# The Pleadings Game

Formalizing Procedural Justice

Thomas F. Gordon*

German National Research Center for Computer Science (GMD)
Institute for Applied Information Technology (FIT)
Artificial Intelligence Research Division
5205 Sankt Augustin 1 / Germany
email: thomas.gordon@gmd.de

## Abstract

The Pleadings Game is a normative formalization and computational model of civil pleading, founded in Robert Alexy's discourse theory of legal argumentation. The consequences of arguments and counterarguments are modelled using Geffner and Pearl's nonmonotonic logic, *conditional entailment*. Discourse is focussed using the concepts of issue and relevance. Conflicts between arguments can be resolved by arguing about the validity and priority of rules, at any level. The computational model is fully implemented and has been tested using examples from Article Nine of the Uniform Commercial Code.

## 1 Introduction

The purpose of pleading is to identify the *issues* to be decided by the court. My model of pleading is more akin to common law practice than to the "modern" law of civil procedure in the United States. At common law, the goal of pleading was to reduce the issues to be tried to a minimum. In the modern law of civil procedure, the parties do not explicitly make legal arguments during pleading, but merely assert or deny "essential" facts which are believed to entitle them to legal relief, such as monetary compensation for damages, or are believed to constitute a defense.

*Many thanks to Wolfgang Bibel, and to my colleagues Joachim Hertzberg, Gehard Brewka and Franco di Primio. Thanks also to Ronald Loui and Trevor Bench-Capon for referring me to important related work.

The model deviates from the modern law of civil procedure as my goal is a normative model of pleading, founded on first principles. The model is inspired by Robert Alexy's discourse theory of legal argumentation [1], which explains how judicial discretion can be restricted without resorting to mechanical jurisprudence or conceptualism. I show how some constraints on procedural justice can be monitored or checked using purely formal methods.

Alexy admits that one purpose of having explicitly formulated a set of discourse norms for legal argumentation was to "reveal their shortcomings the more plainly" [1, pg. 17]. In trying to formalize some of these norms, some of these shortcomings became apparent. Although there is no room to adequately justify them, here are my versions of the norms to be formalized: 1) No party may contradict himself. 2) A party who concedes that a rule is valid must accept its application to every set of objects which satisfy its antecedent. 3) A claim may be supported by an argument only if the claim has been denied and is still an issue. 4) A party may deny a claim only if it is not a necessary consequence of his own claims. 5) A supporting argument may be rebutted by a counterargument which is at least as strong, if the claim supported is still an issue. 6) A rebuttal may be defeated by a stronger counterargument, if the claim supported by the rebutted argument is still an issue.

To facilitate an intuitive understanding of the formalization, consider the following hypothetical exchange of allegations, loosely based on Article Nine of the Uniform Commercial Code, which covers secured transactions. No familiarity with commercial law should be required to appreciate this example.

The plaintiff, Smith, and the defendant, Jones, have both loaned money to Miller for the purchase of an oil tanker, which is the collateral for both loans. Miller has defaulted on both loans, and the practical question is which of the two lenders will first be paid from the proceeds of the sale of the ship. These facts are uncontested. One subsidiary issue is whether Smith *perfected* his security interest in the ship or not. (Roughly, the interest is perfected when sufficient steps have been taken to make it effective.) This is where we enter the pleadings.

> **Plaintiff.** My security interested in Miller's ship was perfected.
>
> **Defendant.** I do not agree.
>
> **Plaintiff.** A security interest in goods may be perfected by taking possession of the collateral (UCC § 9-305). I have possession of Miller's ship.
>
> **Defendant.** What makes you think ships are goods for the purposes of Article 9? Also, prove you have possession.
>
> **Plaintiff.** Except for money and instruments, movable things are goods, according to UCC § 9-105-h.
>
> **Defendant.** Although a ship is surely movable, I do not agree that this is sufficient for being a good according to the UCC. Furthermore, according to the Ship Mortgage Act, a security interest in a ship may only be perfected by filing a financing statement.
>
> **Plaintiff.** I have filed a financing statement. But I do not agree that this is required by the Ship Mortgage Act. Moreover, even if you are right, the UCC would take precedence, as it is newer than the Ship Mortgage Act.
>
> **Defendant.** But the Ship Mortgage Act is Federal Law, which takes precedence over state law such as the UCC, even if the state law was enacted later.

At the end of this exchange several issues have been identified. The parties disagree about whether or not Smith has possession of the ship, and whether he has filed a financing statement. These are factual issues. They also disagree about whether ships are goods in the sense of Article Nine, and whether the Ship Mortgage Act requires filing to perfect a security interest in a ship. These are legal issues. There is also the issue about whether the Ship Mortgage Act has priority over the UCC. The plaintiff argued that it does not, using the principle of *Lex Posterior*, which gives the newer rule priority. The defendant responded with the principle of *Lex Superior*, which gives the rule supported by the higher authority priority. Finally, there may be an issue about which of these two principles has priority.

It is tempting to stratify these issues into three levels. The legal and factual issues would be at the object level. The principles for resolving conflicts at the object level, such as *Lex Superior*, would be at the meta level. And the rules for ordering these principles would be at the meta-meta level. A simpler approach is taken in my model: all rules are first-order objects. Conflicts are resolved by partially ordering rule instances. Levels can be simulated, if desired, by giving all rules at some level priority over all lower level rules. An advantage of this approach is that one is not limited to an a priori number of levels.

The rest of this paper is organized as follows. The next section explains how rules are objectified and defeasible reasoning is handled, using *conditional entailment*. Sections 3 and 4 describe the formal system for the Pleadings Game and its implementation, respectively. Section 5 discusses related work. Section 6 concludes and suggests subjects for future research.

## 2 A Language for Explicit Exceptions

Conditional entailment is a relatively young nonmonotonic logic, developed by Geffner and Pearl [10], with a nice model-theoretic semantics, based on Shoham's framework [23], which has been proven to satisfy a number of useful properties.

A *default theory* is a pair $\langle K, E\rangle$, where $K$, the *background context*, is itself a pair $\langle L, D\rangle$. $E$ and $L$ are sets of closed formulas from some first-order language, representing the case-specific *evidence* and the nondefeasible generic knowledge of the domain, respectively. $D$ is a set of defaults. A default is a $\langle p, q\rangle$ pair, which I will denote `(<=` $q$ $p$`)`, where $p$ and $q$ are formulas which may contain free variables. Let us call $p$ and $q$ the *antecedent* and *consequent* of the default, respectively. A *default instance* is created by systematically replacing free variables by closed terms. Let $\mathcal{D}$ be the set of all instances of $D$. The *assumptions* of a default theory are the set of consequents of all default instances in $\mathcal{D}$.

Conditional entailment is defined relative to some monotonic consequence relation, $\models$. An *argument* is a set of assumptions $\Delta$ consistent with $E \cup L$. $\Delta$ is an argument *supporting* a proposition $\phi$ just when $\Delta \cup E \cup L \models \phi$. Two arguments *conflict* if and only if their union is inconsistent with $E \cup L$. Conflicting arguments are ranked as follows. Let $\alpha$ be the antecedent for the default instance of some assumption $\delta$. Then, $\delta$ is necessarily *preferred* to some assumption in an argument $\Delta$ if and only if $\{\alpha, \delta\} \cup L \cup \Delta \models$ `false`. *It is important to notice here that the so-called evidence, $E$, is not included in this preference test.*

Conditional entailment does not dictate how to represent assumptions, or which formulas to include in $L$. Usually, some convenient notation for default rules is designed, from which $D$ and $L$ may be automatically generated. The representation used by Geffner and Pearl does not allow statements within a first-order language to be about default rules, or instances of default rules, as the rule names are represented by predicates. In Article Nine of the UCC, we would like to be able to represent sections such as Section 9-302, which states (in part):

> (1) A financing statement must be filed to perfect all security interests except the following: (a) a security interest in collateral in possession of the secured party under Section 9-305. ...

Here, Section 9-302 refers explicitly to Section 9-305, by name. Also, principals for resolving conflicts between rules, such as *Lex Posterior*, require one to be able to state properties of rules, such as the date of their enactment.

For these reasons, I have adopted another approach, in which Section 9-302, e.g., could be represented by

```
(rule UCC-9-302 (p s g)
  if (not (filed s))
  then (not (perfected s))
  unless (applies (inst UCC-9-305 (parms p s g))))
```

and Section 9-305, which states that a security interest in goods may be perfected by taking possession of the goods, might be represented by

```
(rule UCC-9-305 (p s g)
  if (and (secured-party s p)
          (collateral s g)
          (goods g)
          (possession g p))
  then (perfected s))
```

Notice that this language allows an optional *explicit exception* to be stated in the rule. Rather than defining the mapping into defaults and formulas of $L$, to save space allow me to just show how the rule for Section 9-302, above, would be translated. The sentences of $L$ for this rule are:

```
(all (p s g)
  (if (and (not (filed s))
           (backing UCC-9-302)
           (ap (inst UCC-9-302 (parms p s g)))
  (applies (inst UCC-9-302 (parms p s g)))))

(all (p s g)
  (if (applies (inst UCC-9-302 (parms p s g)))
      (not (perfected s))))

(all (p s g)
  (if (applies (inst ucc-9-305  (parms p s g)))
      (not (ap (inst UCC-9-302 (parms p s g))))))
```

Its default schema is

```
(<= (ap (inst UCC-9-302 (parms p s g)))
    (and (not (filed s))
         (backing UCC-9-302)))
```

The term

```
(inst UCC-9-302 (parms p s g))
```

names an instance of the default, whereas the term **UCC-9-302** names the default itself. The assumption that this default is applicable to some set of objects is represented by an instance of the atomic formula

```
(ap (inst UCC-9-302 (parms p s g)))
```

The **backing** predicate is introduced to facilitate a discussion about whether or not a rule in this language is an valid representation of some legal rule or piece of common sense knowledge.[1] Nothing can be derived from a rule which is not backed and applicable. The **applies** predicate is just a convenient shorthand for the conjunction of the antecedent of the rule, and its backing and applicability conditions.

Conditional entailment uses *specificity* to implicitly order conflicting default rules. For example, a rule for consumer goods will automatically take precedence over a rule for goods in general. However, it is common in statutes for sections to include explicit exceptions as well. To preserve the structure of Article Nine, the importance of which is stressed in [11] and [5], the question arises whether explicit exceptions can be mapped to the syntactic test used to determine specificity in the proof theory of conditional entailment. Conveniently, the answer is yes.

To cancel the applicability of a default instance named $\delta$, when some condition q is satisfied, it is sufficient to add **(if q (not $\delta$))** to $L$. The problem of explicitly ranking default instances is more subtle. To give an assumption $\gamma$ priority over another assumption $\delta$, it is not sufficient to assert **(if $\gamma$ (not $\delta$))**, as this is equivalent to **(if $\delta$ (not $\gamma$))**. The key to understanding how to explicitly order default instances is contained in the syntactic test for determining whether one assumption $\delta$ is preferred to another in some set of assumptions $\Gamma$, described previously. To encode an explicit preference for $\delta$ over another assumption $\gamma$, it is sufficient to add the formula **(if (and p $\gamma$) (not $\delta$))** to the set of nondefeasible sentences $L$, where **p** is the antecedent of the default instance for $\delta$.

[1] Backing is Toulmin's term for the relationship between a rule, which he calls warrants, and its justification, [25]. In the case of legal rules, their backing is usually their authority.

## 3 The Pleadings Game

Conditional entailment has now been explained sufficiently to understand how it is used in my formalization of civil pleading, called the **Pleadings Game**. It is indeed a formal two-player game, comparable in some ways to Lorenzen's Dialogue Logic [9]. There is the formal equivalent of a playing board, precisely defined moves, and criteria for deciding when the game is over and which, if any, party is the winner. Unlike Dialogue Logic, however, which is a proof theory for Intuitionistic Logic, and therefore entirely analytic, the Pleadings Game is for identifying the issues of what Toulmin calls "substantial arguments" [25, pg. 123]. The Pleadings Game may be viewed as a formalization of Toulmin's theory of practical argumentation.

First, some definitions, presented top-down. The playing board of the game is called the record.

**Definition 1 (The Record)** *The* record *is a triple, $\langle b, \pi, \delta \rangle$, where $b$ is the* background *of the game, and $\pi$ and $\delta$ are each a tuple $\langle O, C, D \rangle$ of the* open, conceded *and* denied statements *of the plaintiff and defendant, respectively.*

**Definition 2 (Background)** *A* background *is a triple $\langle \phi, S, R \rangle$, where $\phi$ is a formula, $S$ is a set of formulas and $R$ is a set of rules, in the rule language described in the previous section. $\phi$ is the* main claim, *the claim the plaintiff ultimately would like to prove. $S$ is a set of formulas about which the parties agree. Both $R$ and $S$ may be empty. Together, $S$ and $R$ determine the* background context *$K$ for conditional entailment. $K = \langle L \cup S, D \rangle$, where $L$ and $D$ are generated from the rules in $R$, using the mapping described in the previous section.*

The main claim is also the *ultimate issue* of the case, so long as it is an issue.

The only players of the pleadings game are the *plaintiff* and the *defendant*. The moves available during pleading are assertions of various kinds of statements:

**Definition 3 (Statements)** *There are four kinds of* statements, *defined inductively as follows:*

1. *If* `p` *is a formula, then* `(claim p)` *is a statement.*
2. *If* `A` *is a set of formulas and* `p` *is a formula, then* `(argument A p)` *is a statement.*
3. *If* `A` *and* `C` *are sets of formulas and* `p` *is a formula, then* `(rebuttal A p C)` *is a statement.*
4. *If* `s` *is a statement, then* `(denial s)` *is also a statement.*
5. *Nothing else is a statement*

These statements are not moves of the game. Rather, the moves are assertions about statements.

**Definition 4 (Assertions)** *There are four kinds of* assertions, *defined as follows:*

1. *If* `s` *is a statement, then* `(concede s)` *is an assertion.*
2. *If* `s` *is a statement, then* `(deny s)` *is an assertion.*
3. *If* `s` *is a statement and* `A` *is a set of formulas, then* `(defend s A)` *is an assertion.*
4. *If* `r` *is a rule, then* `(declare r)` *is an assertion.*
5. *Nothing else is an assertion.*

Statements and assertions completely define the type of moves permitted, but we have yet to give the rules of the game prescribing when an assertion of a particular type may be made, and with what effect on the record. Each rule has a precondition. If this precondition is satisfied by the record, then the player *may* choose to apply the rule. Application of a rule modifies the record, according to the *effects* defined for the rule. The rules define when an assertion is *permitted*, not *obligated*. However this does not imply that no assertions are obligatory. As will be described in more detail below, when discussing control and termination, a party is required to answer every *relevant* statement, not yet answered, on each turn. Pleading terminates when no relevant statements remain to be answered.

The preconditions of moves use Geffner and Pearl's relationships of supporting, defeating and protected arguments. To support discussion about facts, rules and priorities, the definitions of these relationships need to be extended. Whereas their versions are restricted to sets of assumptions, they will be defined here for arbitrary sets of formulas.

**Definition 5 (Arguments)** *An* argument *is a set of formulas. An* argument *for some proposition is a pair $\langle \Phi, \delta \rangle$, where $\Phi$ is a set of formulas and $\delta$ is a formula. Given a default theory $\langle K, E \rangle$, where $K = \langle L, D \rangle$, an argument $\Phi$ is a* supporting argument *for $\delta$ if and only if $L \cup E \cup \Phi \models \delta$. The* claims *of an argument are all of its formulas which are not assumptions.*

**Definition 6 (Counterarguments)** *An argument $\Gamma$ is a* counterargument *to another argument $\Phi$ in a default theory $\langle K, E\rangle$ if and only if* $\Gamma \cup \Phi \cup L \cup E \models$ **false**. *$\Gamma$ is a* defeating counterargument *of $\Phi$ if and only if they are counterarguments and every assumption in $\Gamma$ is preferred to some assumption in $\Phi$. $\Gamma$ is* protected from *$\Phi$ if and only if $\Gamma$ contains a subargument which defeats $\Phi$. Finally, a counterargument $\Gamma$ of $\Phi$ is a* rebuttal *if and only if $\Phi$ is not protected from $\Gamma$.*

Now we are ready to discuss the discourse rules of the game. There are nine of them. There is insufficient room to describe them formally. A precondition of all rules is that the statement being responded to be in the set of open statements of the opponent. An effect of most moves is to move the statement from the opponent's open set to either his denied statements or conceded statements. If a response by the opponent to this new assertion is possible, then another effect is to add a statement to the open set of the party making the move.

Notice that at most one response is permitted to each statement. After moving it from the open set, no other response is allowed. This rule imposes a *resource limit* on the parties. Although there may be many supporting arguments, rebuttals or defeaters, each player has the burden of constructing the best possible argument at each move.

In these rules, two concepts will be used for which there is no room to explain in detail. First, recall that conditional entailment is defined relative to some monotonic consequence relation. (See Section 2.) The Pleadings Game uses two such consequence relations, to be denoted known and $\models$. The **known** relation is a finite, and therefore decidable, consequence relation. $\models$ may be any stronger consequence relation, such as full first-order classical logic. The purpose of the weaker consequence relation, **known**, is to lessen the burden of proof, when appropriate, and to commit players to at least some of the consequences of their claims.

Second, the concept of an issue is used to focus pleading. Indeed the main purpose of pleading is to identify issues. To focus pleading, a response is permitted to an open statement only so long as it is *relevant*. A statement is relevant if and only if the claim it is about is at issue.

Previous versions of the known relation and issue concept are described in detail in [12]. Some changes to the definitions there were made necessary by the adoption of conditional entailment and by the realization that abduction of the form used there to compute issues is intractable. Any well-founded definition of issue in the context of pleading should be tractable, as the burden of computing intractable tasks should be fairly distributed among the parties, rather than borne by the mediator. To achieve this, I have developed a weaker form of abduction, which uses the **known** relation. See [13] for details.

Rather than just listing the rules, let's demonstrate them using an actual transcript of a game for the Smith vs. Jones hypothetical. In my comments, the discourse rules are stated in italics. The plaintiff sets up the game by filing a complaint:

```
p: (argument bg
     (set (all (i1 i2)
            (if (and (preferred i1 i2)
                     (ap i2))
                false))))
p: (complaint (perfected s1) bg)
```

The **argument** command defines an argument. Here bg is the background set of formulas assumed to be accepted by both players before the game begins. It contains just one formula, which states it is inconsistent for a rule instance i2 to be applicable if another rule instance i1 is preferred.

The **complaint** command adds the main claim, that s1 is perfected, to the set of open statements of the plaintiff, and declares bg to be part of the background. It's the defendants turn:

```
d: (deny (claim (perfected s1)))
```

The defendant denies the main claim. Let's call the party making a move the "proponent". *A claim* c *may be denied only if it is not* **known** *to be entailed by the claims of the proponent. One effect of this move is to add* (denial (claim c)) *to the open statements of the proponent, to give the opponent an opportunity to make a supporting argument.* There are no further relevant statements to answer, so it's the plaintiff's turn.

```
p: (rule ucc-9-305 (p s g)
     if (and (secured-party s p)
             (collateral s g)
             (goods g)
             (possession g p))
     then  (perfected s))
```

This declares a rule, ucc-9-305, which the plaintiff believes is an adequate representation of UCC § 9-305. The (rule <symbol> ...) form associates the symbol with the rule and then declares it. The rule is intended to state that goods may be perfected by taking possession. *The rule is translated into a set of formulas and a default and added to the background context. The name of the rule may not have been used previously, for some other rule. There are no* *other preconditions, and no response is required or permitted to the mere declaration of rules.* What may be controversial is the claim that such a rule is *backed*, for example by legal authority. Recall that nothing can be derived from a rule which is not backed. A discussion about backing may be held just as for any other claim.

```
p: (argument a1 (apply ucc-9-305 (smith s1 ship1)))
```

Next, the plaintiff defines an argument, `a1`, using a function, `apply`, which constructs an argument by applying a rule to a tuple of terms. This is just a convenient utility.

```
p: (defend (denial (claim (perfected s1))) a1)
```

The plaintiff responds to the defendant's denial of his claim by asserting a supporting argument. *When asserting* `(defend (denial (claim c)) A)`, *the claim* `c` *must be at* `issue`. *No formula in the argument may have been claimed by the opponent, but denied or not yet answered by the proponent. The argument* `A` *may not be* `known` *to be inconsistent with the previous claims of the proponent. Finally, the proponent has the burden of proving that the argument is a supporting argument for* `c`. *One effect of this move is to concede all open claims of the opponent which are* `known` *to be entailed by this argument and the other claims of the proponent. Also, all claims (i.e. non-assumptions) of this argument which are not* `known` *to be entailed by the proponent's previous claims are asserted as new claims, to be answered individually by the opponent, as is the statement* `(argument A c)`.

```
d: (deny (claim (goods ship1)))
d: (concede (claim (collateral s1 ship1)))
d: (deny (claim (possession ship1 smith)))
d: (concede (claim (secured-party s1 smith)))
```

Here, the defendant first simply denies and concedes some of the new claims from the plaintiffs supporting argument, `a1`. *A claim may be conceded only if it is not* `known` *to be inconsistent with the claims of the proponent, because of the discourse norm against self-contradiction. Denials may not be conceded. A party is not permitted to retract claims.*

```
d: (rule sma-1 (s g)  ; ship mortgage act
     if (and (collateral s g)
             (ship g)
             (not (filed s))
             (perfected s))
     then false)
```

A rule representing the (hypothetical) § 1 of the Ship Mortgage Act is declared. It states that it is inconsistent to suppose that a security interest in a ship is perfected if a financing statement has not been filed.

```
d: (argument r1
     (set (ship ship1)
          (collateral s1 ship1)
          (backing sma-1)
          (ap (inst sma-1 (parms s1 ship1)))
          (not (filed s1))))
```

The defendant defines an argument, `r1`, explicitly. The convenient `apply` function could not be used here, as the defendant does not want to concede that `s1` is perfected, which is the plaintiff's main claim. (A `difference` function would have been of assistance here, but hasn't been implemented.)

```
d: (defend (argument a1 (perfected s1)) r1)
```

The defendant rebuts `a1` with `r1`. *To assert* `(defend (argument A c) R)` *the formula* `c` *must be an* `issue`, *no formulas in* `R` *may be unconceded claims of the opponent, and* `R` *must not be* `known` *to be inconsistent with the previous claims of the proponent. The proponent has the burden of proving that* `R` *is a counterargument which is not known to be defeated by* `A`. *If* `R` *is empty,* `A` *itself is shown to be inconsistent. Rebuttals and defeating counterarguments, described below, accept the claims of the argument they counter for the sake of argument, without conceding them. The effects of a rebuttal are similar to those of supporting arguments: All open claims* `known` *to be entailed by the rebuttal are conceded, all claims in the rebuttal which are not* `known` *to be entailed by the previous claims of the proponent are asserted as new claims, and finally, the statement* `(rebuttal A c R)` *is asserted.*

There are no further relevant statements to be answered, so it's the plaintiff's turn again.

```
p: (deny (denial (claim (possession ship1 smith))))
p: (deny (claim (not (filed s1))))
p: (deny (claim (backing sma-1)))
p: (concede (claim (ship ship1)))
p: (rule ucc-9-105-h (x)
     if (movable x)
     then (goods x)
     unless (money x))
p: (argument a2 (apply ucc-9-105-h (ship1)))
p: (defend (denial (claim (goods ship1))) a2)
```

In his last turn, the defendant denied that ships are goods and that the plaintiff has possession. Here, the plaintiff first denies the denial of his possession claim. *Denying a denial, without asserting further arguments, just has the effect of leaving the statement open for trial.* Next, he supports his claim that ships are goods by arguing that movable things are goods, according to UCC § 9-105(h).

```
p: (rule lex-posterior (r1 p1 a1 d1 r2 p2 a2 d2)
     if (and (conflicting (inst r1 p1) (inst r2 p2))
             (authority r1 a1 d1)
             (authority r2 a2 d2)
```

```
        (before d2 d1))
    then (preferred (inst r1 p1) (inst r2 p2))
    unless (applies (inst lex-superior
                          (parms r2 p2 a2 d2
                                 r1 p1 a1 d1))))
p: (argument d1
     (apply lex-posterior
       (ucc-9-305 (parms smith s1 ship1) ca 1972
        sma-1 (parms s1 ship1) us 1960)))
p: (defend (rebuttal a1 (perfected s1) r1) d1)
```

Here, the plaintiff tries to defeat the rebuttal to his argument that a security interest in a ship can be perfected by possession, by arguing that the UCC takes precedence over the Ship Mortgage Act, because it is newer.[2] He admits that *Lex Superior* has priority over *Lex Posterior*, by including it as an exception.

*A* `(defend (rebuttal A c R) D)` *move asserts a defeating counterargument* `D` *to the rebuttal* `R` *of* `A`. *The precondition and effects of this move are analogous to those for rebuttals, except that the proponent must show that* `D` *defeats* `R`. *If* `D` *is equivalent to* `A`, *then* `R` *is not a rebuttal of* `A`, *but this had not been* **known**. *The burden of proving defeat rests on the party in whose interest it is to show defeat. As* `D` *must defeat* `R`, *and not merely be protected from it,* `D` *may not include any assumptions which are not preferred to some assumption in* `R`. *An effect of this move is to assert the stronger argument* `(argument (union A D) c)`, *giving the opponent another opportunity to construct another rebuttal.*

It's the defendant's turn.

```
d: (deny (denial (claim (not (filed s1)))))
d: (deny (denial (claim (backing sma-1))))
d: (deny (claim (backing ucc-9-105-h)))
d: (concede (claim (movable ship1)))
d: (concede (claim (backing lex-posterior)))
d: (concede
     (claim (conflicting
              (inst ucc-9-305 (parms smith s1 ship1))
              (inst sma-1 (parms s1 ship1)))))
d: (concede (claim (authority ucc-9-305 ca 1972)))
d: (concede (claim (authority sma-1 us 1960)))
d: (concede (claim (before 1960 1972)))
```

First, several new claims are denied or conceded, including the plaintiff's backing claims for his representations of UCC § 9-105(h) and *Lex Posterior*. So these potential legal issues have been avoided. Notice that claims *about* UCC § 9-305 and the Ship Mortgage Act have been made and conceded, such that § 9-305 is California law enacted in 1972, without leaving the object-level.

```
d: (concede (argument a2 (goods ship1)))
```

[2]The long parameter lists of the `lex-posterior` and `lex-superior` rules could have been avoided by using an existential quantifier in their antecedents. A variable need be a rule parameter only if it occurs in the conclusion or exception of the rule.

The defendant concedes the argument that ships are goods, because they are movable, although he had initially denied the claim that `ship1` is goods. *An argument may be conceded at any time. The argument conceded is already known to be correct, as this is a precondition of making the argument in the first place. Conceding an argument gives up the opportunity to make a counterargument.*

Conceding this argument does not violate the norm against self-contradiction, because the defendant never claimed that ships are not goods. He only demanded that the plaintiff bear his burden of proving that `ship1` is goods. Denying a claim `p` is not the same as claiming`(not p)`.

```
d: (rule lex-superior (r1 p1 a1 d1 r2 p2 a2 d2)
     if (and (conflicting (inst r1 p1) (inst r2 p2))
             (authority r1 a1 d1)
             (authority r2 a2 d2)
             (higher a1 a2))
      then (preferred (inst r1 p1) (inst r2 p2)))
d: (argument r3
     (apply lex-superior
        (sma-1 (parms s1 ship1) us 1960
         ucc-9-305 (parms smith s1 ship1) ca 1972))))
d: (defend (argument (union a1 d1) (perfected s1)) r3)
```

Here the defendant accepts the plaintiff's invitation to rebut *Lex Posterior* using the *Lex Superior* exception.

It's the plaintiff's turn again.

```
p: (deny (denial (claim (backing ucc-9-105-h))))
p: (concede (claim (backing lex-superior)))
p: (concede
     (claim (conflicting
          (inst sma-1 (parms s1 ship1))
          (inst ucc-9-305 (parms smith s1 ship1)))))
p: (concede (claim (higher us ca)))
p: (concede (rebuttal (union a1 d1)(perfected s1) r3))
```

The plaintiff just denies or concedes the remaining open relevant statements, including the rebuttal `r3`. *Any rebuttal may be conceded, but at the cost of losing the opportunity to assert a defeating counterargument.* This was the last rule of the game to be explained. Each of the nine rules was used at least once.

After this last move, there are no relevant statements left to answer, for either party, so the game is over. There are several auxiliary commands for querying the state of the record. These commands can be executed at any time during the game. For example, the `issues` command lists all claims which are currently issues, and the `show` command lists all formulas of an argument. The symbol `facts` is bound to the set of formulas which are currently accepted by both parties. At the end of this particular game, the issues and facts are:

```
> (issues)
((possession ship1 smith)(perfected s1)(goods ship1)
 (backing ucc-9-105-h)(backing sma-1)(not (filed s1)))

> (show facts)
((secured-party s1 smith)(ship ship1)
 (collateral s1 ship1)(movable ship1)
 (before 1960 1972)(higher us ca)
 (authority ucc-9-305 ca 1972)
 (authority sma-1 us 1960)
 (conflicting
     (inst sma-1 (parms s1 ship1))
     (inst ucc-9-305(parms smith s1 ship1)))
 (conflicting
     (inst ucc-9-305(parms smith s1 ship1))
     (inst sma-1 (parms s1 ship1)))
 (backing lex-superior)(backing ucc-9-305)
 (backing lex-posterior))
```

A few auxiliary rules of the game need to be made explicit . A party must continue making moves as long as there are *relevant* open statements of the opposing party to be answered. (A statement is relevant only if the formula it concerns is an issue.) Irrelevant statements may also be answered, if some move is applicable, which permits irrelevant claims to be conceded or denied. The game is over when no relevant statements remain to be answered. The plaintiff is the "winner" if no issues remain and the main claim is conditionally entailed by the default theory constructed during pleading, where conditional entailment is defined for this purpose using the weaker **known** consequence relation, rather than $\models$. The defendant wins if no issues remain and the main claim is not conditionally entailed, again using the **known** relation. The winner, if there is one, is entitled to a *summary judgment*. If neither party wins, the game ends in a draw, and the parties can look forward to other language games, not modelled here, such as discovery and trial.[3] The example game ended in a draw.

An important property of the game is that neither player can prevent its termination. Each moves adds only a finite number of statements to those to be answered and each statement may be answered at most once. Also, either party can end the game on any turn, by denying or conceding all remaining relevant statements.

[3]Conditional entailment is a "skeptical" nonmonotonic logic; a formula is entailed only if it is true in *all* admissible models. However, when the Pleadings Game ends in a draw, there are arguments both pro and contra the main claim. A "credulous" logic can be simulated by giving the judge discretion to decide the main issue as she pleases, in this case. In [13] I define a trial discourse game, where the discretion of the judge is restricted by the structure of the pleading's record. Together the pleading and trial games define a model of rational decision making which lies between credulous and skeptical nonmonotonic logics.

## 4 The Computational Model

The Pleadings Game has been fully implemented. The underlying logic used is not classical first-order predicate logic, but rather McCarty's Clausal Intuitionistic Logic [19]. McCarty's system extends the definite Horn clause subset of classical logic with a monotonic form of negation, which is needed for rebuttals and defeating counterarguments.

There is also a theorem prover for conditional entailment, based on the implementation described by Geffner and Pearl in [10]. Both implementations use a reason maintenance system to compute minimal sets of inconsistent arguments (called "nogoods"), from which the minimal rebuttals and defeating counterarguments can be generated. However, Geffner and Pearl use an ATMS, with its exponential worst-case complexity. The implementation here uses my Minimal Reason Maintenance System [13], which implements the tractable theory of abduction mentioned above.

The implementation of the Pleadings Game itself is relatively straightforward, given the services provided by the other modules mentioned.

## 5 Related Work

In addition to Lorenzen's system, there have been several other dialogue logics, by Lorenz [15], Rescher [21], and Mackenzie [17]. Like Lorenzen's Logic, Lorenz's and Mackenzie's systems do not support substantial arguments. Lorenz was the first to suggest resource bounds, by restricting the permitted number of responses. Mackenzie's system was the first to constrain moves by previous statements, using a "commitment store". His system was also novel in allowing players to retract claims. Rescher's system was the first to handle defeasible arguments and the first to rank them by specificity.

Several hypertext systems, such as [18] and [22], have used Toulmin diagrams for organizing and browsing arguments. Unlike the Pleadings Game, logical dependencies are not used to constrain or facilitate the development of the argument graphs. These systems also do not distinguish the roles or interests of the parties; thus the idea of regulating argument moves using discourse norms does not appear.

Layman Allen has designed many logical games for several players, such as the Plain Language Game [2]. This is an early example of a resource allocation game which divides up the burden of proof among the players. Moves of game included making claims about what statements could be proved within certain resource limits, asking questions of a neutral judge, and challenging claims of the opponent.

In AI, Trevor Bench-Capon and his colleagues have developed two discourse games, applied to the problem of improving the explanations of expert systems. One is based on Mackenzie's system [6], the other on Toulmin's theory [7]. Ronald Loui and William Chen have designed an argument game using Loui's LMNOP nonmonotonic logic [16].

The well-known AI models of case-based legal argumentation, such as [8], [3] and [24], may be viewed as cognitive models of the reasoning processes of competent attorneys. In contrast, the Pleadings Game is a model of discourse norms.

Finally, at least one other project [14] shares Alexy's position about the significance of discourse theory for legal argumentation.

## 6 Conclusion and Future Work

To my knowledge, the Pleadings Game is the first formal normative model of argumentation in which 1) the concepts of issue and relevance are used to focus the discourse; 2) a tractable inference relation is used to commit players to consequences of their claims; 3) Toulmin's framework is not restricted to propositional claims; 4) the goal of the game is the identification of issues, rather than deciding the main claim; and 5) conflicts between arguments may be resolved by arguing about the validity and priority of rules, at any "level".

The value of AI models of legal reasoning is twofold: They enable a new methodology for legal philosophy and may provide key technology for new kinds of computer applications. The insights gained from AI models can be used in legal education to improve the quality of legal practice, whether or not lawyers ever use computer systems in their daily work. Playing the Pleadings Game in law school may be instructive.

The Pleadings Game demonstrates that a machine can monitor a discussion, helping ensure that discourse norms are not violated. In his book on Procedural Justice [4, pg. 5], Michael Bayles explains John Rawls' distinction between pure, perfect and imperfect procedural justice [20]. The kind of justice which can be achieved by fair discourse norms can only be imperfect. Unlike the idea of a "computer judge", there would seem to be little basis for fundamental opposition to the idea of a *mediation system*. The human judge is retained, as a "player" with a particular role, whose discretion is restricted by the rules of the "game".

There is an important difference between a mediation system and legal expert systems, as they are usually conceived. In expert systems, the knowledge base is intended to be a single, consistent theory of some domain, with which users have little opportunity to disagree. A mediation system supports a discussion about alternative theories.[4] A theory is constructed during the game. Experts still have a role to play. They can prepare formal theories, comparable to traditional treatises, which the players can mold into arguments.

Opportunities for future work in the field of legal discourse games are plentiful. An AI system which plays the Pleadings Game, or supports a person playing the game, would be interesting. Perhaps a unifying *normative* theory of legal argument, making use of both statutes and cases, could be couched in discourse theoretic terms. Then there are other legal language games to attend to, such as discovery, trial, appeal, and arbitration. Arbitration is especially attractive, as its goal is compromise and consensus, rather than a complete win for one party at the expense of the other.

## References

[1] Robert Alexy. *A Theory of Legal Argumentation.* Claredon Press, Oxford, 1989.

[2] Layman E. Allen. The Plain Language Game: Legal Writing Made Clear by Structuring it Well. In *Proceedings of the International Workshop on Formal Methods in Law*, Sankt Augustin, 1982. German National Research Center for Computer Science (GMD).

[3] Kevin D. Ashley. *Modeling Legal Argument: Reasoning with Cases and Hypotheticals.* MIT Press, 1990.

[4] Michael D. Bayles. *Procedural Justice; Allocating to Individuals.* Kluwer Academic Publishers, 1990.

[5] T.J.M. Bench-Capon and F.P. Coenen. Isomorphism and Legal Knowledge Based Systems. *Artificial Intelligence and Law*, 1(1):65–86, 1992.

[6] T.J.M. Bench-Capon, P.E.S. Dunne, and P.H. Leng. Interacting with Knowledge Systems Through Dialogue Games. In *Proceedings of the 11th Annual Conference on Expert Systems and their Applications (vol. 1)*, pages 123–130, Avignon, 1991.

[4]This idea is also expressed in [7].

[7] T.J.M. Bench-Capon, P.E.S. Dunne, and P.H. Leng. A Dialogue Game for Dialectical Interaction with Expert Systems. In J.C. Rault, editor, *Proceedings of AVIGNON-92 (vol. 1)*, Nanterre, 1992.

[8] L. Karl Branting. Representing and Resuing Explanations and Legal Precedents. In *Proceedings of the Second International Conference on Artificial Intelligence and Law*, pages 103–110. ACM, 1989.

[9] Walter Felscher. Dialogues as a Foundation for Intuitionistic Logic. In D. Gabby and F. Günthner, editors, *Handbook of Philosophical Logic; Vol. III: Alternatives in Classical Logic*, pages 341–372. D. Reidel, 1986.

[10] Hector Geffner and Judea Pearl. Conditional Entailment: Bridging Two Approaches to Default Reasoning. *Artificial Intelligence*, 53(2-3):209–244, 1992.

[11] Thomas F. Gordon. The Role of Exceptions in Models of the Law. In R. Traunmüller H. Fiedler, editor, *Formalisierung im Recht und Ansätze juristicher Expertensysteme*, pages 52–59. J. Schweitzer Verlag, Munich, 1986.

[12] Thomas F. Gordon. An Abductive Theory of Legal Issues. *International Journal of Man-Machine Studies*, 35:95–118, July 1991.

[13] Thomas F. Gordon. The Pleadings Game: An Artificial Intelligence Model of Procedural Justice. (to appear), 1993.

[14] J.C. Hage, G.P.J. Span, and A. Lodder. A Dialogical Model of Legal Reasoning. In et. al. C.A. Grütters, editor, *Legal Knowledge Based Systems: Information Technology and Law, JURIX '92*, Lelystad, The Netherlands, 1992. Koninklijke Vermande.

[15] Kuno Lorenz. *Arithmetik und Logik als Spiele*. PhD thesis, Kiel, 1961.

[16] Ronald Loui and William Chen. An Argument Game. Technical Report WUCS-92-47, Department of Computer Science, Washington University, 1992.

[17] J. D. Mackenzie. Question-Begging in Non-Cumulative Systems. *Journal of Philosophical Logic*, 8:159–177, 1979.

[18] Catherine C. Marshall. Representing the Structure of a Legal Argument. In *The Second Internatinal Conference on Artificial Intelligence and Law*, pages 121–127. ACM, June 1989.

[19] L. Thorne McCarty. Clausal Intutitionistic Logic, II. Tableau Proof Procedures. *The Journal of Logic Programming*, 5:93–132, 1988.

[20] John Rawls. *A Theory of Justice.* Harvard University Press, 1971.

[21] Nicholas Rescher. *Dialectics.* State University of New York, Albany, 1977.

[22] Wolfgang Schuler and John B. Smith. Author's Argumentation Assistant (AAA): A Hypertext-Based Authoring Tool for Argumentative Texts. In A. Rizk, N. Streitz, and J. Andre, editors, *Hypertext: Concepts, Systems and Applications.* Cambridge University Press, 1990.

[23] Yoav Shoham. A Semantical Approach to Non-monotonic Logics. In Matthew L. Ginsberg, editor, *Readings in Nonmonotonic Reasoning*, pages 227–250. Morgan Kaufmann, 1987.

[24] David B. Skalak and Edwina L. Rissland. Arguments and Cases: An Inevitable Intertwining. *Aritificial Intelligence and Law*, 1(1):3–45, 1992.

[25] Stephan E. Toulmin. *The Uses of Argument.* Cambridge University Press, 1958.