Formalizing Defeasible Logic in CAKE

Ewa Madalińska-Bugaj *
Institute of Informatics, Warsaw University
Witold Łukaszewicz †
Dept. of Computer Science, Linköping University
and College of Economics and Computer Science
TWP, Wyzwolenia 30, 10-106 Olsztyn, Poland

Abstract

Due to efficiency, Defeasible Logic is one of the most interesting non-monotonic formalisms. Unfortunately, the logic does not properly deal with cyclic defeasible rules.

In this paper, we propose a new tractable variant of Defeasible Logic, where the above shortcoming is repaired.

1 Introduction

During the last two decades, non-monotonic logics have been given much attention in the AI literature. Unfortunately, most of these formalisms have been never used in the real applications due to their computational complexity. A positive exception is *defeasible logic* (DL, for short), designed by Nute [6; 7] and later studied by many researchers.[1] As shown in [5], DL has linear time complexity what makes it very attractive from the practical point of view.

Proof theory of DL is heavily influenced by a notion of unprovability. More specifically, to defeasibly prove that a formula α holds in DL, we must first constructively show that $\neg\alpha$ is defeasibly unprovable. Unfortunately, the unprovability mechanism used in DL is too weak when cyclic defeasible rules are involved. To illustrate the point, consider the following scenario.

1. John is 18-year-old.
2. Typically, 18-year-olds are unmarried.
3. Typically, adults are married.
4. Typically, adults are employed.
5. Typically, employees are adult.

Given the above facts, we intuitively feel that "John is unmarried" should be provable. However, this conclusion cannot be derived in DL. The problem is that we are unable to show that "John is married" is unprovable. The unprovability of the last fact can be shown only by showing the unprovability of "John is adult". Unfortunately, this cannot be shown due to the circularity between "adult" and "employee": to show the unprovability of "John is adult", we have first to show the unprovability of "John is employed" and vice versa.

In this paper, we formalize defeasible logic in the framework of the CAKE method [3; 4]. The resulting formalism is tractable and properly deals with circular defeasible rules.

The paper is organized as follows. In the next two sections we provide a brief introduction to standard defeasible logic and CAKE method, respectively. In section 4, we show how defeasible theories should be translated into CAKE programs. Finally, section 5 contains conclusions.

2 Description of Defeasible Logic

2.1 General description

DL is a non-monotonic formalism based on rules and allowing a priority relation on them. A theory in this logic consists of five components: facts, strict rules, defeasible rules, defeaters and a superiority relation.

Facts are classical logic statements describing indisputable facts, for example "Marco is Italian" what is represented as

$$Italian(Marco) \tag{1}$$

or "Marco is a communist" represented as

$$Communist(Marco). \tag{2}$$

Strict rules are rules in a classical sense, i.e. if premises of a rule hold then so do the conclusion. An example of a strict rule is

$$Italian(x) \rightarrow European(x). \tag{3}$$

Defeasible rules are understood in the following way: if premises are satisfied and the contrary of the conclusion is not known that the conclusion should be drawn. An example of such a rule is "Italians are typically catholic" which is represented as

$$Italian(x) \Rightarrow Catholic(x). \tag{4}$$

*Supported in part by KBN grant 8 T11C 00919.

†Supported in part by the WITAS project grant under the Wallenberg Foundation, Sweden and KBN grant 8 T11C 00919.

[1] Actually, many variants of defeasible logic have been proposed in the literature. In this paper, we focus on the most popular variant, introduced in [1; 2].

The rule (4) together with (1) allows us to conclude that Marco is a catholic.

Consider now the rule

$$Communist(x) \Rightarrow \neg Catholic(x). \quad (5)$$

Given (5) and (2) we can conclude that Marco is not a catholic. However, considering the theory consisting of (1), (2), (4) and (5), we cannot conclude if Marco is a catholic or not.

Defeaters are rules that are not used to draw any conclusions, but to prevent conclusions of some defeasible rules. An example of a defeater rule is

$$Communist(x) \rightsquigarrow \neg Catholic(x). \quad (6)$$

Given (2) and (6) we cannot conclude that Marco is not a catholic. However, these rules, together with (1) and (4), prevent the conclusion that $Marco$ is a catholic.

The superiority relation among rules is used to make one rule stronger than another one, i.e. to make that one rule may override the conclusion of another one. It is assumed that this relation is cycles free and is defined *locally*, i.e. between rules with complementary heads. For example, assume that we have two rules (4) and (5) and the superiority relation (5) $>$ (4). These, together with (1) and (2), allow us to conclude that Marco is not a catholic, because the conclusion of (4) is overriden by the conclusion of (5).

2.2 Formal definition

Our presentation of DL is based on [1; 5].

We deal with a fixed subset of first-order language containing a finite set of constants and a finite set of relations. Both facts and rules may contain free variables. Such facts (rules) are interpreted as schemata of facts (rules), i.e. as the set of all instances over the set of constants of the considered language.

A *defeasible theory* D is a triple $\langle F, R, > \rangle$, where F is a finite set of literals (called *facts*), R is a finite set of rules, and $>$ is an acyclic superiority relation on R. Rules are of the form: $r : A(r) \hookrightarrow C(r)$, where r is a unique *label* of the rule, $A(r)$ – its *antecedent* (body) which is a finite set of literals, $C(r)$ – its *consequent* (head) which is a single literal and $\hookrightarrow$ depends on a kind of a rule:

$$\hookrightarrow = \begin{cases} \rightarrow & \text{if it is a strict rule} \\ \Rightarrow & \text{if it is a defeasible rule} \\ \rightsquigarrow & \text{if it is a defeater} \end{cases}$$

A supriority relation $>$ is a binary relation on R. $r_1 > r_2$ states that r_1 overrules r_2 if both rules are applicable. The relation $>$ is assumed to be acyclic and is defined between rules with complementary heads.

We denote by R_s, R_d, R_{dft} the subsets of strict rules, defeasible rules and defeaters of R, respectively. We write $R[q]$ to denote a set of all rules from R with q in their heads.

2.3 Proof theory

DL is defined by a proof theory. A *conclusion* of a theory D is a tagged literal which have one of the following forms:

- $+\Delta q$ which means that q is definitely provable in D.
- $-\Delta q$ which means that we have proved that q is not definitely provable in D.
- $+\delta q$ which means that q is defeasibly provable in D.
- $-\delta q$ which means that we have proved that q is not defeasibly provable in D.

Now, we define the notion of provability. It is based on the concept of a *derivation* in $D = \langle F, R, > \rangle$. A derivation is a finite sequence $P = (P(1), \ldots, P(n))$ of tagged literals constructed by inference rules of four kinds. Below, $P(1..i)$ denotes the initial part of the sequence P of length i.

$+\Delta$: We may append $P(i+1) = +\Delta q$ if either
(1) $q \in F$ or
(2) $\exists r \in R_s[q]\ \forall a \in A(r) : +\Delta a \in P(1..i)$.

$-\Delta$: We may append $P(i+1) = -\Delta q$ if
(1) $q \notin F$ and
(2) $\forall r \in R_s[q]\ \exists a \in A(r) : -\Delta a \in P(1..i)$.

$+\delta$: We may append $P(i+1) = +\delta q$ if either
(1) $+\Delta q \in P(1..i)$ or
(2) (2.1) $\exists r \in R_{sd}[q]\ \forall a \in A(r) : +\delta a \in P(1..i)$ and
(2.2) $-\Delta \neg q \in P(1..i)$ and
(2.3) $\forall s \in R[\neg q]$ either
(2.3.1) $\exists a \in A(s) : -\delta a \in P(1..i)$ or
(2.3.2) $\exists t \in R_{sd}[q]$ such that $\forall a \in A(t) : +\delta a \in P(1..i)$ and $t > s$.

$-\delta$: We may append $P(i+1) = -\delta q$
(1) $-\Delta q \in P(1..i)$ and
(2) (2.1) $\forall r \in R_{sd}[q] \exists a \in A(r) : -\delta a \in P(1..i)$ or
(2.2) $+\Delta \neg q \in P(1..i)$ or
(2.3) $\exists s \in R[\neg q]$ such that
(2.3.1) $\forall a \in A(s) : +\delta a \in P(1..i)$ and
(2.3.2) $\forall t \in R_{sd}[q]$ either $\exists a \in A(t) : -\delta a \in P(1..i)$ or $t \not> s$.

2.4 Examples

Example 2.1 Consider a theory $\langle F, R, > \rangle$, where
$F = \{Italian(Marco), Communist(Marco)\}$
$R = \{r_1 : Italian(x) \Rightarrow Catholic(x), r_2 : Communist(x) \Rightarrow \neg Catholic(x)\}$
$> = \{r_2 > r_1\}$

Below is a derivation of $+\delta \neg Catholic(Marco)$.
$P = (+\Delta Italian(Marco), +\delta Italian(Marco), +\Delta Communist(Marco), +\delta Communist(Marco), -\Delta Catholic(Marco), +\delta \neg Catholic(Marco))$

Example 2.2 A DL theory corresponding to the scenario from section 1 is given below.

$Y18(J)$
$A(x) \Rightarrow E(x)$
$E(x) \Rightarrow A(x)$
$A(x) \Rightarrow M(x)$
$Y18(x) \Rightarrow \neg M(x)$.

We leave it to the reader to show that the conclusion $+\delta\neg M(J)$ cannot be derived.

3 The CAKE system

The CAKE system has been developed to reason in the framework of distributed and incomplete information[2]. The system is based on *a rough relation* paradigm. More specifically, a rough relation R is a triple $\langle R^+, R^-, R^\pm\rangle$, where R^+ denotes a set of tuples **known** to satisfy R, R^- denotes a set of tuples **known** not to satisfy R and $R^\pm$ denotes a set of remaining tuples.

The basic notion of CAKE is that of an *information granule*. Any granule is responsible for a single rough relation. However, if a relation is given by many sources, it should be distributed among many granules. To point out the source of information, any relation will be prefixed by a name of a granule.

Formally, an information granule can be viewed as a set of facts and rules allowing to compute the relation the granule is responsible for. The following example will help to illustrate the formalism sketched above.

Example 3.1 We are given two databases, DB_1 and DB_2, containing information about a relation $C(x,y)$ meaning that a place x is directly connected with a place y. DB_1 contains $C(j,m)$, $C(m,r)$ and DB_2 contains $C(m,k)$, $C(j,l)$. Let D_1 and D_2 be names of information granules representing these databases. D_1 consists of the facts $D_1.C^+(j,m)$ and $D_1.C^+(m,r)$, whereas D_2 consists of $D_2.C^+(m,k)$ and $D_2.C^+(j,l)$[3].

Suppose now that we want to combine information from D_1 and D_2 and to represent indirect connections too. For this purpose, we introduce a new granule, D, provided with the following rules.

$$D_1.C^+(x,y) \wedge \neg D_2.C^-(x,y) \Rightarrow D.C^+(x,y) \quad (7)$$

$$D_1.C^-(x,y) \wedge \neg D_2.C^+(x,y) \Rightarrow D.C^-(x,y) \quad (8)$$

$$D_2.C^+(x,y) \wedge \neg D_1.C^-(x,y) \Rightarrow D.C^+(x,y) \quad (9)$$

$$D_2.C^-(x,y) \wedge \neg D_1.C^+(x,y) \Rightarrow D.C^-(x,y) \quad (10)$$

$$D.C^+(x,y) \wedge D.C^+(y,z) \Rightarrow D.C^+(x,z) \quad (11)$$

The rules (7)-(10) import consistent information from D_1 and D_2. It should be stressed that $\neg D_i.C^+(x,y)$ is equivalent to $D_i.C^-(x,y) \vee D_i.C^\pm(x,y)$ and $\neg D_i.C^-(x,y)$ is equivalent to $D_i.C^+(x,y) \vee D_i.C^\pm(x,y)$. Using the facts stored by D_1 and D_2, together with the rules (7)-(11), we can get $D.C^+(j,k)$. This intuitively means that the granule D answers "yes" to the query whether there is a connection between j and k.

[2]CAKE is an acronym which stands for Computer Aided Knowledge Engineering. In this paper we only use a subset of this formalism. In particular, we don't use diagrams which allow to visualise the reasoning process.

[3]Only positive and negative parts of relation are explicitly represented in facts and rules.

3.1 Syntax of CAKE

We start with an alphabet of classical first-order logic containing an enumerable set of individual variables and finite sets of individual constants and relation symbols[4].

An atomic formula of CAKE, *C-atom* for short, is an expression of the form $A.R^+(\overline{u})$ or $A.R^-(\overline{u})$, where A is a name of an information granule, R is a relation name and $\overline{u}$ is a tuple of individual variables and/or constant symbols.

A *literal* is a C-atom or an expression of the form $\neg\alpha$, where α is a C-atom.

A CAKE *rule* is any expression of the form

$$\alpha_1 \wedge \ldots \wedge \alpha_n \Rightarrow \alpha \quad (n \geq 0) \quad (12)$$

where $\alpha_1, \ldots, \alpha_n$ are literals and α is C-atom. The formula α is called the *head* of the rule, whereas $\alpha_1 \wedge \ldots \wedge \alpha_n$ is called its *body*. It is assumed that each variable occurring in a head of the rule occurs in its body.

A rule with the empty body is called a *fact*. In the sequel, we shall write α instead of $\Rightarrow \alpha$. Note that facts cannot contain individual variables.

Let $(x_1, \ldots, x_n)$ be a tuple of all free variables occurring in a rule (12). *A ground instance* of a rule (12) is any rule obtainable from (12) by uniformly replacing each occurrence of x_i by c_i, where c_i is an individual constant, for $1 \leq i \leq n$.

Any finite set of rules and facts is called a CAKE *program*.

3.2 Semantics of CAKE

In this section, we provide a fixpoint semantics of CAKE programs. We start with semi-positive programs.

Definition 3.2 A CAKE program Π is said to be *semi-positive* if, whenever a literal l occurs negatively in the body of a rule of Π, then the relation symbol of l occurs in facts only.

Let Π be a semi-positive CAKE program and K be a set of ground C-atoms. A ground C-atom α is *an immediate consequence* of Π and K if either $\alpha \in K$ or $\alpha_1 \wedge \ldots \wedge \alpha_n \Rightarrow \alpha$ is a ground instance of a rule from Π such that

1. if α_i is a positive literal, then $\alpha_i \in K$ and
2. if α_i is negative literal of the form $\neg\beta_i$, then $\beta_i \notin K$.

We define an operator $\mathcal{C}_\Pi$ such that, for any set of ground C-atoms K, $\mathcal{C}_\Pi(K)$ is the set of all immediate consequence of Π and K.

Let Π be a semi-positive CAKE program. The set of consequences of Π is the smallest fixpoint, written lfp, of an operator $\mathcal{C}_\Pi$. For a semi-positive CAKE program Π, $\mathsf{lfp}(\mathcal{C}_\Pi)$ always exists and is given by the following construction.

X:=K, where K is the set of all facts from Π;

while X$\neq \mathcal{C}_\Pi$(X) do X:=$\mathcal{C}_\Pi$(X)

The above semantics can be generalized to stratified CAKE programs.

[4]Note that CAKE does not permit function symbols.

Definition 3.3 By a *stratification* of a CAKE program Π we mean a sequence of CAKE programs $\Pi^1, \ldots, \Pi^n$, with Π^1 possibly empty, such that

- $\Pi^1 \cup \ldots \cup \Pi^n = \Pi$ and for all $1 \leq i \neq j \leq n$, $\Pi^i \cap \Pi^j = \emptyset$
- for any relation X of Π and $1 \leq i \leq n$:
  - if X occurs positively in Π^i, then all the rules with X in their heads are in $\Pi^1 \cup \ldots \cup \Pi^i$
  - if X occurs in Π^i under negation, then all the rules with X in their heads are in $\Pi^1 \cup \ldots \cup \Pi^{i-1}$

Given a stratification $\Pi^1, \ldots, \Pi^n$ of Π, each Π^i is called a *stratum* of the stratification. A program is called *stratified* if it has a stratification.

Let Π be a stratified CAKE program and let $\Pi^1, \ldots, \Pi^n$be its stratification. We define an operator $\mathcal{CS}_\Pi(X)$ by

$$\begin{aligned} \mathcal{CS}_\Pi(X) &= \mathcal{CS}^n_\Pi(X), \text{ where} \\ \mathcal{CS}^1_\Pi(X) &= \mathcal{C}_{\Pi^1}(X) \\ \mathcal{CS}^i_\Pi(X) &= \mathcal{C}_{\Pi^i \cup \mathsf{lfp}(\mathcal{CS}^{i-1}_\Pi)}(X). \end{aligned}$$

The set of consequences of Π is the smallest fixpoint of an operator $\mathcal{CS}_\Pi$. Its existence follows from the fact that, for $1 < i \leq n$, the program $\Pi^i \cup \mathsf{lfp}(\mathcal{CS}^{i-1}_\Pi)$ is semi-positive.

3.3 Voting

It is often the case that different granules provide contradictory information about a particular relation. In such a situation, a granule which combines information (as D in Example 3.1) from contradictory sources, has to solve the inconsistency problem. In Example 3.1 the problem is solved by the rules (7)-(10). In general case, i.e. if there are many different granules responsible for a relation, it is often convenient to introduce an additional granule, called voting granule, and equip it with rules solving inconsistencies. The *standard voting mechanism* is based on the following idea.

- if at least one granule responsible for relation R answers True to the query and none of the granules answers False, the final answer to the query is True
- if at least one granule answers False to the query and none answers True, the final answer to the query is False.
- otherwise, the answer to the query is Unknown.

The idea is formalized as follows. Let M be the set of all granules delivering the relation R. The CAKE voting granule V attached to relation R consists of the following rules.

$$\bigvee_{N \in M} N.R^+(\overline{x}) \wedge \bigwedge_{N \in M} \neg N.R^-(\overline{x}) \Rightarrow V.R^+(\overline{x})$$

$$\bigvee_{N \in M} N.R^-(\overline{x}) \wedge \bigwedge_{N \in M} \neg N.R^+(\overline{x}) \Rightarrow V.R^-(\overline{x})$$

It is also possible to change the standard voting policy, for instance, by providing priorities among granules, but this subject will not be discussed here.

4 Translating defeasible theories into CAKE programs

In this section, we show how defeasible theories are to be translated into CAKE programs. The following notation will be useful.

Let l be a literal of the form $P(\overline{u})$ or $\neg P(\overline{u})$, where P is a relation symbol and $\overline{u}$ is a tuple of individual variables and/or constant symbols. We write $[l]$ to denote the expression $P^+(\overline{u})$ if l is positive and $P^-(\overline{u})$otherwise. For instance, $[Q(x,a)]$ is $Q^+(x,a)$, whereas $[\neg R(x,y)]$ is $R^-(x,y)$.

We start by providing translation of defeasible theories into CAKE programs under the assumption that the considered theories contain neither priorities nor defeaters.

Definition 4.1 Let $D = \langle F, R, \emptyset \rangle$ be a priority-free defeasible theory over a language $\mathcal{L}_\mathcal{D}$ such that R contains no defeater rules. The CAKE program, CR, associated with D is the smallest set of rules defined as follows.

1. If $l \in F$, then $\Delta[l] \in CR$.
2. If $l_1 \wedge \cdots \wedge l_n \rightarrow l$ is a strict rule from R, then $(\Delta[l_1] \wedge \cdots \wedge \Delta[l_n] \Rightarrow \Delta[l]) \in CR$.
3. For each n-ary relation symbol P occurring in $\mathcal{L}_\mathcal{D}$, CR contains the rules $\Delta P^+(\overline{x}) \Rightarrow \delta P^+(\overline{x})$ and $\Delta P^-(\overline{x}) \Rightarrow \delta P^-(\overline{x})$, where $\overline{x}$ is an n-tuple of individual variables.
4. Let $P_1, \ldots P_n$ be the set of all relation symbols occurring in the heads of strict and defeasible rules from R. For each $1 \leq i \leq n$:
   (a) Suppose that $r^i_1, \ldots r^i_k$ is the set of all rules containing P_i in their heads.
   CR contains the following rules: $\Delta P^+(\overline{x}) \Rightarrow \delta_j P^+(\overline{x})$ and $\Delta P^-(\overline{x}) \Rightarrow \delta_j P^-(\overline{x})$, for $(1 \leq j \leq k)$.
   For each r^i_j $(1 \leq j \leq k)$ of the form $l_1 \wedge \cdots \wedge l_m \Rightarrow l$ (or $l_1 \wedge \cdots \wedge l_m \rightarrow l$), CR contains the rule $\delta[l_1] \wedge \cdots \wedge \delta[l_m] \wedge \neg\delta_j[\neg l] \Rightarrow \delta_j[l]$.
   (b) In addition, CR contains the following pair of voting rules

$$\bigvee_{j=1}^{k} \delta_j P^+_i(\overline{x}) \wedge \bigwedge_{j=1}^{k} \neg\delta_j P^-_i(\overline{x}) \Rightarrow \delta P^+_i(\overline{x})$$

$$\bigvee_{j=1}^{k} \delta_j P^-_i(\overline{x}) \wedge \bigwedge_{j=1}^{k} \neg\delta_j P^+_i(\overline{x}) \Rightarrow \delta P^-_i(\overline{x}). \quad \blacksquare$$

Remark 4.2 The points 4(a) and 4(b) can be simplified for those relation symbols from $P_1, \ldots, P_n$ which occur in the head of exactly one rule. Assume that P_i enjoys this property and suppose that P_i occurs in the head of the rule of the form $l_1 \wedge \cdots \wedge l_m \Rightarrow l$ (or $l_1 \wedge \cdots \wedge l_m \rightarrow l$). In this case, the CAKE rule $\delta[l_1] \wedge \cdots \wedge \delta[l_m] \wedge \neg\delta_j[\neg l] \Rightarrow \delta_j[l]$ from point 4(a) can be replaced by $\delta[l_1] \wedge \cdots \wedge \delta[l_m] \wedge \neg\delta[\neg l] \Rightarrow \delta[l]$ and the voting rules from point 4(b) can be omitted. $\blacksquare$

Example 4.3 Consider the defeasible theory $D = \langle F, R, \emptyset \rangle$, where $F = \{Q(n), R(n)\}$ and $R = \{Q(x) \Rightarrow P(x), R(x) \Rightarrow \neg P(x)\}$.[5] The CAKE program corresponding to D is the following.

$$\Delta Q^+(n) \tag{13}$$
$$\Delta R^+(n) \tag{14}$$
$$\Delta R^+(x) \Rightarrow \delta R^+(x) \tag{15}$$
$$\Delta R^-(x) \Rightarrow \delta R^-(x) \tag{16}$$
$$\Delta Q^+(x) \Rightarrow \delta Q^+(x) \tag{17}$$
$$\Delta Q^-(x) \Rightarrow \delta Q^-(x) \tag{18}$$
$$\Delta P^+(x) \Rightarrow \delta P^+(x) \tag{19}$$
$$\Delta P^-(x) \Rightarrow \delta P^-(x) \tag{20}$$
$$\Delta P^+(x) \Rightarrow \delta P^+(x) \tag{21}$$
$$\Delta P^-(x) \Rightarrow \delta P^-(x) \tag{22}$$
$$\Delta P^+(x) \Rightarrow \delta_1 P^+(x) \tag{23}$$
$$\Delta P^-(x) \Rightarrow \delta_1 P^-(x) \tag{24}$$
$$\Delta P^+(x) \Rightarrow \delta_2 P^+(x) \tag{25}$$
$$\Delta P^-(x) \Rightarrow \delta_2 P^-(x) \tag{26}$$
$$\delta Q^+(x) \wedge \neg\delta_1 P^-(x) \Rightarrow \delta_1 P^+(x) \tag{27}$$
$$\delta R^+(x) \wedge \neg\delta_2 P^+(x) \Rightarrow \delta_2 P^-(x) \tag{28}$$
$$(\delta_1 P^+(x) \vee \delta_2 P^+(x)) \wedge (\neg\delta_1 P^-(x) \wedge \neg\delta_2 P^-(x)) \Rightarrow \delta P^+(x) \tag{29}$$
$$(\delta_1 P^-(x) \vee \delta_2 P^-(x)) \wedge (\neg\delta_1 P^+(x) \wedge \neg\delta_2 P^+(x)) \Rightarrow \delta P^-(x). \tag{30}$$

The partition P_1={(13)-(26)}, P_2={(27),(28)}, P_3={(29),(30)} provides a stratification of the above set. The computation of the relations specified by the rules is given below.

$$\{\Delta Q^+(n), \Delta R^+(n)\}$$
$$\{\Delta Q^+(n), \Delta R^+(n), \delta Q^+(n), \delta R^+(n)\}$$
$$\{\Delta Q^+(n), \Delta R^+(n), \delta Q^+(n), \delta R^+(n), \delta_1 P^+(n), \delta_2 P^-(n)\}. \tag{31}$$

Observe that according to intuitions neither $\delta P^+(n)$ nor $\delta P^-(n)$ is a member of the set (31). ■

Theorem 4.4 The CAKE program CR corresponding to a priority-free defeasible theory D without defeaters is stratified.

Proof Let CR_1, CR_2 and CR_3 be sets of CAKE rules specified in points 1-3, 4(a) and 4(b), respectively (see Definition 4.1). It is easily verified that the partition CR_1, CR_2 and CR_3 provides a stratification of CR. ■

We now consider defeasible theories with priorities, but without defeater rules.

Definition 4.5 Let $D = \langle F, R, > \rangle$ be a defeasible theory such that R contains no defeater rules. The CAKE program associated with D is specified as in Definition 4.1 with voting rules from point 4(b) replaced by

$$\bigvee_{j=1}^{k} (\delta_j P_i^+(\overline{x}) \wedge \bigwedge_{l=1}^{k} [\delta_l P_i^-(\overline{x}) \Rightarrow (r_j^i > r_l^i)]) \Rightarrow \delta P_i^+(\overline{x}) \tag{32}$$

$$\bigvee_{j=1}^{k} (\delta_j P_i^-(\overline{x}) \wedge \bigwedge_{l=1}^{k} [\delta_l P_i^+(\overline{x}) \Rightarrow (r_j^i > r_l^i)]) \Rightarrow \delta P_i^-(\overline{x}). \tag{33}$$

Here $r_j^i > r_l^i$ evaluates to $True$ or $False$, depending on whether $r_j^i > r_l^i$ holds or not.[6] It should be stressed that in the case where the superiority relation $>$ is empty, the rules (32) and (33) are equivalent to the rules (29) and (30), respectively. ■

Example 4.6 Consider a prioritized version of the theory from Example 4.3 given by $D = \langle \{Q(n), R(n)\}, \{r_1, r_2\}, \{r_1 > r_2\} \rangle$, where $r_1 = (Q(x) \Rightarrow P(x))$ and $r_2 = (R(x) \Rightarrow \neg P(x))$. The CAKE program corresponding to D is as in Example 4.3 with (29) and (30) replaced by

$$[[\delta_1 P^-(x) \wedge \delta_2 P^+(x) \Rightarrow (r_2 > r_1)] \vee [\delta_2 P^-(x) \wedge (\delta_1 P^+(x) \Rightarrow (r_1 > r_2))]] \Rightarrow \delta P^+(x) \tag{34}$$

$$[[\delta_1 P^+(x) \wedge \delta_2 P^-(x) \Rightarrow (r_2 > r_1)] \vee [\delta_2 P^+(x) \wedge (\delta_1 P^-(x) \Rightarrow (r_1 > r_2))]] \Rightarrow \delta P^-(x) \tag{35}$$

respectively.

The set of rules $\{(13) - (28), (34) - (35)\}$ is stratified by the partition $P_1 = \{(13) - (26)\}$, $P_2 = \{(27) - (28)\}$, $P_3 = \{(34) - (35)\}$. The computation of the relations specified by this set is given below.

$$\{\Delta Q^+(n), \Delta R^+(n)\}$$
$$\{\Delta Q^+(n), \Delta R^+(n), \delta Q^+(n), \delta R^+(n)\}$$
$$\{\Delta Q^+(n), \Delta R^+(n), \delta Q^+(n), \delta R^+(n), \delta_1 P^+(n), \delta_2 P^-(n)\}$$
$$\{\Delta Q^+(n), \Delta R^+(n), \delta Q^+(n), \delta R^+(n), \delta_1 P^+(n), \delta_2 P^-(n), \delta P^+(n)\}. \tag{36}$$

Note that according to intuitions $\delta P^+(n)$ belongs to the set (36). ■

Theorem 4.7 A CAKE program corresponding to a prioritized defeasible theory without defeaters is stratified.

Proof Similar to that of Theorem 4.4. RL_3 should now be defined as the set of voting rules specified in Definition 4.5. ■

We now provide a translation of arbitrary defeasible theories into CAKE programs.

Definition 4.8 Let $D = \langle F, R, > \rangle$ be an arbitrary defeasible theory. The CAKE program associated with D is specified as in Definition 4.1, with voting rules from point 4(b) replaced by (32) and (33) from Definition 4.5 (if $> \neq \emptyset$), together with the following modifications and additions.

[5] This is the Nixon diamond theory with n, P, Q, R standing for $Nixon$, $Pacifist$, $Quaker$ and $Republican$, respectively.

[6] Note that since the relation $>$ is assumed to be acyclic, $r_j^i > r_l^i$ cannot be both $True$ and $False$.

- For each defeater rule of the form $l_1 \wedge \cdots \wedge l_m \rightsquigarrow l$, the set CR contains $\delta[l_1] \wedge \cdots \wedge \delta[l_m] \Rightarrow d[l]$
- In point 4(a), the CAKE rule $\delta[l_1] \wedge \cdots \wedge \delta[l_m] \wedge \neg\delta_j[\neg l] \Rightarrow \delta_j[l]$ should be replaced by $\delta[l_1] \wedge \cdots \wedge \delta[l_m] \wedge \neg\delta_j[\neg l] \wedge \neg d[\neg l] \Rightarrow \delta_j[l]$.■

Example 4.9 Let $D = \langle\{HSD(j)\}, \{HSD(x) \Rightarrow A(x), A(x) \Rightarrow E(x), HSD(x) \rightsquigarrow \neg E(x)\}$[7],$\emptyset\rangle$. The CAKE program corresponding to D is the following.

$$\Delta HSD^+(j) \quad (37)$$
$$\Delta HSD^+(x) \Rightarrow \delta HSD^+(x) \quad (38)$$
$$\Delta HSD^-(x) \Rightarrow \delta HSD^-(x) \quad (39)$$
$$\Delta A^+(x) \Rightarrow \delta A^+(x) \quad (40)$$
$$\Delta A^-(x) \Rightarrow \delta A^-(x) \quad (41)$$
$$\Delta E^+(x) \Rightarrow \delta E^+(x) \quad (42)$$
$$\Delta E^-(x) \Rightarrow \delta E^-(x) \quad (43)$$
$$\delta HSD^+(x) \wedge \neg\delta A^-(x) \wedge \neg dA^-(x) \Rightarrow \delta A^+(x) \quad (44)$$
$$\delta A^+(x) \wedge \neg\delta E^-(x) \wedge \neg dE^-(x) \Rightarrow \delta E^+(x) \quad (45)$$
$$\delta HSD^+(x) \Rightarrow dE^-(x). \quad (46)$$

Note that the partition $P_1 = \{(37) - (43), (46)\}$ and $P_2 = \{(44) - (45)\}$ provides a stratification of the above set of rules. The computation of the relations specified by this set is given below.

$\{\Delta HSD(j)\}$
$\{\Delta HSD(j), \delta HSD(j)\}$
$\{\Delta HSD(j), \delta HSD(j), dE^-(j)\}$
$$\{\Delta HSD(j), \delta HSD(j), dE^-(j), \delta A^+(j)\} \quad (47)$$

Observe that according to our intuitions $\delta A^+(j)$ is a member of the set (47), but $\delta E^+(j)$ is not. ■

Remark 4.10 In general, CAKE programs corresponding to defeasible theories with defeater rules need not be stratified.

Consider for instance, the defeasible theory $D = \langle\emptyset, R, \emptyset\rangle$, where $R = \{\Rightarrow P(a), \Rightarrow Q(a), \Rightarrow R(a), P(a) \rightsquigarrow \neg Q(a), Q(a) \rightsquigarrow \neg R(a), R(a) \rightsquigarrow \neg P(a)\}$. We leave it to the reader to show that the CAKE program corresponding to D is not stratified. ■

There still remains the problem to determine what is derivable in our version of DL. The details follow.

Definition 4.11 Let $D = \langle F, R, >\rangle$ be a defeasible theory and let stratified progran Π be its translation.

The formula $+\Delta R(\overline{u})$ $(+\delta R(\overline{u}), +\Delta\neg R(\overline{u}), +\delta\neg R(\overline{u}))$ is derivable from D iff $\Delta R^+(\overline{u})$ $(\delta R^+(\overline{u}), \Delta R^-(\overline{u}), \delta R^-(\overline{u}))$ belongs to $\mathsf{lfp}\mathcal{CS}_\Pi$.

The formula $-\Delta R(\overline{u})$ $(-\delta R(\overline{u}), -\Delta\neg R(\overline{u}), -\delta\neg R(\overline{u}))$ is derivable from D iff $\Delta R^+(\overline{u})$ $(\delta R^+(\overline{u}), \Delta R^-(\overline{u}), \delta R^-(\overline{u}))$ does not belong to $\mathsf{lfp}\mathcal{CS}_\Pi$.■

Example 4.12 Consider the translation of the theory from Example 2.2.

$\Delta Y18^+(J)$
$\Delta A^+(x) \Rightarrow \delta A^+(x)$
$\Delta A^-(x) \Rightarrow \delta A^-(x)$
$\Delta E^+(x) \Rightarrow \delta E^+(x)$
$\Delta E^-(x) \Rightarrow \delta E^-(x)$
$\Delta M^+(x) \Rightarrow \delta M^+(x)$
$\Delta M^-(x) \Rightarrow \delta M^-(x)$
$\Delta M^+(x) \Rightarrow \delta_1 M^+(x)$
$\Delta M^-(x) \Rightarrow \delta_1 M^-(x)$
$\Delta M^+(x) \Rightarrow \delta_2 M^+(x)$
$\Delta M^-(x) \Rightarrow \delta_2 M^-(x)$
$\Delta Y18^+(x) \Rightarrow \delta Y18^+(x)$
$\Delta Y18^-(x) \Rightarrow \delta Y18^-(x)$
$\delta A^+(x) \wedge \neg\delta E^-(x) \Rightarrow \delta E^+(x)$
$\delta E^+(x) \wedge \neg\delta A^-(x) \Rightarrow \delta A^+(x)$
$\delta A^+(x) \wedge \neg\delta_1 M^-(x) \Rightarrow \delta_1 M^+(x)$
$\delta Y18^+(x) \wedge \neg\delta_2 M^+(x) \Rightarrow \delta_2 M^-(x)$
$\delta_1 M^-(x) \wedge \neg\delta_2 M^+(x) \Rightarrow \delta M^-(x)$
$\delta_1 M^+(x) \wedge \neg\delta_2 M^-(x) \Rightarrow \delta M^+(x)$
$\delta_2 M^-(x) \wedge \neg\delta_1 M^+(x) \Rightarrow \delta M^-(x)$
$\delta_2 M^+(x) \wedge \neg\delta_1 M^-(x) \Rightarrow \delta M^+(x)$

The computation of the relations specified by this set is given below.

$\{\Delta Y18(J)\}$
$\{\Delta Y18(J), \delta Y18(J)\}$
$$\{\Delta Y18(J), \delta Y18(J), \delta M^-(J)\} \quad (48)$$

Note that according to our intuition, $\delta M^-(J)$ is a member of the set (48) while this formula is not derivable in original version of DL.

5 Conclusions

In this paper, we have provided a new version of defeasible logic which properly deals with cyclic defeasible rules. Like all logics expressible using the CAKE method, our formalism is tractable. In contrast to the original version of DL, our version is defined for defeasible theories leading to stratified CAKE programs. However, this restriction is now widely accepted both in the theory of data bases and in logic programming.

References

[1] G. Antoniou, D. Billington, G. Governatori and M. Maher. Representation Results for Defeasible Logic. *ACM Transactions on Computational Logic*, 2, 255-287, 2001.

[2] D. Billington. Defeasible Logic is Stable. *Journal of Logic and Computation*, 3, 370-400, 1993.

[3] P. Doherty, W. Łukaszewicz and A. Szalas. CAKE: A Computer Aided Knowledge Engineering Technique. *Proceedings of the 15th European Conference on Artificial Intelligence*, IOS Press, July, Amsterdam, 2002.

[7] Here D, A, E and j stand for *HighSchoolDropout*, *Adult*, *Employed* and *John*, respectively.

[4] P. Doherty, P, W. Łukaszewicz, A. Skowron and A. Szalas. Knowledge Engineering: A Rough Set Approach. Physica Verlag, 2003, to appear.

[5] M.Maher. Propositional Defeasible Logic has Linear Complexity. *Theory and Practice of Logic Programming*, 1 (6) 691-711, 2001.

[6] D. Nute. Defeasible Reasoning and Decision Support Systems. *decision Support Systems*, 4, 97-110, 1998.

[7] D. Nute. Defeasible Logic. In D. M Gabbay, C. J. Hogger and J. A. Robinson (eds.): *Handbook of Logic in Artificial Intelligence and Logic Programming*, vol. 3, Oxford University Press, 1994, 353-395.