranking were more extensive, then other attacks might also fail to defeat the argument. We could make different assumptions here, namely, that the argument survives *any attack other than one which is higher in ranking.* Notice that unlike Dung [5], where an attack on an attacker eliminates the attack, we have other means to render the attacker harmless.

More generally, we can defeat or survival relative to ranking schemes RS as defined along the lines of Definition exampleSpec:survivalRuleAttack01. This is an abstract version of the reasoning relation given in Example Specification 4.

***Definition** 42.* surviveRS(a_1, a_2, RS) if and only if $\neg$ $\exists$A $\exists$B [A(a_1, a_2) $\wedge$ RS(A(a_1, a_2), B(a_1, a_2))], otherwise defeated a_1, where RS is a ranking scheme relation, and A and B are either attacking or supporting relations.

This definition and previous definitions give schematics which can be articulated in more detail for particular applications. For our purposes here, we take the reasoning relations which determine defeat with ranking as in Definition (42) to be the `surviveCriteria`; all such definitions define a *surviveCriteriaSet.* However, for simplicity, we assume we only have *attacking* relations, since we have not fully discussed *support* relations.

4.4 Procedural Contexts

We can use the previous definitions to define procedural contexts, which determine *what sorts of attacks can be made and what is success relative to contexts.* As the context changes overall or with respect to the court of consideration, so too changes the decision. However, for brevity, we do not formally present the specification of procedural context and context change, as we have presented the key elements already. A Procedural context is comprised of a set of contexts, which are ranked. We have a set of arguments which are associated with the contexts (e.g. contexts in which the argument is decided or considered). We have a set of attack (and support) relations between the arguments; these can be keyed to highly specific properties of the arguments. We can rank the argument relations as well as determine which of two arguments wins given the context. We have seen that as we change contexts of consideration, we

change the outcome of the argument network. The same point can be made for changing the procedural context.

5. COMPARISONS

Dungian argumentation theory is abstract in several respects. The arguments are atomic elements in the theory, there is only one attack relation between arguments, and the success of an attack is relative solely to whether or not it is attacked. This means that attacks cannot be ranked, the

reason why one argument attacks another argument cannot be expressed, and the burden of proof cannot be assigned to different participants. Similarly, given a set of arguments that hold together, it cannot be said whether the arguments coherently support a particular position since we do not know what the arguments are about. Our approach begins to address these limitations.

Bench-Capon [3] does not articulate arguments, nor make use of context. Atkinson [2], although articulating a particular type of argument and providing a range of types of attack in order to generate an argumentation framework, does not differentiate between attacks on the basis of their origin when evaluating arguments with the framework. Like VAFs, our account gives a way of distinguishing successful from unsuccessful attacks: when combined with values, it provides an extra dimension on which distinction can be made. We also adapt ideas from Gordon and Walton [6] for articulated arguments. However, they do not follow a Dungian analysis of attack relations, nor do they account for procedural contexts.

Unlike research on precedent (cf. Rissland and Ashley [10]), we assume that we have identified a suitable precedent to use in the argument relations rather than defining criteria to identify it.

This work is broadly related to the Argument Interchange Format of Chesnevar et. al. [4]. The clearest similarities are that we use abstract arguments and specify arguments in relations. However, they do not attempt an analysis of precedence or procedure. Our proposal is more specific about the definitions and differs in other ways we do not have the space here to discuss.

6. FUTURE WORK

The formal tools presented in the paper can be applied to a range of other issues of interest to artificial intelligence and law. Not only may it help to clarify important issues, as for example it has with respect to the Dungian framework, but it would also bring together disparate problems into one formal language. For example, the articulation of arguments can be used as the language for Case-Based reasoning along the lines of Rissland and Ashley [10]; two cases are compared in terms of argument structures, argument relations, and winning strategies. By the same token, we can represent Burden of Proof (Prakken and Sartor [9]) on the various attack relations using a property of arguments to represent its proponents and opponents. We also believe that Standards of Proof could be represented in the language. The analysis proposed here can complement value-based argumentation (Bench-Capon [3]). If we introduce additional relations between arguments and statements, we can also account for many of the issues raised by critical questions (Atkinson [2]), where we regard an attack as the *question*, the survival or defeat of an argument as an *answer* to the question which is mediated by the rankings of the procedural context.

7. CONCLUSIONS

The paper provides a formal analysis which can be used to accommodate judicial procedures and precedent within an argumentation framework based approach. The fundamental idea is to define the procedures in each context with different ways to make and win a debate relative to the legal hierarchy. We develop an articulated argumentation theory, where arguments are basic entities. We specify relations between context, statements of various sorts, reasoning relations, and arguments. Given such specific attacks, we can define a spectrum of criteria for winning an attack relative to a context. We define

procedural contexts in terms of these elements such that as the procedural context changes, so changes the decision.

8. ACKNOWLEDGMENTS

During the writing of this paper, the authors gratefully acknowledge the support of the Estrella Project, which is a European AI and Law project comprised of academics, companies, and public administrators. Errors rest with the authors.

9. REFERENCES

[1] L. Amgoud and C. Cayrol. On the acceptability of arguments in preference-based argumentation. In *Proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 1–7, San Francisco, CA, 1998. Morgan Kaufmann.

[2] K. Atkinson. *What Should We Do?: Computational Representation of Persuasive Argument in Practical Reasoning*. PhD thesis, Department of Computer Science, University of Liverpool, Liverpool, United Kingdom, 2005.

[3] T. J. M. Bench-Capon. Persuasion in practical argument using value-based argumentation frameworks. *J. Log. Comput.*, 13(3):429–448, 2003.

[4] C. Chesnevar, J. McGinnis, S. Modgil, I. Rahwan, C. Reed, G. Simari, M. South, G. Vreeswijk, , and S. Willmott. Towards an argument interchange format. *The Knowledge Engineering Review*, 21(4):293–316, 2006.

[5] P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2):321–358, 1995.

[6] T. Gordon and D. Walton. The carneades argumentation framework: Using presumptions and exceptions to model critical questions. In P. E. Dunne and T. Bench-Capon, editors, *Computational Models of Argument: Proceedings of COMMA 2006*, pages 195–207, Amsterdam, 2006. IOS Press.

[7] D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. The MIT Press, 2000.

[8] J. H. C. Morris. English judgments in foreign currency: A "procedural" revolution. *Law and Contemporary Problems, Vol. 41, No. 2*, 41(2):44–53, 1977.

[9] H. Prakken and G. Sartor. Presumptions and burdens of proof. In T. van Engers, editor, *Legal Knowledge and Information Systems. JURIX 2006: The Nineteenth Annual Conference*, pages 21–30, Amsterdam, 2006. IOS Press.

[10] E. L. Rissland and K. D. Ashley. A note on dimensions and factors. *Artif. Intell. Law*, 10(1-3):65–77, 2002.

[11] S. Toulmin. *The Uses of Argument*. Cambridge University Press, 1958.

[12] D. Walton. *Argumentation Schemes for Presumptive Reasoning*. Erlbaum, Mahwah, N.J., 1996.