draft – 2010-06-30

Using Defeasible Logic Programming for Argumentation-Based Decision Support in Private Law

C. BEIERLE [a], B. FREUND [a], G. KERN-ISBERNER [b], M. THIMM [b]
[a] *FernUniversität in Hagen, Germany*
[b] *Technische Universität Dortmund, Germany*

Abstract. Legal reasoning is one of the most obvious application areas for computational models of argumentation as the exchange of arguments and counterarguments is the established means for making decisions in law. In this paper we employ Defeasible Logic Programming (DeLP) for representing legal cases and for giving decision-support, exemplary for private law. We give a formalization of legal provisions that can be used easily by judges for supporting their decision process and present a working system that resembles the decision-making in legal reasoning, in particular, with respect to the *burden of proof*.

1. Introduction

Although AI and Law is a long-established discipline (see [3] for an overview of past developments and present problems), the number of expert systems in actual use by judges—if any—is very low. Instead, legal professionals use computers for writing, communication, and as databases substituting their traditional libraries. There are only few tools to support the decision-making itself. This is especially remarkable since legal reasoning is essentially rule-based, and therefore seems to invite automation. The low motivation of judges for using expert systems can partly be explained by a certain conservatism that characterizes the legal system. But the reluctance to embrace counsel based on computation runs deeper than tradition and nostalgia. To rely on an expert system, a judge—more than any other kind of expert—needs to have not only a basic, but a profound understanding of the formal model and inference mechanism the system is based on. Although new forms of logic and models of argumentation have been studied in legal theory as well (e. g. [14,15,8,19,3]), they never really affected the curriculum. Classical legal theory, based on aristotelian-scholastic reasoning, is still all the logical training judges routinely get. It forms the core of most textbooks on legal method and lies at the heart of legal reasoning as it is taught at law schools both in common law [2] (US) and civil law countries [11] (China), [4] (Germany), [13] (Switzerland). It is therefore the natural key to the comprehension and acceptance of expert systems by judges.

In this paper, we will focus on these general principles that govern judicial legal reasoning, and elaborate on its argumentative inference structures, in particular with respect to the *burden of proof*. We will show that defeasible logic programming (DeLP) [5] suits the judicial way of reasoning especially well, and present an argumentative system for decision support in private law that has been implemented based on DeLP. The expert system introduced here, LiZ, focuses on this perspective of a judge, and thus is to be taken with a grain of salt when compared to the more sophisticated models recently used in AI and law which have been mentioned above. In contrast to these, it does not strive to model externally the way legal experts actually think (or argue), but it takes the internal position of a judge. Moreover, by using DeLP to implement the model, it becomes possible to add defeasible reasoning at the level of subsuming (or classifying) cases while at the same time respecting the classical structure of the law. The system is not confined to material law rules, but takes into account the procedural aspects of the law paramount for the work of a judge, particularly, the burden of proof, necessary indications to the parties, intermediary decisions to manage the process and so on. However, we abstract as far as possible from idiosyncratic aspects of specific jurisdictions. In line with current research in comparative law [20] we focus on a common core of legal procedure, ensuring broader applicability of the formal model. In general, various kinds of burden of proof can be distinguished; for a recent and thorough analysis and survey see [7].

In Sec. 2 we give a brief introduction to legal reasoning in private law and afterwards extract and compare its essential aspects with methods for defeasible argumentation. In Sec. 3 we establish a model of legal reasoning by exemplifying some legal concepts, and further refine our model by formalizing knowledge representation in Sec. 4. Section 5 recalls the relevant technical aspects of DeLP, and Sec. 6 presents our approach of an argumentation-based decision-support system using DeLP. In Sec. 7 we conclude.

2. Legal reasoning and defeasible argumentation

Legal reasoning is rule-based. This holds true for all legal systems and all areas of law. Moreover, the structure of the rules, whether contained in judgements, statutes, legal literature or even in practitioners' minds, is more ore less universal. A simple private law case, as it might appear in a student textbook, shall help to illustrate this structure.

Example 1. Consider the following scenario: *Bobby Buyer (B) has contracted with Sally Seller (S), the local car dealer, for the delivery of a brand new car. However, at the agreed date, S does not deliver. Can B demand delivery of the car from S?*

To begin with, we need a rule to decide the case. Let's assume B and S live in land where the *UNIDROIT Principles* [9] apply. Then our case would be governed by the following article (Art. 7.2.2 of the UNIDROIT Principles):

> *"Where a party who owes an obligation other than one to pay money does not perform, the other party* ***may require performance****, unless*
>
> *(a) performance is impossible in law or in fact;*
> *(b) performance or, where relevant, enforcement is unreasonably burdensome or expensive;*
> *(c) the party entitled to performance may reasonably obtain performance from another source;*
> *(d) performance is of an exclusively personal character; or*
> *(e) the party entitled to performance does not require performance within a reasonably time after it has, or ought to have, become aware of the non-performance."*

Thanks to the clear-cut way in which this provision is designed, it is rather simple to extract the rule incorporated in it, or, more precisely, the conditions and the consequence of the rule. In a semi-formal notation, one might put it like this:

```
[ obligation(X,Y,O) and                  // meaning "X owes O to Y"
not_money(O) and                         // true if O isn't a sum of money
no_performance(X,Y,O) and                // X hasn't delivered O to Y
not (performance_impossible(X,O) or      // X cannot deliver O
  performance_too_burdensome(X,O) or
  alternative_source_available(O) or
  personal_obligation(X,Y,O) or
  time_limit_exceeded(X,Y,O) ) ]         // Y didn't bring his claim in time
⇝ right_to_performance(Y,X,O)            // Y can demand performance (i. e., delivery)
```

To answer the question of the case, one would aim at gathering information about all atoms mentioned in the body of the rule and then derive formally whether the formula specified in the head holds.

However, it should be emphasized that classical logic is not an appropriate framework for processing rules as the one given in Ex. 1, for two reasons: First, legal rules have usually been amended, over time, by exceptions and counter-exceptions. Therefore, legal rules are not strict, but *defeasible*, as we can rarely rely on them not having exceptions (indicated by *unless* in Art. 7.2.2.) [17,6]. Second, there is the nature of the situation in which legal cases typically arise. Usually there are two or more parties involved, one bringing a claim and the other one defending against it. Since both of them have their stakes in it, they can hardly be expected to be objective in their description of the situation. Yet they are often the only ones with first-hand knowledge about the relevant facts, and thus are the natural starting point when collecting these facts. This raises the question of how to reconcile the two conflicting views presented by the parties to the judge. Since a judge *must* ultimately find a decision, the conflict has to be solved. A way of dealing with contradictory and missing information has to be found, as it is not possible to send the plaintiff home with a mere "*sorry, I don't know*". So, legal reasoning cannot be *logical* (or else would trivialize, or run into blazing contradictions), but it has to be *rational*, and, what is particularly important in this domain, *justifiable*. The question is what consequences may correctly be inferred from a set of defeasible, possibly contradicting legal rules. Legal theory provides some guidance, as it has developed requirements of formal or procedural justice. These requirements will be mimicked by restrictions for inference mechanisms applicable to our model of law. All inference mechanisms that observe these restrictions—and the results that can be inferred by applying them—will be called *justifiable*. The goal is not to completely determine the inference, but to set out a frame within which several inference mechanisms can exist. We assume the following restrictions to form the basic requirements for justifiability:

Strictness Strict legal rules—like e. g. the first article of the basic law in Germany: “The dignity of a human being is inviolable.”—are to be treated as in classical logic (accordingly, a knowledge base containing contradicting *strict* rules is corrupt).

Specificity Exceptions to a rule have a higher priority than the rule itself, as the idea is that an exception, if its conditions are met, defeats the rule.

Equality Equal cases are to be treated equally.

The last requirement, “Equality”, means that the inference mechanism is not randomized, so that using the same knowledge base and entering the same facts repeatedly yields the same result. This, of course, is only one of the implications of “Equality”. The concept goes further in theory, but insofar cannot be properly formalized here. The reason is that the expert system cannot decide whether two real-life situations are “alike”. It only asks for the relevant facts, which are then to be entered by the user. In other words the inference mechanism guarantees formal equality (same input, same output), while the user has to guarantee material equality (similar situations, same input).

Beyond justifiability, there are some standard conflict rules [21]. The following three, given in their original latin form, date back to medieval times, when the *ius commune* governed continental Europe, and still serve as important guidelines for the administration of justice:

1. Lex specialis derogat legi generali. (*The special rule defeats the general rule.*)
2. Lex superior derogat legi inferiori. (*The superior rule defeats the inferior rule.*)
3. Lex posterior derogat legi priori. (*The younger rule defeats the older rule.*)

The idea of the first conflict rule is that a rule having additional conditions is based on more information and therefore preferable to one based on less information. This is meant by a “special” rule, which applies only to a part of the situations where the “general” rule applies, because it has additional preconditions. The second conflict rule makes use of a hierarchy of laws, where e. g. the Constitution is at the top. The rationale behind the third conflict rule is similar to that of the first one: a legal rule that has been created at a later stage, for example by an act of parliament, is based on more information because the lawgiver of the future knows more than the lawgiver of the past.

There are many more formal (and, of course, material) criteria for justice proposed in legal theory, but none of them have received universal acceptance. Taking the aspects described above as a specification of minimal requirements for a justifiable inference mechanism, defeasible argumentation appears to be an optimal framework for a system that is able to support judicial decisions. The standard version of DeLP [5], which we used for the LiZ system, is an adequate inference engine for defeasible argumentation. It turns out that the additional assumptions made in DeLP to add plausibility to the results inferred (like “Generalized Specificity” as a comparision criterion for arguments and the “concordance” of argumentation lines) are remarkably similar to the standard conflict rules used in legal theory. We will come back to this issue later, but first we go on with a general formalization of the legal concepts of “burden of proof” and “legal trees”.

3. Legal trees and the burden of proof

Gathering information on a legal case is not always simple. Considering again the rule formalized in Ex. 1, already the first condition *obligation*(X,Y,O) is problematic. Though it seems clear that S has promised to B to deliver the car and is thereby obliged to do so, so that *obligation*(S,B,CAR) should be true, an obligation per definition belongs to the *legal sphere*. It is not tangible or measurable as a real-life object, there is no physical detector for it. Instead, we need another legal rule to tell us when an obligation arises. In fact, we find such a rule in the UNIDROIT Principles [9, Art. 1.3], a specialized version of which would contain the following rules (again in semi-formal notation):

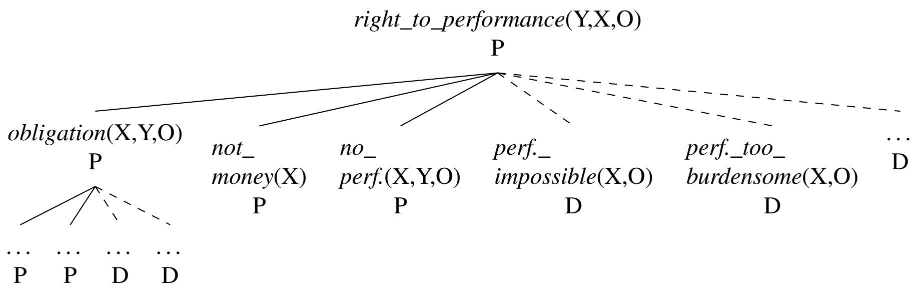

Figure 1. Part of the *legal tree* for the claim *right_to_performance*(Y,X,O). The solid lines correspond to *if*, a dashed line to *unless*. The labels P (plaintiff) and D (defendant) indicate the *burden of proof*.

(1) *sales_contract*(X,Y,PRICE,O) $\rightsquigarrow$ *obligation*(X,Y,money) // the buyer's obligation
(2) *sales_contract*(X,Y,PRICE,O) $\rightsquigarrow$ *obligation*(Y,X,O) // the seller's obligation

In applying Art. 7.2.2 to the example, we would take recourse to the second one of the above rules to determine whether a non-monetary obligation between B and S exists. We will call rules which are invoked in this way (i. e., to explain, define or otherwise determine the conditions of another rule) *secondary rules*, as opposed to the *primary rules* that directly raise a legal claim as their consequence (in this case: "*Can B demand delivery of the car?*", which is answered by Art. 7.2.2).

The secondary rule tells us that if the condition *sales_contract*(B,S,PRICE,O) is met, we can derive *obligation*(S,B,O). The question whether such a contract exists is almost a matter of fact, since we would expect that the sales contract exists as a real-life object. Therefore, in breaking down our initial question "*Can B demand delivery?*", we have reached a rather basic level were the legal question is turned into tiny factual bits of information we can provide easily. Of course, this is only the ideal (see [12] for an account of the age-old criticism of this method and [10] for a recent investigation of the relationship between legal argumentation and language). Often questions remain which are not answered so easily (like "*is the enforcement unreasonably burdensome?*", see Art. 7.2.2 lit. b), and would provide even greater difficulty if they should be answered by artificial intelligence. Yet the method just described has proven workable and is practiced by judges and lawyers around the globe. Due to the complexity of the law itself, this method carries far enough potential for computer-assistance even without the possibility to automatically find the basic factual information finally needed to decide the case.

Breaking down a rule by using secondary rules leads to a tree structure, as Fig. 1 illustrates. The conditions of the primary rule are "explained" by secondary rules, the conditions of which may in turn be explained by further secondary rules, and so on. This goes on until there are no more rules available for any of the conditions that have not already been explained. It should then be possible to determine these conditions—represented by the leafs of the tree—with relative ease by collecting appropriate "real-life" evidence. These conditions pose the decisive questions of the case. Answering them means to collect the "relevant facts", which then allows deciding on the secondary rules step by step—moving from the leafs to the root of the tree—and finally deciding whether the primary rule applies to the given case or not.

Collecting "relevant facts" imposes the next issue that has to be dealt with in order to integrate all information for a given case. As discussed before, usually, there are multiple parties involved in legal reasoning and the question at hand is, what to do when these parties provide contradictory or incomplete information? Or more general, which of the parties is responsible in bringing forward essential information needed to solve a given case in an adequate manner? Courts all over the world have developed more or less the same solution to this problem. The approach is rather simple. First, a kind of *assumption* is created for every single one of the relevant facts. If the parties cannot help to clarify the fact in question, the fact is either assumed to be given or absent. This provides a way of dealing with incomplete information. Second, if the parties do say something but contradict each other, evidence is collected from both sides. If the evidence suffices to establish the fact (or its absence), this result becomes part of the basis for the decision. If however the evidence remains inconclusive, again recourse is taken to the assumption created earlier to circumvent the contradiction presented by the parties.

The legal term associated with the creation of the assumption is *burden of proof* (see [16], [18], or [7] for an AI and Law perspective on this subject, and e. g. [1] for an analysis from within legal theory). Saying that a party has the burden to prove fact “A” means that the assumption for this fact is “NOT A”. Saying a party has the burden to prove “NOT A” means that the assumption is “A”. Logically it is not important which of the parties carries the burden, but which content the burden has (to prove “A” or to prove “NOT A”). However, legal experts are not used to talking about the content at all. They don’t have to. For a human being, the content of the burden can easily be derived from the knowledge of who carries it. A party would only try to either prove a fact or to prove the absence of it, depending on what suits the party best. When the plaintiff likes to prove "A", the defendant will always like to prove “NOT A”, and both will be clear from the context. Therefore, knowing who carries the burden of proof regarding a certain fact is equivalent to knowing the content of the burden, and thus the corresponding assumption. In anglo-american terminology, this relationship is depicted very clearly: when one party has the burden of proof, the other one is said to have the *benefit of assumption* (i. e., it benefits if the assumption is resorted to).

Naturally, jurists have developed rules to determine the burden of proof. The general rule is that a party carries the burden of proof for those facts that, if given, would favor the party, i. e. facts that would either (a) support the party’s claim or (b) support its defense against the other party’s claim. However, this general rule is sometimes tricky to apply. The question it raises is whether a certain fact is the negative condition of a claim or rather the positive condition of a defense to it (or vice versa), which may be hard to say since the wording of the law can be ambivalent. From the perspective of classical logic, presupposing an omniscient observer, it wouldn’t make a difference. Yet, in a court scenario with the possibilities of contradictory and incomplete information, it makes a difference in determining the assumptions and therefore in deciding most real cases. To illustrate this, we will use the concept of an exception to a rule. An exception to a rule can be derived from the rule by adding further conditions and negating the consequence. For example, if [A **and** B **and** C] $\rightsquigarrow$ Z is a rule, then [A **and** B **and** C **and** D] $\rightsquigarrow$ NOT Z is an exception.

Back to the problem of the burden of proof we find that exceptions usually constitute the basis for a defense against a claim, whereas exceptions to exceptions in turn support the claim, and so on; this kind of proof burden corresponds to the *burden of production* in [7]. That means, when the plaintiff claims Z under the rule above, he has the burden of proof for A, B, and C. In turn, the burden of proof for D lies with the defendant, as it only occurs as part of the exception which naturally only the defendant would like to support. The assumptions would be “NOT A”, “NOT B”, “NOT C” and “NOT D”. If, however, we dropped the exception and modified the strict rule by adding the additional literal from the exception in negated form as a condition, we would get the single (strict) rule: [A **and** B **and** C **and** (**not** D)] $\rightsquigarrow$ Z.

In this scenario, the plaintiff claiming Z would have to prove A, B, C and also "NOT D", as all these are conditions for the rule supporting his claim. The resulting assumptions would be “NOT A”, “NOT B”, “NOT C” and "D", so in contrast to the first scenario, “D” would be assumed. If the plaintiff can show A, B, and C, but the parties disagree on D, and neither of them can provide sufficient evidence, then in the latter scenario the plaintiff loses the case (for failing to establish the conditions of the rule supporting the claim), while winning in the first scenario (where the defendant fails to establish the conditions of the exception). Generally, the absence of a fact is often hard to show, which is why negated conditions are kept to a minimum in legal rules (although they do exist). Instead they are put into exceptions, where they occur in positive form.

Until now, we have only applied the concept of burden of proof to primary rules, but it can be elegantly extended to the whole *legal tree* introduced in Fig. 1. To do so, we start at the root of the tree and mark the root and the conditions of the primary rule with a “P” for plaintiff, and the conditions of exceptions to the primary rule with a “D” for defendant. Then we work our way down the tree: In the illustration given in Fig. 1, this means that the burden of proof is inherited from the parent node along solid lines, but inverted along dashed lines. Thus, not only the relevant facts needed to solve a given case but also the respective burden of proof for each of the facts (and, therefore, the assumptions that follow) can easily and automatically be calculated. The only knowledge necessary is the rules (and their exceptions).

4. Legal Knowledge Modelling and Reasoning

We will now further extend and formalize our model. As a general starting point we will allow rules to be either strict or defeasible, while the latter will be the standard.

Legal provisions In order to represent legal rules in a concise fashion, we will represent exceptions within their corresponding rule. The concept of rules and exceptions correctly reflects the theoretical structure of the law, but at a somewhat low level. For example, a single article from any given statute usually contains more than one rule (and can contain multiple exceptions, cf. Art. 7.2.2). Therefore, a meta-structure comprising all those rules seems convenient. In our model, rules with the same consequence are incorporated in a *legal provision* (cf. Def. 1 below). Grouping rules with the same consequence in this way also helps to deal with alternatives. In accordance to the way legal experts conceive and apply legal rules, alternative conditions, corresponding to a logical disjunction, are removed by introducing extra rules for every alternative. Therefore, in our legal model, the conditions of a rule will always be *cumulative*, i. e. a conjunction of literals.

Subsumption When applying a legal rule, after breaking it down as far as possible to its constituents (i. e. the conditions of the lowest secondary rules in the “legal tree”), there inevitably comes the task of *subsuming* the given case under these constituents. Although there are no more legal rules available to decide whether these conditions are fulfilled, one will usually find further lexical rules to do so. Taking a trivial example, to decide whether an obligation is “non-monetary” (see Art. 7.2.2), “money” could be looked up in a dictionary, which would probably enumerate the forms in which money comes along or its characteristics. Such definitions are nothing else than further rules. Our model therefore allows for *subsumption rules* to be added to the database. They have the same form as legal rules and are treated likewise in the inference process. They are only named differently to make clear that they do not share the same authority due to their non-legal origin.

Open concepts and antagonistic rules As mentioned, legal rules in the knowledge base are usually meant to be *defeasible*, i. e. open to exceptions and refutation. Although modelling law by using defeasible rules is probably against the intuition of many legal experts, it is suitable for the reasons given above. Moreover, defeasible rules can also be used for modelling the arguments that are used in law in dealing with open concepts like "burdensome" (cf. Art. 7.2.2), "reliable", "adequate" etc. For instance, as adequacy is a very broad and fuzzy concept, indicators have been developed for the adequacy and inadequacy of damages to make the handling of the law easier and more foreseeable. Such arguments can be seen as *antagonistic* rules, where the consequence of one rule is the negation of the consequence of the other.

In the knowledge base, not only rules with the same consequence, but also their antagonistic counterparts become part of one single *legal provision*:

Definition 1 (legal provision)**.** A *legal provision* is a triple $P = (c, \mathcal{B}, \overline{\mathcal{B}})$ with $\mathcal{B} = \{B_1, \ldots, B_k\}$ and $\overline{\mathcal{B}} = \{B_{k+1}, \ldots, B_{k+l}\}$ where $k \geq 1$ is the number of legal rules having c as their consequence, and $l \geq 0$ is the number of corresponding antagonistic rules. Each B_i is of the form $B_i = ((b_1, \ldots, b_n), E_1, \ldots, E_m)$ where the b_j are the *conditions* and E_j sequences of *exceptions* for the i-th rule with $n \geq 1$, and $m \geq 0$.

For instance, in an attribute-value pair notation for legal provisions (in LiZ, an XML representation is used), Art. 7.2.2 is given by the following knowledge base entry:

Art. 7.2.2 = (consequence = *right_to_performance*(Y,X,O),
rules = ((conditions = (*obligation*(X,Y,O), *not_money*(X),
no_performance(X,Y,O)),
exception = (*performance_impossible*(X,O)),
exception = (*performance_too_burdensome*(X,O)),...)))

A *legal knowledge base* is a set $KB = \{P_1, \ldots, P_n\}$ of legal provisions. For each claim c, such a knowledge base uniquely determines a legal tree $ltree(c, KB)$ with root c and whose further nodes and arrows are constructed from KB. For the general case, we extended the legal tree construction as illustrated in Fig. 1 accordingly:

- Multiple rules with the same consequence: Outgoing arrows of a node are grouped together if they belong to the same rule.
- Conjunctions of exceptions: The respective arrows are grouped accordingly.
- Antagonistic rules: Introduce *not-if* and *not-unless* arrows.

Note that the computation of the burden of proof within the legal tree carries over smoothly to antagonistic rules: The burden of proof is inverted along *not-if* arrows, but inherited along *not-unless* arrows.

We continue with a brief excursus to some technical details of DeLP that will be used to represent and to reason with the model formalized above.

5. Defeasible Logic Programming

The basic elements of *Defeasible Logic Programming* (DeLP) are facts and rules. A single atom h or a negated atom $\sim h$ (in DeLP negation is denoted by $\sim$) is called a literal or *fact*. Rules are divided into strict rules of the form $h \leftarrow B$ and defeasible rules of the form $h \prec B$ where h is a literal and B is a set of literals. A literal h is *derivable* from a set of facts, strict rules, and defeasible rules X, denoted by$X \mid\!\sim h$, iff it is derivable in the classical rule-based sense treating strict and defeasible rules equally. A set X is *contradictory*, denoted $X \mid\!\sim \perp$, iff both $X \mid\!\sim h$ and $X \mid\!\sim {\sim h}$ holds for some h. A *defeasible logic program* (*de.l.p.*) P is a tuple $P = (\Pi, \Delta)$ with a non-contradictory set of strict rules and facts Π and a set of defeasible rules Δ. Using rules and facts arguments can be constructed as follows.

Definition 2 (Argument, Subargument)**.** Let $h \in \mathcal{L}$ be a literal and let $P = (\Pi, \Delta)$ be a *de.l.p.*. Then $\langle \mathcal{A}, h \rangle$ with $\mathcal{A} \subseteq \Delta$ is an *argument* for h, iff $\Pi \cup \mathcal{A} \mid\!\sim h$, $\Pi \cup \mathcal{A} \not\mid\!\sim \perp$, and $\mathcal{A}$ is minimal with respect to set inclusion. An argument $\langle \mathcal{B}, q \rangle$ is a *subargument* of an argument $\langle \mathcal{A}, h \rangle$, iff $\mathcal{B} \subseteq \mathcal{A}$.

Two literals h and h_1 *disagree* regarding a *de.l.p.* $P = (\Pi, \Delta)$, iff the set $\Pi \cup \{h, h_1\}$ is contradictory. An argument $\langle \mathcal{A}_1, h_1 \rangle$ is a *counterargument* to an argument $\langle \mathcal{A}_2, h_2 \rangle$ at a literal h, iff there is a subargument $\langle \mathcal{A}, h \rangle$ of $\langle \mathcal{A}_2, h_2 \rangle$ such that h and h_1 disagree.

In order to deal with counterarguments, a central aspect of defeasible logic programming is a formal comparison criterion among arguments. Bearing in mind the context of legal reasoning *Generalized Specificity* [5] seems to be the most appropriate choice. According to this criterion an argument is preferred to another argument, iff the former one is more *specific* than the latter, i. e., (informally) iff the former one uses more facts or less rules. For example, $\langle \{c \prec a, b\}, c \rangle$ is more specific than $\langle \{{\sim c} \prec a\}, {\sim c} \rangle$, denoted by $\langle \{c \prec a, b\}, c \rangle \succ \langle \{{\sim c} \prec a\}, {\sim c} \rangle$, cf. [5]. Then an argument $\langle \mathcal{A}_1, h_1 \rangle$ is a *defeater* of an argument $\langle \mathcal{A}_2, h_2 \rangle$, iff there is a subargument $\langle \mathcal{A}, h \rangle$ of $\langle \mathcal{A}_2, h_2 \rangle$ such that $\langle \mathcal{A}_1, h_1 \rangle$ is a counterargument of $\langle \mathcal{A}_2, h_2 \rangle$ at literal h and either $\langle \mathcal{A}_1, h_1 \rangle \succ \langle \mathcal{A}, h \rangle$ (*proper defeat*) or $\langle \mathcal{A}_1, h_1 \rangle \not\succ \langle \mathcal{A}, h \rangle$ and $\langle \mathcal{A}, h \rangle \not\succ \langle \mathcal{A}_1, h_1 \rangle$ (*blocking defeat*).

When considering sequences of arguments, the definition of defeat is not sufficient to describe a conclusive argumentation line since it disregards the dialectical structure of argumentation, cf. [5].

Definition 3 (Acceptable Argumentation Line)**.** Let $P = (\Pi, \Delta)$ be a *de.l.p.*. Let $\Lambda = [\langle \mathcal{A}_1, h_1 \rangle, \ldots, \langle \mathcal{A}_m, h_m \rangle]$ be a sequence of some arguments. Λ is called an *acceptable argumentation line*, iff 1.) Λ is a finite sequence, 2.) every argument $\langle \mathcal{A}_i, h_i \rangle$ with $i > 1$ is a defeater of its predecessor $\langle \mathcal{A}_{i-1}, h_{i-1} \rangle$ and if $\langle \mathcal{A}_i, h_i \rangle$ is a blocking defeater of $\langle \mathcal{A}_{i-1}, h_{i-1} \rangle$ and $\langle \mathcal{A}_{i+1}, h_{i+1} \rangle$ exists, then $\langle \mathcal{A}_{i+1}, h_{i+1} \rangle$ is a proper defeater of $\langle \mathcal{A}_i, h_i \rangle$, 3.) $\Pi \cup \mathcal{A}_1 \cup \mathcal{A}_3 \cup \ldots$ is non-contradictory (*concordance of supporting arguments*), 4.) $\Pi \cup \mathcal{A}_2 \cup \mathcal{A}_4 \cup \ldots$ is non-contradictory (*concordance of interfering arguments*), and 5.) no argument $\langle \mathcal{A}_k, h_k \rangle$ is a subargument of an argument $\langle \mathcal{A}_i, h_i \rangle$ with $i < k$.

In DeLP a literal h is *warranted*, if there is an argument $\langle \mathcal{A}, h \rangle$ which is non-defeated in the end. To decide whether $\langle \mathcal{A}, h \rangle$ is defeated or not, every acceptable argumentation line starting with $\langle \mathcal{A}, h \rangle$ has to be considered.

Definition 4 (Dialectical Tree)**.** Let $P = (\Pi, \Delta)$ be a *de.l.p.* and let $\langle \mathcal{A}_0, h_0 \rangle$ be an argument. A *dialectical tree* for $\langle \mathcal{A}_0, h_0 \rangle$, denoted $\mathcal{T}_{\langle \mathcal{A}_0, h_0 \rangle}$, is defined as follows:

1. The root of $\mathcal{T}$ is $\langle \mathcal{A}_0, h_0 \rangle$.
2. Let $\langle \mathcal{A}_n, h_n \rangle$ be a node in $\mathcal{T}$ and let $\Lambda = [\langle \mathcal{A}_0, h_0 \rangle, \ldots, \langle \mathcal{A}_n, h_n \rangle]$ be the sequence of nodes from the root to $\langle \mathcal{A}_n, h_n \rangle$. Let $\langle \mathcal{B}_1, q_1 \rangle, \ldots, \langle \mathcal{B}_k, q_k \rangle$ be the defeaters of $\langle \mathcal{A}_n, h_n \rangle$. For every defeater $\langle \mathcal{B}_i, q_i \rangle$ with $1 \leq i \leq k$ such that the argumentation line $\Lambda' = [\langle \mathcal{A}_0, h_0 \rangle, \ldots, \langle \mathcal{A}_n, h_n \rangle, \langle \mathcal{B}_i, q_i \rangle]$ is acceptable, the node $\langle \mathcal{A}_n, h_n \rangle$ has a child $\langle \mathcal{B}_i, q_i \rangle$. If there is no such $\langle \mathcal{B}_i, q_i \rangle$, the node $\langle \mathcal{A}_n, h_n \rangle$ is a leaf.

In order to decide whether the argument at the root of a given dialectical tree is defeated or not, it is necessary to perform a *bottom-up*-analysis of the tree. Every leaf of the tree is marked "undefeated" and every inner node is marked "defeated", if it has at least one child node marked "undefeated". Otherwise it is marked "undefeated". Let $\mathcal{T}^*_{\langle \mathcal{A},h \rangle}$ denote the marked dialectical tree of $\mathcal{T}_{\langle \mathcal{A},h \rangle}$. We call a literal h *warranted*, iff there is an argument $\langle \mathcal{A}, h \rangle$ for h such that the root of the marked dialectical tree $\mathcal{T}^*_{\langle \mathcal{A},h \rangle}$ is marked "undefeated". Then $\langle \mathcal{A}, h \rangle$ is a *warrant* for h.

6. Using DeLP for Legal Reasoning

One advantage of the representation for legal knowledge described in Sec. 4 is the similarity to a *de.l.p.*. The legal knowledge can be transformed into DeLP rules straightforwardly. The relevant facts of the case can then be added as DeLP facts.

For example, Art. 7.2.2 in DeLP rules becomes:

R1: *right_to_performance*(Y,X,O) $\prec$ *obligation*(X,Y,O), *not_money*(X), *no_performance*(X,Y,O).
R2: ~*right_to_performance*(Y,X,O) $\prec$ *obligation*(X,Y,O), *not_money*(X),
no_performance(X,Y,O), *performance_impossible*(X,O).
R3: ~*right_to_performance*(Y,X,O) $\prec$ *obligation*(X,Y,O), *not_money*(X),
no_performance(X,Y,O), *performance_too_burdensome*(X,O).

Likewise, secondary and antagonistic rules are translated to DeLP, for instance:

R4: *performance_too_burdensome*(X,O) $\prec$ *performance_req_excessive_expense*(X,O).
R5: ~*performance_too_burdensome*(X,O) $\prec$ *obstacles_were_foreseeable*(X,O).
R6: *performance_req_excessive_expense*(X,O) $\prec$ *severe_short_of_goods*(O), *out_of_stock*(X,O).

For the general case, let $P = (c, \mathcal{B}, \overline{\mathcal{B}})$ be a legal provision as in Def. 1 and let $B = ((b_1, \ldots, b_n), E_1, \ldots, E_m)$ with $E_i = (e_{i,1}, \ldots, e_{i,r_i})$. If $B \in \mathcal{B}$, the following $m + 1$ DeLP rules are generated:

$$\begin{array}{l} c \prec b_1, \ldots, b_n. \\ {\sim}c \prec b_1, \ldots, b_n, e_{1,1}, \ldots, e_{1,r_1}. \\ \quad \cdots \\ {\sim}c \prec b_1, \ldots, b_n, e_{m,1}, \ldots, e_{m,r_m}. \end{array}$$

For $B \in \overline{\mathcal{B}}$, the DeLP rules generated are obtained from the rules above by replacing c (resp. ${\sim}c$) in each rule head by ${\sim}c$ (resp. c).

When passing the rules {R1, ..., R6} together with the facts *obligation*(X,Y,O), *not_money*(O), *no_performance*(X,Y,O), *severe_short_of_goods*(O), *out_of_stock*(X,O), *obstacles_were_foreseeable*(X,O) to the DeLP interpreter and asking for the status of the claim *right_to_performance*(Y,X,O), it generates the dialectical tree shown in Fig. 2. Since the root of this tree is undefeated, DeLP warrants its claim.

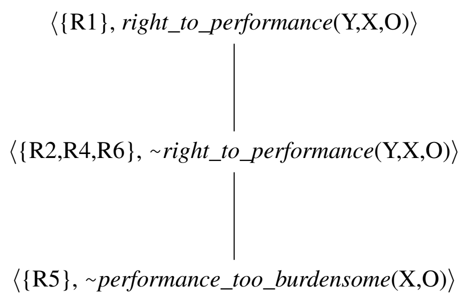
⟨{R1}, *right_to_performance*(Y,X,O)⟩
⟨{R2,R4,R6}, ~*right_to_performance*(Y,X,O)⟩
⟨{R5}, ~*performance_too_burdensome*(X,O)⟩

Figure 2. Example of a dialectical tree generated by DeLP

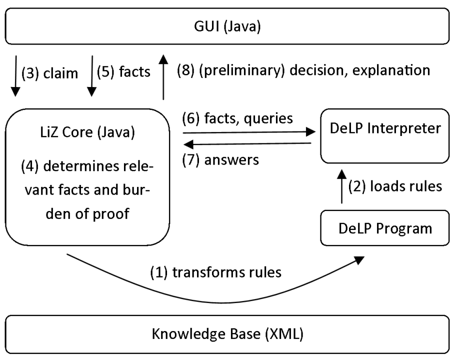

Figure 3. Illustration of the LiZ system. The system is interactive, steps 5 to 8 are repeated during the legal procedure until no more relevant facts are submitted.

It is obvious that the inference mechanism of DeLP is *justifiable* according to our definition in Sec. 2. This is true regardless of the concrete comparison criterion used to decide between conflicting rules (DeLP allows for different criteria), as long as this criterion prefers exceptions over the corresponding rule. Above that, the standard criterion of “generalized specificity” is particularly suitable for modeling legal reasoning. The concept of “generalized specificity” is twofold. First, it prefers “more precise” rules, where rule A is more precise than rule B if the conditions of B are a true subset of the conditions of A. Secondly, it prefers more concise arguments, where an argument is more concise when it makes less use of rules than another. This concept corresponds remarkably well to the widely acclaimed standard rules of interpretation in legal theory that we recalled in Sec. 2. Indeed, generalized specificity incorporates the same idea as the *lex specialis*.

As an extension to this work, also the ideas of *lex superior* and *lex posterior* could be easily implemented on the basis of DeLP. For this, only the priority relation between arguments has to be changed, e. g., by combining generalized specificity with information about time and law hierarchies in a lexicographic way.

The structure of the LiZ system itself is illustrated in Fig. 3. It interactively follows the course of a civil action from the judge’s point of view. First, the user enters general information about the case, especially the kind of claim. Then, LiZ searches the knowledge base for rules that might support such a claim, collects all the secondary rules needed to interpret them, and constructs and shows the legal tree(s). It also determines the burden of proof for the relevant facts. The user can then successively enter the relevant facts as presented by plaintiff and defendant through the GUI. Whenever a new fact is entered, LiZ updates the three fact sets (view of the plaintiff/defendant, view to be taken by the judge), transforms them to DeLP facts and passes them on to the DeLP interpreter, along with a query as to whether the claim would be supported under each of the fact sets. The results are transformed into plain text as used by judges, the user can base his or her intermediary decisions on them (e. g., ask the defendant to produce further evidence to avoid losing the case). Thus, the system shows the decision that would have to be made at any given time, which becomes the final decision once all facts have been entered. Currently, the LiZ database contains rules from the law of obligations of the German Civil Code, but the design of LiZ is in no way specifically tailored to the German jurisdiction.

7. Conclusions and Further Work

We presented an approach to an argumentation-based system that supports a judge in deciding private law cases. Starting from a focus on general principles governing judicial legal reasoning, we argued that the defeasibility inherent in laws and rules as well as the way rules are aggregated, compared, and preferred match the reasoning behavior of DeLP, together with the power of generalized specificity that is used as comparison criterion among arguments. The introduced knowledge representation model based on the notion of legal provisions has been implemented with an XML representation in the LiZ system. In particular, LiZ automatically determines the burden of proof that is essential for argumentation, exploits the defeasible reasoning facilities of the DeLP system, and provides interface functionalities as required from a private law judge point of view. As part of our future work, we plan to modify the priority relation among arguments in such a way as to take superiority of laws into account, or to favor more recent laws. Moreover, the modelling of further and more intricate cases will be necessary for a thorough usability and acceptance evaluation of the LiZ system.

References

[1] A. Aarnio. *The Rational as Reasonable*. D. Reidel Publ. Co., Dordrecht, 1987.

[2] R. J. Aldisert. *Logic for Lawyers*. National Institute for Trial Advocacy, 3rd edition, 1997.

[3] T. Bench-Capon and H. Prakken. Introducing the logic and law corner. *J. of Logic and Computation*, 18(1):1–12, 2008.

[4] E. Bund. *Juristische Logik und Argumentation*. Rombach & Co., Freiburg im Breisgau, 1983.

[5] Alejandro J. García and Guillermo R. Simari. Defeasible logic programming: An argumentative approach. *Theory and Practice of Logic Programming*, 4(1):95–138, 2004.

[6] D. M. Godden and D. Walton. Defeasibility in judicial opinion: Logical or procedural? *Informal Logic*, 28(1):6–19, 2008.

[7] T. F. Gordon and D. Walton. Proof burdens and standards. In I. Rahwan and G. Simari, editors, *Argumentation in Artificial Intelligence*, pages 239–260–144. Springer, 2009.

[8] J. Horovitz. *Law and Logic*. Springer, 1972.

[9] International Institute for the Unification of Private Law. *UNIDROIT Principles of International Commercial Contracts*. UNIDROIT, Rome, 2nd edition, 2004.

[10] M. Klatt. *Making the law explicit*. Hart Publishing, Oxford, 2008.

[11] X. Minghui. On the inference rules in legal logic. *Social Sciences in China*, 30:58–74, 2009.

[12] K. Olivecrona. *Law as Fact*. Steven and Sons, London, 1971.

[13] E. Ott. *Die Methode der Rechtsanwendung*. Schulthess Polygraphischer Verlag AG, Zürich, 1979.

[14] A. Peczenik. *On Law and Reason*. Springer Science + Business Media B.V., 2009.

[15] A. Podlech, editor. *Rechnen und Entscheiden, Mathematische Modelle juristischer Argumentation*. Duncker & Humblot, Berlin, 1977.

[16] H. Prakken. A formal model of adjudication dialogues. *AI and Law*, 16(3):333–359, 2008.

[17] H. Prakken and G. Sartor. The role of logic in computational models of legal argument: A critical survey. In A. C. Kakas and F. Sadri, editors, *Computational Logic: Logic Programming and Beyond*, volume 2408 of *Lecture Notes in Computer Science*, pages 342–381. Springer, 2002.

[18] G. Sartor. Defeasibility in legal reasoning. In Z. Bankowski, I. White, and U. Hahn, editors, *Informatics and the Foundations of Legal Reasoning*, pages 119–157. Kluwer, 1995.

[19] M. J. Sergot, F. Sadri, R. A. Kowalski, F. Kriwaczek, P. Hammond, and H. T. Cory. The British nationality act as a logic program. *Commun. ACM*, 29(5):370–386, 1986.

[20] V. Varano. Some reflections on procedure, comparative law, and the common core approach. *Global Jurist Topics*, 3(2):395–418, 2003.

[21] E. Vranes. The definition of 'norm conflict' in international law and legal theory. *The European Journal of International Law*, 17(2):395–418, 2006.