Artificial Intelligence and Law **2:** 239–292, 1994

The Pleadings Game
An Exercise in Computational Dialectics

THOMAS F. GORDON
German National Research Center for Computer Science (GMD), Institute for Applied Information Technology, Sankt Augustin, Germany, email: thomas. gordon@gmd.de

(Received 26 November 1993; accepted 20 April 1994)

Key words: argumentation, dialectics, issue spotting, mediation systems, nonmonotonic logic, pleading

Abstract. The Pleadings Game is a normative formalization and computational model of civil pleading, founded in Roberty 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 in 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 Pleadings Game is a formal, normative model of a particular type of legal proceeding. Pleading is the first of a series of proceedings which can occur along the way toward the decision of a civil case. Very roughly, but sufficient for our purposes, this series of proceedings can be depicted as in Figure 1. The purpose of pleading is to identify the legal and factual issues of the case. What is the conflict about? The purpose of *discovery* is to gather evidence which may be relevant for deciding the factual issues, such as documents and the written statements of witnesses. The purpose of *trial* is to decide the legal and factual issues. Of course, the evidence gathered during discovery is presented at the trial. Finally, the purpose of *appeal* is to review the decision and procedure of the trail court. There may be several levels of appeal, ending in a review by the Supreme Court.

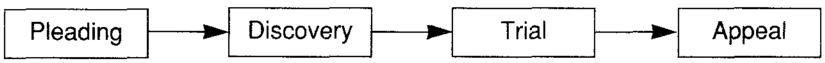


Fig. 1. Series of Civil Proceedings.

One of the main questions of legal philosophy concerns how judicial discretion can be rationally restricted, so as to preserve the balance of power between the legislative and judicial branches of government. My model of pleading is inspired by Robert Alexy's discourse theory of legal argumentation [Alexy 1989], which explains how judicial discretion can be fairly restricted without resorting to mechanical jurisprudence or conceptualism. The Pleadings Game shows how some constraints on procedural justice can be monitored or checked by purely formal methods.

A model of pleading alone would not demonstrate how judicial discretion may be restricted by fair procedural rules, as the judge does not participate in pleading. However, as shown above, pleading is but a part of a series of proceedings, including the trial and appellate proceedings where judges do make decisions. A very basic formal *Trial Game* will also be defined, to demonstrate how a judge's discreton may be sensibly restricted by the issues, and the dependencies between issues, identified during pleading.

The Pleadings Game is a contribution to a new subfield of Artificial Intelligence, which may be called "Computational Dialectics". Its subject matter is the design and implementation of systems which mediate and regulate the flow of messages between agents in distributed systems, so as to facilitate the recognition and achievement of common goals in a rational, effective, and fair way. The design space for systems of this kind is large. The dimensions of discourse games in this space include the purpose of the game, the types of data exchanged in speech acts, the types of speech acts or "moves", the number and roles of the players, the rights and obligations or "commitments" of the players, the kinds of resources and the means of their distribution, and finally the kinds of issues which may be raised and addressed during the game. The Pleadings Game can be characterized along these dimensions as follows:

- The purpose of the game is to identify the legal and factual issues of a case.
- The "data" of the game are defeasible rules and sentences of first-order logic.
- Unlike in most nonmonotonic logics, defeasible rules, nondefeasible rules and "evidence" may be asserted during the game.
- There are four kinds of speech acts, for conceding, denying and defending claims, and for declaring defeasible rules.
- Players are committed to the *known* consequences of their claims. The concept of an *issue* is used to focus the discussion, by prohibiting further arguments about statements which have become irrelevant.
- The only resource is the number of arguments which may be made to support or challenge claims.
- There are two players, the plaintiff and defendant. The burden of proving claims is divided among the players, depending on their role.
- Finally, potential issues include not only substantial claims, but also the validity and priority of defeasible rules. The discourse rules of the game, however, are not subject to dispute during the game.

Even at this abstract level, certain novel features of the game may be apparent. To my knowledge it is the first formal game in which: (1) defeasible reasoning is modeled as a dialogue between two speakers; (2) inference is used to commit players to the known consequences of their claims; (3) the goal of the game is to identify issues, rather than decide them; (4) discourse is focused using the concept of an issue, and (5) the speakers may argue about the validity and priority of defeasible rules.

The Pleadings Game, which is presented fully in this article, is the main result of my dissertation [Gordon 1993]. Of course there is insufficient room here to cover all of the material in the dissertation. In particular, see the full dissertation for a comparative analysis of several standpoints regarding the limits of judicial discretion from legal philosophy, include those of Hart, Rödig and Alexy, a comparison of various formal models of defeasible reasoning and argumentation, beginning with Lorenzen's Dialogue Logic, and a fairly detailed description of an implementation of the Pleadings Game in the Standard ML programming language. The dissertation also explains my choices of Alexy's theory of legal argumentation as the philosophical basis for the Pleadings Game, Geffner and Pearl's logic of conditional entailment as the basis for defeasible reasoning, and Thorne McCarty's Clausal Intuitionistic Logic in the computational model.

The remaining sections of this article are organized as follows: (2) Civil pleading is described in more detail and a legal example in the area of commercial law is presented. (3) A formal language for defeasible rules is defined, and shown to be sufficient and convenient for representing the kinds of relationships between legal rules found in Article Nine of the Uniform Commercial Code (UCC) of the United States. The semantics of this language is given by a mapping into default theories of Geffner and Pearl's logic of *conditional entailment* [Geffner 1992]. (4) The Pleadings Game itself is defined. (5) The Game is demonstrated using the same commercial law example. (6) To show how judicial discretion may be limited by pleading, I present a simple Trial Game. (7) The theory of issues used in these games depends on the structure of the dialectical graph for the main claim. These dialectical graphs are discussed here. (8) The theory of issues is presented. (9) An implementation is briefly described. (10) Some related work is mentioned. (11) Finally, I discuss the possible relevance of the Pleadings Game for legal applications and point out some directions for future research.

2. Civil Pleading

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 minium. There appears to be no limit to the number of pleadings which could be filed by the parties [Black 1979]:

Pleading.... The process performed by the parties to a suit or action, in alternately presenting written statements of their contention, each responsive to what precedes, and each serving to narrow the field of controversy, until there evolves a single point, affirmed on one side and denied on the other, called the "issue", upon which they then go to trial.

The individual allegations of the respective parties to an action at common law, proceeding from them alternately, in the order and under the distinctive names following: The plaintiff's *declaration*, the defendant's *plea*, the plaintiff's *replication*, the defendant's *rejoinder*, the plaintiff's *surrejoinder*, the defendant's *rebutter*, the plaintiff's *surrebutter*; after which they have no distinctive names.

In modern legal systems, typically the rules of civil procedure do not require the parties to explicitly make legal arguments during pleading; rather, they 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. The number of pleadings which may be filed is reduced, in the usual case, to three:

1. The plaintiff begins by filing a *complaint*, in which the facts are asserted which he believes entitles him to legal relief. The complaint also includes a demand for some specific relief.
2. The defendant may then file an *answer*, in which each of the assertions in the complaint is admitted or denied, or a *motion to dismiss for failure to state a claim*, also known as a *demurrer*, in which it is asserted that the law does not entitle the plaintiff to the relief demanded, even if the facts asserted are true. If the defendant files an answer, he may also assert facts which he believes constitute an *affirmative defense*. In our terminology, these are facts which make an exception to some rule applicable.
3. Finally, if the defendant's answer includes an affirmative defense, the plaintiff may file a *reply*, to answer any claims made by the defendant in an affirmative defense, or a demurrer, to assert that the facts claimed in the answer do not constitute a defense.

Notice that this procedure does not account for the possibility that the plaintiff may assert additional facts in his reply which are believed to constitute, in effect, a defense to the defense. There are often exceptions to exceptions in the law, but modern civil procedure terminates pleading, somewhat arbitrarily, after the plaintiff's reply. Unlike common law pleading, the purpose here is not so much to refine and limit the issues to be tried, but only to establish whether or not there is a genuine legal conflict.

The model deviates from the modern law of civil procedure as my goal is a normative model of pleading, founded on first principles, inspired by Robert Alexy's discourse theory of legal argumentation.

Having decided that the basic purpose of pleading should be to identify the factual and legal issues of a case, we can begin to consider which discourse norms would promote this purpose. The norms proposed by Alexy were not designed with the goals of a particular legal proceeding in mind. So we may not adopt them uncritically. Moreover, 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" [Alexy 1989, p. 17]. In trying to formalize some of these norms, some of these shortcomings became apparent. Rather than trying to formalize all of the norms Alexy proposes, I have selected a few to start with, which appear both relevant to pleading and amenable to formalization:

(1.1) No speaker may contradict him or herself.

(1.3) Every speaker who applies a predicate F to an object a must be prepared to apply F to every other object which is like a in all relevant respects.

(2) Every speaker must give reasons for what he or she asserts when asked to do so, unless he or she can cite reasons which justify a refusal to provide a justification.

(2.2) (a) Everyone may problematize any assertion.

(3.3) Whoever has put forward an argument is only obliged to produce further arguments in the event of counter-arguments.

Again, these few principles by no means exhaust those proposed by Alexy. There would seem to be no way to formalize the constraint, e.g., that "Every speaker may only assert what he or she actually believes."

It might be helpful to restate these norms in a way which more closely resembles the formalization:

1. No party may contradict himself.
2. A party who concedes that a rule is valid must be prepared to apply the rule to every set of objects which satisfy its antecedent.
3. An argument supporting an issue may be asserted only when the issue has been denied by the opponent.
4. A party may deny any claim made by the opponent, if it is not a necessary consequence of his own claims.
5. A party may rebut a supporting argument for an issue he has denied.
6. A party may defeat the rebuttal of a supporting argument for one of his own claims, if the claim is an issue.

The first principal just confirms Alexy's rule 1.1.

The second principal is an operational restatement of Alexy's rule 1.3. Two objects (more precisely, two sets of objects) are considered to be alike "in all relevant respects", relative to some rule, when they both satisfy the antecedent of the rule. Notice, however, that an exception may be applicable to one of the objects, but not the other. Thus, after all rules have been considered, it may well be that the predicate of the conclusion of the general rule is "applied" to only one of objects, even though the antecedent of the rule is satisfied by both.

Neither Alexy nor I require an uncontested claim to be supported by an argument. However, Alexy's principal 2 obliges the proponent of the challenged claim to support it with an argument, whereas my third principal permits, but does not require, a supporting argument to be made. Moreover, Alexy leaves open whether or not an argument may be made when the claim has not been challenged. In my system, an argument is permitted only when the claim has been denied and is still an issue.

The fourth principal is my version of Alexy's principal 2.2(a). Alexy permits "everyone to problematize any assertion." My principal is more restrictive. Only the parties may deny claims, not just anyone. And the claim may be denied only if it is not a necessary consequence of his own claims. It might be thought that Alexy's principal 1.1 implies this second condition. However, although principal 1.1 prohibits a speaker from asserting statements which contradict his previous claims, it is unclear whether this principal prohibits the speaker from denying the necessary consequences of his claims.

My fifth and sixth principals correspond to Alexy's principal 3.3. As in the principal for supporting arguments, counterarguments are permitted in my system, rather than obligatory, as in Alexy's. (Strictly speaking, Alexy only seems to require defeating counterarguments to be asserted, but leaves open whether rebuttals are also obligatory.) As for supporting arguments, rebuttals and defeating counterarguments are allowed only so long as the claim they are about is still an issue.

To complete this section, and to facilitate an intuitive understanding of the Pleadings Game model, consider the following hypothetical exchange of allegations, concerning an Article Nine priority conflict between two secured transactions.

The plaintiff, Smith, and the defendant, Jone, have both loaned money to Miller for the purchaser 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. This is where we enter the pleadings.

Plaintiff. My security interest 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 the Pleadings Game: 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 next section describes the rule language of the Pleadings Game, which realizes this approach.

3. A Language for Explicit Exceptions

The consequences of arguments and counterarguments in the Pleadings Game are modelled using Geffner and Pearl's nonmonotonic logic of "conditional entailment" [1992].

(There is no room here to justify the choice of conditional entailment over the multitude of other logics for defeasible reasoning. See [Gordon, 1993] for a fuller discussion of this issue.) One of the features of Geffner and Pearl's theory of conditional entailment is that it uses specificity information to order conflicting defeasible rules. This is seen as an advantage. It is argued that the task of maintaining a set of defeasible rules using systems, such as Reiter's Default Logic, which require the user to rank conflicting rules explicitly, is unduly burdensome and error prone. Others, such a McCarty and Cohen [1992], have argued just the opposite, claiming that it can be easy to write and maintain sets of defeasible rules using explicit exceptions. Ideally, the system would provide the means for representing defeasible legal rules in a way which reflects the way statutes are usually written. Article Nine of the Uniform Commercial Code requires support for *both* implicit and explicit exceptions. However, one empirical observation can be made at this point: a brief scan of Article Nine shows that explicit exceptions predominate. In fact, it is difficult to find examples of implicit exceptions. Here are a few (paraphrased) examples of explicit exceptions of the kind we would like to be able to handle:

§9–102. (1) Except as provided in Section 9–104 on excluded transactions, this Article applies...

§9–105. (h) "Goods" inlcude all things movable... but does not include money or instruments.

§9–302. (1) A financing statement must be filed to perfect all security instruments except the following: (a) a security interest in collateral in possession of the secured party under Section 9–305.

Notice that there are two types of explicit exceptions in these few examples: (1) those which state that one section does not apply when another section does apply, such as the first and third examples; and (2) those which create an exception by explicitly stating the conditions under which the section does not apply, such as the second example, where money and instruments are excepted from the definition of goods.

As Geffner and Pearl's system is attractive for several other reasons, such as its support for partially ordered defaults, its model-theoretic semantics, and its interesting argument-oriented proof theory, the question arises whether it is possible to somehow encode explicit exceptions using conditional entailment. The answer is yes. This section describes one method. Moreover, the method has the advantage, as will be demonstrated, that rules can be structured in a way that is quite similar to the way statutes are written.

The rest of this section assumes familiarity with Geffner and Pearl's theory of conditional entailment [Geffner 1992].

Let us begin by defining a first-order language. First, there is a set of *symbols*, S, and a triple $\langle R, F, C \rangle$, where R, F, and C are members of 2^S, the power set of S. R, F and C are sets of *predicate*, *function* and *constant* symbols, respectively. Each symbol in R and F is associated with an integer giving its *arity*. A *constant* may be viewed as a function symbol whose arity is 0, but the use of a separate set C for constants allows the same symbol to be used as both a constant and a function with some other arity. There is also a set of symbols for the usual quantifiers, the connectives and a set of variables. The language **L** is the set of terms and formulas determined by ⟨*R*, *F*, *C*⟩. To facilitate the layout of rather complex formulas in examples from Article Nine, a Lisp-like notation will be used.

DEFINITION 1 (Terms). The set of terms of **L** is inductively defined, as follows:
1. A variable is a term.
2. If `c` is a constant symbol in *C*, then `c` is a term.
3. If `f` is a function symbol in *F*, with arity `n`, and `t1` ... `tn` are terms, then `(f t1 ... tn)` is a term.
4. Nothing else is a term.

A *closed term* has no variables.

DEFINITION 2 (Atomic Formulas). An expression `(r t1 ... tn)` is an *atomic formula* if `r` is a predicate symbol in *P* of arity `n` and `t1` ... `tn` are terms. The symbol `false` is an atomic formula.

DEFINITION 3 (Formulas). The set of formulas of **L** is inductively defined as follows:
1. An atomic formula is a formula.
2. If `p` is a formula, then `(not p)` is a formula.
3. If `p` and `q` are formulas, then `(if p q)` is a formula.
4. If `p1` ... `pn` are formulas, then `(and p1 ...pn)` and `(or p1 ...pn)` are formulas.
5. If `p` is a formula and `x1` ... `xn` are variables, then `(all (x1 ... xn) p)` and `(exists (x1 ...xn) p)` are formulas.
6. Nothing else is a formula.

The basic set of connectives included here does not exhaust the possibilities, but any first-order predicate logic sentence can be expressed using these.

What is the semantics of **L**? Geffner and Pearl's system of conditional entailment is defined relative to some monotonic consequence relation. In their paper, this relation was assumed to be the ordinary entailment relation of classical logic, but any consequence relation may be used. To model legal argumentation, it will be necessary to use two different consequence relations, denoted $\models$ and known, where known is a weaker, decidable subset of $\models$. The `known` relation will be defined precisely below. For the purposes of the formalization of legal argument, it is unimportant which monotonic consequence relation is chosen for $\models$. Let us assume in the rest of this article that it is classical entailment. (In the computational model described in Section 9, we will use a somewhat weaker logic, to avoid some of the computational complexity of classical logic.)

Recall that a *default theory* for conditional entailment is a pair ⟨*K*, *E*⟩, where *K*, the *background context*, is itself a pair ⟨*L*, *D*⟩. *E* and *L* are sets of closed formulas from **L**, representing the case-specific *evidence* and the nondefeasible generic knowledge of the domain, respectively. D is a set of *defaults*. A default is a ⟨*p*, *q*⟩ pair, denoted $p \Rightarrow q$, 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 of *L*. The *assumptions* of a default theory are the set of consequents of all instances of the defaults in *D*. These assumptions may be viewed as names of instances of defaults.

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 [Gordon 1986] and [Bench-Capon 1992a], 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 δ, when some condition `q` is satisfied, it is sufficient to add `(if q (not δ))` to L. The problem of explicitly ranking default instances is more subtle. To give an assumption γ priority over another assumption δ, it is not sufficient to assert `(if γ (not δ))`, as this equivalent to `(if δ (not γ))`. The key to understanding how to explicitly order default instances is contained in the syntactic test for determining whether one assumption δ is preferred to another in some set of assumptions Γ in all admissable priority orderings [Geffner 1992, p. 220]. If the default instance for δ is $p \Rightarrow \delta$, then it is preferred to some assumption in Γ in all admissable orderings if and only if $\Gamma \cup \{p\} \models$ `(not δ)`. Thus, to encode an explicit preference for δ over another assumption γ, it is sufficient to add the formula `(if (and p γ) (not δ))` to the set of nondefeasible sentences L, where `p` is the antecedent of the default instance for δ.

Rather than encoding defeasible rules directly in this format, we develop a more convenient notation for such rules, whose semantics is given by a mapping into defaults and sentences of K:

DEFINITION 4 (Rules). A *rule* is a quintuple $\langle \delta, (x_1, \ldots, x_n), a, c, e \rangle$, where d is a constant symbol naming the rule, $(x_1, \ldots, x_n)$ is a vector of n distinct variables for the *parameters* of the rule, and a, c, and e are formulas, which may include free variables, for the *antecedent*, *consequent*, and *exception* of the rule, respectively. Every free variable in a, c, and e is a member of $(x_1, \ldots, x_n)$. For convenience, a rule will be denoted by `(rule δ` $(x_1, \ldots, x_n)$ `if a then c unless e)`, or, if e is `false`, simply as `(rule δ` $(x_1, \ldots, x_n)$ `if a then c)`.

For example, §9–105 of Article Nine, about which things are goods, might be represented in this format as:

```
(rule s9-105 (x)
  if (movable x)
  then (goods x)
  unless (or (money x) (instrument x))).
```

Here, `s9-105` is the name of the rule, `x` is its only parameter, `(goods x)` is its consequent, `(movable x)` is its antecedent and `(or (money x) (instrument x))` is its exception.

The semantics of rules is given by the following mapping into defaults and formulas of a default theory for conditional entailment.

DEFINITION 5 (Interpretation of Rules). Let the language **L** include four distinguished unary predicates (`applies`, `antecedent`, `ap`, and `backing`) and two function symbols (`inst` and `parms`). Then the *interpretation* of a rule ⟨`delta`, $(x_1, \ldots, x_n)$, `a`, `c`, `e`⟩ is a background context ⟨*L*, *D*⟩, where L is the set of formulas:

```
{(all (x1...xn)
   (if (and a (backing delta))
      (antecedent (inst delta (parms x1...xn))))),

 (all (x1...xn)
   (if (and a
          (backing delta)
          (ap (inst delta (parms x1...xn))))
      (applies (inst delta (parms x1...xn))))),

 (all (x1...xn)
   (if (applies (inst delta (parms x1...xn)))
      c)),

 (all (x1...xn)
   (if (and e
          (ap (inst delta (parms x1...xn))))
      false))}
```

and *D* is a singleton set containing the default

```
(antecedent (inst delta (parms x1...xn))) ⇒
(ap (inst delta (parms x1...xn))).
```

Given a rule *r*, whose interpretation is ⟨*L*, *D*⟩, let `strict` (*r*) = *L* and `defaults` (*r*) = *D*. As a notational convenience, let `I`(*R*) denote the interpretation of a set of rules *R*. If *R* is $\{r_1, \ldots, r_n\}$, then

$$\mathtt{I}(\{r_1,\ldots,r_n\}) = \langle \bigcup_1^n \mathtt{strict}\,(r_i), \bigcup_1^n \mathtt{defaults}\,(r_i)\rangle.$$

Thus, the interpretation of the example above, for §9–105, is the background context consisting of the strict sentences

```
(all (x)
     (if (and (movable x)
              (backing s9-105))
       (antecedent (inst s9-105 (parms x)))))

(all (x)
     (if (and (movable x)
              (backing s9-105)
              (ap (inst s9-105 (parms x))))
     (applies (inst s9-105 (parms x)))))
```

```
(all (x)
     (if (applies (inst s9-105 (parms x)))
         (goods x)))

(all (x)
     (if (and (or (money x) (instrument x))
              (ap (inst s9-105 (parms x))))
         false))
```

and the default

```
(antecedent (ap (inst s9-105 (parms x)))) ⇒
(ap (inst s9-105 (parms x))).
```

If the exception had been `false`, rather than `(or (money x) (instrument x))`, the last sentence of the background context in this example would have been

```
(all (x)
     (if false
         (not (ap (inst s9-105 (parms x))))))
```

which is a tautology. Thus, this last sentence can be safely omitted from the translation when a rule has no exception.

Notice that I have deviated somewhat from Geffner and Pearl's suggested encoding of defeasible rules. Their encoding of a defeasible rule, such as movable things are (normally) goods, is the sentence

```
(all (x)
     (if (and (movable x)
              (s9-105 x))
         (goods x)))
```

and a default `(movable x)` ⇒ `(s9-105 x)`, where a ground instance of the atomic formula `(s9-105 x)` is the name of an instance of the rule. The set of such names make up the set of *assumptions* maximized by conditional entailment.

In my encoding, on the other hand, following a suggestion by Ulrich Junker [1992], rule instances are named by ground *terms*, rather than ground atomic formulas, and the set of assumptions consists of ground instances of applicability sentences such as `(ap (inst s9-105 (parms x)))`. The advantages of this approach will be discussed next, after this definition:

DEFINITION 6 (Rule Instance). A *rule instance* is obtained from a rule ⟨`delta`, $(x_1, \ldots, x_n)$, a, c, e⟩ by systematically substituting every parameter in `x1 ... xn` by a ground term from **L**. The term `(inst delta (parms t1...tn))`, where `t1...tn` are ground terms, is the *name of the rule instance* obtained by substituting `t1...tn` for the parameters `x1...xn` in the rule named `delta`. The formula `(ap (inst delta (parms t1...tn)))` is the *applicability assumption* for this rule instance.

This change in the naming convention for rule instances has no effect on Geffner and Pearl's theoretical results concerning conditional entailment. But it is of practical importance for us here, as it allows us to reason about rule instances without leaving first-order logic. Rules, and rule instances, are *reified* in this system. It is possible to write rules making statements about rules. Moreover, properties of rules can be expressed. For example, when legal statutes are represented as rules in this form, we can state such properties as the date of their enactment, their authority or source, and the article of the code in which they appear. As it turns out, this ability makes it possible to handle all of the kinds of exceptions and principles for resolving conflicts between rules we have identified in Article Nine.

For a first example of the utility of reification, notice that a *backing condition* is included in the translation of rules into background formulas. In the example for §9–105, the relevant formula was

```
(all (x)
     (if (and (movable x)
              (ap (inst s9-105 (parms x)))
              (backing s9-105))
         (applies (inst s9-105 (parms x))))).
```

Reification allows Toulmin's distinction between warrants and backing to be handled. The role of warrant is played by rules. One type of challenge to a conclusion of a rule is to attack the rule by requesting its backing. In the legal context, this is usually reduced to the issue of the *authority* for the rule. Using Susskind's terminology, the rule is a law "statement" which may be backed by an authoritative law "formulation" [Susskind 1987, pp. 36–37]. Pragmatically, there is a discourse rule allowing the parties to agree that a rule is backed, just as any other claim may be conceded, without making an issue out of this. However, the opponent may instead challenge the rule, in which case the proponent will have to present an argument for the backing claim, just as is the case for every other disputed claim, in order to use the rule to derive its consequent. In our example, the `(backing s9-105)` claim may be conceded, but if it is denied, the proponent will have to support the backing claim with an argument in order to use `s9-105` to support `(goods t)`, for any constant `t`. The discourse rules are described in detail below.

It is the rule, `s9-105` here, which must be backed, not just a rule instance, such as `(inst s9-105 (parms t))`. If the backing condition for a rule is falsified, e.g., if `(not (backing s9-105))` is conditionally entailed by the default theory, then the rule is effectively cancelled.

For another example of the utility of reification, let us represent §9–302, in which §9–305 is referred to explicitly in the exception. Recall that §9–302 states, in part:

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

Here, one needs to also be familiar with §9–305 to interpret §9–302; the "security interest in collateral in possession of the secured party" phrase is not an additional condition on the applicability of §9–302, but rather a redundant restatement of part of the antecedent of §9–305, presumably intended to help the reader identify or remember §9–305.

Now, §9–302 can be represented by the following rule:

```
(rule s9-302 (p s g)
   if (an (perfected s)
          (not (filed s)))
   then false
   unless (applies (inst s9-305 (parms p s g)))).
```

Five means of resolving conflicts between legal rules are used in Article Nine: (1) authority (2) time (3) express exceptions (4) specificity, and (5) scoping and applicability rules. I have already shown how express exceptions can be handled in our rule language. Implied exceptions, where rules are ordered by their specificity, are handled by conditional entailment. This leaves authority, time and scoping provisions to be discussed. The reification of rules makes it easy to represent these other conflict resolution principles as well.

This next example will show one way of using both authority and time to order conflicting rules. According to the principal known as *Lex Superior*, rules with higher authority have precedence over rules with lower authority. For example, federal law has priority over state law. On the other hand, the principal of *Lex Posterior* states later rules have priority over conflicting earlier rules. Moreover, both of these principles are themselves defeasible. When they conflict, *Lex Superior* prevails over *Lex Posterior*. For example, a federal law has priority over a conflicting later state law.

We begin with a few formulas about rules. One rule instance conflicts with another if they are not both applicable.

```
(all (r1 r2)
     (if (not (and (ap r1) (ap r2)))
         (conflicting r1 r2))).
```

Using our convention for representing assumptions, the formula for encoding an explicit preference for the default instance δ over the default instance γ would be `(if (and (antecedent` δ`) (ap` γ`)) (not (ap` δ`)))`. To allow explicit preferences to be asserted in a more convenient manner, let us add the following formula to **L**, introducing a preferred predicate.

```
(all (x y)
     (if (preferred x y)
         (if (and (antecedent x)
                  (ap y))
             (not (ap x)))))
```

which is equivalent to

```
(all (x y)
     (if (and (preferred x y)
              (antecedent x)
              (ap y)
              (ap x))
         false)).
```

To see how this can be used, let us now state the rule for ordering rule instances by authority. Let us assume there are predicates for ranking authority and ordering dates. `(higher a1 a2)` is intended to mean that `a1` is a higher authority than `a2`. `(before d1 d2)` is intended to mean that the date `d1` is before `d2`.

```
(rule lex-superior (r1 p1 r2 p2)
  if  (exists (a1 d1 a2 d2)
          (and (conflicting (inst r1 p1)
                            (inst a1 p2)
               (authority r1 a1 d1)
               (authority r2 a2 d2)
               (higher a1 a2)))
  then (preferred (inst r1 p1) (inst r2 p2))).
```

The `(conflicting (inst r1 p1) (inst r2 p2))` condition in this example is not unimportant. Without it, the previous formula for preferred above would imply a conflict between the two rule instances, even if they were not otherwise in conflict. It would have made it impossible to apply state and federal law together in one argument.

As explained previously, in the interpretation of rules of this formalism, one condition of the antecedent of a default instance `(inst r parms)` is the proposition `(backing r)`, meaning that the rule is well supported or founded. This next formula simply states that legal authority is one form of backing. `(authority r a d)` is intended to mean that the rule `r` was enacted by the authority `a` on the date `d`.

```
(all (r a d)
     (if (authority r a d)
         (backing r))).
```

The following representation of the rule for ordering conflicting rules by time is similar to the rule for ordering them by authority, but includes an exception for rules backed by higher authority.

```
(rule lex-posterior (r1 p1 r2 p2)
  if (exists (a1 d1 a2 d2)
         (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 r1 p1)))).
```

Alternatively, we could omit the exception from `lex-posterior` and add another rule ordering instances of `lex-posterior` and `lex-superior` in cases of conflict:

```
(rule lex-posterior (r1 p1 r2 p2)
  if (exists (a1 d1 a2 d2)
        (and (conflicting r1 r2)
             (authority r1 a1 d1)
             (authority r2 a2 d2)
             (before d2 d1)))
  then (preferred (inst r1 p1) (inst r2 p2))

(rule superior-over-posterior (p1 p2)
  if (conflicting (inst lex-superior p1)
                  (inst lex-posterior p2))
  then (preferred (inst lex-superior p1)
                  (inst lex-posterior p2))).
```

Notice that this later rule orders the `lex-superior` and `lex-posterior` rules, and thus only indirectly the "object-level" rules in conflict.

These two versions of `lex-posterior` are not equivalent. In the first version, the backing for `lex-posterior` is also backing for the principal that authority has priority over time. That is, anyone who asserts this version of the rule may not dispute that it is subordinate to the `lex-superior` rule. In the second version, the principal of authority over time is factored out, and represented independently in a rule of its own. Thus, in this version, a party could accept `lex-posterior` while disputing `superior-over-posterior`. Moreover, the exception in the first version is not defeasible. The second version leaves open the possibility of there being exceptions to the principle that authority takes precedence over time.

Finally, here is an example showing how to use this scheme to represent a scoping provision. §9–102 (2) of Article Nine states the article does not apply to statutory liens except as provided in §9–301.

This may be represented using a rule such as:

```
(rule s9-102-2 (e)
  if (exists (r p c d)
        (and (substantive r)
             (event (inst r p) e)
             (security-interest e)
             (collateral e c)
```

```
                    (statutory-lien c)
                    (authority r A9 d)
                    (ap (inst r p))))
      then false
      unless (applies (inst s9-301 (parms e)))).
```

The predicates `substantive`, `event`, `security-interest`, `collateral`, and `statutory-lien` here are domain specific. `(substantive r1)` is intended to mean that a rule `r` is one of substantive law, rather than, e.g., a scoping provision. `(event r e)` is intended to mean that `r` is a rule instance involving an event `e`. `(security-interest e)` means that the event creates a security interest. `(collateral e c)` means `c` is the collateral of the security interest, and `(statutory-lien c)` is intended to mean that c is a statutory lien. `A9` is a constant naming Article Nine.

Notice that §9–102 is self-referential – it purports to regulate the scope of the article in which it itself appears – but this section is certainly not intended to mean that it, §9–102, is not applicable to statutory liens. Presumably it means that only those sections of Article Nine affecting interests in the collateral are not applicable to such liens. The `(substantive r)` condition of the representation of §9–102 makes this condition explicit.

4. The Pleadings Game

Now we are ready to formally define the Pleadings Game itself. It is structured as a formal, two-person game, comparable to Lorenzen's Dialogue Logic [Felscher 1986]. We will be defining the "playing board", the moves permitted by the rules of the game, and a termination criterion for determining when the game is over. It will also be defined what it means to "win" this game. When there are issues remaining at the end of the pleadings, then neither party wins and the case proceeds to trial.

To begin, the game is played against a particular *background.* (Recall that `I` is the interpretation function of Definition 5, above.)

DEFINITION 7 (Background). A *background* is a triple $\langle \phi, S, R\rangle$, where ϕ is a formula, S is a set of formulas and R is a set of rules. ϕ is the *main claim*, the claim the plaintiff ultimately would like to prove. S and R are formulas and rules, respectively, about which the parties agree. Both sets may be empty. Together, S and R determine the *initial background context* K_0 for conditional entailment. $K_0 = \langle L \cup S, D\rangle$, where $\langle L, D\rangle = \mathtt{I}\ (R)$.

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

The only players of the game, during pleading, are the *plaintiff* and the *defendant*. The judge, or court, enters the game only during the trial. The moves available during pleading are assertions of various kinds of statements:

DEFINITION 8 (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 9 (Assertions). There are also 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 playing board. The playing board here will be called the *record*, which is a legal term for the pleadings and other documents filed at the court by the parties. 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 below 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.

Before describing these production rules, the structure and certain properties of the record, used in the preconditions of the rules, need to be defined.

DEFINITION 10 (Statements of a Party). The *statements* of a party are a triple $\langle O, D, C\rangle$, where O, D and C are each sets of statements. O is the set of *open* statements, which have not yet been responded to by the opponent. D is the set of statements which the opponent has *denied.* Finally, C is the set of statements which the opponent has *conceded.*

DEFINITION 11 (The Record). The *record* is also a triple, $\langle b, \pi, \delta\rangle$, where b is the background $\langle \Phi, S, R\rangle$ of the game, and π and δ are the statements of the plaintiff and defendant, respectively.

DEFINITION 12 (Argument). An *argument* is a set of formulas. An *argument for some proposition* is a pair $\langle \Phi, \delta\rangle$, where Φ is a set of formulas and δ is a formula. Given a default theory $\langle K, E\rangle$, where $K = \langle L, D\rangle$, an argument Φ is a *supporting argument* for δ 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 13 (Counterarguments). An argument Γ is a *counterargument* to another argument Φ in a default theory $\langle K, E\rangle$ if and only if $\Gamma \cup \Phi \cup L \cup E \models$ `false`. Γ is a *defeating counterargument* of Φ if and only if they are counterarguments and every assumption in Γ is preferred to some assumption in Φ. Γ is *protected from* Φ if and only if Γ contains a subargument which defeats Φ. Finally, a counterargument Γ of Φ is a *rebuttal* if and only if Φ is not protected from Γ.

The definition of *supporting* argument here is similar to Geffner and Pearl's definition of argument [Geffner 1992, p. 225], except that an argument may consist of arbitrary formulas, not just assumptions, and $L \cup E \cup \Phi$ is not required here to be consistent. The definitions of counterarguments, defeating counterarguments and rebuttals are the same as Geffner and Pearl's. They are repeated here for the sake of completeness.[1]

Here are some auxiliary functions and values used in the rules for assertions. Each of these is defined relative to the current state of the record. A phrase such as "the open statements of the plaintiff", e.g., means his open statements in the current record.

opponent: `statements` $\rightarrow$ `statements`. Maps the statements of one party into the statements of the opposing party. Recalling that π and δ are the statements of the plaintiff and defendant, respectively, `opponent` $(\pi) = \delta$ and `opponent` $(\delta) = \pi$.

rules: 2^{rule}. The union of the rules of the initial background and all rules subsequently declared by either party. The rules declared by a party are $\{r \mid (\texttt{declare}\ r) \in C\}$, where C is the set of conceded statements of the party. Let Π be the rules stated by the plaintiff and Δ be the rules stated by the defendant. Then, `rules` $= R \cup \Pi \cup \Delta$.

K: $2^L \times 2^\Delta$. The current *background context.* $K = \langle L \cup S, D\rangle$, where $\langle L, D\rangle = \mathrm{I}$ `(rules)`.
L: 2^L. The nondefeasible part of the current background context, $\langle L, D\rangle = K$.

facts: 2^L. These are the conceded claims of both parties. They are what Geffner and Pearl, in their system for conditional entailment, call "evidence". The conceded claims of a party are $\{f \mid (\texttt{claim}\, f) \in C\}$, where C is the set of conceded statements of the party. Let Π be the conceded claims of the plaintiff, and Δ be the conceded claims of the defendant. Then `facts` $= \Pi \cup \Delta$.

Θ : 2^L. The current *context*, the union of the nondefeasible sentences of the background context and the facts. $\Theta = L \cup$ `facts`.

claims: `statements` $\rightarrow 2^L$. The claims of a party. If the statements made by the party are $\langle O, D, C\rangle$, then `claims` $(\langle O, D, C\rangle) = \{f \mid (\texttt{claim}\, f) \in (O \cup D \cup C\}$.

arguments: 2^A. The arguments which have been made by the parties. If the statements made by a party are $\langle O, D, C\rangle$, his arguments are

[1] These definitions differ slightly from the ones I used in [Gordon 1991]. There an argument is what I have chosen to call a valid argument here. A rebuttal there is what Geffner and Pearl call counterarguments, as defined above. A counterargument there was a special kind of rebuttal.

$\{\langle A, c\rangle \mid$ (argument $A\ c) \in (O \cup D \cup C)\ \} \cup$
$\{\ \langle A \cup R$, false$\rangle \mid$(rebuttal $A\ c\ R) \in (O \cup D \cup C)\}$.

Let Π be the arguments made by the plaintiff and Δ be the arguments made by the defendant. Then arguments $= \Pi \cup \Delta$.

Two predicates, issue and known play an important role in this model. known is explained next, but a proper explanation of the issue predicate requires more space than I want take here. For now, the following should suffice. issue (ψ, ϕ) is true if and only if the formula ψ is *relevant* to the goal of proving the formula ϕ. (See Section 8, below, for further details.)

The known relation, which is a subset of $2^L \times L$, is relatively easy to define. The intended meaning of known (Γ, ϕ), is that ϕ is known to be entailed by Γ and some set of premises Θ. Intuitively, one would expect a knowledge relation to be decidable, and tractably so. One's own knowledge can be efficiently retrieved. If a difficult problem must first be solved, then intuitively one can be said to know the answer only after the problem has been solved. Thus, the knowledge relation should be some tractably decidable subset of the entailment relation.

DEFINITION 14 (Known Relation). Recall that a supporting argument for a proposition is a pair $\langle\Gamma, \phi\rangle$ where $\Gamma \cup \Theta \models \phi$ in the context Θ. Given this context and a set of supporting arguments A, a formula ϕ is *known* to be entailed by an argument Γ, denoted known$_{\langle A, \Theta\rangle}$ (Γ, ϕ), if and only if:
1. ϕ is a member of $\Gamma \cup \Theta$,
2. known $\langle A, \Theta\rangle$ $(\Gamma$, false$)$, or
3. there exists an argument $\langle\psi, \phi\rangle \in A$, such that, for every formula $\psi \in \psi$, known$_{\langle A, \Theta\rangle}$ (Γ, ψ).

When the context and set of arguments is clear, we will simply write known (Γ, ϕ), without the subscript, as a notational convenience.

As arguments are isomorphic to propositional Horn clauses, not only is the problem of deciding whether known (Γ, ϕ) holds decidable, its worst-case time complexity is *linear* [Dowling 1984].

A well-defined knowledge relation surely requires more than tractability. For an extreme example, the empty set is clearly also a tractably decidable subset of any entailment relation, but it would not seem plausible to suggest that nothing is known. Arguably, the knowledge relation should be a *consequence relation*, satisfying the usual Tarskian properties:

Inclusion. $\Gamma \sqsubset$ closure (Γ), where the closure of Γ is the set of all formulas ϕ, such that known$_{\langle A, \Theta\rangle}(\Gamma, \phi)$.

Idempotence. closure(Γ) = closure(closure(Γ)).

Monotonicity. If $\Delta \sqsubset \Gamma$ then closure$(\Delta) \subset$ closure(Γ).

The `known` relation is a consequence relation in Tarski's sense.

We are ready to define the rules controlling assertions. There are three types of assertions (concessions, denials and defenses); the rules for each type will be presented together. Altogether there are ten rules; they will be numbered for future reference. A precondition of all rules is that the statement being responded to be in the set of open statements of the opponent. If the party making the move is x, this set is denoted by O in $\langle O, D, C\rangle$ = `opponent` (x). An effect of most moves is to delete the statement from O and add it to either the denied statements, D, or the conceded statements, C. If a response by the opponent to this new assertion is possible, then another effect is to add the assertion to the set of upon statements of the party making the move.

No response is required or permitted to the mere declaration of rules, i.e., to an assertion of the form `(declare r)`, where `r` is a rule. 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. These backing claims, which have the form `(claim (backing r))`, may be conceded or denied in the same way as other claims.

The syntax used here for production rules is informal.[2] Each rule consists of a *statement pattern*, which is matched against the statement for which this move is a response, a set of *preconditions* which must be satisfied if the move is to be applicable, and a sequence of *effects*, which are executed in the order they appear when the move is applied. In every rule, p denotes statements of the party making the move, which is intended to be mnemonic for *proponent*, and o denotes the statements of the *opponent* of p, i.e., o = `opponent`(p). Also, as a notional convenience, the open, denied and conceded statements of the parties are denoted using subscripts. For example, if $\langle O, D, C\rangle = p$ then $O_p = O$. In effects, $\leftarrow$ denotes assignment.

There are three rules for concessions. Claims, arguments and rebuttals may be conceded; denials may not be conceded. Conceding the denial of one's own assertion would violate the principle against self-contradiction. For the same reason, a claim may be conceded only if it is not known to be inconsistent with the other claims of the party making the concession. There are no further preconditions on concessions; no artificial barriers are placed on the willingness of the parties to agree. By conceding an argument or rebuttal, the party gives up the opportunity to make a counterargument.

1. **move** `(concede (claim c))`
   preconditions
   - `(claim c)` $\in O_o$
   - $\neg$`known (claims` $(p) \cup \{c\}$`, false)`

   effects
   1. $O_o \leftarrow O_o - \{$`(claim c)`$\}$
   2. $C_o \leftarrow C_o \cup \{$`(claim c)`$\}$

[2] Although it would also have been possible to formalize these rules as operators in a planning language, such as STRIPS [Fikes 1971, Lifschitz 1987], these productions rules are simpler and sufficient for our purposes here.

2. **move** `(concede (argument A c))`
   preconditions
   - `(argument A c)` $\in O_o$

   effects
   1. $O_o \leftarrow O_o -$ { `(argument A c)` }
   2. $C_o \leftarrow C_o \cup$ { `(argument A c)` }

3. **move** `(concede (rebuttal A c R))`
   preconditions
   - `(rebuttal A c R)` $\in O_o$

   effects
   1. $O_o \leftarrow O_o -$ { `(rebuttal A c R)` }
   2. $C_o \leftarrow C_o \cup$ { `(rebuttal A c R)` }.

There are only two rules for denials. Only claims and denials may be denied. Arguments and rebuttals may not be denied as we will require the proponent to prove an argument or rebuttal when it is asserted. (See below.) A party may deny a claim only if it is not known to be entailed by his own claims. This is another use of the principle against self-contradiction. By denying a denial the party leaves any issues raised by the claim for trial and forfeits the opportunity to make an argument supporting his claim.

Denying a statement is not the same as claiming the negation of the statement, because of the division of the burden of proof. The pragmatic effect of a denial is to request the other party to bear his burden of proof. Conversely, a concession relieves the opponent from the burden of proving one of his claims.

4. **move** `(deny (claim c))`
   preconditions
   - `(claim c)` $\in O_o$
   - $\neg$`known (claims` $(p), c)$

   effects
   1. $O_o \leftarrow O_o -$ { `(claim c)` }
   2. $D_o \leftarrow D_o \cup$ { `(claim c)` }
   3. $O_p \leftarrow O_p \cup$ { `(denial (claim c))` }

5. **move** `(deny (denial s))`
   preconditions
   - `(denial s)` $\in O_o$

   effects
   1. $O_0 \leftarrow O_0 -$ { `(denial s)` }.

Declaring a rule simply adds it to the rules of the background of the record. The only precondition is that there not already be another rule of the same name. (In the membership test below, two rules are considered equal if they have the same name.)

6. **move** `(declare r)`
   preconditions
   - `r` $\notin$ `rules`

   effects
   1. $C_p \leftarrow C_p \cup \{$ `(declare r)` $\}$.

Finally, there are several defenses. Rather than merely denying or conceding a statement asserted by the opponent, a defense makes an argument against the statement. If the opponent has denied some claim, the defense is an argument supporting the claim. If the opponent has made an argument, the defense is a rebutting counterargument. If the opponent has asserted a rebuttal, the defense is either a counterargument defeating the rebuttal, or a rebuttal of the rebuttal.

A defense usually asserts new claims. For example, when defending a claim with a supporting argument `(argument A c)`, one not only asserts that `c` is a necessary consequence of A, but claims that each of the formulas in `A` is true. Each claim is asserted at most once. If it is known that previous claims entail some formula in `A`, then the formulas of these previous claims are the only ones which are open to debate. No claim is asserted for the new, entailed formula. This rule encourages making claims which are only as strong as required to prove an argument. Stronger claims than necessary should be held back until they are required. For example, rather than claiming that the collateral is consumer goods, in a context where a good of any kind would suffice, one should claim only that the collateral is a good. If one fails to show that collateral is a consumer good, there may not be another opportunity to show that it is a good for some other reason.

One effect of a defense is to concede all of the open claims of the opponent known to be entailed by the argument of the defense. This is just a matter of convenience, as one precondition of denials is that the claim to be denied not be known to be entailed by the claims of the party making the denial. Thus, the only move permitted is to concede the claim, which is thus done automatically by defenses.

Applicability assumptions in arguments are also not asserted as claims. A rule is challenged by denying its backing, not by contesting the applicability assumption of one of its instances. Conversely, a rule is supported by an argument for its backing, not by arguments for an applicability assumption. These assumptions are included in arguments for technical reasons related to the way conditional entailment realizes defeasible reasoning. They are maximized by conditional entailment. See Geffner and Pearl's paper [1992] for details.

A rebuttal may, but need not, assert further claims. When no claims are made, the rebuttal asserts the argument being rebutted is inconsistent. For example, one rebuttal of an argument `(argument A c)` would be a proof that `A` is inconsistent. Similarly, a rebuttal `(rebuttal A c R)` may itself be rebutted by a proof that `R` is inconsistent. Here we see one reason why rebuttals have their own syntactic form, rather than being represented as `(argument` $(A \cup R)$ `false)`. Although to make a rebuttal one must prove that $(A \cup R)$ is inconsistent, the opponent should be given the opportunity to counter by proving that `R` alone is inconsistent. This opportunity is preserved by representing rebuttals so as to distinguish `A` and `R`.

Rebuttals and defeating counterarguments accept the argument against which they are a defense "for the sake of argument." Intuitively, when making one of these defenses, one says "Even if all the facts you claim are true, which I deny, you are still wrong, because of these other facts." For example, suppose one party has argued that Article Nine does not apply to the transaction because the security interest was created by a statutory lien, §9–102 (2). This argument gives rise to three statements, each of which may be answered. The first is the claim that the interest was created by a statutory lien. The second is the argument that if an interest was created by a statutory lien then it is excluded from Article Nine. The third is that Article Nine provides authority for the proposition that statutory liens are excluded. (This is the question of backing.) The opponent can respond to each of these statements. He can deny that there is a statutory lien, deny that Article Nine excludes statutory liens and at the same time argue, in a rebuttal, that even if statutory liens are excluded, this security interest is covered by Article Nine, appealing to e.g., §9–310. The only limitation here is that a party may not both deny that there is a statutory lien and claim that it is a special type of statutory lien.

The burden of proof is divided between the parties in this system. The party making an argument `(argument A c)` must prove $\Theta \cup \mathtt{A} \models \mathtt{c}$. However, he need not also show that $\Theta \cup \mathbf{A}$ is consistent. The opponent of the argument has the burden of proving its inconsistency, by asserting a rebuttal. Similarly, the party asserting a rebuttal `(rebuttal A c R)` must prove $\Theta \cup \mathtt{A} \cup \mathtt{R} \models \mathtt{false}$, but his opponent may prove that `R` alone is the source of the inconsistency: $\Theta \cup \mathtt{R} \models \mathtt{false}$. Recall that a rebuttal is a counterargument which is not defeated by some subset of the argument it is intended to rebut. Rather than requiring the proponent of the rebuttal to prove that all parts of this test are met, here too the burden of proof is divided between the parties. The proponent of the rebuttal must only show that it is a counterargument which is not *known* to be defeated by some subset of the other argument. The opponent has the burden of proving, using the stronger entailment relation, $\models$, that the supposed rebuttal is in fact defeated by some subset of the other argument. When asserting a defeating counterargument, however, the party making the assertion has the full burden of proving that it is a defeating counterargument.

The burden of proving that the preconditions of a discourse rule are satisfied is to be borne by the party making the move, not by the person (or machine) mediating the game, to assure that the rules are followed. When a precondition of a discourse rule requires proof of consistency, for example, *the party making the move is required to present the proof in such a form that it may be efficiently checked.* That is, the problem of checking the proof is required to be decidable and tractable. The rules of the Pleadings Game do not specify how this condition is to be met. There are at least two possibilities: (1) The proof can be packaged as a sequence of applications of inference rules in some calculus, i.e., as a path through the search space. To check the proof, one need only confirm that each inference rule application is correct. (2) The proof can be represented by a set of settings for some reference theorem prover, such as a particular heuristic evaluation function, selection of a search strategy, and such resource bounds as a depth or time limit. To check the proof then, the mediator need only run the theorem prover using these settings. The proponent is considered to have met his burden of proof only if the reference theorem prover succeeds in finding a proof using these settings.

A defense to a claim `(claim c)`, argument `(argument A c)` or rebuttal `(rebuttal A c R)` may be made only if the formula `c` is an *issue*. This restriction prevents the assertion of irrelevant claims and arguments, which would unduly prolong pleading. Irrelevant claims and arguments *may* be denied or conceded, to remove them from the set of open statements requiring a response. However, one must be careful when choosing between conceding or denying statements, even when they are irrelevant. If the claim is conceded, only those claims are permitted later which are not known to be inconsistent with it and the other claims made or conceded by the party. If the claim is denied, the party may not later make claims known to be entailed by it.

The purpose of the prohibition against making arguments using claims which one has denied is to discourage obstinacy. A claim should be denied only in good faith. However, good faith is not something which can be formally determined. Thus this rule brings with it the risk that a party will be prohibited from using a claim denied in good faith, even when he has become convinced by arguments made to support the claim. To plug a potential loophole in this rule, an open claim must be conceded before it may be used in an argument. Otherwise, it would have been possible to avoid this rule by first making the argument and then denying the claim used in the argument.

For similar reasons, a rule may be applied in an argument only if the party has not denied that the rule has backing.

In this system, exactly one answer to any statement is permitted. Technically, this is realized by permitting responses only to statements in the set of open statements of the opponent. As one effect of every move is to delete this statement from the open set, at most one answer is possible. This restriction also applies to defenses. For example, just one argument may be made supporting a claim denied by the opponent. Each party has the burden of making the best argument available. The purpose of this rule is to prevent a party from making spurious arguments, intended only to delay resolution of the conflict. Conceivably, this rule could be relaxed somewhat, by allowing some small number of defenses, rather than just one.

There is no formal requirement that arguments be minimal. That is, to make an argument `(argument A c)`, the proponent need not first prove that there is no subset `B` of `A` such that $\Theta \cup B \models c$. There are two reasons for this. First, as the proponent has the burden of proving all the formulas in `A`, it is in his own self-interest to keep `A` as small as possible. A formal check is not necessary here to promote the goal of avoiding unnecessary claims. Second, when a rebuttal is defeated by asserting a counterargument, it is known not to be minimal in this sense, as it is a superset of the argument rebutted. The purpose of the additional formulas in the counterargument is not to prove the claim, but to defeat the rebuttal.

7. **move** `(defend (denial (claim c)) A)`
   preconditions
   - `(denial (claim c))` $\in O_o$
   - `issue` (c, ϕ)
   - $\neg(\exists\, \phi \in A$. `(claim` ψ`)` $\in O_o \cup D_o)$
   - $\neg$`known (claims` $(p) \cup$ `A, false)`
   - $\Theta \cup$ `A` $\models$ `c`

   effects
   1. $O_o \leftarrow O_o -$ {`(denial (claim c))`}
   2. for each `(claim` ϕ`)` $\in O_0$, if `known (claims` $(p) \cup$ `A`, ϕ) then
      a. $O_o \leftarrow O_o -$ {`(claim` ϕ`)`}
      b. $C_o \leftarrow C_o \cup$ {`(claim` ϕ`)`}
   3. for each $\phi \in$ `A`, if
      a. $\neg$`known (claims` (p), ϕ), and
      b. ϕ is not an applicability assumption
      then $O_p \leftarrow O_p \cup$ { `(claim` ϕ`)` }
   4. $O_p \leftarrow O_p \cup$ { `(argument A c)` }

8. **move** `(defend (argument A c) R)`
   preconditions
   - `(argument A c)` $\in O_o$
   - `issue` (c, ϕ)
   - $\neg(\exists\, \psi \in$ R. `(claim` ψ`)` $\in (O_o \cup D_o))$
   - $\neg$`known (claims` $(p) \cup$ R, `false)`
   - $\Theta \cup$ `A` $\cup$ `R` $\models$ `false`
   - `A` does not contain a subset which is known to be a defeating counterargument to `R`.

   effects
   1. $O_o \leftarrow O_o -$ { `(argument A c)` }
   2. for each `(claim` ϕ`)` $\in O_o$ if `known (claims` $(p) \cup$ R, ϕ) then
      a. $O_o \leftarrow O_o -$ { `(claim` ϕ`)` }
      b. $C_o \leftarrow C_o \cup$ { `(claim` ϕ`)` }
   3. for each $\phi \in$ R, if
      a. $\neg$`known (claims` (p), ϕ), and
      b. ϕ is not an applicability assumption
      then $O_p \leftarrow O_p \cup$ { `(claim` ϕ`)` }
   4. $O_p \leftarrow O_p \cup$ { `(rebuttal A c R)` }

9. **move** `(defend (rebuttal A c R)` {})
   preconditions
   - `(rebuttal A c R)` $\in O_o$
   - `issue` (c, ϕ)
   - $\Theta \cup$ `R` $\models$ `false`

effects

1. $O_o \leftarrow O_o - \{$`(rebuttal A c R)`$\}$
2. $C_p \leftarrow C_p \cup \{$`(argument R false)`$\}$

10. **move** `(defend (rebuttal A c R) D)`

preconditions

- `(rebuttal A c R)` $\in O_0$
- `issue` (c, ϕ)
- $\neg(\exists\, \phi \in$ D. (`claim` $\phi) \in (O_0 \cup R_0))$
- $\neg$ `known (claims` $(p) \cup$ D, `false)`
- `D` is a defeating counterargument of R.

effects

1. $O_o \leftarrow O_o - \{$`(rebuttal A c R)`$\}$
2. for each (`claim` $\phi) \in O_o$ if `known (claims` $(p) \cup$ D, ϕ) then
   a. $O_o \leftarrow O_o - \{$(`claim` $\phi)\}$
   b. $C_o \leftarrow C_o \cup \{$(`claim` $\phi)\}$
3. for each $\phi \in$ D, if
   a. `known (claims` $(p) \cup$ D, ϕ), and
   b. ϕ is not an applicability assumption
   then $O_p \leftarrow O_p \cup \{$(`claim` $\phi)\}$
4. $O_p \leftarrow O_p \cup \{$`(argument (A` $\cup$ `D) c)`$\}$.

This completes the definition of the set of discourse rules. We now turn to the subject of control and termination. How does pleading begin? What are a player's obligations at each turn? When is pleading over? Can one of the parties "win" at this stage, without the case proceeding to trial? These are the main questions to be answered next.

As is usual in games, the players take turns making moves until some termination criterion is satisfied. The defendant takes the first turn. (Recall that the plaintiff's claim is considered to be part of the initial background context.) On each turn, a player *must* continue to make moves until no *relevant* statement remains to be answered. However, an irrelevant statement *may* also be answered, so long as some move is applicable to it. The pleading phase of the game is over when no relevant statements remain to be answered at the beginning of a party's turn to move. Thus, to state these rules precisely, we first need to define relevance.

DEFINITION 15 (Relevance of Statements). A statement is *relevant* if and only if the formula it is about is an issue. There are four cases, one for each kind of statement. A statement of the form `(claim c)`, `(argument A c)` or `(rebuttal A c D)` is relevant if and only if `c` is an issue. Finally, `(denial s)` is relevant if and only if `s` is relevant.

Now let us formulate the rules for pleading as a recursive procedure:

```
procedure plead (p : statements);
   var s : statements;
   var m : assertion;
```

```
begin
 if $\exists s \in O_o$. relevant ($s$) then
   begin
     while $\exists s \in O_o$. relevant ($s$) do
       begin
         s ← choose a statement in $O_o$;
         m ← choose a move applicable to s;
         execute m;
       end;
     plead (opponent p)
   end
end;
```

There are two choice points in this procedure, the choice of a statement to answer and the choice of a move applicable to that statement. Although there are only finitely many open statements to answer, for any such statement there are infinitely many applicable moves. (For example, any new rule may be declared and the domain of rules is infinite.) Heuristic methods will thus be required to play the game, whether it is to be played by a person or an AI system.

The pleadings game is set up by the plaintiff first making a claim and the parties agreeing, by some unspecified procedure, to sets of background sentences and rules. One possibility is that they will agree to include a whole "knowledge base" about some area of law. In the worst case these sets will be empty, and the game begins with a "clean slate". Here is the initial state of the *background context*. Notice that the only open statement of the plaintiff initially is the main claim.

```
const background = $\langle c, S, R \rangle$ ;
var plaintiff : statements ← $\langle \{$ (claim c) $\}, \emptyset, \emptyset \rangle$;
var defendant : statements ← $\langle \emptyset, \emptyset, \emptyset \rangle$;
```

The game is then started by the defendant taking the first turn:

```
plead (defendant);
```

At the end of pleading, the plaintiff is the "winner" if and only if there are no issues and the main claim, c, is conditionally entailed by the default theory $\langle K$, facts$\rangle$, where K is the final background context. For this purpose, the monotonic consequence relation for testing conditional entailment is the weaker, decidable known relation, not $\models$. The defendant "wins" if the main claim is not conditionally entailed, also using the known relation, and there are no issues to be decided. If neither party has won at this stage, the pleadings game ends in a "draw" and the case proceeds to trial.

In terms of the law of civil procedure, a party would be entitled to a *summary judgment* if he wins the pleadings game. A summary judgment is to be granted in favor of a party when [Black 1979, p. 1287] "there is no genuine issue of material fact and he is entitled to prevail as a matter of law."

5. A Detailed Example

This section demonstrates how the Pleadings Game is played, using the hypothetical case of Smith vs. Jones presented near the beginning of this article. The code shown in this section is from an actual transcript of a game played using the implementation described briefly in Section 9. Some of the commands available for interacting with the system will also be explained here, when they are first used.

Recall that Miller is the debtor on two loans secured by his ship, one from the plaintiff, Smith, and the other from the defendant, Jones. Miller has defaulted on both loans, and the practical question is which of the two security interests has priority. However, for the purposes of this example we will focus on a subsidiary question, whether or not Smith had perfected his security interest, as he claims. Let us suppose that pleading begins by Smith filing a complaint in which he claims that his security interest, `s1`, is perfected.

```
p: (argument bg
       (set (all (x y)
                 (if (and (preferred x y)
                          (antecedent x)
                          (ap y)
                          (ap x))
                     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 the formula discussed above, for conveniently expressing a preference between two rules.

The `complaint` command sets up the game by adding 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 is the defendant's turn:

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

d: (issues)
((perfected s1))

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

The defendant begins by first asking for information. The `statements` command lists the relevant open statements of the opponent. Here the only such statement is the main claim. The `issues` command lists the current set of issues, which also is just the claim the `s1` is perfected. The next move denies that the security interest is perfected, applying rule 4 of the Pleadings Game. 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, open statements to answer, so it's the plaintiff's turn.

```
p: (statements)
(denial (claim (perfected s1)))

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))
```

The only statement to answer is the defendant's denial of the main claim. The plaintiff then declares a rule, `ucc-9-305`, which he believes is an adequate representation of UCC §9–305, which says that a security interest in goods may be perfected by taking possession. The `(rule <symbol>...)` form associates the symbol with the rule and then declares it, applying rule 6 of the Pleadings Game. The rule is intended to state that goods may be perfected by taking possession. Rules are 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.

```
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, applying rule 7 of the Pleadings Game. The system checks that the argument is indeed a supporting argument, using a theorem prover. When asserting a supporting argument the claim 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 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 the claim. 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 for the argument itself. It is the defendant's turn.

```
d: (statements)
(claim (backing ucc-9-305))
(claim (secured-party s1 smith))
(claim (collateral s1 ship1))
(claim (goods ship1))
(claim (possession ship1 smith))

(argument
        ((possession ship1 smith)
         (goods ship1)
         (collateral s1 ship1)
         (secured-party s1 smith)
         (backing ucc-9-305)
         (ap (inst ucc-9-305 (parms smith s1 ship1))))
        (perfected s1))
```

The defendant first asks which statements of the plaintiff are open and relevant. Notice that the claims of the plaintiff's supporting argument, `a1`, were added to his set of open statements, as was the argument itself.

```
    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 simply denies and concedes some of these new claims, using rules 1 and 4 of the game. 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)
          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`, applying rule 8 of the Pleadings Game. The system that `r1` is a counterargument to `a1` which is not known to be protected from it. 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 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 is the plaintiff's turn again.

```
p: (statements)
(denial (claim (goods ship1)))
(denial (claim (possession ship1 smith)))
(claim (if (filed s1) false))
(claim (backing sma-1))
(claim (ship ship1))
(rebuttal
     ((possession ship1 smith)
      (goods ship1)
      (collateral s1 ship1)
      (secured-party s1 smith)
      (backing ucc-9-305)
      (ap (inst ucc-9-305 (parms smith s1 ship1))))
      (perfected s1)

     ((ship ship1)
      (collateral s1 ship1)
      (backing sma-1)
      (ap (inst sma-1 (parms s1 ship1)))
      (not (filed s1))))

p: (issues)
((ship ship1)
 (possession ship1 smith)
 (goods ship1)
 (perfected s1)
 (backing sma-1)
 (not (filed s1)))
```

Again, the system is first queried about the relevant open statements of the opponent, here the defendant, and the current set of issues.

```
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, using rule 5 of the game. 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)
      ie (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 defeats 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.[3] He admits that *Lex Superior* has priority over *Lex Posterior*, by including it as an exception. The assertion of defeating counterarguments is regulated by rule 10 of the game.

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

[3] 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.

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.

There are no further relevant open statements, so it is the defendant's turn.

```
d: (issues)
((before 1960 1972)
 (authority ucc-9-305 ca 1972)
 (authority sma-1 us 1960)
 (conflicting
 (inst ucc-9-305 (parms smith s1 ship1))
 (inst sma-1 (parms s1 ship1)))
 (movable ship1)
 (possession ship1 smith)
 (goods ship1)
 (perfected s1)
 (backing ucc-9-105-h)
 (backing lex-posterior)
 (backing sma-1)
 (not (filed s1)))

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

```
         (authority ucc-9-305 ca 1972)
         (conflicting
           (inst ucc-9-305 (parms smith s1 ship1))
           (inst sma-1 (parms s1 ship1)))
        (possession ship1 smith)
        (goods ship1)
        (collateral s1 ship1)
        (secured-party s1 smith)
        (backing lex-posterior)
        (backing ucc-9-305)
         (ap (inst lex-posterior
              (parms ucc-9-305 (parms smith s1 ship1) ca 1972
                    sma-1 (parms s1 ship1) us 1960)))
         (ap (instance ucc-9-305 (parms smith s1 ship1))))
         (perfected s1))
```

As usual, the defendant first lists the current issues and relevant open statements.

```
     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)))
```

Next, several new claims by the plaintiff are denied or conceded. The backing claims for the representations of UCC §9–105(h) and *Lex Posterior* are conceded, 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)))
```

Here, the defendant concedes the argument that ships are goods, because they are movable, although he had initially denied the claim that `ship1` is a good. The concession of arguments is controlled by rule 2 of the game. 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: (statements)
(denial (claim (backing ucc-9-105-h)))
(claim (backing lex-superior))
(claim (conflicting
          (inst sma-1 (parms s1 ship1))
          (inst ucc-9-305 (parms smith s1 ship1))))
(claim (higher us ca))

(rebuttal
 ((before 1960 1972)
  (authority sma-1 us 1960)
  (authority ucc-9-305 ca 1972)
  (conflicting
      (inst ucc-9-305 (parms smith s1 ship1))
      (inst sma-1 (parms s1 ship1)))
  (possession ship1 smith)
  (goods ship1)
  (collateral s1 ship1)
  (secured-party s1 smith)
  (backing lex-posterior)
  (backing ucc-9-305)
  (ap (inst lex-posterior
        (parms ucc-9-305 (parms smith s1 ship1) ca 1972
             sma-1 (parms s1 ship1) us 1960)))
    (ap (inst ucc-9-305 (parms smith s1 ship1))))
    (perfected s1)

   ((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)))
    (backing lex-superior)
    (applicable
    (instance lex-superior
      (parms sma-1 (parms s1 ship1) us 1960
             ucc-9-305 (parms smith s1 ship1) ca 1972)))))

  p: (issues)
  ((higher us ca)
   (conflicting
        (inst sma-1 (parms s1 ship1))
        (inst ucc-9-305 (parms smith s1 ship1)))
   (possession ship1 smith)
   (goods ship1)
   (perfected s1)
   (backing ucc-9-105-h)
   (backing lex-superior)
   (backing sma-1)
   (not (filed s1)))
```

First the plaintiff lists the current issues and open, relevant statements.

```
  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))
```

Then the plaintiff just denies or concedes the remaining open relevant statements, including the rebuttal `r3`. According to rule 3 of the game, any rebuttal may be conceded, but at the cost of losing the opportunity to assert a defeating counterargument.

After this last move, there are no relevant statements left to answer, for either party, so the game is over. There are issues remaining, so the Pleadings Game ends in a draw:

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

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:

```
>(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))
```

6. The Trial Game

In this model of legal argumentation, if the pleadings game was played to a draw, the case proceeds immediately to trial. The model does not account for the motions and "devices", such as those designed to discover evidence, which may take place after pleading and before trial.

Another simplification is that the only player at trial is the court. In practice, the parties, through their attorneys, present evidence at the trial, for which there are elaborate procedural rules. Also, the fact-finding and law-finding roles of the court are merged here, although they may be divided between a judge and jury in practice.

The court has a relatively "passive" role in this model. The legal and factual issues are completely determined by the parties, during pleading. The court's role is restricted to choosing which of the claims to accept. It is not free to make arguments on its own initiative.

Although there is only one player, the court, the trial is also modeled as a game, although it is much simpler than the one for pleading. Here the game board is called the proceedings, which includes the record of the pleadings.

DEFINITION 16 (Proceedings). The *proceedings* is a triple $\langle r, A, R\rangle$, where r is the record of the pleadings, and A and R are sets of formulas for claims *accepted* and *rejected* by the court, respectively.

There are only two moves; the court may decide to accept or reject some claim.

DEFINITION 17 (Decisions). There are two kinds of *decisions*, defined inductively as follows:

- If `c` is a formula, then `(accept c)` is a decision.
- If `c` is a formula, the `(reject c)` is a decision.
- There are no other decisions.

The accepted formulas become part of the facts. Technically, the definition of `facts` used during pleading is modified to include the accepted claims as well as those conceded by the parties. The issues are to be determined using this new definition.

DEFINITION 18 (Facts at Trial). The **facts** are the conceded claims of both parties and the claims accepted by the court. If the record is ⟨b, π, δ⟩, then

$$\texttt{facts} = \{f \mid (\texttt{claim}\,f) \in (C_\pi \cup C_\delta) \vee f \in \mathrm{A}\}.$$

The court is not free to choose just any disputed claim to decide. Rather, the preconditions of the moves are designed so as to focus the court's attention on those issues with the most potential for reducing the number of issues which must ultimately be addressed to decide the case and to assure that no relevant arguments made by the parties are ignored. The concept of an *active issues* is introduced for this purpose. It plays a role analogous to that of relevant statements during pleading. Active issues are defined in Section 8.

Here are the production rules prescribing when a decision is applicable, and the effect of making the decision.

move `(accept c)`
preconditions

- `c` is an active issue.

effects

- $A \leftarrow A \cup \{\mathrm{c}\}$

move `(reject c)`
preconditions

- `c` is an active issue.

effects

- $R \leftarrow R \cup \{\mathrm{c}\}$.

The trial is over when no further move is possible, i.e., when no active issues remain. The prescribed procedure for conducting the trial is:

```
const r : record $\langle b, \pi, \delta\rangle$, where $b$ is $\langle c, L, D\rangle$ ;
var A : $2^L \leftarrow \emptyset$ ;
var R : $2^L \leftarrow \emptyset$ ;
while there are applicable decisions do
  begin
    select an applicable decision;
    make the decision
  end
```

At the end of the trial, the court shall enter judgment for the plaintiff if and only if the main claim, c, is conditionally entailed by the default theory $\langle K,$ facts$\rangle$, where K is the final background context of the record. Otherwise the court shall enter judgment for the defendant. As in determining whether a party is entitled to a summary judgment after pleading, conditional entailment here is to be determined using the decidable known consequence relation, not the stronger $\models$ relation.

How much discretion does the court have in this model? On the one hand, the court is free to decide issues in any way it chooses. Here the model surely gives the court too much discretion. A more realistic model should at least address the law of evidence. For example, one constraint might be that at least some evidence supporting the claim must be presented at the trial before the court may accept the claim. On the other hand, the parties have complete control over the delineation of the factual and legal issues in this model. The court is not permitted to make arguments on its own initiative. Moreover, the court must address all of the arguments made by the parties, so long as they are relevant.

In the chapter on legal philosophy in [Gordon 1993], I argued that legal judgments need not be formulated as deductive proofs. The main point was that deductive proof alone is neither sufficient nor necessary for limiting judicial discretion and subjecting judgments to review. It is not sufficient, as the court should be obligated to address the arguments made by the parties, and should not be permitted to merely assume that which is to be proven. It is not necessary, as elliptical arguments, in which uncontested premises remain unstated, should be permitted. The model presented here remains faithful to these points.

7. Dialectical Graphs

The issues surrounding a claim depend on the dialectical structure of arguments pro and contra the claim. This section precisely defines this dialectical structure. Recall that an argument is a set of formulas. Let A denote the domain of arguments. The following functions will be defined here:

supports: $L \rightarrow 2^A$. The set of minimal supporting arguments for some formula.

rebuttals: $A \rightarrow 2^A$. The set of minimal rebuttals of an argument.

defeaters: $A \rightarrow 2^A$. The set of minimal defeating counterarguments of an argument.

Given a formula ψ, these functions can be used to generate the complete *dialectical graph* of arguments and rebuttals for ψ. Each node of the graph is a set of formulas. The root of the graph is the singleton set $\{\psi\}$. The rest of the graph of arguments can be partitioned into *layers*. The arguments of the first layer are the minimal supporting arguments of ψ. The next layer consists of the rebuttals of these supporting arguments. The third layer consists of the defeating counterarguments of these rebuttals, and so on.

Figure 2 shows one path through the dialectical graph for the example about whether or not a security interest in a ship had been perfected.

Let us begin by defining the `supports` function. This will be shown to be a well-founded, but tractable, form of abduction.

The `supports`: L $\rightarrow 2^A$ function maps a formula to the *minimal*, consistent arguments known for the formula. As a first attempt, we might consider defining `supports` as follows. Given a set of arguments A and a context Θ, $\Gamma \in$ `supports` (ϕ) if and only if:

1. `known` (Γ, ϕ), and
2. $\neg$`known` (Γ, `false`), and
3. $\neg(\exists\, \psi\,.\, \psi \subset \Gamma \wedge$ `known` $(\psi, \phi))$.

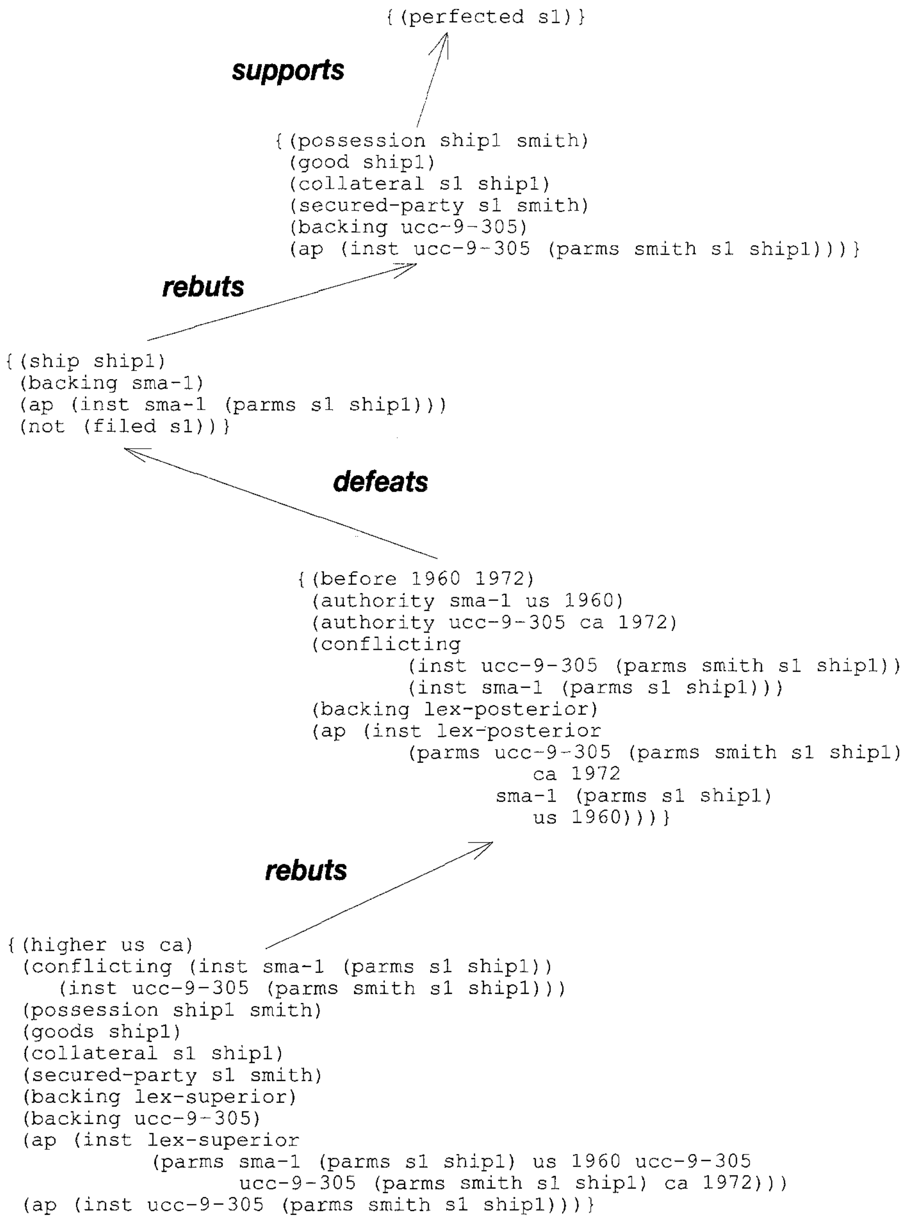
The problem of computing the `supports` of a formula using this definition can easily be shown to be isomorphic to the problem of finding the abductive "explanations" of a proposition from a set of propositional Horn clauses. More precisely, let $\langle A$, `supports`$\rangle$ be a structure, where A is a set of arguments. The `supports` function of this structure can be simulated by the `explanations` function of a $\langle C$, `explanations`$\rangle$ structure, where C is a set of propositional Horn clauses.

A literal is either a propositional letter p, called a positive literal, or its negation $\neg p$, a negative literal. Propositional Horn clauses are often defined as sets of literals with at most one positive literal. A clause $\{p, \neg q_1, \ldots, \neg q_n\}$, for $n \geq 0$, is interpreted to be equivalent to a formula in disjunctive normal form $p \vee \neg q_1 \vee \ldots \vee \neg q_n$. Such a formula is of course equivalent to the material implication $q_1 \wedge \ldots \wedge q_n \rightarrow p$. A definite Horn clause has exactly one positive literal. In the following, we will restrict our attention to definite clauses, which can be more conveniently represented as pairs $\langle\{q_1, \ldots, q_n\}, p\rangle$, where n may be 0.

With these preliminaries, the `explanations` function can be defined as follows.

DEFINITION 19 (Propositional Horn Clause Abduction). Given a set of propositional, definite Horn clauses, C, the *abductive explanations* of a propositional letter p, denoted `explanations`(p), are all sets of propositional letters Γ such that:

1. $\Gamma \cup C \models p$, and
2. $\Gamma \cup C$ is satisfiable, and
3. There does not exist a subset Δ of Γ such that $\Delta \cup C \models p$.



```
{(perfected s1)}

{(possession ship1 smith)
 (good ship1)
 (collateral s1 ship1)
 (secured-party s1 smith)
 (backing ucc-9-305)
 (ap (inst ucc-9-305 (parms smith s1 ship1)))}

{(ship ship1)
 (backing sma-1)
 (ap (inst sma-1 (parms s1 ship1)))
 (not (filed s1))}

{(before 1960 1972)
 (authority sma-1 us 1960)
 (authority ucc-9-305 ca 1972)
 (conflicting
        (inst ucc-9-305 (parms smith s1 ship1))
        (inst sma-1 (parms s1 ship1)))
 (backing lex-posterior)
 (ap (inst lex-posterior
        (parms ucc-9-305 (parms smith s1 ship1)
                  ca 1972
               sma-1 (parms s1 ship1)
                  us 1960)))}

{(higher us ca)
 (conflicting (inst sma-1 (parms s1 ship1))
    (inst ucc-9-305 (parms smith s1 ship1)))
 (possession ship1 smith)
 (goods ship1)
 (collateral s1 ship1)
 (secured-party s1 smith)
 (backing lex-superior)
 (backing ucc-9-305)
 (ap (inst lex-superior
          (parms sma-1 (parms s1 ship1) us 1960 ucc-9-305
                 ucc-9-305 (parms smith s1 ship1) ca 1972)))
 (ap (inst ucc-9-305 (parms smith s1 ship1)))}
```

Fig. 2. One Path in a Dialectical Graph.

This definition is similar to the definition of "minimal supports" given by Reiter and de Kleer in [1987, p. 184], except that their definition is for arbitrary clauses, not just definite Horn clauses.

Now, it is easy to create a ⟨C, `explanations`⟩ structure to simulate the supports function of ⟨A, `supports`⟩. First a bijective function f mapping the arguments of A to propositional Horn clauses is constructed. This only requires naming each formula in A

by a propositional letter. Let f^{-1} be the inverse function of f. Then Γ is a minimal supporting argument of a formula f, for which there is an argument in A, if and only if there is a set of propositional letters $\{p_1, \ldots, p_n\}$ such that:

$$\Gamma = \bigcup_{1}^{n} f^{-1} p_i$$

where

$$\{p_1, \ldots, p_n\} \in \texttt{explanations}\ (f\ \phi)).$$

Other than showing that the problem of determining the minimal supporting arguments of a formula can be viewed as an abduction problem, this result is interesting for another reason. As discussed by Selman and Levesque [1990] this problem of generating the abductive explanations of a propositional letter from a set of propositional Horn clauses is NP-complete. (In the worst case, there can be exponentially many such explanations.) Thus, the problem of finding the minimal supporting arguments of a formula, using the above definition, is also NP-complete. Selman and Levesque have shown that the problem of finding a *single*, nontrivial explanation from a set of propositional Horn clauses, can be solved in time $O\ (kn)$, where k is the number of propositional letters and n is the number of occurrences of literals in the set of Horn clauses. But this result is of no assistance to us here, as we are interested in testing whether a formula is an issue, which requires, in the worst case, all minimal supporting arguments to be examined.

This result is unsettling. One could just accept that there is no efficient algorithm for computing minimal supporting arguments, and thus issues, and leave it at that. However, this contradicts my intuitions about the concept of an issue. In a dispute about some subject, the parties know what issues have been raised. Recalling the issues is not a hard problem. Constructing arguments and rebuttals can be hard, or even undecidable, but keeping track of the arguments which have been made, and the dependencies between arguments and issues, should be an easy task. To use the metaphor of civil procedure, this should be a job for the *clerk* of the court. The arguments made in the pleadings of the parties are filed with the clerk, who should then be able to report about the record of the pleadings, including listing the issues. This should be a simple, bookkeeping task. The burden of solving any undecidable or hard problems should be distributed fairly among the players, rather than delegated to the clerk. Thus, rather than accept the intractability of the problem of generating supporting arguments, I view this as an indication of a potential flaw in the model of argumentation.

The source of this unwanted computational complexity must be the third prong of the provisional definition of `supports` above, regarding minimality. The first two conditions just check whether a tuple is a member of the `known` relation, which is a tractable problem of linear complexity, as mentioned previously. To gain tractability, let us weaken this minimality requirement. Instead of requiring that there not be a subset ψ of Γ such that the formula of interest, ϕ, is known to be entailed by ψ, let us instead require merely that no smaller argument for ϕ than Γ has been made by one of the parties. This shifts the burden of finding minimal arguments from the clerk to the parties, giving us the following definition.

DEFINITION 20 (Supports). Given a set of arguments A and a context Θ, an argument $\Gamma \in$ `supports` (ϕ) if and only if:

1. `known` ($\Gamma\ \phi$),
2. $\neg$`known` (Γ, `false`), and
3. $\neg(\exists\ \psi\ .\ \psi \subset \Gamma \wedge \langle\psi, \phi\rangle \in A)$.

What has been lost by adopting this tractable version of the idea of minimal supporting arguments? It brings with it the risk that the parties will waste resources arguing about claims which could have been shown to be irrelevant to the main claim. However, as each party bears the burden of proving his claims, there is a natural incentive to avoid irrelevant claims. Thus, from a pragmatic point of view, there is no need for the clerk to try to solve the same hard problem that presumably one of the parties has already tried to solve.

Using this tractable definition of minimal supporting arguments, has its relationship to abduction also been lost? Perhaps minimal supporting arguments could be considered an incomplete form of abduction, in Selman's sense [1990]. However, this characterization is not quite satisfactory, as the adjective "complete" here seems out of place. The minimal supporting arguments of a formula are not a subset of some "complete" set of abductive explanations. (Indeed they are a *superset* of the minimal supporting arguments generated using the definition isomorphic to propositional Horn clause abduction.) Instead, I will argue that the `supports` function is simply a tractable instance of a more abstract concept of abduction.

To clarify this idea, a suitably abstract definition of abduction is needed. There is no one authoritative definition of abduction. According to the *Encyclopedia of Philosophy* [Edwards 1972, vol. 5, p. 57], abduction is:

(1) A syllogism whose major premise is known to be true but whose minor premise is merely probable. (2) C.S. Pierce's name for the type of reasoning that yields from a given set of facts an explanatory hypothesis for them.

The first definition here would not usually be called abduction today, at least not within the field of Artificial Intelligence. Rather, one would speak of probabilistic, default or defeasible reasoning. The second definition has found wider acceptance [Charniak 1985, Konolige 1990, Poole 1990]. Here, abduction is the process of inferring potential causes or explanations of observed effects from general knowledge about how the world functions. That is, according to this view, abduction is another name for diagnosis. It is a kind of task, or a class of problems, rather than a particular kind of *formal* reasoning. From this point of view, it is not incoherent to speak of solving abduction problems deductively [Eshghi 1988].

Others, however, contrast deduction, abduction and induction, using purely formal criteria. Hector Levesque, e.g., has adopted this view [1989, p. 1061]:

Using the terminology of C. S. Pierce, given sentences α, β and $\alpha \rightarrow \beta$, there are three operations one can consider: from α and $\alpha \rightarrow \beta$, one might *deduce* β; from α and β, one might *induce* $\alpha \rightarrow \beta$; and from β and $\alpha \rightarrow \beta$, one might *abduce* α.

Abduction can be thought of as a form of hypothetical reasoning. To ask what can be abduced from β is to ask for an α which, in conjunction with background knowledge, is sufficient to account for β. When α and β are about the physical world, this normally involves finding a cause α for an observed effect β.... But not all abduction is concerned with cause and effect. If we happen to know that Marc is 3 or 4 years old, the fact that he is not yet 4 does not *explain* his being 3, although it does imply it, given what is known....

Levesque goes on to propose a very general description of abduction, in terms of the "simplest explanations" of a proposition in an "epistemic" state. Every particular form of abduction has a "belief operator" and a partial order on explanations, where the smallest explanations in the order are considered to be the "simplest". For example, Levesque proposes the number of distinct literals in an explanation as a measure of simplicity.

I would like to propose a similar signature for abduction here. Let A denote the domain of arguments, 2^L. An abduction structure is a quadruple $\langle L, \models, <, e\rangle$, where:

1. L is some logical language (i.e., a possibly infinite set of formulas);
2. $\models$ is a consequence relation on $A \times L$;
3. $<: L \rightarrow 2^{A \times A}$ maps a formula to a strict partial order on arguments (denoted $<_\phi$, for a formula ϕ) and
4. $e : A \times L \rightarrow 2^A$ is a function satisfying the following properties. $\Gamma \in e(\psi, \phi)$ only if:
   a. $\Gamma \cup \psi \models \phi$,
   b. It is not the case that $\Gamma \cup \psi \models$ `false`, and
   c. There does not exist an argument Δ such that
      i. $\Delta \cup \psi \models \phi$,
      ii. It is not the case that $\Delta \cup \psi \models$ `false`, and
      iii. $\Delta <_\phi \Gamma$.

The principal innovation here is to make the ordering on "explanations" dependent on the formula to be explained.

This signature is general enough to capture several common forms of abduction. Propositional Horn clause abduction, for example, is an abduction structure $\langle L, \models, <, e\rangle$, where L is restricted to propositional Horn clauses, and $\Delta <_\phi \Gamma$ if and only if $\Delta \subset \Gamma$. e here is a partial function: $e(C, p)$ is defined only if C is a set of definite Horn clauses and p is a propositional literal. Recall that a signature like this states only the defining properties of a class of structures. Particular instances may satisfy additional constraints.

The ordering relation on explanations used for propositional Horn clause abduction is partially *semantic*, as it prefers the smallest sets of formulas which *entail* the formula to be explained. Entailment is a semantic concept. However, in many applications of abduction, the preference relation on explanations is defined by purely syntactic criteria. Levesque goes so far as to argue that syntactic criteria are *necessary* to capture some notion of simplicity [Levesque 1989, p. 1063].

David Poole's Theorist system also uses a syntactic method to order explanations; the preferred explanations consist only of formulas drawn from a set of *hypotheses* [Poole 1985, Poole 1990]. A Theorist structure is a quadruple $\langle L, F, H, f\rangle$, where L is a logical language, F is a set of closed formulas from L, called the "facts", and H is a set of hypotheses. A *scenario* is a subset D of H such that $D \cup F$ is (classically) consistent. The *explanations* of a formula ϕ in Poole's system, $f(\phi)$, are every scenario D such that $D \cup F$ (classically) entails ϕ. Thus, a Theorist structure $\langle L, F, H, f\rangle$ is an abduction structure $\langle L, \models, <, e\rangle$, where $\models$ is classical entailment, $\Delta <_\phi \Gamma$ is always false, and $e(F, \phi) = f(\phi)$.

Notice that explanations are not ordered in Theorist. Also, as in the previous example, *e* here is a partial function; it is defined only for *F*.

As a final example, here is the abduction structure for my theory of minimal supporting arguments. Given a set of arguments *A* and a context Θ, the abduction structure is

$$\langle L, \texttt{known}_{\langle A,\Theta\rangle}, <, \texttt{supports}\rangle$$

where *L* is the set of first-order formulas occurring in *A* and $\Delta <_{\phi} \Gamma$ if and only if $\Delta \subset \Gamma$ and $\langle \Delta, \phi\rangle \in A$.

The `rebuttals` and `defeaters` functions remain to be defined. These are the smallest known arguments which are presumed to rebut an argument, or known to defeat an argument, respectively.

These definitions use Geffner and Pearl's syntactic test for whether an assumption is necessarily preferred to some assumption in a set of assumptions, except that here we use the decidable `known` relation in the test, instead of $\models$.

Once it is known that one argument defeats another, no additional amount of inference will require this conclusion to be retracted, as the `known` relation is monotonic. Recall, however, that one argument is a rebuttal of another only if this other argument does not contain a defeating counterargument of the first argument. Thus, it is only *presumed* that one argument rebuts another when using the weak `known` relation to check whether or not the other argument is protected. If after additional reasoning it becomes known that it is protected, then this presumption should be retracted.

DEFINITION 21 (Rebuttals and Defeaters). Let Γ and ψ be arguments. Γ and ψ are *known counterarguments* if and only if `known` (Γ ∪ ψ, `false`). Γ is a *known defeating counterargument* of ψ if and only if they are known counterarguments and for every assumption $\delta \in \Gamma$ it is the case that `known` ({α, δ} ∪ ψ', `false`), where α is the antecedent of the default instance for the assumption δ and ψ' is the set of assumptions in ψ. Γ is *known to be protected* from ψ if and only if Γ contains a subargument which is known to defeat ψ. Γ is a *presumed rebuttal* of ψ if and only if they are known counterarguments and ψ is not known to be protected from Γ.

Γ ∈ `rebuttals` (ψ) if and only if:

1. Γ is presumed to be a rebuttal of ψ, and
2. There does not exist a subset of Γ which is presumed to be a rebuttal of ψ.

Γ ∈ `defeaters` (ψ) if and only if:

1. Γ is known to be a defeater of ψ, and
2. There does not exist a subset of Γ which is known to be a defeater of ψ.

8. The Concept of an Issue

The concept of an *issue* has been used in a number of places: in the preconditions of defenses, the definition of relevant statements and in the *active issues* of the Trial Game.

In this section the concept of issue will be formally defined. The theory of issues presented here is a refinement of previous work of mine [Gordon 1989, Gordon 1991]. There are however some important differences, due first of all to the adoption of Geffner and Pearl's system of conditional entailment as the basis for defeasible reasoning.[4] Using conditional entailment, the concept of issue can no longer be restricted to formulas in minimal arguments supporting some formula. Propositions in rebuttals and defeating counterarguments must also be considered. Recall for example that defeating counterarguments are supersets of minimal supporting arguments.

Minimality still has a role to play however. Now we are interested in minimal counterarguments, as well as minimal supporting arguments. For example, if A is an argument and D is a rebuttal of A, then intuitively a proposition ϕ should be an issue *because of D* only if $D - \{\phi\}$ is not a rebuttal of A.

There is another complication to be considered. Suppose that $\{a, b\}$ is the only argument supporting c. Both a and b have been denied. In a defense to the denial of a, a supporting argument $\{d, e\}$ has been made. Both of these claims have also been denied. No counterarguments have been made. What are the issues concerning c? Using the theory of issues in [Gordon 1991], the answer would be just $\{a, b\}$, as these are the only members of minimal supporting arguments of c. In the context of our discourse model of argumentation, this seems intuitively wrong. The formulas of the argument supporting a surely should also be issues, to allow the proponent an opportunity to defend them. These considerations lead to a new definition of issue.

The issues of a claim depend on arguments in the dialectical graph for the claim. The exact structure of the dialectical graph is unimportant. The function *join* constructs the union of all the arguments in the dialectical graph for a claim by iterating over the successive layers of the graph, starting with the minimal supporting arguments of the claim.

DEFINITION 22 (Join). The function $\texttt{succ}: 2^A \rightarrow [2^A]$ generates the *successor layers* of a set of supporting arguments. Let S be a set of supporting arguments $\{A_1, \ldots, A_n\}$. If S is empty then $\texttt{succ}(S)$ is the empty sequence. Otherwise, let R be the union of the rebuttals of each argument in S,

$$R = \bigcup_1^n \texttt{rebuttals}(A_i)$$

and D be the union of the defeaters of each rebuttal

$$D = \bigcup_1^n \texttt{defeaters}(R_i)$$

Then, $\texttt{succ}(S) = R, D, \texttt{succ}(D)$.

Let append denote sequence catenation. Now $\texttt{join}(\phi) = \bigcup_1^n \Delta_i$, where

$$[\Delta_1, \ldots, \Delta_n] = \texttt{append}\ (\texttt{supports}(\phi),\ \texttt{succ}\ (\texttt{supports}(\phi))).$$

[4] I had thought defeasible legal reasoning could be modelled using abduction. However, it is now clear that the kind of defeasible reasoning possible with abduction is too limited for modeling the variety of exceptions found in Article Nine.

A formula ψ is an issue with respect to some claim ϕ if and only if ψ is a claim which is not known to be derivable from Θ, the context of conceded formulas, and is a member of the dialectical graph for ϕ or, recursively, an issue with respect to some formula in the graph.

DEFINITION 23 (Issues). A formula ψ is an *issue* relative to a *goal* formula ϕ, denoted `issue` (ψ, ϕ), if and only if
1. ψ is a claim,
2. $\neg$`known` $_{\langle A,\Theta\rangle}(\emptyset, \psi)$, and
   a. $\psi = \phi$,
   b. $\psi \in$ `join`(ϕ), or
   c. $\exists\, \rho \in$ `join`(ϕ). `issue` (ψ, ρ).

"One small blemish" [Gordon 1991, p. 106] of my former theory is corrected here. There, rebuttals did not raise issues. Only the propositions of arguments not known to be inconsistent were issues, and rebuttals are, by definition, inconsistent. The new definition checks whether a formula is a member or any rebuttals, as well as supporting arguments.

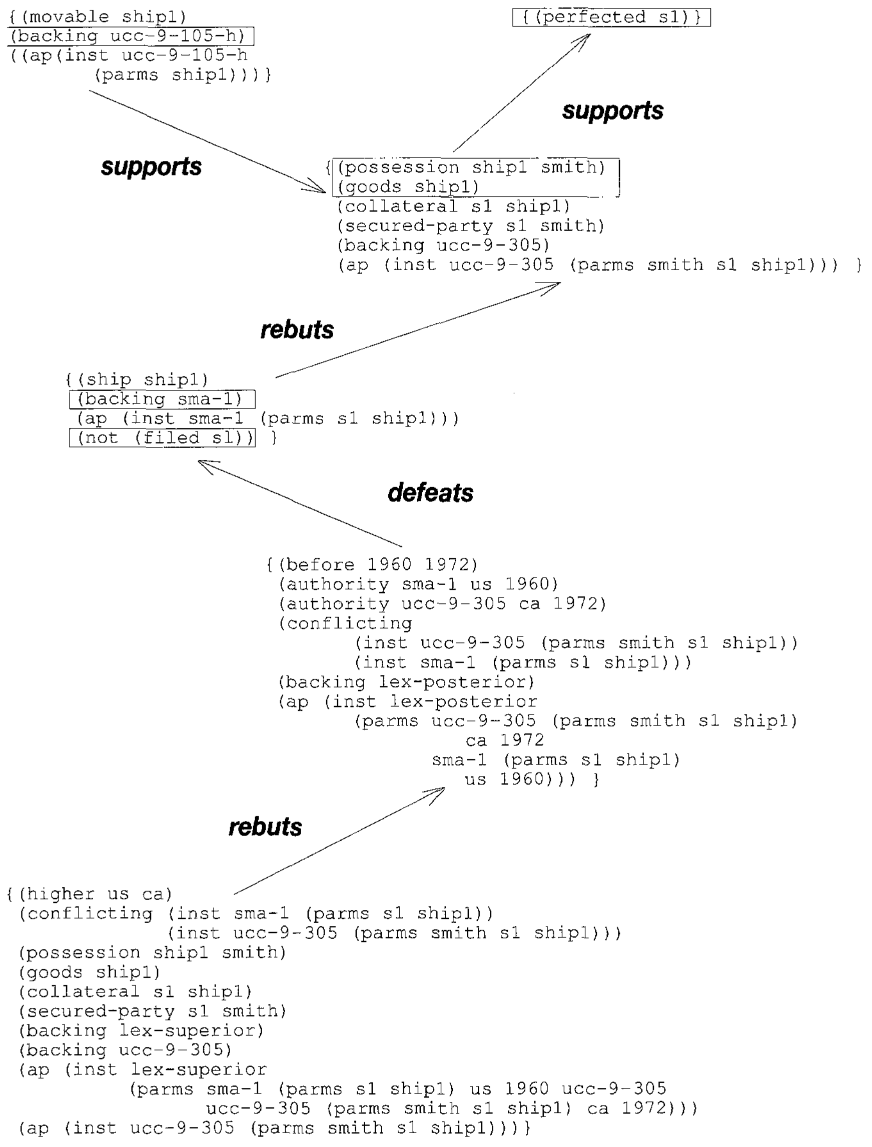
In Figure 3, the issues regarding `(perfected s1)` have been highlighted. The figure shows the relevant portion of the dialectical graph for `(perfected s1)`, as well as the graph for one of its issues, `(goods ship1)`. Notice that the question of whether or not §9–105(h) is authority for the proposition that movable things are goods is an issue relative to `(perfected s1)` because it is an issue in the dialectical graph for `(goods ship1)`. All of the other formulas in these graphs are not issues, either because they are known to be a consequence of the context (i.e., the conceded formulas), or because they are applicability assumptions.

In the model of the trial, the court may decide an issue only if it is *active*. This restriction is designed to further two goals:

1. To minimize costs and avoid the decision of unnecessary legal issues, those issues should be decided first which have the most potential of making other issues moot. This can be achieved by requiring the issues of supporting arguments to be decided before those of rebuttals. If an issue of the supporting argument is decided against its proponent, the rebuttal becomes moot. Thus, the issues of higher levels in the dialectic graph should be considered before those of lower levels.

2. The court should be obligated to address relevant arguments brought forward by the parties. Thus, if an argument Δ for an issue ϕ has been made, the issues of Δ should be decided before ϕ may be decided directly. However, once the issues of Δ are decided, ϕ may cease to be an issue and no longer require a decision.

To understand how both of these goals can be achieved, we need to first define the *leaves* of an argument. Given a formula and the `supports` function, an *and/or graph* of formulas can be generated. (If the formula has no supporting arguments, then the only node in the graph will be the root node, for the formula itself.) Notice that these and/or graphs are orthogonal to the dialectical graph of arguments concerning a formula. Each formula in the dialectical graph has its own and/or graph. A *leaf* of such an and/or graph is a node which has no successors in the graph. The leaves of an argument are the union of the leaves of the and/or graphs for each formula in the argument.



```
{(movable ship1)                                        {(perfected s1)}
(backing ucc-9-105-h)
((ap(inst ucc-9-105-h
       (parms ship1)))}

                                           {(possession ship1 smith)
                                            (goods ship1)
                                            (collateral s1 ship1)
                                            (secured-party s1 smith)
                                            (backing ucc-9-305)
                                            (ap (inst ucc-9-305 (parms smith s1 ship1))) }

    {(ship ship1)
     (backing sma-1)
     (ap (inst sma-1 (parms s1 ship1)))
     (not (filed s1)) }

                               {(before 1960 1972)
                                (authority sma-1 us 1960)
                                (authority ucc-9-305 ca 1972)
                                (conflicting
                                      (inst ucc-9-305 (parms smith s1 ship1))
                                      (inst sma-1 (parms s1 ship1)))
                                (backing lex-posterior)
                                (ap (inst lex-posterior
                                      (parms ucc-9-305 (parms smith s1 ship1)
                                               ca 1972
                                            sma-1 (parms s1 ship1)
                                               us 1960))) }

{(higher us ca)
 (conflicting (inst sma-1 (parms s1 ship1))
              (inst ucc-9-305 (parms smith s1 ship1)))
 (possession ship1 smith)
 (goods ship1)
 (collateral s1 ship1)
 (secured-party s1 smith)
 (backing lex-superior)
 (backing ucc-9-305)
 (ap (inst lex-superior
          (parms sma-1 (parms s1 ship1) us 1960 ucc-9-305
                 ucc-9-305 (parms smith s1 ship1) ca 1972)))
 (ap (inst ucc-9-305 (parms smith s1 ship1)))}
```

Fig. 3. The issues regarding `(perfected s1)`.

DEFINITION 24 (Leaves of an Argument). Let $\{\phi_1, \ldots, \phi_n\}$ be an argument. The `leaves`: $2^L \rightarrow 2^L$ of an argument are defined recursively as follows:

$$\texttt{leaves}(\{\phi_1, \ldots, \phi_n\}) = \bigcup_1^n \texttt{l}(\phi_i)$$

where the auxiliary function `l` is defined as follows.

If `supports` $(\phi) = \emptyset$ then `l` $(\phi) = \{\phi\}$. Otherwise let $\{S_1, \ldots, S_n\}$ = `supports`(ϕ) in

$$\texttt{l}(\{S_1, \ldots, S_n\}) = \bigcup_1^n \texttt{leaves}(S_i).$$

The active issues of a *layer* of a dialectical graph of arguments are the claims of the union of the leaves of all arguments of that layer. The active issues of the *trial proceedings* are the active issues of the first layer with active issues of the complete dialectical graph for the main claim.

DEFINITION 25 (Active Issues). Let `first`: $2^A \rightarrow A \times [2^A] \rightarrow A$ be a function which iterates over the 2^A sequence looking for the first set of arguments for which the $2^A \rightarrow A$ function maps to a non-empty argument. If one is found, then `first` returns this argument, otherwise the empty argument is returned. Let `active`: $2^A \rightarrow 2^L$ be the following function. If L is a layer $\{\Gamma_1, \ldots, \Gamma_n\}$ of a dialectical graph of arguments, then `active` (L) is the set of *issues* in

$$\bigcup_1^n \texttt{leaves}(\Gamma_i).$$

Now, let ϕ be the main claim of the trial proceedings, S = `supports` (ϕ) and

$$[L_1, \ldots, L_n] = \texttt{append}\ (S, \texttt{succ}\ (S)).$$

The *active issues* of the proceedings are

$$\texttt{first}\ (\texttt{active}, [L_1, \ldots, L_n]).$$

To return to our example, Figure 4 shows the and/or graph for the arguments of the first layer of the dialectical graph for the main claim `(perfected s1)`, which was shown above in Figure 3. The issues of this and/or graph, which happens to be a tree, are highlighted. (All of the other formulas have been conceded or are applicability assumptions.) As the graph does contain issues, they are the only issues which are active at the beginning of trial, and must be decided first.

One of these issues is a legal question, `(backing ucc-9-105-h)`, and the other is a question of fact, `(possession ship1 smith)`. In this model, issues of fact and law are distinguished syntactically. The only legal issues are backing claims. The question of possession could have involved legal issues but the parties didn't make arguments raising such issues, so it remains a question of fact in this case. The Trial Game does not impose an order on legal and factual issues. In a more elaborate model, presumably factual issues should be tried first, perhaps by a jury, to avoid deciding legal questions unnecessarily. In our example, if the trier of fact decides that the plaintiff does not have possession of the ship, then all other issues become moot and the plaintiff loses.

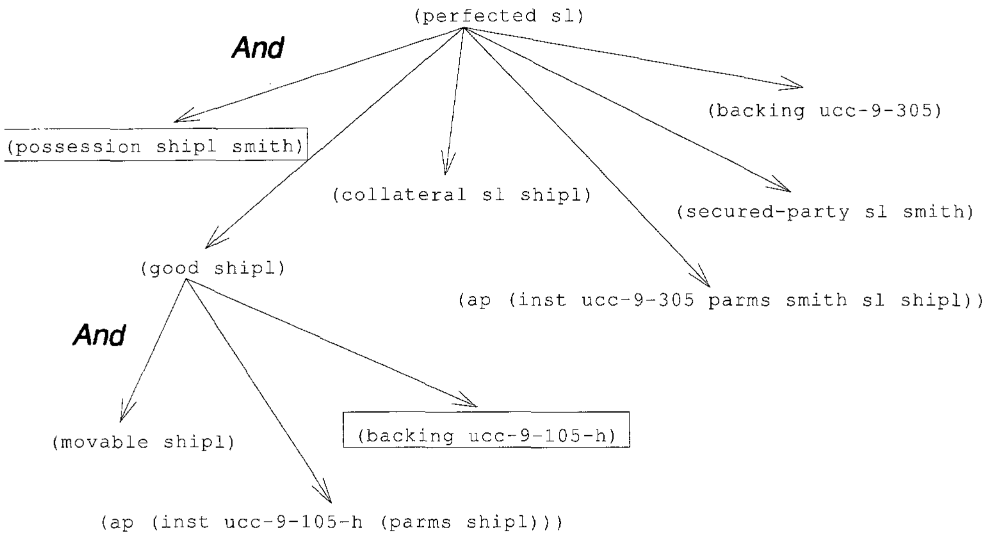


Fig. 4. Active Issues at the Beginning of the Trial Game.

9. 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 [1988]. 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 [1992]. 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 ad Pearl use an ATMS, with its exponential worst-case complexity. The implementation here uses my Minimal Reason Maintenance System [Gordon 1993], 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.

10. Related Work

In addition to Lorenzen's system [Felscher 1986], there have been several other dialogue logics, by Lorenz [1961], Rescher [1977], and Mackenzie [1979]. 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 [Marshall 1989] and [Schuler 1990], 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 [Allen 1982]. This is an early example of a resource allocation game which divides up the burden of proof among the players. Moves of the 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 [Bench-Capon 1991], the other on Toulmin's theory [Bench-Capon 1992b]. Ronald Loui and William Chen have designed an argument game using Loui's LMNOP nonmonotonic logic [Loui 1992].

The AI models of case-based legal argumentation, such as [Branting 1989, Ashley 1990, Skalak 1992], may be viewed as cognitive models of the reasoning processes of competent attorneys. In contrast, the Pleadings Game is a model of discourse norms.

For alternative logic-based approaches to modelling defeasible legal reasoning, one should read Henry Prakken's monograph [1993] and several of the papers in the proceedings of the Fourth International Conference on Artificial Intelligence in Law, especially those by Prakken [1994], Nitta [1994], Sartor [1994] and Loui [1994]. Finally, see also [Hage 1992], which includes another model of two-party, adversarial argumentation with its roots in Alexy's discourse theory for legal argumentation. All of the works mentioned in this paragraph were developed at about the same time as my Pleadings Game model: it remains for future work to compare their relative merits.

11. 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 [Bayles 1990, p. 5], Michael Bayles explains John Rawls' distinction between pure, perfect and imperfect justice [Rawls 1971]. 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 in 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.[5] 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.

Acknowledgements

Many thanks to my thesis advisors, Wolfgang Bibel and L. Thorne McCarty, to Barbara Becker, Gerhard Brewka, Joachim Hertzberg, Christoph Lischka, Lothar Philipps and Franco di Primio, who read and provided helpful comments on various parts of the thesis, and to Trevor Bench-Capon and Ronald Loui, for pointing out significant related work.

References

Allen, L. E. 1982. 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: German National Research Center for Computer Science (GMD).

Alexy, R. 1989. *A Theory of Legal Argumentation.* Oxford: Claredon Press.

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

Bayles, M. D. 1990. *Procedural Justice; Allocating to Individuals.* Kluwer Academic Publishers.

Bench-Capon, T. J. M., Dunne, P. E. S. & Leng, P. H. 1991. Interacting with Knowledge Systems Through Dialogue Games. In Proceedings of *AVIGNON*-92, vol. 1, 123–130. Avignon.

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

Bench-Capon, T. J. M., Dunne, P. E. S. & Leng, P. H. 1992b. A Dialogue Game for Dialectical Interaction with Expert Systems. In Proceedings of *The 11th Annual Conference on Expert Systems and their Applications*, vol. 1. Avignon.

Black, H. C. 1979. *Black's Law Dictionary.* West Publishing Company.

Branting, L. K. 1989. Representing and Reusing Explanations and Legal Precedents. In Proceedings of *The Second International Conference on Artificial Intelligence and Law*, 103–110. Vancouver: Association for Computing Machinery.

[5] This idea is also expressed in [Bench-Capon 1992].

Charniak, E. & McDermott, D. 1985. *Introduction to Artificial Intelligence*. World Student Series. Reading, Massachusetts: Addison-Wesley.

Dowling, W. F. & Gallier, J. H. 1984. Linear-Time Algorithms for Testing the Satisfiability of Propositional Horn Formulae. *Journal of Logic Programming* 3: 267–284.

Edwards, P. editor. 1972. *The Encyclopedia of Philosophy*, volume 1, Macmillan Pub. Co., Inc. & The Free Press.

Eshghi, K. & Kowalski, R. A. 1988. *Abduction as Deduction*. Technical report, Dept. of Computing, Imperial College of Science and Technology, London.

Felscher, W. 1986. Dialogues as a Foundation for Intuitionistic Logic. In *Handbook of Philosophical Logic, Vol. III: Alternatives to Classical Logic*. eds. D. Gabbay & F. Günther, 341–372. D. Reidel.

Fikes, R. E. & Nillson, N. J. 1971. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligense* 2: 189–208.

Geffner, H. & Pearl, J. 1992. Conditional Entailment: Bridging Two Approaches to Default Reasoning. *Artificial Intelligence* 53(2–3): 209–244.

Gordon, T. F. 1986. *The Role of Exceptions in Models of the Law*. Formalisierung im Recht und Ansätze juristischer Expertensysteme, eds. R. Traunmüller & H. Fiedler, 52–59. Munich: J. Schweitzer Verlag.

Gordon, T. F. 1989. Issue Spotting in a System for Searching Interpretation Spaces. In Proceedings of *The Second International Conference on Artificial Intelligence and Law*, 157–164. Vancouver: Association for Computing Machiery.

Gordon, T. F. 1991. An Abductive Theory of Legal Issues. *International Journal of Man-Machine Studies* 35: 95–118.

Gordon, T. F. 1993. *The Pleadings Game: An Artificial Intelligence Model of Procedural Justice*. Ph.D. diss., Fachbereich Informatik, Technische Hochschule Darmstadt.

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

Junker, U. 1992. *Relationship Between Assumptions*. Ph.D. diss., Kaiserslautern.

Konolige, K. 1990. A General Theory of Abduction. In *Working Notes of the AAAI Spring Symposium on Automated Abduction*, 62–66.

Levesque, H. J. 1989. A knowledge-Level Account of Abduction. In Proceedings of *The International Joint Conference on Artificial Intelligence*, 1061–1067. Detroit.

Lifschitz, V. 1987. On the Semantics of STRIPS. In *Reasoning about Action and Plans*, eds. M. Georgeff and A. Lansky, Morgan Kaufmann.

Lorenz, K. 1961. *Arithmetik und Logik als Spiele*. Ph.D. diss., Kiel.

Loui, R. & Chen, W. 1992. An Argument Game. Technical Report WUCS-92-47, Dept. of Computer Science, Washington University.

Loui, R., Norman, J., Olson, J. & Merrill, A. 1994. A Design for Reasoning with Policies, Precedents, and Rationales. In Proceedings of *The Fourth International Conference on Artificial Intelligence and Law*, 202–211. Amsterdam: Association for Computing Machinery.

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

Marshall, C. C. 1989. Representing the Structure of a Legal Argument. In Proceedings of the *Second International Conference on Artificial Intelligence and Law*, 121–127. Vancouver: Association for Computing Machinery.

McCarty, L. T. 1988. Clausal Intuitionistic Logic, II. Tableau Proof Procedures. *The Journal of Logic Programming*, 5: 93–132.

McCarty, L. T. & Cohen, W. W. 1992. *The Case for Explicit Exceptions*.

Nitta, K., Wong, S. & Ohtake, Y. 1994. A Computational Model of Trial Reasoning. In Proceedings of *The Fourth International Conference on Artificial Intelligence and Law*, 20–29. Amsterdam: Association for Computing Machinery.

Poole, D. 1985. On the Comparison of Theories: Preferring the Most Specific Explanation. In Proceedings of *The International Joint Conference on Artificial Intelligence*, 144–147, Los Angeles.

Poole, D. 1990. Hypo-deductive Reasoning for Abduction, Default Reasoning and Design. In Working Notes of the *AAAI Spring Symposium on Automated Abduction*, 106–110.

Prakken, H. 1993. *Logical Tools For Modelling Legal Argument*. In Proceedings of *The Fourth International Conference on Artificial Intelligence and Law*, 1–9. Amsterdam: Association for Computing Machinery.

Rawls, J. 1971. *A Theory of Justice*. Harvard University Press.

Rescher, N. 1977. *Dialectics*. State University of New York, Albany.

Reiter, R. & de Kleer, J. 1987. Foundations of Assumption-Based Truth Maintenance Systems, Preliminary Report. In Proceedings of *Sixth National Conference on Artificial Intelligence*, 183–188.

Sartor, G. 1994. A Simple Computational Model for Non-Monotonic and Adversarial Legal Reasoning. In Proceedings of *The Fourth International Conference on Artificial Intelligence and Law*, 192–201. Amsterdam: Association for Computing Machinery.

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

Selman, B. 1990. Computing Explanations. In Working Notes of the *AAAI Spring Symposium on Automated Abduction*, 82–84.

Selman, B. & Levesque, H. J. 1990. Abductive and Default Reasoning: A Computational Core. In Proceedings of the *Eighth National Conference on Artificial Intelligence*, 343–348.

Skalak, D. B. & Rissland, E. L. 1992. Arguments and Cases: An Inenvitable Intertwining. *Artificial Intelligence and Law* 1(1): 3–45.

Susskind, R. E. 1987. *Expert Systems in Law*. Oxford.

Toulmin, S. E. 1958. *The Uses of Argument*. Cambridge University Press.