A Defeasible Logic for Clauses

Author

Billington, David

Published

2011

Journal Title

Lecture Notes in Computer science

DOI

10.1007/978-3-642-25832-9_48

A Defeasible Logic for Clauses

David Billington

School of Information and Communication Technology, Nathan campus,
Griffith University, Brisbane, Queensland 4111, Australia.
d.billington@griffith.edu.au

Abstract. A new non-monotonic logic called clausal defeasible logic (CDL) is defined and explained. CDL is the latest in the family of defeasible logics, which, it is argued, is important for knowledge representation and reasoning. CDL increases the expressive power of defeasible logic by allowing clauses where previous defeasible logics only allowed literals. This greater expressiveness allows the representation of the Lottery Paradox, for example. CDL is well-defined, consistent, and has other desirable properties.
Keywords: Defeasible logic, Non-monotonic reasoning, Knowledge representation and reasoning, Artificial intelligence.

1 Introduction

Non-monotonic reasoning systems represent and reason with incomplete information where the degree of incompleteness is not quantified. A very simple and natural way to represent such incomplete information is with a defeasible rule of the form "antecedent $\Rightarrow$ consequent"; with the meaning that provided there is no evidence against the consequent, the antecedent is sufficient evidence for concluding the consequent. Creating such rules is made easier for the knowledge engineer as each rule need only be considered in isolation. The interaction between the rules is the concern of the logic designer.

Reiter's normal defaults [23] have this form, with the meaning that if the antecedent is accepted and the consequent is consistent with our knowledge so far then accept the consequent. Of course the consequent could be consistent with current knowledge and yet there be evidence against the consequent. This results in multiple extensions. However multiple extensions are avoided by interpreting a defeasible rule as "if the antecedent is accepted and all the evidence against the consequent has been nullified then accept the consequent". This interpretation forms the foundation of a family of non-monotonic logics all based on Nute's original defeasible logic [21]. (Different formal definitions of "accepted", "evidence against", and "nullified" have been used by different defeasible logics.)

Unlike other non-monotonic reasoning systems, these defeasible logics use Nute's very simple and natural "defeasible arrow" to represent incomplete information. This simplicity and naturalness is important when explaining an implementation to a client. All defeasible logics have a priority relation on rules and use classical negation rather than negation-as-failure.

A key feature of these defeasible logics is that they all have efficient easily implementable deduction algorithms [10, 20, 22]. Indeed defeasible logics have been used in an expert system, for learning and planning [22], in a robotic dog which plays soccer [4, 5], in a robotic poker player [6], to improve the accuracy of radio frequency identification [11], to model the behaviour of autonomous robots [9], and to facilitate the encoding of software requirements [8] so they can be automatically translated into a programming language [7]. Defeasible logics have been advocated for various applications including modelling regulations and business rules [1], agent negotiations [13], the semantic web [2, 3, 25], modelling agents [16], modelling intentions [15], modelling dynamic resource allocation [17], modelling contracts [12], legal reasoning [18], modelling deadlines [14], and modelling dialogue games [24]. Moreover, defeasible theories, describing policies of business activities, can be mined efficiently from appropriate datasets [19].

The unique features and diverse range of practical applications show that defeasible logics are useful and their language is important for knowledge representation and reasoning. Using defeasible logic as the inference engine in an expert system is obvious. But it is less obvious to use defeasible logic to deal with the error-prone output of sensors (possibly in a robot), because this can be done using classical logic. The advantages of using defeasible logic are that the system can be developed incrementally, there are fewer rules, and the rules are simpler [4, 5, 11].

In this article we shall define a new defeasible logic, called clausal defeasible logic (CDL) and explain how it works. The main purpose of CDL is to increase the expressive power of defeasible logic by allowing clauses where previous defeasible logics only allowed literals.

The rest of the paper has the following organisation. Section 2 gives an overview of CDL. The formal definitions of CDL are in Section 3. An example is considered in Section 4. The results in Section 5 show that CDL is well-defined, consistent, and has other desirable properties. A summary forms Section 6.

2 Overview of Clausal Defeasible Logic (CDL)

CDL reasons with both factual and plausible information. The factual information is represented by strict rules of the form $A \to c$ where A is a finite set of literals and c is a clause. If all the literals in A are proved then c can be deduced, no matter what the evidence against c is.

The plausible information is represented by defeasible rules, warning rules, and a priority relation, $>$, on these rules.

Defeasible rules have the form $A \Rightarrow c$ where A is a finite set of clauses and c is a clause. If all the clauses in A are proved and all the evidence against c has been nullified then c can be deduced. For example, "Birds usually fly." can be represented by $\{b\} \Rightarrow f$.

Warning rules, for example $A \rightsquigarrow \neg l$, warn against concluding usually l, but do not support usually $\neg l$. For example, "Sick birds might not fly." can be represented by $\{s, b\} \rightsquigarrow \neg f$. The idea is that a bird being sick is not sufficient

evidence to conclude that it usually does not fly; it is only evidence against the conclusion that it usually flies.

There is an acyclic priority relation between non-strict rules; $r_2 > r_1$ means that r_2 is preferred over r_1.

CDL has four major proof algorithms, μ, ρ, π, and β, which cater for different intuitions about what should follow from a reasoning situation. The μ algorithm is monotonic and uses only strict rules. CDL restricted to μ is essentially classical propositional logic. The ambiguity blocking algorithm is denoted by β. There are two ambiguity propagating algorithms denoted by ρ and π, ρ being more reliable than π.

The task of proving a formula is done by a recursive proof function P. The input to P is the proof algorithm to be used, the formula to be proved, and the background. The background is an initially empty storage bin into which is put all the clauses that are currently being proved as P recursively calls itself. The purpose of this background is to detect loops. The output of P is one of the following proof-values $+1$, 0, or -1. The $+1$ means that the formula is proved in a finite number of steps, 0 means that the proof got into a loop which was detected in a finite number of steps, and -1 means that in a finite number of steps it has been demonstrated that there is no proof of the formula and that this demonstration does not get into a loop.

3 Clausal Defeasible Logic (CDL)

Atm is a non-empty countable set of (propositional) atoms, and $Lit = Atm \cup \{\neg a : a \in Atm\}$ is the set of all literals. If C is a clause or a set of clauses then $Lit(C)$ denotes the set of literals in C. The set of all tautologies is denoted by $Taut$. The set of all clauses which are resolution-derivable from the set C of clauses is denoted by $Res(C)$.

The **complement**, $\sim l$, of a literal l is defined as follows. If a is an atom then $\sim a$ is $\neg a$, and $\sim\neg a$ is a. If L is a set of literals then $\sim L = \{\sim l : l \in L\}$.

Definition 1. A **rule**, r, is any triple $(A(r), arrow(r), c(r))$, such that $A(r)$ is a finite set of non-empty clauses called the set of **antecedents** of r, $arrow(r) \in \{\rightarrow, \Rightarrow, \rightsquigarrow\}$, $c(r)$ is a non-empty clause called the **consequent** of r, and if $arrow(r)$ is $\rightarrow$ then $A(r)$ is a finite set of literals.

Strict rules use the **strict arrow**, $\rightarrow$, and are written $A(r) \rightarrow c(r)$. **Defeasible rules** use the **defeasible arrow**, $\Rightarrow$, and are written $A(r) \Rightarrow c(r)$. **Warning rules** use the **warning arrow**, $\rightsquigarrow$, and are written $A(r) \rightsquigarrow c(r)$.

Definition 2. Let R be any set of rules, R_s be any set of strict rules, C be any set of clauses, L be any set of literals, c be any clause, and l be any literal.

$R_s = \{r \in R : r \text{ is a strict rule}\}$. $R[\vee L] = \{r \in R : c(r) = \vee L\}$.
$R_d = \{r \in R : r \text{ is a defeasible rule}\}$. $R[c; 1] = \{r \in R[c] : |A(r)| \leq 1\}$.
$R_w = \{r \in R : r \text{ is a warning rule}\}$. $R[C] = \{r \in R : c(r) \in C\}$.
$R_{dw} = R_d \cup R_w$. $Cl(R_s) = \{\vee(L \cup \sim A) : A \rightarrow \vee L \in R_s\}$.
$Ru(C) = \{\sim(L - K) \rightarrow \vee K : \vee L \in C \text{ and } \{\} \subset K \subseteq L\}$.

Cl converts a strict rule to a clause. Conversely, Ru converts a clause with n literals to $2^n - 1$ strict rules.

Definition 3. If R is a set of rules then $>$ is a **priority relation** on R iff $>$ is an acyclic binary relation on R_{dw}. $R[l;s] = \{t \in R[l] : t > s\}$.

Let C be a set of clauses. We want to remove from C all the clauses which are empty, or tautologies, or are strict superclauses of other clauses. The result is called the reduct of C, $Red(C)$. A set of literals such that each clause in C contributes exactly one literal to the set is called a transversal of C.

To help prove clauses we "move literals from the antecedent to the consequent". For instance the rule $\{a, b\} \Rightarrow c$ generates the rules $\{a\} \Rightarrow \vee\{\sim b, c\}$ and $\{\} \Rightarrow \vee\{\sim a, \sim b, c\}$.

Definition 4. Let R be any set of rules. The set, $Gen(R)$, of **rules generated from** R is defined by $Gen(R) = \{(A(r)\text{–}C)\ arrow(r)\ \vee(L \cup \sim T) : r \in R,\ C \subseteq A(r),\ c(r) = \vee L$, and T is a transversal of $C\}$.

Definition 5. Let R be a finite set of rules. The ordered pair $(R, >)$ is called a **clausal defeasible theory** (**cdt**) iff DT1, DT2, and DT3 all hold.
DT1) $R_s = Ru(Red(Res(Cl(R_s))))$.
DT2) $R_{dw} = Gen(R_{dw})$.
DT3) $>$ is a priority relation on R_{dw}.

Definition 6. Let $\Theta = (R, >)$ be a cdt. The set $Ax(\Theta)$ of **axioms** of Θ is defined by $Ax(\Theta) = Cl(R_s)$. Define $A^*(R_s[l;1]) = \{A(r) : r \in R_s[l;1]\} \cup \{\{l\}\}$.

To cater for various intuitions we will introduce the following proof algorithms: μ, ρ, π, β, π', and ρ', which are explained after their formal definition.

Definition 7. Suppose $\lambda \in \{\rho, \pi, \beta\}$. Then λ' is the **co-algorithm** of λ, where $\beta' = \beta$. Moreover we define $(\lambda')' = \lambda$.

A **clausal defeasible logic** consists of a clausal defeasible theory Θ and its **proof function**, P. To define P we shall define some auxiliary functions and the proof algorithms $\mu, \rho, \pi, \beta, \pi', \rho'$. For non-empty sets max and min have their usual meaning. But we also define $\max\{\} = -1$, and $\min\{\} = +1$. We now define P, its auxiliary functions, and the proof algorithms.

Definition 8. Suppose $\lambda \in \{\mu, \rho, \pi, \beta, \pi', \rho'\}$, C is a finite set of clauses, and L is a finite set of literals.
$P\lambda$set) $P(\lambda, C, B) = \min\{P(\lambda, c, B) : c \in C\}$.
$P\lambda$taut) If $\vee L$ is a tautology then $P(\lambda, \vee L, B) = +1$.
$P\lambda$mt) $P(\lambda, \vee\{\}, B) = -1$.

Definition 9. Suppose $\lambda \in \{\rho, \pi, \beta, \pi', \rho'\}$, L is a non-empty finite set of literals, $\vee L$ is not a tautology, and l is a literal.

$P\mu$cl) If $\{\} \subset K \subseteq L$ and $\vee K \in Cl(R_s)$ then $P(\mu, \vee L, B) = +1$; else $P(\mu, \vee L, B) = -1$.

$P\lambda 0$) If $\vee L \in B$ then $P(\lambda, \vee L, B) = 0$.

$P\lambda$pc) If $\vee L \notin B$ and $|L| \geq 2$ then $P(\lambda, \vee L, B) = \max(\{P(\lambda, l, B) : l \in L\} \cup \{P(\lambda, A(r), \{\vee K\} \cup B) : K \subseteq L,\ |K| \geq 2, \text{ and } r \in R_s[\vee K; 1] \cup R_d[\vee K]\})$.

$P\lambda$lit) If $l \notin B$ then $P(\lambda, l, B) = \max(\{P(\lambda, A(r), \{l\} \cup B) : r \in R_s[l; 1]\} \cup \{Plaus(\lambda, l, B)\})$.

Definition 10. Suppose $\lambda \in \{\rho, \pi, \beta\}$, l is a literal, B is a finite set of clauses, and $l \notin B$.

$Plaus\rho'$) $Plaus(\rho', l, B) = \max\{P(\rho', A(r), \{l\} \cup B) : r \in R_d[l]\}$.

$Plaus\lambda$) $Plaus(\lambda, l, B) = \min(\{For(\lambda, l, B)\} \cup \{Nulld(\lambda, l, B, I) : I \in A^*(R_s[\sim l; 1])\})$.

$Plaus\pi'$) $Plaus(\pi', l, B) = \max\{Evid(\pi', l, B, r) : r \in R_d[l]\}$.

$For\lambda$) $For(\lambda, l, B) = \max\{P(\lambda, A(r), \{l\} \cup B) : r \in R_d[l]\}$.

$Evid\pi'$) $Evid(\pi', l, B, r) = \min(\{P(\pi', A(r), \{l\} \cup B)\} \cup \{Nulldr(\pi', l, B, r, I) : I \in A^*(R_s[\sim l; 1])\})$.

$Nulld\lambda$) $Nulld(\lambda, l, B, I) = \max\{Discred(\lambda, l, B, q) : q \in I\}$.

$Nulldr\pi'$) $Nulldr(\pi', l, B, r, I) = \max\{Discredr(\pi', l, B, r, q) : q \in I\}$.

$Discred\lambda$) $Discred(\lambda, l, B, q) = \min\{Dftd(\lambda, l, B, s) : s \in R_s[q; 1] \cup R_{dw}[q]\}$.

$Discredr\pi'$) $Discredr(\pi', l, B, r, q) = \min\{Dftd(\pi', l, B, s) : s \in R[q; r]\}$.

$Dftd\lambda$) Now suppose $\lambda \in \{\rho, \pi, \beta, \pi'\}$. Then $Dftd(\lambda, l, B, s) = \max(\{P(\lambda, A(t), \{l\} \cup B) : t \in R_d[l; s]\} \cup \{-P(\lambda', A(s), \{l\} \cup B)\})$.

Definition 11. Suppose Θ is a cdt, P is the proof function of Θ, c is a clause, and $\lambda \in \{\mu, \rho, \pi, \beta, \pi', \rho'\}$. We define $\Theta(\lambda+) = \{c : P(\lambda, c, \{\}) = +1\}$, and $\Theta(\lambda-) = \{c : P(\lambda, c, \{\}) = -1\}$.

We shall now give some insight into the above proof algorithms. Note that min and max behave like quantifiers. Suppose $\lambda \in \{\mu, \rho, \pi, \beta, \pi', \rho'\}$.

A set of clauses, C, is proved by proving each clause in C. So for $P(\lambda, C, B)$ to be $+1$ each $P(\lambda, c, B)$ must be $+1$, where $c \in C$. Hence $P\lambda$set.

Now consider proving a clause. If a clause, $\vee L$, is a tautology then we declare it proved, whether or not it is in the background. Hence $P\lambda$taut. The empty clause is disproved, whether or not it is in the background. Hence $P\lambda$mt. So suppose $\vee L$ is not a tautology and $L \neq \{\}$.

Apart from the cases considered above, a clause $\vee L$ is proved by proving any non-empty subclause of $\vee L$.

The μ algorithm declares $\vee L$ to be proved if a non-empty subclause of $\vee L$ is an axiom; otherwise it is disproved. Hence $P\mu$cl. So suppose $\lambda \in \{\rho, \pi, \beta, \pi', \rho'\}$.

If $\vee L$ is in the background B then we are already in the process of trying to prove $\vee L$. So we are now in a loop. Hence $P(\lambda, \vee L, B) = 0$ and so $P\lambda 0$. So suppose $\vee L$ is not in B.

To prove a proper clause, $\vee L$, we must either prove a literal which occurs in L, (hence $\{P(\lambda, l, B) : l \in L\}$), or prove a subclause $\vee K$ of $\vee L$ which is proper.

To prove $\vee K$ we need to prove every clause in the set of antecedents of a strict or defeasible rule whose consequent is $\vee K$, provided that the set of antecedents of the strict rule contains at most 1 literal. The need for this restriction is shown by Example 1. We have the strict rule r_7: $\{\neg s_1, \neg s_2\} \rightarrow s_3$, and also by Evaluations $E1$ and $E2$ we can prove both $\neg s_1$ and $\neg s_2$. But the fact that $\neg s_1$ and $\neg s_2$ can be independently proved is not evidence for s_3. Hence $P\lambda$pc.

To prove a literal l we must prove every clause in the set of antecedents of a strict or defeasible rule whose consequent is l. If a strict rule is used then evidence against l need not be considered, and its set of antecedents must contain at most 1 literal, $\max\{P(\lambda, A(r), \{l\} \cup B) : r \in R_s[l; 1]\}$. If a defeasible rule is used then evidence against l must be considered, $Plaus(\lambda, l, B)$. Hence $P\lambda$lit.

The ρ' algorithm ignores all evidence against l. Hence $Plaus\rho'$.

Now consider the ρ, π, and β algorithms. Suppose $\lambda \in \{\rho, \pi, \beta\}$. These algorithms must consider evidence for l, $For(\lambda, l, B)$ and $For\lambda$, and nullify the evidence against l, $\{Nulld(\lambda, l, B, I) : I \in A^*(R_s[\sim l; 1])\}$. Hence $Plaus\lambda$.

The evidence against l consists of any set of literals which is inconsistent with l. The set of all such sets is $A^*(R_s[\sim l; 1])$. The evidence against l is nullified by, for each I in $A^*(R_s[\sim l; 1])$, finding a literal, q, in I such that every rule, s, whose consequent is q, is defeated. The only restriction is that if s is a strict rule then its set of antecedents must contain at most 1 literal. Hence $Nulld\lambda$ and $Discred\lambda$. A rule, s, is defeated by either using the λ' algorithm to disprove a clause in its set of antecedents, $-P(\lambda', A(s), \{l\} \cup B)$, or by using team defeat. That is by finding a defeasible rule t whose consequent is l (a member of the team for l) and which is superior to s, and then proving every clause in the set of antecedents of t, $\{P(\lambda, A(t), \{l\} \cup B) : t \in R_d[l; s]\}$. Hence $Dftd\lambda$.

Finally the π' proof algorithm is similar to the π algorithm, except that we choose one defeasible rule r supporting l and only consider rules s which are superior to r as evidence against l, and hence need defeating.

4 Example

The following example shows how the lottery example can be represented.

Example 1. (The 3-Lottery example)

Consider a fair 3-sided die. Let s_i denote side i. Then for each i in $\{1, 2, 3\}$ it is plausible that the outcome of a roll of this die is not s_i. So for each i in $\{1, 2, 3\}$ we have $\{\} \Rightarrow \neg s_i$. Moreover for each i in $\{1, 2, 3\}$ it is plausible that the outcome of a roll of this die is in $\{s_1, s_2, s_3\} - \{s_i\}$. So for each i in $\{1, 2, 3\}$ we have $\{\} \Rightarrow \vee(\{s_1, s_2, s_3\} - \{s_i\})$. Furthermore the outcome of a roll of this die is exactly one of s_1, s_2, or s_3. So we have $\vee\{s_1, s_2, s_3\}$, $\neg\wedge\{s_1, s_2\}$, $\neg\wedge\{s_1, s_3\}$, $\neg\wedge\{s_2, s_3\}$. Converting these facts to clauses gives: $\vee\{s_1, s_2, s_3\}$, $\vee\{\neg s_1, \neg s_2\}$, $\vee\{\neg s_1, \neg s_3\}$, $\vee\{\neg s_2, \neg s_3\}$.

The cdt $\Theta = (R, >)$ which captures this situation is defined as follows.

The strict rules are: r_1: $\{\} \rightarrow \vee\{s_1, s_2, s_3\}$, r_2: $\{\neg s_1\} \rightarrow \vee\{s_2, s_3\}$, r_3: $\{\neg s_2\} \rightarrow \vee\{s_1, s_3\}$, r_4: $\{\neg s_3\} \rightarrow \vee\{s_1, s_2\}$, r_5: $\{\neg s_2, \neg s_3\} \rightarrow s_1$,

r_6: $\{\neg s_1, \neg s_3\} \to s_2$, r_7: $\{\neg s_1, \neg s_2\} \to s_3$, r_8: $\{\} \to \vee\{\neg s_1, \neg s_2\}$,
r_9: $\{s_1\} \to \neg s_2$, r_{10}: $\{s_2\} \to \neg s_1$, r_{11}: $\{\} \to \vee\{\neg s_1, \neg s_3\}$, r_{12}: $\{s_1\} \to \neg s_3$,
r_{13}: $\{s_3\} \to \neg s_1$, r_{14}: $\{\} \to \vee\{\neg s_2, \neg s_3\}$, r_{15}: $\{s_2\} \to \neg s_3$, r_{16}: $\{s_3\} \to \neg s_2$.

The defeasible rules are: r_{17}: $\{\} \Rightarrow \neg s_1$, r_{18}: $\{\} \Rightarrow \neg s_2$, r_{19}: $\{\} \Rightarrow \neg s_3$,
r_{20}: $\{\} \Rightarrow \vee\{s_1, s_2\}$, r_{21}: $\{\} \Rightarrow \vee\{s_1, s_3\}$, r_{22}: $\{\} \Rightarrow \vee\{s_2, s_3\}$.

There are no warning rules and the priority relation, $>$, is empty.

We show that if λ is in $\{\rho, \pi, \beta\}$ then λ is able to prove every clause in $U(3) = \{\neg s_1, \neg s_2, \vee\{s_1, s_2\}\}$. We note $A^*(R_s[s_1; 1]) = \{\{s_1\}\}$.

Evaluation $E1$

1) $P(\lambda, \neg s_1, \{\}) = \max\{P(\lambda, \{s_2\}, \{\neg s_1\}), P(\lambda, \{s_3\}, \{\neg s_1\}), \mathit{Plaus}(\lambda, \neg s_1, \{\})\}$, by $P\lambda$lit
2) $\mathit{Plaus}(\lambda, \neg s_1, \{\}) = \min\{\mathit{For}(\lambda, \neg s_1, \{\}), \mathit{Nulld}(\lambda, \neg s_1, \{\}, \{s_1\})\}$, by $\mathit{Plaus}\lambda$
3) $\mathit{For}(\lambda, \neg s_1, \{\}) = P(\lambda, \{\}, \{\neg s_1\})$, by $\mathit{For}\lambda$
4) $= \min\{\} = +1$, by $P\lambda$set.
5) $\therefore \mathit{Plaus}(\lambda, \neg s_1, \{\}) = \mathit{Nulld}(\lambda, \neg s_1, \{\}, \{s_1\})\}$, by lines 4, 3, and 2 of $E1$.
6) $= \mathit{Discred}(\lambda, \neg s_1, \{\}, s_1)$, by $\mathit{Nulld}\lambda$
7) $= \min\{\} = +1$, by $\mathit{Discred}\lambda$
8) $\therefore P(\lambda, \neg s_1, \{\}) = +1$, by lines 7, 6, 5, and 1 of $E1$.

A similar evaluation, say $E2$, shows that $P(\lambda, \neg s_2, \{\}) = +1$. Finally we show that $\vee\{s_1, s_2\}$ is provable by any λ in $\{\rho, \pi, \beta\}$.

Evaluation $E3$

1) $P(\lambda, \vee\{s_1, s_2\}, \{\}) = \max\{P(\lambda, \{s_1\}, \{\}), P(\lambda, \{s_2\}, \{\}),$
$P(\lambda, \{\neg s_3\}, \{\vee\{s_1, s_2\}\}), P(\lambda, \{\}, \{\vee\{s_1, s_2\}\})\}$, by $P\lambda$pc
2) $P(\lambda, \{\}, \{\vee\{s_1, s_2\}\}) = \min\{\} = +1$, by $P\lambda$set
3) $\therefore P(\lambda, \vee\{s_1, s_2\}, \{\}) = +1$, by lines 2 and 1 of $E3$.

EndExample1

5 Results

Our first result shows that the proof function P really is a function. (The proofs of these and other results are in the full article available from the author.)

Theorem 1. Let $\Theta = (R, >)$ be a clausal defeasible theory. Then P is a function with co-domain $\{+1, 0, -1\}$.

The next theorem says that CDL has a linear proof hierarchy, and also the reverse for disproof. This shows that proofs with different levels of confidence can be achieved without using numbers.

Theorem 2. Suppose Θ is a clausal defeasible theory.
(1) $\Theta(\mu+) \subseteq \Theta(\rho+) \subseteq \Theta(\pi+) \subseteq \Theta(\beta+) \subseteq \Theta(\pi'+) \subseteq \Theta(\rho'+)$.
(2) $\Theta(\rho'-) \subseteq \Theta(\pi'-) \subseteq \Theta(\beta-) \subseteq \Theta(\pi-) \subseteq \Theta(\rho-) \subseteq \Theta(\mu-)$.

Our final result is that CDL is consistent; that is, each pair of provable clauses is satisfiable, which as the lottery example shows is the desired result.

Theorem 3. Suppose $\Theta = (R, >)$ is a clausal defeasible theory, $Ax(\Theta)$ is consistent, and $\lambda \in \{\mu, \rho, \pi, \beta\}$. If $\{c_1, c_2\} \subseteq \Theta(\lambda+)$ then $\{c_1, c_2\}$ is satisfiable.

6 Summary

In Section 1 we argued that, among non-monotonic logics, the family of defeasible logics is important for knowledge representation and reasoning. Defeasible logics are powerful enough for a diverse range of practical applications, and yet their language has a unique combination of expressiveness, simplicity, and naturalness.

The rules of previous defeasible logics were constructed from literals, but the non-strict rules of CDL are constructed from clauses. This greater expressiveness allows the representation of the Lottery Paradox. The results listed in Section 5 are some of the more important results required to show that CDL is well-defined and consistent.

Acknowledgement

The author wishes to thank Andrew Rock, René Hexel, and Vladimir Estivill-Castro for their assistance and advice; and Michael Maher and Kewen Wang for their comments on a draft of an earlier version of this article.

References

1. G. Antoniou, D. Billington, and M. J. Maher. On the analysis of regulations using defeasible rules. In *Proceedings of the 32nd Hawaii International Conference on Systems Science*. IEEE Press, January 1999.
2. Grigoris Antoniou. Nonmonotonic rule system on top of ontology layer. In *ISWC*, volume 2432 of *Lecture Notes in Computer Science*, pages 394–398. Springer, 2002.
3. N. Bassiliades, G. Antoniou, and I. Vlahavas. Dr-device: A defeasible logic system for the semantic web. In *2nd Workshop on Principles and Practice of Semantic Web Reasoning*, volume 3208 of *Lecture Notes in Computer Science*, pages 134–148, 2004.
4. D. Billington, V. Estivill-Castro, R. Hexel, and A. Rock. Non-monotonic reasoning for localisation in robocup. In *Proceedings of the 2005 Australasian Conference on Robotics and Automation*, 2005.
5. D. Billington, V. Estivill-Castro, R. Hexel, and A. Rock. Using temporal consistency to improve robot localisation. In *Proceedings of the 10th RoboCup International Symposium*, volume 4434 of *Lecture Notes in Artificial Intelligence*, pages 232–244. Springer, 2007.
6. D. Billington, V. Estivill-Castro, R. Hexel, and A. Rock. Architecture for hybrid robotic behavior. In E. Corchado, X. Wu, E. Oja, A. Herrero, and B. Baruque, editors, *Hybrid Artificial Intelligence Systems (HAIS09)*, volume 5572 of *Lecture Notes in Artificial Intelligence*, pages 145–156. Springer, June 2009.
7. D. Billington, V. Estivill-Castro, R. Hexel, and A. Rock. Modelling behaviour requirements for automatic interpretation, simulation and deployment. In *2nd International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR2010)*, volume 6472 of *Lecture Notes in Artificial Intelligence*, pages 204–216. Springer, November 2010.

8. D. Billington, V. Estivill-Castro, R. Hexel, and A. Rock. Non-monotonic reasoning for requirements engineering: State diagrams driven by plausible logic. In *5th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE2010)*, Communications in Computer and Information Science (CCIS). Springer-Verlag, July 2010.
9. D. Billington, V. Estivill-Castro, R. Hexel, and A. Rock. Plausible logic facilitates engineering the behaviour of autonomous robots. In R. Fox and W. Golubski, editors, *The IASTED International Conference on Software Engineering 2010*, pages 41–48. ACTA Press, February 2010.
10. D. Billington and A. Rock. Propositional plausible logic: Introduction and implementation. *Studia Logica*, 67(2):243–269, 2001.
11. P. Darcy, B. Stantic, and R. Derakhshan. Correcting stored rfid data with non-monotonic reasoning. *International Journal of Principles and Applications of Information Science and Technology*, 1(1):65–77, December 2007.
12. G. Governatori. Representing business contracts in ruleml. *International Journal of Cooperative Information Systems*, 14(2-3):181–216, 2005.
13. G. Governatori, M. Dumas, A. H. ter Hofstede, and P. Oaks. A formal approach to protocols and strategies for (legal) negotiation. In *Proceedings of the 8th International Conference on Artificial Intelligence and Law*, pages 168–177. ACM Press, 2001.
14. G. Governatori, J. Hulstijn, R. Riveret, and A. Rotolo. Characterising deadlines in temporal modal defeasible logic. In *Proceedings of the 20th Australian Joint Conference on Artificial Intelligence*, volume 4830 of *Lecture Notes in Artificial Intelligence*, pages 486–496. Springer, 2007.
15. G. Governatori and V. Padmanabhan. A defeasible logic of policy-based intention. In T. D. Gedeon and L. C. C. Fung, editors, *AI 2003: Advances in Artificial Intelligence*, volume 2903 of *Lecture Notes in Artificial Intelligence*, pages 414–426. Springer, 2003.
16. G. Governatori and A. Rotolo. Defeasible logic: Agency, intention and obligation. In A. Lomuscio and D. Nute, editors, *Deontic Logic in Computer Science*, volume 3065 of *Lecture Notes in Artificial Intelligence*, pages 114–128. Springer, 2004.
17. G. Governatori, A. Rotolo, and S. Sadiq. A model of dynamic resource allocation in workflow systems. In K. D. Schewe and H. E. Williams, editors, *Database Technology 2004. Conference Research and Practice of Information Technology*, volume 27, pages 197–206. Australian Computer Science Association, ACS, 2004.
18. G. Governatori, A. Rotolo, and G. Sartor. Temporalised normative positions in defeasible logic. In *Proceedings of the 10th International Conference on Artificial Intelligence and Law*, pages 25–34. ACM Press, 2005.
19. B. Johnston and G. Governatori. An algorithm for the induction of defeasible logic theories from databases. In K.D. Schewe and X. Zhou, editors, *Database Technology 2003. Conference Research and Practice of Information Technology*, volume 17, pages 75–83. Australian Computer Science Association, Australian Computer Science Association, 2003.
20. M. J. Maher, A. Rock, G. Antoniou, D. Billington, and T. Miller. Efficient defeasible reasoning systems. *International Journal of Artificial Intelligence Tools*, 10(4):483–501, 2001.
21. D. Nute. Defeasible reasoning. In *Proceedings of the 20th Hawaii International Conference on System Science*, pages 470–477. University of Hawaii, 1987.
22. D. Nute. Defeasible logic. In *Proceedings of the 14th International Conference on Applications of Prolog*, volume 2543 of *Lecture Notes in Artificial Intelligence*, pages 151–169. Springer, 2003.

23. Raymond Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.
24. S. Thakur, G. Governatori, V. Padmanabhan, and J. E. Lundstrom. Dialogue games in defeasible logic. In *Proceedings of the 20th Australian Joint Conference on Artificial Intelligence*, volume 4830 of *Lecture Notes in Artificial Intelligence*, pages 497–506. Springer, 2007.
25. K. Wang, D. Billington, J. Blee, and G. Antoniou. Combining description logic and defeasible logic for the semantic web. In *Proceedings of the Rules and Rule Markup Languages for the Semantic Web Workshop (RuleML2004)*, volume 3323 of *Lecture Notes in Computer Science*, pages 170–181. Springer, November 2004.