An algorithm for the induction of defeasible logic theories from databases

Benjamin Johnston **Guido Governatori**

School of Information Technology and Electrical Engineering
The University of Queensland
Brisbane, Queensland, Australia
Email: superhero@benjaminjohnston.com.au, guido@itee.uq.edu.au

Abstract

Defeasible logic is a non-monotonic logic with applications in rule-based domains such as law. To ease the development and improve the accuracy of expert systems based on defeasible logic, it is desirable to automatically induce a theory of the logic from a training set of precedent data. Empirical evidence suggests that minimal theories that describe the training set tend to be more faithful representations of reality. We show via transformation from the hitting set problem that this global minimization problem is intractable, belonging to the class of NP optimisation problems. Given the inherent difficulty of finding the optimal solution, we instead use heuristics and demonstrate that a best-first, greedy, branch and bound algorithm can be used to find good theories in short time. This approach displays significant improvements in both accuracy and theory size as compared to recent work in the area that post-processed the output of an Apriori association rule-mining algorithm, with comparable execution times.

Keywords: Defeasible Logic, Machine Learning, Association Rules

1 Introduction

Expert and decision support systems are slowly making inroads into the legal community, but unfortunately they currently appear to be limited in terms of either the difficulty of their construction or their inability to justify their reasoning processes to the user. Existing systems could be roughly classified into two broad categories: expert systems that are constructed by manual encoding of knowledge (Zeleznikow & Hunter 1994), and classification tools that are automatically trained from precedent data using machine learning or data mining techniques (Zeleznikow & Stranieri 1997, Brüninghaus & Ashley 1999). Unsurprisingly, the expense involved in employing human experts and the difficulty that experts have in expressing the reasoning behind their "intuition" can eliminate the option of building expert systems in spite of the benefits that such a system can offer. Other approaches focus on automatically inducing models from precedent data using AI techniques such as neural networks, decision trees and association rules. One challenge of this approach is that the stringent demands for verification and correctness in law can eliminate such approaches on the grounds that the underlying model is a "black-box". Instead, it is desirable to attempt to find a middle ground between these approaches - using an underlying model more typical of expert systems, but facilitating construction via automatic induction.

Recent work in the field of non-monotonic logics suggests the suitability of the formalism as an underlying model for such reasoning, that turns out (as we will show) to be conducive to automatic induction. Non-monotonic logics, such as defeasible logic, were originally developed to simplify reasoning with incomplete information (Ginsberg 1993). In contrast to monotonic logics whereby a conclusion of a theory remains valid irrespective of how many assertions are added to the theory, non-monotonic logics can reach tentative conclusions that may be overridden (and replaced with a contrary conclusion) in light of additional information. Defeasible logic is one of many non-monotonic logics in use, but is particularly desirable for use in information systems because it matches the non-monotonicity of legal reasoning and is computationally efficient without sacrificing too much expressiveness. The extension of a defeasible logic theory has been shown to be computable in linear time (Antoniou, Billington, Governatori & Maher 2001, Maher 2001), as opposed to the NP-hardness or even undecidability of most non-monotonic and monotonic logics (Prakken 1997, Ginsberg 1993). While some expressiveness is sacrificed in using defeasible logic over first order logic, it still remains quite suited to legal domains. Antoniou et al (1999) have demonstrated the extremely high correspondence between regulatory documents and their equivalent expression as defeasible logic theories, in some cases the correspondence is almost 1-1 between sentences in legal documents and logical rules.

2 Defeasible Logic

In this section we will present a formal explanation of defeasible logic. Because we are focussing our attention to a specific application of defeasible logic, for simplicity our terminology slightly deviates from that used in other papers. A more complete definition of propositional defeasible logic appears in (Antoniou, Billington, Governatori, Maher & Rock 2000, Nute 1993).

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 conflicting 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

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

- a partially ordered relation[1] (called the superiority relation) that indicates the relative strength of each rule.

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

- 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.

We now formalize the ideas and terminology in the example, and present a proof theory.

A *defeasible logic theory* T is a pair $(R, \succ)$ where R is a finite set of rules and $\succ$ is a partially ordered[2] 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 likewise 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 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); that is, 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.

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 (Antoniou, Billington, Governatori & Maher 2000, Antoniou, Billington, Governatori, Maher & Rock 2000), 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 $t \not\succ s$.
```

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

If we consider only theories for which the set of all possible premises is disjoint from the set of all conclusions, then it is the case that all inferences can be performed in a single step. Because a single-step mode of operation precludes the need for recursive evaluation of the backing of a rule (i.e., there is no need to defeasibly prove the truth of the antecedents, beyond checking facts), we can simplify the above inference rules to give the following simpler proof mechanism:

$+\partial$:
If $T, F \vdash +\partial p$ then
(1) $\exists r \in R_d[p] \forall q \in \mathit{Ante}(r) : q \in F$ and
(2) $\forall s \in R[{\sim}p]$ either
    (a) $\exists q \in \mathit{Ante}(s) : {\sim}q \in F$ or
    (b) $\exists t \in R_d[p]$ such that
        $\forall q \in \mathit{Ante}(t) : q \in F$ and $t \succ s$.

$-\partial$:
If $T, F \vdash -\partial p$ then
(1) $\forall r \in R_d[p]\ \exists q \in \mathit{Ante}(r) : {\sim}q \in F$ or
(2) $\exists s \in R[{\sim}p]$ such that
    (a) $\forall q \in \mathit{Ante}(s) : q \in F$ and
    (b) $\forall t \in R_d[p]$ either
        $\exists q \in \mathit{Ante}(t) : {\sim}q \in F$ 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 (Antoniou et al. 2001, Billington 1993).

3 Complexity of the Induction Problem

Given the apparent similarities of defeasible logic with legal expression and reasoning, we turn our attention to the problem of inducing a defeasible logic theory from precedent data (i.e., a set of training examples). That is, we require an algorithm that, given a training dataset of cases and their corresponding conclusions, will produce a theory that, when applied to each element of the training dataset, will reach the same conclusions. 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 with respect to propositional variable a. 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 $\mathit{Var}(D) = a$, where a is the propositional variable that appears in every conclusion.

For convenience we will assume that all datasets are consistent. Whenever this assumption is invalid, it can be corrected by pre-processing the dataset. 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 4 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 *Given any consistent dataset D, there exists at least one theory that describes the dataset.*

Proof. By Construction.
Let $a = \mathit{Var}(D)$, we construct a theory $T = (R, \succ)$, from rules

$$\begin{aligned} R \ = \ & \{F \Rightarrow c | (F, c) \in D \wedge c \neq ?a\} \cup \\ & \{F \rightsquigarrow a | (F, c) \in D \wedge c = ?a\} \cup \\ & \{F \rightsquigarrow \neg a | (F, c) \in D \wedge c = ?a\} \end{aligned}$$

and superiority relation

$$\succ = \{(r1, r2) | r1, r2 \in R \wedge \mathit{Ante}(r2) \subset \mathit{Ante}(r1)\}.$$

This theory describes the dataset, for given any record $r = (F, c) \in D$, we have $T, F \vdash c$. By the reasoning mechanism of defeasible logic, we know that only the rules in

$$\begin{aligned} \{r | r \in R \wedge \mathit{Ante}(r) \subseteq F\} = \\ \{r | r \in R \wedge \mathit{Ante}(r) = F\} \cup \\ \{r | r \in R \wedge \mathit{Ante}(r) \subset F\} \end{aligned}$$

can be applied in the reasoning process. The dataset is consistent (by assumption), and the theory has been constructed from each element of the dataset, so we know that the theory T will contain either exactly one defeasible rule with $\mathit{Ante}(r) = F$ and conclusion c, or exactly two defeaters with $\mathit{Ante}(r) = F$ which will give conclusion $?a = c$. This means that applying the rules in

$$\{r | r \in R \wedge \mathit{Ante}(r) = F\}$$

will give us the correct conclusion. It is clear that the rules in

$$\{r | r \in R \wedge \mathit{Ante}(r) \subset F\}$$

will have no effect on this outcome, for the superiority relation defines those rules where $\mathit{Ante}(r) \subset F$, to be weaker than the rules where $\mathit{Ante}(r) = F$. □

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 the 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. Finding a minimal theory also simplifies the comprehension effort required to have a human expert verify a theory, and more closely matches our expectations that human experts in their own practice would themselves typically avoid reasoning with convoluted rules or on an individual case-based-reasoning approach. Unfortunately, though, finding the minimal theory turns out to be intractable, we show this by transformation from the hitting set problem.

Garey and Johnson report that the hitting set problem is NP-complete (by transformation from the vertex cover problem by Karp in 1972). The problem is as follows:

Input Given a collection C of subsets of a finite set S, and a positive integer $k \leq |S|$

Problem Is there a subset $S' \subseteq S$ with $|S'| \leq k$ such that S' contains at least one element from each subset in C.

Theorem 2 *Given a dataset D and 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-hard. (We refer to this problem as the Describing Theory Problem.)*

Proof. If we assume the existence of an algorithm for solving the describing theory problem, we can solve the hitting set problem (Problem SP8 in (Garey & Johnson 1979)) via transformation to the inputs of such an algorithm:

- The collection of the hitting set problem can be encoded in polynomial time to a dataset, then
- An algorithm that solves the describing theory problem can be used to reach a decision for the hitting set problem.

The describing theory problem is therefore NP-hard, because if there exists a polynomial-time algorithm that can solve the describing theory problem, we can solve the hitting set problem in polynomial time, and therefore *all* NP-complete problems in polynomial time. This transformation is possible because of a correspondence between a minimum hitting set and a minimum describing theory.

Given a collection C of subsets of a finite set S, and a positive integer $k \leq |S|$, these inputs are encoded as follows:

- Create a new propositional variable, a, not appearing in S,
- Construct a dataset
$$D = \{(E, a) | E \in C \wedge E \neq \emptyset\} \cup \{(\emptyset, \neg a)\},$$
- Execute an algorithm that solves the describing theory problem, with input D and $k' = k + 1$, if the algorithm returns "true", then there exists a hitting set for C of size k.

Noting the correspondence between a minimum hitting set and a minimum describing theory, the correctness of the above encoding can be seen:

The dataset, D, of the above construction has several interesting properties. Any theory that describes the dataset must contain the rule $\emptyset \Rightarrow \neg a$. This is because when reasoning about the record $d = (\emptyset, \neg a) \in D$, the only applicable rules are those with $Ante(r) = \emptyset$, so because we need to reach conclusion $\neg a$ given the facts $F = \emptyset$, we must have a rule of the form $\emptyset \Rightarrow \neg a$. Furthermore, any other rule in the theory will be stronger than the rule $\emptyset \Rightarrow \neg a$ and will be of the form $F \Rightarrow a$ for $F \neq \emptyset$ because all the remaining records $d \in D \backslash \{(\emptyset, \neg a)\}$ have the conclusion a. While it is possible for a theory to describe the dataset with additional rules such as defeaters or rules of the form $F \Rightarrow \neg a$ for $F \neq \emptyset$, a simpler theory can be produced by eliminating such redundant rules and for this reason a minimal theory would not contain such rules.

Now, we note that when reasoning about some facts, $F \neq \emptyset$, with the rules of a minimal theory $T_{min} = (R, \succ)$ as above, a given rule $r \in R$ is applicable if $Ante(r) \subseteq F$. In fact, a minimal describing theory of a dataset D is a theory such that for each record $d = (F, c) \in D$ where $F \neq \emptyset$, it holds that

$$\exists r \in R \mid Ante(r) \subseteq F$$

(this property is quite close to the definition of a hitting set). Finally, by noting that if we have a rule $r = (Q \Rightarrow a)$ that applies to a record $d = (F, c) \in D$, we can choose any element $s \in Q$ to produce a new rule $r' = (\{s\} \Rightarrow a)$ so that it is still the case that

$$Ante(r') \subseteq F \wedge Ante(r') \neq \emptyset.$$

Thus, if we are given a minimal theory T_{min} for a dataset that has been constructed from a hitting set problem, we can simplify each rule (with the positive conclusion a) in the theory T_{min} to give a new theory

$$T'_{min} = (R', \succ')$$

so that the premise of the rule is a singleton set. This simplification neither adds nor removes rules, so we still have a minimal theory – but by taking the union of the premises of each rule we have a minimum hitting set

$$S' = \cup\{Ante(r) | r \in R'\}.$$

We see that this is indeed the case because for each record $d = (F, c) \in D$, it holds that

$$\exists r \in R' \mid Ante(r) \subseteq F$$

and since each $Ante(r)$ is singleton, this is equivalent to

$$\exists s \in S' \mid s \in F$$

(i.e., S' is a hitting set). By the same reasoning, we can construct a minimal describing theory from a minimum hitting set by creating a rule with singleton premise set for each element of the hitting set (of course, in addition to the rule $\emptyset \Rightarrow \neg a$).

That is, given a minimal theory T_{min}, size $k' = |R|$, of such a dataset, we can produce a minimum hitting set size,

$$k = S' = k' - 1,$$

of the corresponding collection C, by selecting from the premise of each rule one element. And likewise, we can produce a minimal theory from a minimum hitting set. So that if there is a theory of size $|R| \leq k'$ that describes the dataset, then there will exist a hitting set of size

$$|S'| \leq k = k' - 1$$

for the collection. □

Theorem 3 *Given a dataset D and 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.*

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

Figure 2: Defeasible Theory Search Algorithm

Proof. The decision procedure can be solved in NP time. An algorithm is as follows:

1. Non-deterministically, generate a theory T of size $|R| \leq k'$ over the propositional variables in D.
2. For each record $(F, c) \in D$, use the linear time algorithm in (Maher 2001) to check $T, F \vdash c$.
3. If the theory T describes the dataset D, succeed.

The correctness of this algorithm can be seen immediately, for it simply checks that the nondeterministically generated theory of step 1 describes the dataset. Clearly, this algorithm runs in nondeterministic polynomial time. Given that the decision procedure is also NP-hard (Theorem 2) we conclude that the decision procedure is NP-complete. □

4 A New Algorithm

While the above results pertain to a decision procedure, the problem of finding the *smallest* theory that describes a dataset belongs to the class of NP Optimisation problems (defined in (Ausiello, Crescenzi, Gambosi, Kann, Marchetti-Spaccamela & Protasi 1999)), and is consequently NP-hard. These results follow immediately from the definition of NPO, or can be proven via transformation from Problem SP7 in (Ausiello et al. 1999) using a similar mapping as that used Theorem 2.

The important consequence of these results is that it is unlikely that there is a tractable algorithm that finds the global optimum, but that it is necessary to use heuristics to find an approximate solution.

In light of the inherent difficulty of the problem, it is necessary to take a pragmatic approach in seeking an algorithm that produces “reasonable” theories in “reasonable” time. A new algorithm, HeRO, that uses a greedy, branch-and-bound, best-first search strategy, suits these criteria – producing meaningful output on realistic data.

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:

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

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:

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

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

```
set best gain so far, $bg \leftarrow 0$
set best premises, $bp \leftarrow$ nil
set best conclusion, $bc \leftarrow$ 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, $q \leftarrow \emptyset$
        using $q$ enqueue $\emptyset$ with priority 0
        while $q \neq \emptyset$
                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 $bg \leftarrow g$
                        set $bp \leftarrow p$
                        set $bc \leftarrow c$
                if $mg > bg$
                        foreach refinement $p'$ of $p$
                                $q$.enqueue($p'$)
if $bg > 0$
        return ($bg \Rightarrow c$) and current position
else
        return nil
```

Figure 3: Rule Search Routine

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 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, and in fact, produces a theory that is easier for a human expert to comprehend since such a theory represents an ordered list of rules, as opposed to a digraph of rules that is more difficult to interpret.

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.

5 Comparison with Other Approaches

The HeRO algorithm has been implemented in C#. The importance of set-based operations in the algorithm suggests that an efficient implementation must makes careful use of hash-based data structures that 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 our implementation of HeRO with other algorithms that attempt to accomplish similar ends.

This research was motivated by work by Governatori and Stranieri (2001) that explored the semantic overlap between association rules and rules within defeasible logic, but we have attempted to incorporate elements of inductive logic programming and of association rule mining so as to produce accurate theories. The original intent of the work has been followed by John Avery, in the development of a system that generates defeasible logic rules by analyzing the output of the Apriori (Agrawal & Srikant 1998) 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 (Agrawal & Srikant 1998, Han, Pei & Yin 2000)) for association rule mining, and then post-process the association rules to identify a suitable defeasible logic theory.

John Avery, in private communication, reports some preliminary results from his research implementing an association rule based algorithm. Their algorithm, DefGen, that faithfully follows the original intent of the work by Governatori and Stranieri (2001) has been tested against a Japanese credit application dataset [3] dataset for benchmarking machine learning algorithms that contains 125 records and is supplied with a Lisp domain theory. 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 propositional variables. Avery 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

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

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

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.

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 (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 and in such situations, while comparable to other approaches, this particular methodology suggests no distinct argument for its use.

Bench-Capon (1993) highlights the opportunities for data mining within the legal domain and trains a neural network on a dataset for a fictitious welfare benefit. In a later work (Bench-Capon, Coenen & Leng 2000), he also compares these results with the unprocessed output of an association rule-mining algorithm that has been executed with the same dataset. The synthetic dataset used by Bench-Capon is similar to the Japanese Credit Application Dataset, and gives equally promising theories when HeRO is executed.

The dataset used in the experiments describes a fictional welfare benefit paid to pensioners that suffer expenses visiting a spouse in hospital. Each record has a binary conclusion, whether they receive the benefit or not, and Bench-Capon explains that this payment is contingent upon the satisfaction of 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, in-patient, distance), which form the dataset along with which 52 additional "noise" features that have no influence on the conclusion.

Neural networks trained on 2400 such records obtained the following success rates:

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

Even though these are very promising success rates, the opacity of neural network techniques to verification and analysis makes it difficult to rationalize the use such technologies in legal practice.

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):

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

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):

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

These defeasible theories correspond very closely to the conditions in the actual domain theory. That HeRO has not found every condition appears to be symptomatic of the fact that the dataset is synthetic and that the domain theory contains an n-out-of-m constraint that is difficult to express in defeasible logic. Given that at an accuracy of 99.8%, only 4 records remain misclassified – it appears that the remaining conditions for the welfare benefit are simply not exercised within the dataset. Though, needless to say, 99.8% accuracy is an impressive result even if the other benefits of using defeasible logic over more opaque knowledge representations such as neural networks are disregarded.

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.

6 Future Work

The algorithm presented in this paper bears similarity to algorithms used for inducing theories in the monotonic horn-clause logic used by the Inductive Logic Programming (ILP) community. Though the nonmonotonic nature of defeasible logic renders established ILP algorithms inapplicable without significant post-processing, it is worthwhile further exploring a mechanism for translation between horn-clause logic and defeasible logic.

Several decision tree induction algorithms (such as C4.5(Quinlan 1993)) 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.

7 Conclusions

The challenge of representing and encoding knowledge within legal decision support systems can be solved by the use of defeasible logic. Defeasible logic provides a faithful representation of legal regulation within a formal framework, and now has a mechanism by which the legal knowledge can be automatically encoded from a set of precedents. Though finding a minimal theory to describe a given dataset is intractable, the combination of heuristics in HeRO results in a practical algorithm that generates meaningful theories without excessive runtimes.

References

Agrawal, R. & Srikant, R. (1998), Fast algorithms for mining association rules, *in* M. Stonebraker & J. Hellerstein, eds, 'Readings in Database Systems', 3rd edn, Morgan Kaufmann Publishers, chapter 7, pp. 580–592.

Antoniou, G., Billington, D., Governatori, G. & Maher, M. (1999), On the modelling and analysis of regulations, *in* 'Proceedings of the 10th Australasian Conference on Information Systems', pp. 20–29.

Antoniou, G., Billington, D., Governatori, G. & Maher, M. (2000), A flexible framework for defeasible logics, *in* 'AAAI/IAAI', pp. 405–410.

Antoniou, G., Billington, D., Governatori, G. & Maher, M. (2001), 'Representation results for defeasible logic', *ACM Transactions on Computational Logic* **2**(2), 255–287.

Antoniou, G., Billington, D., Governatori, G., Maher, M. & Rock, A. (2000), A family of defeasible reasoning logics and its implementation, *in* 'Proceedings of the 14th European Conference on Artificial Intelligence', IOS Press, Amsterdam.

Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A. & Protasi, M. (1999), *Complexity and Approximation: Combinatorial Optimization Problems and their Approximability Properties*, Springer-Verlag, chapter Appendix B, p. 426. See also `http://www.nada.kth.se/~viggo/problemlist/`.

Bench-Capon, T. (1993), Neural networks and open texture, *in* 'Proceedings of the Fourth International Conference on Artificial Intelligence and Law', ACM Press.

Bench-Capon, T., Coenen, F. & Leng, P. (2000), An experiment in discovering association rules in the legal domain, *in* 'Proceedings of the Eleventh International Workshop on Database and Expert Systems Applications', IEEE Computer Society, Los Alamitos, California, pp. 1056–1060.

Billington, D. (1993), 'Defeasible logic is stable', *Journal of Logic and Computation* **3**, 370–400.

Brüninghaus, S. & Ashley, K. (1999), Toward adding knowledge to learning algorithms for indexing legal cases, *in* 'The Seventh International Conference on Artificial Intelligence and Law', pp. 9–17.

Garey, M. & Johnson, D. (1979), *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, chapter Appendix A, p. 222.

Ginsberg, M. (1993), AI and nonmonotonic reasoning, *in* D. Gabbay, C. Hogger & J. Robinson, eds, 'Handbook of Logic in Artificial Intelligence and Logic Programming', Vol. 3, Oxford University Press.

Governatori, G. & Stranieri, A. (2001), Towards the application of association rules for defeasible rule discovery, *in* B. Verheij, A. Lodder, R. Loui & A. Muntjewerff, eds, 'Frontiers in Artificial Intelligence and Applications', Vol. 70, IOS Press. Proceedings of JURIX 2001.

Han, J., Pei, J. & Yin, Y. (2000), Mining frequent patterns without candidate generation, *in* W. Chen, J. Naughton & P. Bernstein, eds, '2000 ACM-SIGMOD International Conference on Management of Data', ACM Press, Dallas, Texas, pp. 1–12.

Maher, M. (2001), 'Propositional defeasible logic has linear complexity', *Theory and Practice of Logic Programming* **1**(6), 691–711.

Nute, D. (1993), Defeasible logic, *in* D. Gabbay, C. Hogger & J. Robinson, eds, 'Handbook of Logic in Artificial Intelligence and Logic Programming', Vol. 3, Oxford University Press.

Prakken, H. (1997), *Logical Tools for Modelling Legal Argument*, Kluwer Academic Publishers.

Quinlan, J. (1993), *C4.5: programs for machine learning*, Morgan Kaufmann, San Mateo, California.

Zeleznikow, J. & Hunter, D. (1994), *Building Intelligent Legal Information Systems*, Computer/Law Series, Kluwer Law and Taxation Publishers.

Zeleznikow, J. & Stranieri, A. (1997), Knowledge discovery in the split up project, *in* 'The Sixth International Conference on Artificial Intelligence and Law'.