Induction of Defeasible Logic Theories in the Legal Domain

Benjamin Johnston
superhero@benjaminjohnston.com.au

Guido Governatori
guido@itee.uq.edu.au

School of Information Technology and Electrical Engineering
The University of Queensland
Brisbane, Queensland, Australia

ABSTRACT

Defeasible Logic is a promising representation for legal knowledge that appears to overcome many of the deficiencies of previous approaches to representing legal knowledge. Unfortunately, an immediate application of technology to the challenges of generating theories in the legal domain is an expensive and computationally intractable problem. So, in light of the potential benefits, we seek to find a practical algorithm that uses heuristics to discover an approximate solution. As an outcome of this work, we have developed an algorithm that integrates defeasible logic into a decision support system by automatically deriving its knowledge from databases of precedents. Experiments with the new algorithm are very promising – delivering results comparable to and exceeding other approaches.

1. INTRODUCTION

Some projects have attempted to produce more powerful systems to aid the legal decision making progress, and while these have experienced some success, clearly the legal community has not whole-heartedly taken up these technologies (for they do not form a common part of a lawyer's tool-set, cf. [34, 32]). With this in mind, we seek to produce a system that assists the decision making process of law-makers, lawyers and the public, by providing a tool that offers advice and reasons for its advice, given information about current legal proceedings. The challenge of this work is in the peculiar requirements of the legal domain, namely the importance of extremely high accuracy, the value of judicial independence and the need for presenting both decisions and their rationale in a language that is natural to its users place constraints on the development of information systems that are relatively unique to the domain – constraints that, to date, have yet to be adequately overcome.

By developing legal information systems, we do not seek to replace judges nor remove compassion from the legal system – but the development of such systems should offer the legal community benefits that are difficult to realise without computer technologies. Thus, we have developed our ideas with three clear outcomes in mind:

- To assist in accurate and expedient decision making by providing an efficient tool that suggests and argues decisions,
- To help reach consistent decisions by drawing conclusions consistent with precedents,
- To provide good advice and information at a low cost by offering a tool that can quickly offer strategic and predictive information without expensive construction or analysis costs.

Admittedly, the need for compassion and humanity, and the difficulty of interpreting law gives rise to cases that are far beyond the capabilities of any existing computer systems to reason about. But even though we cannot handle every such case, it is understood that in many fields of legal practice, (particularly such as taxation and traffic law) legislation and precedents are well understood and the typical cases before the courts are "routine". In fact, we have made the assumption that most cases are not "landmark" decisions, but that much of the decision making process of legal practice is routine and potential scope for partial automation. For this reason, it seems natural that a legal information system that takes an encoded form of the facts that are currently before a court (or the situation of a user considering legal action) and returns suggestions that are consistent with precedents along with justifications for the suggestions would prove to be beneficial to the legal community.

Even though we seek only to assist (and not replace) experts, it is still vitally important that any decision support system is accurate and verifiable. We therefore have motivated our work by assuming the following requirements:

- A transparent operation and construction which can present an argument or justification for any conclusion it reaches,
- An internal representation of knowledge that is verifiable and can be comprehended by those who verify it,
- A conservative mode of operation that does not draw conclusions where there is doubt,
- A user-friendly approach suitable for law workers and the general public that does not require training in formal logic to view knowledge within the system or interpret a conclusion of the system,
- Preferably, a low cost method of construction.

The challenge of producing legal information systems is that it is necessary to find a practical trade-off between these requirements. For example, while automatically generating knowledge bases might alleviate the problem of expense (as a result of less need for expert knowledge in generating and structuring the data), this method of construction typically reduces the confidence that

ICAIL-03 June 24–28, Edinbugh, Scotland

users have in the soundness of the system. Similarly, heuristic methods applied to large sets of training examples might improve the accuracy and applicability of the system, but at the cost of a sound and verifiable reasoning method or the ability to examine and comprehend the underlying reasoning involved in reaching a given conclusion. In spite of these inherent difficulties, the potential benefits to the legal community are such that the successful satisfaction of these needs by a legal decision support system (LDSS) is surely an invaluable outcome.

2. PREVIOUS APPROACHES

Existing approaches to constructing legal expert systems fall into two broad categories: manual encoding of rules in expert systems, or automatic induction via machine learning techniques. Manually encoded expert systems have the luxury of being able to use expressive and verifiable internal representations, but it is an expensive and delicate operation to produce such expert systems that may in fact result in a system that is inconsistent with legal practice due to the difficulty in having an expert accurately encode their knowledge. On the other hand, while machine learning can reduce the construction cost, systems based on such technologies may not be able to reconcile automatic construction of knowledge bases with knowledge representations that suitably encode knowledge for reasoning.

2.1 Case Based Reasoning

Case based reasoning systems (such as Hypo [35]) attempt to find related cases under the assumption that a similar case is likely to have a similar conclusion. While such an information system certainly would benefit a lawyer in preparing an argument, the systems do not focus on the rationale of a decision but on the similarity between the cases, so it is difficult for a case based reasoning system to produce a justification for its predictions other than by analogy. Some systems [35] attempt to resolve this difficulty by manually inferring the principles underlying the case, but this is really no more than producing an expert system with analogical reasoning capabilities (and so is as expensive as other approaches to manually constructing expert systems).

2.2 Attribute-Value and Propositional Learners

Propositional learners such as the ID3 and C4.5 decision tree induction algorithms or ASSISTANT have had some success when applied in the legal domain [13]. Unfortunately, propositional rules are difficult to interpret by a lawyer without a computer science background, and even with training, the resultant trees or propositional expressions are still difficult to interpret. Furthermore, the output of such algorithms are typically expressions that seek to classify cases with as few attributes as possible, but do not adequately handle the possibility that those attributes may not even be available at all (i.e., propositional learners do not work satisfactorily with partial information) [33].

2.3 Neural Networks, Bayesian Networks and Other Continuous Model Fitting

Projects such as the Split-Up project [36] which applied neural networks to family law, often achieve high levels of accuracy. But unfortunately, while approaches to model fitting over continuous variables such as neural networks give very high accuracy by learning from examples and have proven successes in disparate fields, in general there are difficulties in applying them successfully in a legal context. The reasoning behind such methods is simply an adaptive non-linear function and, as such, it is extremely difficult to generate explanations from their output, or even to verify their correctness.

2.4 Association Rules

Some work has been done towards using association rules (mined by the Apriori algorithm [1]) as a basis for an LDSS. While this is currently work-in-progress (and in fact was the initial motivation for this project), it seems unlikely that taking this approach alone is going to result in significant success. Association rule mining finds "general associations" between properties, but does not produce rules. A specific field in the legal domain might have several guiding rules with many exceptions and so is guided by not only the associations between assumptions and a conclusion, but also exceptions to the associations – because of this, association rule miners are unlikely to be a general solution to the problem of learning the reasoning principles underlying law. Furthermore, care must be taken to ensure that mined associations are not simply common properties of the domain, but are indeed deciding factors. These difficulties with interpreting association rules, and the challenge of integrating them into a formal model suggests that association rules might be better suited as a heuristic device as opposed to an end in itself.

2.5 Inductive Logic Programming

Inductive logic programming (ILP) is concerned with producing pure negation-free Prolog programs (theories in Horn clause logic) from example sets of inputs and outputs, and usually operate in either a bottom-up approach by generalising examples or a top-down approach specialising the empty program into a suitable theory [25]. In the literature, there appears to be limited application of ILP in the legal domain, which may be due to the monotonicity of Horn clause logic (we will address the monotonicity issue later) being unsuitable for law. The simplicity of theories induced by ILP systems in their typical domains is significant, so we have drawn from the body of work in ILP for our research, but have adapted some of the benefits into a framework that handles the defeasibility of law directly.

There is some existing work with non-monotonic ILP that seeks to overcome some of the limitations of the closed world assumption and monotonicity of the logic used within ILP. ILP approaches have been used [15, 27] to induce theories in Reiter's Default logic – a similar strategy [23] has also been used to learn Extended Logic programs (which represents a subset of default logic). While these approaches are near what we are attempting to achieve, finding the extension of a Default logic theory is an NP-complete problem, and Default logic is a form of expression that we believe is difficult for untrained users to interpret, particularly as a language for describing legal systems.

3. DEFEASIBLE LOGIC

While the aforementioned existing approaches have experienced a certain degree of success, each face deficiencies that we believe will limit the widespread acceptance of the techniques in the legal domain. The challenge seems to be that of finding a balance between having a representation for efficient reasoning and the cost of encoding knowledge into that representation. In [29], Prakken describes several non-monotonic logics that appear to have a more natural correspondence to legal reasoning than other forms of formal expression. Furthermore, Defeasible Logic, a form of non-monotonic reasoning invented by Donald Nute [28] (and related to one of the logics presented by Prakken) has a particularly natural expression that permits the expression of (normative) systems with a close correspondence between plain-language expression and its

Rule				Explanation
$r1$:		$\Rightarrow$	$\neg guilty$	Innocence is presumed
$r2$:	$evidence$	$\Rightarrow$	$guilty$	Evidence can show guilt
$r3$:	$\neg motive$	$\rightsquigarrow$	$\neg guilty$	Lack of motive can suggest innocence
$r4$:	$alibi$	$\Rightarrow$	$\neg guilty$	An alibi can prove innocence
$r4 \succ r3, r3 \succ r2, r2 \succ r1$				

Figure 1: Hypothetical Criminal Law Theory

encoded form [6, 2, 14, 19]. In fact, [18, 20] illustrates the use of defeasible reasoning within a multi-agent system for automated negotiation. Practical applications of defeasible reasoning have been proposed for executable regulations [6, 2], contracts [31] and business rules [21]. The clarity of a defeasible logic theory to an untrained user can be further enhanced during a validation phase – a domain expert can annotate the rules of a defeasible logic theory with plain-English expressions that can be displayed for the user as an "explanation" of the conclusion drawn from the theory.

Defeasible logic, being a non-monotonic logic handles partial knowledge well [29, 17], has a sceptical reasoning process [3, 5], is equivalent to or subsumes many other defeasible logics [3, 7] and has an algorithm for computing its extension in linear time and space [4, 26]. These favourable properties of defeasible logic are important for legal reasoning. Law abounds in cases involving partial knowledge, and any reasonable decision support system must act in a conservative and sceptical way (for it is better to give **no** answer than **an** incorrect answer). The computational efficiency is also important, not only for drawing conclusions from a theory, but also because it lends itself to efficient algorithms for the induction of theories from datasets. While defeasible logic does not allow meta-level reasoning about the purpose or backing of rules, for example, the logic remains a powerful [6, 2, 14, 19] and natural form of expression that is relevant for practical, routine legal practice and rules in a defeasible logic theory correspond to the untrained understanding of a "rule".

For these reasons it seems that defeasible logic would be a suitable representation for knowledge in an LDSS – the challenge that remains is the automatic or semi-automatic creation of such knowledge from precedents.

A defeasible logic theory is a collection of rules that permit us to reason about a set of facts or known truths to reach a set of defeasible conclusions. Because multiple rules may be applicable in any given situation, a defeasible logic theory additionally includes a relation for resolving these conflicts.

For example, consider the theory about criminal law in Figure 1. The theory consists of two components:

- A set of rules that can be used to conclude the guilt or innocence of the defendant in the event of certain facts being presented in the court of law, and
- An acyclic transitive relation over the rules that indicates the relative strength of each rule.[1]

Suppose we are given the theory of Figure 1, and a set of facts, $\{evidence, alibi\}$, that have been presented to the court of law and are assumed true, then we can defeasibly prove the innocence of the defendant, for:

- We note that $r4$ permits us to conclude $\neg guilty$, and
- The necessary conditions for the application of $r4$ hold, namely it is known that *alibi* is a fact (or has been defeasibly proven true), and
- Of the remaining rules, the only one that reaches the contradictory conclusion *guilty* and for which its necessary conditions are satisfied, is $r2$, but $r4$ is stronger than $r2$ (i.e., $r4 \succ r2$) so $r2$ does not override the conclusion[2].

A *defeasible logic theory* T is a pair $(R, \succ)$ where R is a finite set of rules and $\succ$ is a partially ordered[3] relation defining superiority over R.

Rules are defined over *literals*, where a literal is either an atomic propositional variable a or its negation, $\neg a$. Given a literal, p, the *complement*, $\sim p$ of that literal is defined to be a if p is of the form $\neg a$, and $\neg a$ if p is of the form a.

There are two kinds of rules, *defeasible rules* and *defeaters*. *Defeasible rules* can be used to defeasibly prove some conclusion, but *defeaters* can only be used to prevent a conclusion being reached. Typically a third kind of rule is permitted, *strict rules*, which have a more classical meaning in that they are monotonic and cannot be defeated. We disregard strict rules in application to the automatic induction of defeasible theories because it is impossible to conclude a strict correlation with only the partial knowledge possible with finite datasets (in any case, strict rules can be simulated with defeasible rules that are 'high' in the superiority relation such that they can rarely be defeated).

A *defeasible rule* is denoted by $Q \Rightarrow p$ where Q is a set of literals denoting the premises of the rule, and p is a single literal denoting the conclusion upon application of the rule. A rule of this form can be interpreted to say that whenever the literals in Q are known to be facts or to be defeasibly provable, then we can defeasibly prove p (by "defeasibly", we mean that we can only prove p tentatively, and is subject to possible defeat by other, stronger, rules).

A *defeater* is a rule that is denoted by $Q \rightsquigarrow p$ where Q is a set of literals denoting the premises of the rule, and p is a single literal denoting the counter-conclusion that can be used upon application of the rule. A defeater is used to prevent the conclusion p. That is, a rule of this form can be interpreted to say that whenever the literals in Q are known to be facts or to be defeasibly provable, then we can only reach a conclusion that is consistent with p (subject to defeat by other rules); we cannot prove $\sim p$, but *may* prove p if there are other rules supporting this position, otherwise we may reach no conclusion at all. Note that with a pair of rules of the form $Q \rightsquigarrow a$ and $Q \rightsquigarrow \neg a$, each at the same superiority, we can block any conclusion with respect to a.

[1] We have only denoted the relevant mappings, the actual superiority relation would in fact be the least acyclic transitive relation containing the mappings denoted – in this case, the transitive closure of these mappings.

[2] Note also that the rule $r3$ cannot be used to prove innocence, but merely to disprove guilt.

[3] Though, in the general case, the superiority relation is simply a binary relation over the set of rules.

We define $Ante(r) = Q$ where r is a rule of the form $Q \Rightarrow p$ or $Q \rightsquigarrow p$; that is, $Ante(r)$ is the set of premises or antecedents of the rule r (i.e., $Ante(r)$ is the left hand side of the rule).

We reason about a set of facts (of a given case) F with respect to a defeasible theory T to reach defeasible or tentative conclusions of that particular case. A conclusion of a defeasible theory T and facts F is conventionally a tagged literal of one of the following forms:

- $+\partial p$, which is intended to mean that p is defeasibly provable in T over F
- $-\partial p$, which is intended to mean that p is not defeasibly provable in T over F.

We define an entailment relation, $T,F \vdash c$, which indicates that c is a conclusion of the set of facts F with respect to theory T. The entailment relation is defined by the proof mechanism expounded in [3, 5], and which is briefly presented here for completeness.

A proof within a defeasible logic theory T given a set of facts F is a finite sequence $P = \langle p_1, p_2, \ldots \rangle$ of tagged literals satisfying the two inference rules that follow. $P(1..i)$ denotes the initial part of the sequence P, of length i, and $P(i)$ denotes the ith element of P. R_d denotes the set of defeasible rules in R (i.e., those rules that are not defeaters) and $R[q]$ denotes the set of rules in R with conclusion q.

$+\partial$:
If $P(i+1) = +\partial p$ then either
$p \in F$ or
(1) $\exists r \in R_d[p] \forall q \in Ante(r) : +\partial q \in P(1..i)$ and
(2) $\forall s \in R[\sim p]$ either
(a) $\exists q \in Ante(s) : -\partial q \in P(1..i)$ or
(b) $\exists t \in R_d[p]$ such that
$\forall q \in Ante(t) : +\partial q \in P(1..i)$ and $t \succ s$.

$-\partial$:
If $P(i+1) = -\partial p$ then
$p \notin F$ and
(1) $\forall r \in R_d[p]\ \exists q \in Ante(r) : -\partial q \in P(1..i)$ or
(2) $\exists s \in R[\sim p]$ such that
(a) $\forall q \in Ante(s) : +\partial q \in P(1..i)$ and
(b) $\forall t \in R_d[p]$ either
$\exists q \in Ante(t) : -\partial q \in P(1..i)$ or $t \not\succ s$.

Or, in plain English, we use the conclusion of the strongest of the defeasible rules that have all premises satisfied. If no such rule exists, or if there is any defeater with the opposite conclusion that is not stronger, then we have no conclusion.

For simplicity, we disregard the tagging, and instead represent $T,F \vdash +\partial p$ as simply $T,F \vdash p$, and use $T,F \vdash ?a$ to denote the case that both $T,F \vdash -\partial a$ and $T,F \vdash -\partial \neg a$ holds (that is, $T,F \vdash ?a$ denotes the case that nothing can be defeasibly proven with respect to a). With a partially ordered superiority relation, only these three possibilities can occur [4, 12].

4. INDUCING DEFEASIBLE THEORIES FROM DATASETS

When automatically generating a logical theory from precedents, two important considerations are how well the theory models the domain and how well the theory generalises to new cases. While judging what constitutes a good degree of generalisation is a subjective problem that can have no definitive answer, the peculiarities of producing LDSSs are such that we can assume (and note in practice) that a minimal theory that is consistent with and describes the existing precedents is most desirable. This is because having a smaller theory results in rules that cover more examples and therefore should both generalise well and realistically match the human preference for using simple reasoning processes. A further reason for seeking a minimal theory is that, with less rules in the theory, it is likely to be easier to have a domain expert comprehend, provide feedback on and improve the induced theory.

This goal gives rise to two concerns:

- Is there **always** a set of minimal theories for any dataset?
- Is the generation of minimal theories tractable?

Two results are important here; that it is possible to find a theory that describes a dataset, and that finding the optimal solution is an NP optimisation problem. We begin with some definitions first.

We define a *dataset D* as a finite set of pairs of the form (F,c), where F is a set of literals denoting the known truths or facts of the particular precedent, and where c is either a literal or the term $?a$ indicating that no conclusion could be reached or is known with respect to propositional variable a ($?a$ is usually only introduced during pre-processing). We say a dataset is consistent if every conclusion is formed from the same propositional variable, and if for all records $d_1 = (F_1,c_1) \in D$ and $d_2 = (F_2,c_2) \in D$, if $F_1 = F_2$ then $c_1 = c_2$. In other words, a dataset is consistent if no two identical cases have different conclusions. Given a consistent dataset D we define $Var(D) = a$, where a is the propositional variable that appears in every conclusion.

For convenience we will assume that the datasets are consistent. Whenever this assumption is invalid (possibly due to noise, two identical borderline cases with different conclusions, or changing legal practice), it can be corrected by pre-processing it. This might mean deleting the records that are causing the inconsistency, using the result from the recent record, or replacing all such inconsistent records with a new record of the form $(F,?a)$ to indicate that given the particular facts F that are causing the inconsistencies, we do not have any known conclusion.

It should be noted that the definition of a dataset includes the assumption of a single propositional variable to be used over all the conclusions. In applications where multiple conclusions are necessary, this limitation can be trivially overcome by processing each class of conclusions with a different dataset and repeated application of the algorithm presented later, in Section 5 of this paper.

We say a defeasible logic theory, T, has an *accuracy* of x% with respect to a given dataset, D, if for x% of the records $d = (F,c) \in D$, it can be proven that $T,F \vdash c$. We say a theory *describes* a dataset if it is 100% accurate, that is for each record $d = (F,c) \in D$, it can be proven that $T,F \vdash c$.

THEOREM 1. *[24] Given any consistent dataset D, there exists at least one theory that describes the dataset.*

In fact, it turns out that for any given dataset D, there can be many theories that describe the dataset. While we could use the theory generated in the construction used for the proof of Theorem 1, this is unsatisfying because using such a theory becomes no more than a primitive case-based reasoning system. Instead, we look to find a minimal theory, that is a theory that describes the dataset and has the *minimal* number of rules. Because a minimal theory has few rules, we would expect each rule to carry more significance and have greater predictive power than might the rules of a larger theory describing the same dataset. For this reason we expect that a minimal theory would give the best generalisation of the dataset and would be most likely to perform well on unseen cases. We believe that finding a minimal theory also simplifies the comprehension effort required to have a human expert verify a theory, and

```
set theory, T = (∅, ∅)
do
    invoke Rule Search Routine (Figure 3), to find a new rule r
    if r ≠ nil
                set T, to T + r
while r ≠ nil
```

Figure 2: Defeasible Theory Search Algorithm

that simpler representations are more likely to match the reasoning of human experts in their own practice than needlessly convoluted rules or an individual case-based-reasoning approach.

Unfortunately it turns out that the problem of generating a minimal theory is NP-hard. Under the assumption that $P \neq NP$, and given the NP-hardness of the problem we conclude that the generation of minimal theories is intractable. The proof of NP-hardness is by showing that instances of the hitting set problem (SP8 in [16]) or of the minimum hitting set problem (SP7 in [8]) can be transformed to a dataset of precedents in polynomial time, such that the minimal theory of the precedents can be used to solve the decision procedure or optimisation problem respectively. This concern is the subject of the following theorem regarding a decision procedure for which NP-hardness of the corresponding optimisation problem follows immediately:

THEOREM 2. *[24] Given a dataset D and a positive integer $k' \leq |D|$, the problem of deciding whether there exists a theory $T = (R, \succ)$ of size $|R| \leq k'$ that describes D is NP-complete.*

5. GENERATING SMALL THEORIES WITH HEURISTICS

Given the intractability of finding minimal theories, we instead look towards heuristics and clever search strategies to find approximate, "small" theories. Much of the existing work on generation of small logical theories from examples is in the context of inductive logic programming (ILP) and so drawing inspiration from this work, we have developed a new algorithm that uses top-down refinement techniques, similar to work in ILP, to improve the performance on datasets. The algorithm, HeRO, operates in a greedy, best-first, branch-and-bound fashion.

The algorithm starts with an empty theory and iteratively adds rules to the theory in a greedy fashion so as to improve the accuracy of the theory. With every iteration the search space of possible rules is explored using a branch and bound algorithm to select the rule with the highest gain. This greedy mode of operation is not unrealistic because rules that offer a high degree of predictive power should naturally offer the greatest degree of improvement in accuracy of the theory (and indeed, practice confirms that this is the case or at least a suitable approximation of reality). Pseudocode for the high-level operation of the algorithm is detailed in Figure 2.

We now turn our attention to the search algorithm used for selecting rules and their appropriate positions to add to the theory.

If a rule $r = (Q \Rightarrow c)$ or $r = (Q \rightsquigarrow c)$ is added at some position in the superiority relation $\succ$ of a theory $T = (R, \succ)$, to give a new theory T', we define the gain, $gain_{T,r,T'}$, of that rule to be the difference between the number of records, $d = (F, c) \in D$, for which $T', F \vdash c$ and the number of records for which $T, F \vdash c$. That is, the gain of a rule is the increase in the accuracy of a theory that is a result of adding the rule to the theory.

This definition of gain can be equivalently stated in terms of "incorrect conclusions that are corrected by adding the new rule", and "correct conclusions that are blocked by adding the new rule", as follows:

$$\begin{aligned} gain_{T,r,T'} =& \#\{(F,c) \in D | T', F \vdash c\} - \\ & \#\{(F,c) \in D | T, F \vdash c\} \\ =& \#\{(F,c) \in D | T', F \vdash c \wedge T, F \nvdash c\} - \\ & \#\{(F,c) \in D | T', F \nvdash c \wedge T, F \vdash c\} \end{aligned}$$

We derive an upper bound for $gain_{T,r,T'}$ by noting that if the rule r is refined by adding literals to the premises to make the rule more specific, then the number of "incorrect conclusions that are corrected by adding the new rule" must decrease because a subset (but no more) of these "corrections" will still be applicable after refining the rule, and the number of "correct conclusions that are blocked by adding the new rule" will also decrease because for the same reason a subset (and no more) of these "blocks" will still be applicable after refining the rule. Under ideal circumstances, the refinement of a rule would result in no reduction of "corrections", but would eliminate all "blocking". It is this ideal condition that leads us to the upper bound, $maxgain_{T,r,T'}$ for any refinement of the rule r:

$$maxgain_{T,r,T'} = \#\{(F,c) \in D | T', F \vdash c \wedge T, F \nvdash c\}$$

These expressions for $gain_{T,r,T'}$ and $maxgain_{T,r,T'}$ can be further refined if required to support a legal practice that is known to evolve over time. Instead of simply counting records with the # operator, it is possible to compute a weighted sum, with the contribution of each record inversely proportional to the age of the record. This approach places greater emphasis on more recent conclusions, and allows theories to be generated for datasets that may contain evolutionary change.

Now, by either best-first or simply breadth-first search, we can explore the search space by maintaining a variable that holds the best rule found so far, and only exploring those branches of the search space where the upper bound, $maxgain_{T,r,T'}$, is strictly greater than the value of $gain_{T,r,T'}$ for *bestgain*. The *bestgain* can then be added to the current theory T (if it would result in a positive gain), before repeating the search again for the next iteration (or halting if no more rules exist that result in positive gain).

By only considering totally ordered superiority relations, it is possible to obtain an efficient implementation of this algorithm. For each position in the total order, the weaker rules are immediately applied to the dataset to give tentative conclusions and the records in the dataset to which stronger rules apply are discarded (because if we added a rule at this position in the superiority relation it would have no effect on the conclusions of records for which one of the stronger rules is applicable). This initial processing allows the $gain_{T,r,T'}$ and $maxgain_{T,r,T'}$ to be efficiently computed in a single pass over the dataset. Furthermore, additional performance

```
set best gain so far, bg ← 0
set best premises, bp ← nil
set best conclusion, bc ← nil
foreach position in the totally ordered superiority relation
        set weaker ←the existing rules that are weaker than the current position
        set stronger ←the existing rules that are stronger than the current position
        set priority queue, q ← ∅
        using q enqueue ∅ with priority 0
        while q ≠ ∅
                set current premise p = q.dequeue()
                compute preferred conclusion, c, of p,
                        gain, g, of p, and
                        maxgain, mg, of p
                if g > bg
                        set bg ← g
                        set bp ← p
                        set bc ← c
                if mg > bg
                        foreach refinement p' of p
                                q.enqueue(p')
if bg > 0
        return (bg ⇒ c) and current position
else
        return nil
```

Figure 3: Rule Search Routine

gains are possible by associating with each set of premises, the records in the dataset that are applicable (and maintaining this set during each set-based computation). Restricting the algorithm to only totally ordered superiority relations does not appear to result in poorer theories.

Pseudocode for an implementation of the greedy rule search appears in Figure 3. A best-first search appears, but this can be trivially modified to a breadth-first search by replacing the priority queue with a standard queue. Because it is possible to compute the accuracy gain of a rule $r = (Q \Rightarrow p$ and it's negation $r' = (Q \Rightarrow \neg p)$ in a single pass, we compute both simultaneously for a given premise set Q, and return the conclusion, *gain* and *maxgain* of the rule with greater accuracy gain.

6. EXPERIMENTAL RESULTS

The HeRO algorithm has been implemented in Prolog and C#. In this section we evaluate how faithful a theory induced by HeRO is, compared to the known theory underlying the dataset. We do this by manually creating a theory, randomly generating a dataset of 1000 records (including "noise" variables) that is consistent with the theory, and running HeRO on the dataset. HeRO gives impressively accurate results on such theories, and executes within 0.5 seconds. We will analyse a selection of the important outcomes on common patterns identified in [19]:

6.1 Simple Exception

A simple exception occurs when a rule is an exception to a base rule that results in the opposite conclusion. Consider two defeasible rules, $ar1: a \Rightarrow p$ and $ar2: a,b \Rightarrow \neg p$. $ar2$ is an exception to $ar1$ because a is associated with p unless in the presence of b, it is associated with $\neg p$.

Source Theory			
$r1$:	a	$\Rightarrow$	p
$r2$:	a, b	$\Rightarrow$	$\neg p$
$r3$:	a, c	$\rightsquigarrow$	$\neg p$
$r4$:	e	$\Rightarrow$	p
$r2 \succ r1, r3 \succ r1$			

Induced Theory			
$r1$:	a	$\Rightarrow$	p
$r2$:	a, b	$\Rightarrow$	$\neg p$
$r3$:	e	$\Rightarrow$	p
$r4$:	a, b	$\rightsquigarrow$	$\neg p$
$r5$:	a, c	$\rightsquigarrow$	$\neg p$
$r5 \succ r4, r4 \succ r3, r3 \succ r2, r2 \succ r1$			

Though the induced theory is not identical to the source theory, there is an extremely high correspondence (and the two theories are logically equivalent). The only significant difference is the additional rule $r4$ that appears in the induced theory – this rule is to compensate for the fact that the superiority relation of the source theory is a partial order, and HeRO considers only totally ordered superiority relations.

6.2 Aggregate vs Separate

The rule group called Aggregate vs Separate rules depicts situations where a choice must be made in favour of one rule with aggregated antecedents or two or more rules, each with separate antecedents. For example, a and b are, independently, sufficient reasons for p but, very often they occur simultaneously. Hence the choice is between a rule $ar1: a,b \Rightarrow p$ and two rules $ar2: a \Rightarrow p$ and $ar3: b \Rightarrow p$.

Source Theory			
$r1$:	a	$\Rightarrow$	p
$r2$:	b	$\Rightarrow$	p
$\emptyset$			

Induced Theory			
$r1$:	a	$\Rightarrow$	p
$r2$:	b	$\Rightarrow$	p
$r2 \succ r1$			

The problem of separating aggregate rules ($a,b \Rightarrow p$) from separate rules ($a \Rightarrow p$ and $b \Rightarrow p$), and yet even if a and b appear together

Algorithm	DefGen	DefGen	HeRO	HeRO
Minimum support	5	10		
Minimum confidence	100	75		
Target theory size			8	no limit
Accuracy	87%	84%	90%	97%
Rules generated	46	55	8	16
Runtime	0.4s	0.5s	2s	20s

Table 1: Comparison of Results for Credit Application Dataset

extremely often, HeRO correctly identifies the underlying theory as two separate rules.

6.3 General vs Specific

The rule group called General Vs Specific rules depicts situations where a choice must be made between a general rule and one that is more specific.

Source Theory			
$r1$:	a	$\Rightarrow$	p
$r2$:	a, b	$\Rightarrow$	p
$r5 \succ r4$			

Induced Theory			
$r1$:	a	$\Rightarrow$	p
$\emptyset$			

The problem of distinguishing between a general ($a \Rightarrow p$) and a specific ($a, b \Rightarrow p$) rule is relevant because in some contexts the more specific (and complex) rule is preferred, and in others the more general rule is preferred [19]. This problem contradicts our assumption that a minimal theory is best, and we believe that contexts requiring the more specific rule are somewhat atypical. In the source theory above, the second rule is in fact redundant, and HeRO has correctly identified the minimal equivalent theory. This sample case highlights the difference between optimistic and pessimistic modes of operations; if, for example, $r1$ in the source theory was never exercised – that b always occurred with a – then it might be argued that we should not generalize to cases in which b does not occur with a. On real datasets, we have not encountered this problem, but it is possible to alter HeRO to accommodate this requirement.

7. COMPARISON WITH OTHER APPROACHES

The HeRO algorithm has been implemented in C# and Prolog. A C# implementation was necessary due to the importance of set-based operations in the algorithm that require careful use of hash-based data structures to support rapid testing of set membership conditions. Another significant optimisation that offers performance improvement is associating records in the dataset with any set that is a subset of the premises of the record, thus allowing rapid identification of the records of a rule but without requiring excessive maintenance at each constructive set operation.

In this section we compare the implementation of the HeRO algorithm with other approaches to machine learning in the legal domain.

7.1 DefGen

The approach used by DefGen[9] follows the intent of original work by Governatori and Stranieri[19]. This work is motivated by the syntactic similarities, and the certain degree of semantic overlap between mined association rules and defeasible logic theories. The intent of this work is the development of a system that generates defeasible logic rules by analyzing the output of the Apriori [1] association rule mining algorithm. The rationale of such an approach is that it is possible to benefit from the well-understood and highly efficient algorithms (such as [1, 22]) for association rule mining, and then post-process the association rules to identify a suitable defeasible logic theory.

The algorithm has been tested against a Japanese credit application dataset[4] for benchmarking machine learning algorithms that contains 125 records and is supplied with a Lisp domain theory describing 10 attributes of credit applicants and the outcome of their applications. Apriori cannot be applied to datasets containing cardinal features, so these were binned according to categories used within the domain theory. A similar limitation also applies to our algorithm, so the same binning has been used so that the values can be mapped to a set of 27 propositional variables. [9] reports that by changing the minimum support and confidence used by the Apriori algorithm, is possible to obtain theories that offer higher accuracy or that result in fewer rules. He selects two combinations of support and confidence that give a practical balance between theory size and accuracy. In contrast, our algorithm, HeRO, seeks to find only a minimal theory – the only parameterisation possible is the trade of runtime for accuracy by halting the algorithm after a predefined number of iterations. Table 1 compares the two approaches.

Clearly, HeRO offers vast improvements over DefGen in both accuracy and the number of rules generated (fewer rules suggests a greater degree of generalization). Even though these results pertain to initial implementations of the corresponding theoretical works, the dramatic differences between these two approaches appear to be rooted in two problems. First, while defeasible logic can indeed be used to represent associations between premises and conclusions, defeasible logic, by virtue of the superiority relation, can represent "disassociations" (or exceptions to associations) that are difficult to represent or even detect with association rule mining algorithms that work with minimum supports. And second, identifying and collecting association rules into defeasible logic rules is a combinatorial optimisation problem for which finding an optimal theory is also likely to be intractable, and so this approach must also resort to the use of heuristics – only three patterns in defeasible logic were identified and used in the DefGen algorithm, a more complete version of DefGen would need to recognise far more patterns in the mined association rules and then optimise the trade-off between selecting the simpler or more complex patterns.

DefGen offers better runtime performance, and is arguably more likely to be scalable, given highly efficient association rule mining algorithms that exist. This certainly does not rule out HeRO for practical problems – the Bench-Capon dataset discussed below has more records and features, and yet has comparable runtimes. In practice, it appears that the primary limiting factor of HeRO is not the size of the dataset, but the complexity of the underlying theory

[4]Freely Available at the UCI Machine Learning repository, http://www.ics.uci.edu/ mlearn/MLRepository.html

(in such theories the branch-and-bound techniques are less effective). This particular limitation is not of serious concern, for if a particular dataset has a very complex underlying theory then the output of HeRO may be too complex for a human to reasonably verify – in such situations, the accuracy of HeRO is comparable to other approaches, but offers no distinct advantage over other approaches that are unable to be verified.

7.2 Neural Networks and Association Rules

In [10], Bench-Capon justifies the development of legal information systems such as that discussed in this paper, and trains neural networks on existing cases. Indeed, Bench-Capon faces the same problem we discussed in section 2.3 – he admits that a trained neural network is difficult to comprehensively verify, and it is clear that a neural network offers little justification for its answers – much work is necessary to explain to non-technical law workers how to interpret and use the output of a neural network.

Bench-Capon used synthetic datasets describing a fictional welfare benefits scheme paid to pensioners that suffer expenses visiting a spouse in hospital. The decision of whether or not to pay the pensioner is subject to the following conditions:

1. The person should be of pensionable age (60 for a woman, 65 for a man);
2. The person should have paid contributions in four out of the last five relevant contribution years;
3. The person should be a spouse of the patient;
4. The person should not be absent from the UK;
5. The person should not have capital resources amounting to £30,000;
6. If the relative is an in-patient the hospital should be within a certain distance: if an out-patient, beyond that distance.

These six conditions immediately translate into 12 features (age, gender, 5 contributions, spouse, absence, capital resources, distance, in-patient), which form the dataset along with 52 additional "noise" features that have no influence on the conclusion.

The neural networks that Bench-Capon trained on 2400 such records obtained the following success rates [10]:

One hidden layer:	99.25%
Two hidden layers:	98.90%
Three hidden layers:	98.75%

Even though this accuracy is quite high, it is of course difficult to judge how faithfully the conditions for payment are encoded within the network, and so it is hard to rationalize the use of such technology in legal practice if alternatives exist.

To apply the HeRO algorithm to the same dataset, the input features must first be binned to a set of prepositional variables. We do so in the same manner as Bench-Capon when preprocessing for Apriori, using some domain knowledge to intelligently partition the domain of each feature. The following defeasible logic theory is generated by HeRO, with an accuracy of 99.8% (in order of superiority, from weakest to strongest):

$$\begin{aligned} &\Rightarrow && grant \\ distance\_short,\ inpatient &\Rightarrow && \neg grant \\ \neg spouse &\Rightarrow && \neg grant \\ absent &\Rightarrow && \neg grant \\ age\_lt\_60 &\Rightarrow && \neg grant \\ capital\_gt\_3000 &\Rightarrow && \neg grant \end{aligned}$$

It is trivial to alter the algorithm to prefer scepticism when two rules have equal accuracy gain; doing so results in an even simpler theory with the same accuracy (in order of superiority, from weakest to strongest):

$$\begin{aligned} &\Rightarrow && \neg grant \\ spouse,\ \neg absent, \neg age\_lt\_60, & \\ \neg capital\_gt\_3000 &\Rightarrow && grant \\ distance\_short,\ inpatient &\Rightarrow && \neg grant \end{aligned}$$

Clearly, this theory has a very high correspondence to the original conditions – it would be easy for a non-technical law worker to understand this theory (and, in fact, could be manually annotated by a domain expert so that plain-English explanations for conclusions can be presented to the user when the theory is used). In fact, even if we ignore the fact that defeasible logic allows such clearly interpreted theories to be generated, the high accuracy is competitive with the results produced by Bench-Capon's neural networks.

Bench-Capon [10] points out that it is not necessary for an induced theory to encode all rules of an underlying domain theory in order to attain a high degree of accuracy. This is illustrated in the generated theory above: high accuracy is reached even though HeRO does not mine the second rule of the underlying domain theory (an n-out-of-m rule). In order to decrease the accuracy of such incompletely mined models, Bench-Capon creates a second dataset specifically designed to ensure that each condition is exercised in isolation. Bench-Capon reports just 77% accuracy with a neural network approach on his second datase – HeRO performs just as poorly on this dataset too (82% accuracy). Because n-out-of-m rules can only be represented as a set of ${}^{m}\mathbf{C}_{n}$ rules that individually express each condition, such conditions require far larger datasets and more intensive searches to successfully mine (it was therefore not possible to perform an exhaustive search of the rules on the low-end systems used). However, the accuracy of HeRO is comparable to Bench-Capon's neural networks, and it is likely that with a more memory efficient implementation (the current implementation is unable to explore the entire search space when the upper bound of the branch and bound algorithm is too small – that is, when the theory is almost complete, but a small percentage of error remains) it is conceivable that HeRO could mine all the required rules to express the n-out-of-m rule. In practice, in the few cases where n-out-of-m rules are suspected to exist in the domain theory (i.e., when HeRO gives poor results), it may be necessary to manually augment the mined theory with the correct n-out-of-m rule, possibly extending the syntax of defeasible logic to support such a declaration. This is, of course, an area for further research.

In [11] Bench-Capon manually analyses association rules generated from the dataset, identifying both noise variables and some conditions of the domain theory, but does not offer any immediate suggestions for automation. It is not possible to directly compare such work, but that similar trends are identified by HeRO without the manual effort is significant. Furthermore, the techniques he used lend themselves to strategies for coping with scalability. For example, Apriori (or a suitable entropy measure) could be used to preprocess datasets for HeRO in order to identify and remove noise attributes from the search space.

8. FUTURE WORK

A worthwhile extension of this work is to further examine the correspondences between horn-clause logic and defeasible logic. The similarity between the defeasible logic theory minimization problem and ILP is such that there may be benefit in exploring mappings and further correspondences between the two logics – possi-

bly to offer new benefits to both this work and ILP systems. Antoniou et al [7] have demonstrated that defeasible logic subsumes logic programming without negation as failure, and so our algorithm might provide new approaches for the ILP community.

Several decision tree induction algorithms (such as C4.5[30]) can operate in a rule generation mode that produces theories resembling defeasible logic. Such algorithms do not specifically seek to optimise their output as a defeasible theory but it will be worthwhile to compare the relative sizes of the theories generated by of each of these techniques.

An implicit assumption in this work is in the existence of structured datasets that contain propositional attributes. Clearly, this is not always the case: many attributes are continuous, and many domains of law are too complex to express assumptions in a flat propositional structure. We intend to explore the use of semi-structured representations of precedents such as XML as an input to this algorithm, in the hope that it might bring even greater relevance to this work.

Acknowledgements

This research was supported by the University of Queensland under the grant UQRSF ITEE-03/2002001335.

9. REFERENCES

[1] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In Michael Stonebraker and Joseph Hellerstein, editors, *Readings in Database Systems*, chapter 7, pages 580–592. Morgan Kaufmann Publishers, 3rd edition, 1998.

[2] Grigoris Antoniou, David Billington, Guido Governatori, and Michael Maher. On the modeling and analysis of regulations. In *Proceedings of the 10th Australasian Conference on Information Systems*, pages 20–29, 1999.

[3] Grigoris Antoniou, David Billington, Guido Governatori, and Michael Maher. A flexible framework for defeasible logics. In *AAAI/IAAI*, pages 405–410, 2000.

[4] Grigoris Antoniou, David Billington, Guido Governatori, and Michael Maher. Representation results for defeasible logic. *ACM Transactions on Computational Logic*, 2(2):255–287, April 2001.

[5] Grigoris Antoniou, David Billington, Guido Governatori, Michael Maher, and Andrew Rock. A family of defeasible logics and its implementation. In *Proceedings of the 14th European Conference on Artificial Intelligence*, Amsterdam, 2000. IOS Press.

[6] Grigoris Antoniou, David Billington, and Michael J. Maher. The analysis of regulations using defeasible rules. In *Proceedings of the 32nd Hawaii International Conference on Systems Science*. IEEE Press, 1999.

[7] Grigoris Antoniou, Michael Maher, and David Billington. Defeasible logic versus logic programming without negation as failure. *Journal of Logic Programming*, 42(1):47–57, January 2000.

[8] Giorgio Ausiello, Pierluigi Crescenzi, Giorgio Gambosi, Viggio Kann, Alberto Marchetti-Spaccamela, and Marco Protasi. *Complexity and Approximation: Combinatorial Optimization Problems and their Approximability Properties*, chapter Appendix B, page 426. Springer-Verlag, 1999. See also http://www.nada.kth.se/ viggo/problemlist/.

[9] John Avery, Andrew Stranieri, Guido Governatori, and John Zeleznikow. The identification of defeasible rule sets from databases using association rules. Technical Report, University of Ballarat.

[10] Trevor Bench-Capon. Neural networks and open texture. In *Proceedings of the Fourth International Conference on Artificial Intelligence and Law*. ACM Press, 1993.

[11] Trevor Bench-Capon, Frans Coenen, and Paul Leng. An experiment in discovering association rules in the legal domain. In *Proceedings of the Eleventh International Workshop on Database and Expert Systems Applications*, pages 1056–1060, Los Alamitos, California, 2000. IEEE Computer Society.

[12] David Billington. Defeasible logic is stable. *Journal of Logic and Computation*, 3:370–400, 1993.

[13] Stefanie Brüninghaus and Kevin Ashley. Toward adding knowledge to learning algorithms for indexing legal cases. In *The Seventh International Conference on Artificial Intelligence and Law*, pages 9–17, 1999.

[14] Michael Covington. Logical control of an elevator with defeasible logic. *IEEE Transactions on Automatic Control*, 45(7):1347–1349, 2000.

[15] Béatrice Duval and Pascal Nicolas. Learning default theories. In Anthony Hunter and Simon Parsons, editors, *ECSQARU*, number 1638 in LNCS, pages 148–159, London, 1999. Springer.

[16] Michael Garey and David Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, chapter Appendix A, page 222. W. H. Freeman and Company, 1979.

[17] Matthew Ginsberg. AI and nonmonotonic reasoning. In Dov Gabbay, Christopher Hogger, and John Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 3. Oxford University Press, 1993.

[18] Guido Governatori, Marlon Dumas, Arthur H.M. ter Hofstede, and Phillipa Oaks. A formal approach to protocols and strategies for (legal) negotiation. In Henry Prakken, editor, *Procedings of the 8th International Conference on Artificial Intelligence and Law*, pages 168–177. IAAIL, ACM Press, 2001.

[19] Guido Governatori and Andrew Stranieri. discovery. Towards the application of association rules for defeasible rules discovery. In Bart Verheij, Arno Lodder, Ronald Loui, and Antoinette Muntjewerff, editors, *Frontiers in Artificial Intelligence and Applications*, volume 70. IOS Press, 2001. Proceedings of JURIX 2001.

[20] Guido Governatori, Arthur H.M. ter Hofstede, and Phillipa Oaks. Defeasible logic for automated negotiation. In P. Swatman and P.M. Swatman, editors, *Proceedings of CollECTeR*. Deakin University, 2000.

[21] Benjamin N. Grosof, Yannis Labrou, and Hoi Y. Chan. A declarative approach to business rules in contracts: courteous logic programs in XML. In *Proceedings of the first ACM conference on Electronic commerce*, pages 68–77. ACM Press, 1999.

[22] Jaiwei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In Weidong Chen, Jeffrey Naughton, and Philip Bernstein, editors, *2000 ACM-SIGMOD International Conference on Management of Data*, pages 1–12, Dallas, Texas, 05 2000. ACM Press.

[23] Katsumi Inoue and Yoshimitsu Kudoh. Learning extended logic programs. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, volume 1, pages 176–181. Morgan Kaufmann, 1997.

[24] Benjamin Johnston and Guido Goverantori. An algorithm for

the induction of defeasible logic theories from databases. In Klaus-Dieter Schewe and Xiaofang Zhou, editors, *Database Technology 2003*, number 17 in Conference Research and Practice of Information Technology, pages 75–83. Australian Computer Science Association, ACS, 4-7 February 2003.

[25] Nada Lavrăc and Sažo Džeroski. *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood, New York, 1994.

[26] Michael Maher. Propositional defeasible logic has linear complexity. *Theory and Practice of Logic Programming*, 1(6):691–711, November 2001.

[27] Pascal Nicolas and Béatrice Duval. Representation of incomplete knowledge by induction of default theories. In Thomas Eiter, Wolfgang Faber, and Miroslaw Truszczyríski, editors, *Logic Programming and Nonmonotonic Reasoning*, number 2173 in LNAI, pages 160–172. Springer, 2001.

[28] Donald Nute. Defeasible logic. In Dov Gabbay, Christopher Hogger, and John Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 3. Oxford University Press, 1993.

[29] Henry Prakken. *Logical Tools for Modelling Legal Argument*. Kluwer Academic Publishers, 1997.

[30] John Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann, San Mateo, California, 1993.

[31] Daniel M. Reeves, Michael P. Wellman, Benjamin N. Grosof, and Hoi Y. Chan. Automated negotiation from declarative contract descriptions. In *Seventeenth National Conference on Artificial Intelligence, Workshop on Knowledge-Based Electronic Markets(KBEM)*, Austin, Texas, July 30–31 2000. AAAI, AAAI Press.

[32] Gurmak Singh, John O'Donoghue, and Karen Broome. An empirical study in the use of it by small and large legal firms in the uk. *The Journal of Information, Law and Technology*, 1:http://elj.warwick.ac.uk/jilt/02–1/singh.html, 22 March 2002. accessed 9/05/2003.

[33] Sebastian Thrun et al. The MONK's problems: A performance comparison of different learning algorithms. Technical Report CMU-CS-91-197, Carnegie Mellon University, 1991.

[34] David S. Wall. Information technology and the shaping of legal practice in the uk. In *13th BILETA Conference: 'The Changing Jurisdiction'*, page http://www.bileta.ac.uk/98papers/wall.html, 1998. accessed 9/5/2003.

[35] John Zeleznikow and Dan Hunter. *Building Intelligent Legal Information Systems*. Computer/Law Series. Kluwer Law and Taxation Publishers, 1994.

[36] John Zeleznikow and Andrew Stranieri. Knowledge discovery in the split up project. In *The Sixth International Conference on Artificial Intelligence and Law*, 1997.