Fundamenta Informaticae 57 (2003) 193–213
IOS Press

Formalizing Defeasible Logic in CAKE

Ewa Madalińska-Bugaj[*]
Institute of Informatics, Warsaw University
Banacha 2, Warsaw, Poland

Witold Łukaszewicz[†]
Dept. of Computer Science, Linköping University
and College of Economics and Computer Science
TWP, Olsztyn, Poland

Abstract. Due to its efficiency, defeasible logic is one of the most interesting non-monotonic formalisms. Unfortunately, the logic has one major limitation: it does not properly deal with cyclic defeasible rules.

In this paper, we provide a new variant of defeasible logic, using CAKE method. The resulting formalism is tractable and properly deals with circular defeasible rules.

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 [11, 10] and later studied by many researchers.[1] As shown in [9], DL has a linear time complexity (with respect to the number of rules) what makes it very attractive from the practical point of view.

[*] Supported in part by KBN grant 8 T11C 00919.
Address for correspondence: Institute of Informatics, Warsaw University, Banacha 2, Poland
[†] 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 [2, 4].

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.

John is 17-year-old.

Typically, 17-year-olds are unmarried.

Typically, adults are married.

Typically, adults are employed.

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"; to show the unprovability of "John is employed", we have first to show the unprovability of "John is adult".

In this paper, we formalize defeasible logic in the framework of CAKE method [6, 7]. The resulting formalism is tractable and properly deals with circular defeasible rules.

The paper is organized as follows. In the next section, we provide a brief introduction to standard defeasible logic. In section 3, we describe CAKE method for stratified CAKE programs. In section 4, we show how defeasible theories should be translated into CAKE programs. In section 5, we provide a method to deal with non-stratified programs, using well-founded semantics. In section 6, we compute two non-stratified CAKE programs applying the method specified earlier. Finally, section 7 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

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

Defeasible rules are understood in the following way: if premises of a rule hold and the contrary of the conclusion of the rule is unknown, then the conclusion holds. An example of such a rule is “Typically, Italians are catholics” which is represented as

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

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

Consider now the rule

$$Communist(x) \Rightarrow \neg Catholic(x). \tag{5}$$

Given (5) and (2) we can conclude that Marco is not a catholic. However, if the theory under consideration consists of (1), (2), (4) and (5), we cannot determine whether 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

$$Communist(x) \leadsto \neg Catholic(x). \tag{6}$$

This rule have the following intuitive meaning: “for any object x, if x is a communist, then the conclusion that x is a catholic should be blocked.” Thus, 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. 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 [2, 9].

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 list of literals, $C(r)$ – its *consequent* (head) which is a single literal and $\hookrightarrow$ depends on a kind of a rule:

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

A superiority 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 in DL. 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. (Here j, $Y17$, A, E and M stand for $John$, $17-year-old$, $Adult$, $Employed$ and $Married$, respectively.)

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

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 method is under implementation at Department of Computer and Information Science of Linköping University.

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 contains information concerning a single rough relation. However, if a relation is given by many sources, it may be distributed among many granules. To point out the source of information, any relation is 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 basic ideas.

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

[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 do not use diagrams which allow to visualise the reasoning process.

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.

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

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

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

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

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

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

3.1. Syntax of CAKE

We start with an alphabet of the 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

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

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*.

A rule with the empty body is called a *fact*. In the sequel, we shall write α instead of $\Rightarrow \alpha$.

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

Definition 3.1. Let $(x_1, \ldots, x_n)$ be a tuple of all free variables occurring in a rule (12). *A ground instance* of (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 $1 \leq i \leq n$. A set of all ground instances of all rules from a program Π is called the *ground instance* of Π. ■

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

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

3.2. 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 from contradictory sources (as D in Example 3.1), 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 be discussed later.

3.3. 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 the 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[5].

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 a rule's body 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 a rule's body 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

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

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.

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 $\mathcal{L}_D$ such that R contains no defeater rules. The CAKE program, CR, associated with D is the smallest set of rules defined as follows. (Information granules occurring in the CAKE program corresponding to a DL theory

[5]The notion of stratification has been independently proposed by several researchers [3, 5, 8, 13]

will be named by the symbols Δ and δ, the latter possibly with subscripts[6]. To simplify notation, we shall write $\Delta P^{+}(\overline{u})$, $\Delta P^{-}(\overline{u})$, $\delta P^{+}(\overline{u})$, $\delta P^{-}(\overline{u})$, $\delta_i P^{+}(\overline{u})$, $\delta_i P^{-}(\overline{u})$, instead of $\Delta.P^{+}(\overline{u})$, $\Delta.P^{-}(\overline{u})$, $\delta.P^{+}(\overline{u})$, $\delta.P^{-}(\overline{u})$, $\delta_i.P^{+}(\overline{u})$, $\delta_i.P^{-}(\overline{u})$.)

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 the language of the theory 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: $\Delta P^{+}_i(\overline{x}) \Rightarrow \delta_j P^{+}_i(\overline{x})$ and $\Delta P^{-}_i(\overline{x}) \Rightarrow \delta_j P^{-}_i(\overline{x})$, 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

$$\bigvee_{j=1}^{k} \delta_j P^{+}_i(\overline{x}) \wedge \bigwedge_{j=1}^{k} \neg\delta_j P^{-}_i(\overline{x}) \Rightarrow \delta P^{+}_i(\overline{x}) \tag{13}$$

$$\bigvee_{j=1}^{k} \delta_j P^{-}_i(\overline{x}) \wedge \bigwedge_{j=1}^{k} \neg\delta_j P^{+}_i(\overline{x}) \Rightarrow \delta P^{-}_i(\overline{x}). \tag{14}$$

■

Remark 4.1. 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. ■

Example 4.1. Consider the defeasible theory $D = \langle F, R, \emptyset \rangle$, where[7]

$$F = \{Q(n), R(n)\} \text{ and } R = \{Q(x) \Rightarrow P(x), R(x) \Rightarrow \neg P(x)\}.$$

[6]The δ symbol is subscribed if there are many rules with the same relation in their heads.
[7]This is the Nixon diamond theory with n, P, Q, R standing for $Nixon$, $Pacifist$, $Quaker$ and $Republican$, respectively.

The CAKE program, Π, corresponding to D is the following.

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

The partition Π_1={(15)-(26)}, Π_2={(27),(28)}, Π_3={(29),(30)} provides a stratification of the above set. The computation of the smallest fixpoint of the operator $\mathcal{CS}_\Pi$ is given below.

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

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

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

Definition 4.2. 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 $r^i_j > r^i_l$ evaluates to True or False, depending on whether $r^i_j > r^i_l$ holds or not.[8] The $\Rightarrow$ symbol in expression in [ ] stands for logical implication and this expression should be evaluated and abbreviated to single C-atom during translation. It should be stressed that in the case where the superiority relation $>$ is empty, the rules (32) and (33) are equivalent to the rules (13) and (14), respectively.■

Example 4.2. Consider a prioritized version of the theory from Example 4.1 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, Π, corresponding to D is as in Example 4.1 with (29) and (30) replaced by

$$\begin{aligned}&(\delta_1 P^+(x) \wedge (\delta_2 P^-(x) \Rightarrow (r_1 > r_2))\vee \\ &\quad \delta_2 P^+(x) \wedge (\delta_1 P^-(x) \Rightarrow (r_2 > r_1))) \quad \Rightarrow \quad \delta P^+(x) \qquad (34)\end{aligned}$$

$$\begin{aligned}&(\delta_1 P^-(x) \wedge (\delta_2 P^+(x) \Rightarrow (r_1 > r_2))\vee \\ &\quad (\delta_2 P^-(x) \wedge (\delta_1 P^+(x) \Rightarrow (r_2 > r_1))) \quad \Rightarrow \quad \delta P^-(x) \qquad (35)\end{aligned}$$

respectively.

The set of rules $\{(15) - (28), (34) - (35)\}$ is stratified by the partition $\Pi_1 = \{(15) - (26)\}$, $\Pi_2 = \{(27) - (28)\}$, $\Pi_3 = \{(34) - (35)\}$. The computation of the smallest fixpoint of the operator $\mathcal{CS}_\Pi$ is given below.

$$\begin{aligned}&\{\Delta Q^+(n), \Delta R^+(n)\}\\ &\{\Delta Q^+(n), \Delta R^+(n), \delta Q^+(n), \delta R^+(n)\}\\ &\{\Delta Q^+(n), \Delta R^+(n), \delta Q^+(n), \delta R^+(n), \delta_1 P^+(n), \delta_2 P^-(n)\}\\ &\{\Delta Q^+(n), \Delta R^+(n), \delta Q^+(n), \delta R^+(n), \delta_1 P^+(n), \delta_2 P^-(n), \delta P^+(n)\}. \qquad (36)\end{aligned}$$

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

Now, we provide a translation of arbitrary defeasible theories into CAKE programs.

Definition 4.3. Let $D = \langle F, R, >\rangle$ be an arbitrary defeasible theory. The CAKE program associated with D is specified as in Definition 4.1, together with the following modifications and additions.

- Suppose that $r^i_{k+1}, \ldots r^i_{k+p}$ is the set of all defeater rules containing P_i in their heads.
  For each r^i_j $(k+1 \leq j \leq k+p)$ of the form $l_1 \wedge \cdots \wedge l_m \rightsquigarrow l$, the set CR contains the rule $\delta[l_1] \wedge \cdots \wedge \delta[l_m] \Rightarrow d_j[l]$.
- Voting rules from point 4(b) in Definition 4.1 should be replaced by

$$\bigvee_{j=1}^{k} (\delta_j P^+_i(\overline{x}) \wedge \bigwedge_{l=1}^{k}[\delta_l P^-_i(\overline{x}) \Rightarrow (r^i_j > r^i_l)] \wedge \bigwedge_{l=k+1}^{k+p} [d_l P^-_i(\overline{x}) \Rightarrow (r^i_j > r^i_l)]) \Rightarrow \delta P^+_i(\overline{x}) \qquad (37)$$

$$\bigvee_{j=1}^{k} (\delta_j P^-_i(\overline{x}) \wedge \bigwedge_{l=1}^{k}[\delta_l P^+_i(\overline{x}) \Rightarrow (r^i_j > r^i_l)] \wedge \bigwedge_{l=k+1}^{k+p} [d_l P^+_i(\overline{x}) \Rightarrow (r^i_j > r^i_l)]) \Rightarrow \delta P^-_i(\overline{x}). \qquad (38)$$

■

[8]Note that since the relation $>$ is assumed to be acyclic, $r^i_j > r^i_l$ cannot be both True and False.

Example 4.3. Consider the theory given by

$$\langle \{D(j)\}, \{D(x) \Rightarrow A(x), A(x) \Rightarrow E(x), D(x) \leadsto \neg E(x)\}, \emptyset \rangle.$$

(Here D, A, E and j stand for $HighSchoolDropout$, $Adult$, $Employed$ and $John$, respectively.) The CAKE program, Π, corresponding to D is the following.

$$\Delta D^{+}(j) \tag{39}$$
$$\Delta D^{+}(x) \Rightarrow \delta D^{+}(x) \tag{40}$$
$$\Delta D^{-}(x) \Rightarrow \delta D^{-}(x) \tag{41}$$
$$\Delta A^{+}(x) \Rightarrow \delta A^{+}(x) \tag{42}$$
$$\Delta A^{-}(x) \Rightarrow \delta A^{-}(x) \tag{43}$$
$$\Delta E^{+}(x) \Rightarrow \delta E^{+}(x) \tag{44}$$
$$\Delta E^{-}(x) \Rightarrow \delta E^{-}(x) \tag{45}$$
$$\delta D^{+}(x) \Rightarrow dE^{-}(x) \tag{46}$$
$$\delta D^{+}(x) \wedge \neg \delta A^{-}(x) \wedge \neg dA^{-}(x) \Rightarrow \delta A^{+}(x) \tag{47}$$
$$\delta A^{+}(x) \wedge \neg \delta E^{-}(x) \wedge \neg dE^{-}(x) \Rightarrow \delta E^{+}(x). \tag{48}$$

Note that the partition $\Pi_1 = \{(39) - (46)\}$, $\Pi_2 = \{(47) - (48)\}$ provides a stratification of the above set of rules. The computation of the smallest fixpoint of the operator $\mathcal{CS}_\Pi$ is given below.

$$\begin{gathered} \{\Delta D(j)\} \\ \{\Delta D(j), \delta D(j)\} \\ \{\Delta D(j), \delta D(j), dE^{-}(j)\} \\ \{\Delta D(j), \delta D(j), dE^{-}(j), \delta A^{+}(j)\}. \end{gathered} \tag{49}$$

Observe that according to our intuitions $\delta A^{+}(j)$ is a member of the set (49), but $\delta E^{+}(j)$ is not. ■

Remark 4.2. In general, CAKE programs corresponding to defeasible theories need not be stratified. Consider for instance, the defeasible theory $D = \langle \emptyset, R, \emptyset \rangle$, where

$$R = \{\Rightarrow P(a), \Rightarrow Q(a), \Rightarrow R(a), P(a) \leadsto \neg Q(a), Q(a) \leadsto \neg R(a), R(a) \leadsto \neg P(a)\}.$$

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

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

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

Example 4.4. Consider the translation Π of the theory D from Example 2.2.

$$\begin{array}{l}
\Delta Y17^{+}(j)\\
\Delta A^{+}(x) \Rightarrow \delta A^{+}(x)\\
\Delta A^{-}(x) \Rightarrow \delta A^{-}(x)\\
\Delta E^{+}(x) \Rightarrow \delta E^{+}(x)\\
\Delta E^{-}(x) \Rightarrow \delta E^{-}(x)\\
\Delta M^{+}(x) \Rightarrow \delta M^{+}(x)\\
\Delta M^{-}(x) \Rightarrow \delta M^{-}(x)\\
\Delta M^{+}(x) \Rightarrow \delta_1 M^{+}(x)\\
\Delta M^{-}(x) \Rightarrow \delta_1 M^{-}(x)\\
\Delta M^{+}(x) \Rightarrow \delta_2 M^{+}(x)\\
\Delta M^{-}(x) \Rightarrow \delta_2 M^{-}(x)\\
\Delta Y17^{+}(x) \Rightarrow \delta Y17^{+}(x)\\
\Delta Y17^{-}(x) \Rightarrow \delta Y17^{-}(x)\\
\delta A^{+}(x) \wedge \neg\delta E^{-}(x) \Rightarrow \delta E^{+}(x)\\
\delta E^{+}(x) \wedge \neg\delta A^{-}(x) \Rightarrow \delta A^{+}(x)\\
\delta A^{+}(x) \wedge \neg\delta_1 M^{-}(x) \Rightarrow \delta_1 M^{+}(x)\\
\delta Y17^{+}(x) \wedge \neg\delta_2 M^{+}(x) \Rightarrow \delta_2 M^{-}(x)\\
\delta_1 M^{-}(x) \wedge \neg\delta_2 M^{+}(x) \Rightarrow \delta M^{-}(x)\\
\delta_1 M^{+}(x) \wedge \neg\delta_2 M^{-}(x) \Rightarrow \delta M^{+}(x)\\
\delta_2 M^{-}(x) \wedge \neg\delta_1 M^{+}(x) \Rightarrow \delta M^{-}(x)\\
\delta_2 M^{+}(x) \wedge \neg\delta_1 M^{-}(x) \Rightarrow \delta M^{+}(x).
\end{array}$$

The computation of the smallest fixpoint of the operator $\mathcal{CS}_{\Pi}$ is given below.

$$\begin{array}{l}
\{\Delta Y17(j)\}\\
\{\Delta Y17(j), \delta Y17(j)\}\\
\{\Delta Y17(j), \delta Y17(j), \delta M^{-}(j)\}.
\end{array} \tag{50}$$

Note that according to our intuitions, $\delta M^{-}(j)$ is derivable from D. ■

Originally, the CAKE system has been developed for stratified programs only. However, not stratified programs can be also dealt with, using another semantics which is called well-founded. This semantics, which is presented in the next section, is also tractable and can be applied to all CAKE programs. Moreover, well-founded semantics can be viewed as a natural extension of stratified semantics because both agree on stratified programs.

5. Well-founded Semantics for CAKE Programs

Well-founded semantics is based on a notion of three-valued interpretation, in which the truth value of facts can be True, False or Unknown. In the sequel, we shall consider only finite interpretations, i.e. interpretations providing truth values to all ground C-atoms in a given CAKE program Π. Our presentation of well-founded semantics follows [1].

5.1. Three-valued interpretations.

We start by defining a notion of a 3-valued interpretation.

Definition 5.1. Let $ATM(\Pi)$ be a set of all ground C-atoms in a CAKE program Π. A *3-valued interpretation* **S** of (Π) is a mapping from $ATM(\Pi)$ into the set $\{$True,Unknown,False$\}$. ■

We denote by $\mathbf{S}^0$, $\mathbf{S}^{1/2}$, $\mathbf{S}^1$ the set of C-atoms in $ATM(\Pi)$ whose truth value is False, Unknown and True respectively.

It is often convenient to represent a 3-valued interpretation by listing the positive and negative facts and omitting the unknown ones. For instance, the interpretation **S**, where $\mathbf{S}(\delta P^+(a)) =$ True, $\mathbf{S}(\delta P^+(b)) =$ Unknown, $\mathbf{S}(\delta P^-(b)) =$ Unknown and $\mathbf{S}(\delta P^-(a)) =$ False can be represented as $\mathbf{S}=\{\delta P^+(a), \neg\delta P^-(a)\}$.

There is a natural ordering $\prec$ among 3-valued interpretations of a program (Π) defined by

$$\mathbf{S} \prec \mathbf{T} \text{ iff } \mathbf{S}^1 \subseteq \mathbf{T}^1 \text{ and } \mathbf{T}^0 \subseteq \mathbf{S}^0.$$

The least element wrt $\prec$, denoted by $\perp$, is the interpretation where all atoms are false.

Given a 3-valued interpretation **S** we can extend it in a standard way to Boolean combinations of atoms:

$$\begin{aligned}
\mathbf{S}(A \wedge B) &= \min(\mathbf{S}(A), \mathbf{S}(B)) \\
\mathbf{S}(A \vee B) &= \max(\mathbf{S}(A), \mathbf{S}(B)), \\
\mathbf{S}(\neg A) &= \begin{cases} \text{True} & \text{if } \mathbf{S}(A) = \text{False} \\ \text{Unknown} & \text{if } \mathbf{S}(A) = \text{Unknown} \\ \text{False} & \text{if } \mathbf{S}(A) = \text{True} \end{cases}
\end{aligned}$$

where $\min, \max$ are defined according to the truth ordering False $<$ Unknown $<$ True.

Definition 5.2. Given a CAKE program Π and an interpretation **S** of Π, the *positivized version* of Π wrt **S** is the program obtained from ground instance of Π by replacing each negative literal $\neg A$ by $\mathbf{S}(\neg A)$ (i.e. False, True or Unknown). Such program will be called a 3-CAKE program.[9]

[9]Note that 3-CAKE programs do not contain negation symbols.

Example 5.1. Consider a ground CAKE program Π given by

$$\begin{aligned}\neg\delta R^+(a) &\Rightarrow \delta P^+(a)\\ \neg\delta R^+(a)\wedge\delta P^+(a) &\Rightarrow \delta Q^+(a)\\ \neg\delta T^+(a) &\Rightarrow \delta S^+(a)\\ \delta Q^+(a)\wedge\neg\delta S^+(a) &\Rightarrow \delta T^+(a)\\ \neg\delta T^+(a)\wedge\delta P^+(a)\wedge\delta S^+(a) &\Rightarrow \delta U^+(a)\end{aligned}$$

and the interpretation $\perp = \{\neg\delta P^+(a), \neg\delta Q^+(a), \neg\delta R^+(a), \neg\delta S^+(a), \neg\delta T^+(a), \neg\delta U^+(a)\}$. The positivized version of Π wrt ⊥ is the 3-CAKE program, Π_+, given by

$$\begin{aligned}\mathsf{True} &\Rightarrow \delta P^+(a)\\ \mathsf{True}\wedge\delta P^+(a) &\Rightarrow \delta Q^+(a)\\ \mathsf{True} &\Rightarrow \delta S^+(a)\\ \delta Q^+(a)\wedge\mathsf{True} &\Rightarrow \delta T^+(a)\\ \mathsf{True}\wedge\delta P^+(a)\wedge\delta S^+(a) &\Rightarrow \delta U^+(a).\end{aligned}$$

■

5.2. Computing well-founded semantics

In this section, we provide a method of efficiently computing CAKE programs using well-founded semantics. We start with some preliminary notions.

Definition 5.3. Let Π be a 3-CAKE program. The *3-valued immediate consequence operator, 3-$\mathcal{C}_\Pi$*, on an interpretation **S** of Π is a mapping defined as follows, where $A \in ATM(\Pi)$:

$$3\text{-}\mathcal{C}_\Pi(A) \overset{\text{def}}{=} \begin{cases} \mathsf{True} & \text{if } A \text{ is a fact or there is a rule of the form}\\ & R_1\wedge\cdots\wedge R_k \Rightarrow A \text{ such that } \mathbf{S}(R_1\wedge\ldots\wedge R_k)=1\\ \mathsf{False} & \text{if there is no rule with } A \text{ in its head or, for each}\\ & \text{rule of the form } R_1\wedge\cdots\wedge R_k \Rightarrow A,\ \mathbf{S}(R_1\wedge\ldots\wedge R_k)=0\\ \mathsf{Unknown} & \text{otherwise.}\end{cases}$$

■

The immediate consequence operator 3-$\mathcal{C}_\Pi$ has following property ([1]).

Theorem 5.1. Let Π be a 3-CAKE program. Then the sequence $\{(3\text{-}\mathcal{C}_\Pi)^i(\perp)\}_{i>0}$ is non-decreasing and converges to the least fixpoint of 3-$\mathcal{C}_\Pi$. ■

Let Π be a CAKE program and let **S** be a 3-valued interpretation. We write $3\text{-}\mathcal{CS}_\Pi(\mathbf{S})$ to denote the least fixpoint of 3-$\mathcal{C}_{\Pi_+}$, where Π_+ is the positivized version of the ground instance of the program Π wrt **S**.

Example 5.1. (continued)
We compute 3-$\mathcal{CS}_\Pi(\bot)$.[10]

$$\begin{array}{rcl} 3\text{-}\mathcal{C}_{\Pi_+}(\bot) & = & \{\delta P^+(a), \delta S^+(a), \neg\delta Q^+(a), \neg\delta T^+(a), \neg\delta U^+(a), \neg\delta R^+(a)\} \\ (3\text{-}\mathcal{C}_{\Pi_+})^2(\bot) & = & \{\delta P^+(a), \delta S^+(a), \delta Q^+(a), \neg\delta T^+(a), \neg\delta U^+(a), \neg\delta R^+(a)\} \\ (3\text{-}\mathcal{C}_{\Pi_+})^3(\bot) & = & \{\delta P^+(a), \delta S^+(a), \delta Q^+(a), \delta T^+(a), \delta U^+(a), \neg\delta R^+(a)\} \\ (3\text{-}\mathcal{C}_{\Pi_+})^4(\bot) & = & (3\text{-}\mathcal{C}_{\Pi_+})^3(\bot). \end{array}$$

Thus, 3-$\mathcal{CS}_\Pi(\bot) = \{\delta R^+(a), \delta S^+(a), \delta Q^+(a), \delta T^+(a), \delta U^+(a), \neg\delta R^+(a)\}$. ■

We are now ready to define well-founded semantics of a CAKE program. Let Π be a ground CAKE program. First we define the following sequence of interpretations:

$$\begin{array}{rcl} \mathbf{S}_0 & = & \bot \\ \mathbf{S}_{i+1} & = & 3\text{-}\mathcal{CS}_\Pi(\mathbf{S}_i). \end{array}$$

It can be shown that, for all $i > 0$,

$$\mathbf{S}_0 \prec \mathbf{S}_2 \ldots \prec \mathbf{S}_{2i} \prec \mathbf{S}_{2i+2} \prec \ldots \prec \mathbf{S}_{2i+1} \prec \mathbf{S}_{2i-1} \prec \ldots \prec \mathbf{S}_1.$$

Thus, the even subsequence is non-decreasing and the odd one is non-increasing. Since there are finitely many three-valued interpretations for a given program, each of those subsequences becomes constant at some point. Let $\mathbf{S}_*$ denote the limit of the subsequence $\{\mathbf{S}_{2i}\}_{i\geq 0}$ and let $\mathbf{S}^*$ denote the limit of the subsequence $\{\mathbf{S}_{2i+1}\}_{i\geq 0}$. We define a three-valued interpretation of P, denoted by $\mathbf{S}^*_*(A)$, by

$$\mathbf{S}^*_*(A) = \begin{cases} \textsf{True} & \text{if } \mathbf{S}_*(A) = \mathbf{S}^*(A) = 1 \\ \textsf{False} & \text{if } \mathbf{S}_*(A) = \mathbf{S}^*(A) = 0 \\ \textsf{Unknown} & \text{otherwise} \end{cases}$$

The above interpretation is called the *well-founded model* of a program Π. The well-founded semantics assigns to a given CAKE program Π the well-founded model of Π.

Example 5.1. (continued)
We now compute the well-founded model for the program Π. Note that

$$\mathbf{S}_1 = 3\text{-}\mathcal{CS}_\Pi(\bot) = \{\delta P^+(a), \delta S^+(a), \delta Q^+(a), \delta T^+(a), \delta U^+(a), \neg\delta R^+(a)\}.$$

By continuing the application of the operator 3-$\mathcal{CS}_\Pi$, we obtain the following sequence of interpretations.

$$\begin{array}{rcl} \mathbf{S}_2 & = & 3\text{-}\mathcal{CS}_\Pi(\mathbf{S}_1) = \{\delta P^+(a), \delta Q^+(a), \neg\delta R^+(a), \neg\delta S^+(a), \neg\delta T^+(a), \neg\delta U^+(a)\} \\ \mathbf{S}_3 & = & 3\text{-}\mathcal{CS}_\Pi(\mathbf{S}_2) = \mathbf{S}_1 \\ \mathbf{S}_4 & = & 3\text{-}\mathcal{CS}_\Pi(\mathbf{S}_3) = \mathbf{S}_2. \end{array}$$

Hence $\mathbf{S}_* = \mathbf{S}_4 = \{\delta P^+(a), \delta Q^+(a), \neg\delta R^+(a), \neg\delta S^+(a), \neg\delta T^+(a), \neg\delta U^+(a)\}$ and $\mathbf{S}^* = \mathbf{S}_3 = \{\delta P^+(a), \delta S^+(a), \delta Q^+(a), \delta T^+(a), \delta U^+(a), \neg\delta R^+(a)\}$. Thus, $\mathbf{S}^*_* = \{\delta P^+(a), \delta Q^+(a), \neg\delta R^+(a)\}$. ■

[10]Recall that the positivized version of Π wrt $\bot$ is the 3-CAKE program Π_+.

There remains the problem of determining what is derivable from a defeasible theory under the well-founded semantics. The details follow.

Definition 5.4. Let $D = \langle F, R, > \rangle$ be a defeasible theory and let Π be its translation. Suppose further that $\mathbf{S}_*^*$ is the well-founded model of Π.

A tagged literal $+\Delta R(\overline{u})$ (resp. $+\delta R(\overline{u}), +\Delta\neg R(\overline{u}), +\delta\neg R(\overline{u})$) is wf-derivable from D iff $\Delta R^+(\overline{u})$ (resp. $\delta R^+(\overline{u})$, $\Delta R^-(\overline{u}), \delta R^-(\overline{u})$) belongs to $\mathbf{S}_*^*$.

A tagged literal $-\Delta R(\overline{u})$ (resp.$-\delta R(\overline{u})$,$-\Delta\neg R(\overline{u}), -\delta\neg R(\overline{u})$) is wf-derivable from D iff $\Delta R^+(\overline{u})$ (resp. $\delta R^+(\overline{u})$, $\Delta R^-(\overline{u}), \delta R^-(\overline{u})$) does not belong to $\mathbf{S}_*^*$. ■

Example 5.2. Reconsider the defeasible theory from example 4.1 and its translation into CAKE program Π given by rules (15)-(30). We compute the well-founded model for this program.

$$\begin{aligned}
\mathbf{S}_0 &= \bot = \{\neg\Delta Q^+(n), \neg\Delta R^+(n), \neg\Delta P^+(n), \neg\Delta Q^-(n), \neg\Delta R^-(n), \neg\Delta P^-(n),\\
&\quad \neg\delta Q^+(n), \neg\delta R^+(n), \neg\delta P^+(n), \neg\delta Q^-(n), \neg\delta R^-(n), \neg\delta P^-(n), \neg\delta_1 P^+(n),\\
&\quad \neg\delta_1 P^-(n), \neg\delta_2 P^+(n), \neg\delta_2 P^-(n)\}.\\
\mathbf{S}_1 &= 3\text{-}\mathcal{CS}_\Pi(\mathbf{S}_0) = \{\Delta Q^+(n), \Delta R^+(n), \delta R^+(n), \delta Q^+(n), \delta_1 P^+(n), \delta_2 P^-(n),\\
&\quad \delta P^+(n), \delta P^-(n), \neg\Delta Q^-(n), \neg\Delta R^-(n), \neg\Delta P^+(n), \neg\Delta P^-(n),\\
&\quad \neg\delta Q^-(n), \neg\delta R^-(n), \neg\delta_1 P^-(n), \neg\delta_2 P^+(n)\}.\\
\mathbf{S}_2 &= 3\text{-}\mathcal{CS}_\Pi(\mathbf{S}_1) = \{\Delta Q^+(n), \Delta R^+(n), \delta R^+(n), \delta Q^+(n), \delta_1 P^+(n), \delta_2 P^-(n),\\
&\quad \neg\delta P^+(n), \neg\delta P^-(n), \neg\Delta Q^-(n), \neg\Delta R^-(n), \neg\Delta P^+(n), \neg\Delta P^-(n),\\
&\quad \neg\delta Q^-(n), \neg\delta R^-(n), \neg\delta_1 P^-(n), \neg\delta_2 P^+(n)\}.\\
\mathbf{S}_3 &= 3\text{-}\mathcal{CS}_\Pi(\mathbf{S}_2) = \mathbf{S}_2.
\end{aligned}$$

Thus, $\mathbf{S}_*^* = \mathbf{S}_3$.■

Comparing Examples 4.1 and 5.2, it is immediately seen that the set of tagged literals derivable from the considered theory D is equal to the set of tag literals that are wf-derivable from D. As the next theorem shows, this is a general case for stratified CAKE programs.

Theorem 5.2. Let D be a stratified defeasible theory. A tagged literal α is derivable from D iff it is wf-derivable from D. ■

6. Examples

In this section, we apply well-founded semantics to two non-stratified defeasible theories.

Example 6.1. Consider the defeasible theory $D = \langle \emptyset, R, \emptyset \rangle$, where $R = \{\Rightarrow P(a), \Rightarrow Q(a), \Rightarrow R(a), P(a) \rightsquigarrow \neg Q(a), Q(a) \rightsquigarrow \neg R(a), R(a) \rightsquigarrow \neg P(a)\}$. The CAKE program Π, corresponding to D is the following.

$$\begin{aligned}
&\Delta P^+(a) \Rightarrow \delta P^+(a)\\
&\Delta Q^+(a) \Rightarrow \delta Q^+(a)\\
&\Delta R^+(a) \Rightarrow \delta R^+(a)
\end{aligned}$$

$$\begin{array}{l}
\Delta P^-(a) \Rightarrow \delta P^-(a)\\
\Delta Q^-(a) \Rightarrow \delta Q^-(a)\\
\Delta R^-(a) \Rightarrow \delta R^-(a)\\
\neg\delta P^-(a) \wedge \neg dP^-(a) \Rightarrow \delta P^+(a)\\
\neg\delta Q^-(a) \wedge \neg dQ^-(a) \Rightarrow \delta Q^+(a)\\
\neg\delta R^-(a) \wedge \neg dR^-(a) \Rightarrow \delta R^+(a)\\
\delta P^+(a) \Rightarrow dQ^-(a)\\
\delta Q^+(a) \Rightarrow dR^-(a)\\
\delta R^+(a) \Rightarrow dP^-(a).
\end{array}$$

The program Π is not stratified. We now compute the well-founded model of Π. We start with

$$\begin{aligned}
\mathbf{S}_0 = \bot = \{&\neg\Delta P^+(a), \neg\Delta Q^+(a), \neg\Delta R^+(a), \neg\Delta P^-(a), \neg\Delta Q^-(a), \neg\Delta R^-(a), \neg\delta P^+(a),\\
&\neg\delta Q^+(a), \neg\delta R^+(a), \neg\delta P^-(a), \neg\delta Q^-(a), \neg\delta R^-(a), \neg dP^-(a), \neg dQ^-(a), \neg dR^-(a)\}.
\end{aligned}$$

By applying 3-$\mathcal{CS}_\Pi$ operator, we obtain the following sequence of interpretations.

$$\begin{array}{rcl}
\mathbf{S}_1 & = & \{\neg\Delta P^+(a), \neg\Delta Q^+(a), \neg\Delta R^+(a), \neg\Delta P^-(a), \neg\Delta Q^-(a), \neg\Delta R^-(a), \delta P^+(a),\\
& & \delta Q^+(a), \delta R^+(a), \neg\delta P^-(a), \neg\delta Q^-(a), \neg\delta R^-(a), dP^-(a), dQ^-(a), dR^-(a)\}\\
\mathbf{S}_2 & = & \mathbf{S}_0\\
\mathbf{S}_3 & = & \mathbf{S}_1.
\end{array}$$

Thus $\mathbf{S}_* = \mathbf{S}_3$ and $\mathbf{S}^* = \mathbf{S}_2$. In consequence,

$$\begin{array}{rcl}
\mathbf{S}_*^* & = & \{\neg\Delta P^+(a), \neg\Delta Q^+(a), \neg\Delta R^+(a), \neg\Delta P^-(a), \neg\Delta Q^-(a), \neg\Delta R^-(a),\\
& & \neg\delta P^-(a), \neg\delta Q^-(a), \neg\delta R^-(a)\}. \blacksquare
\end{array}$$

Example 6.2. Consider the defeasible theory $D = \langle F, R, \emptyset\rangle$, where $F = \{Q(a)\}$ and $R = \{Q(x) \Rightarrow P(x), R(x) \Rightarrow \neg P(x), P(x) \Rightarrow Q(x), S(x) \Rightarrow \neg Q(x)\}$. The ground CAKE program Π corresponding to D is the following.

$$\begin{array}{l}
\Delta Q^+(a)\\
\Delta Q^+(a) \Rightarrow \delta Q^+(a)\\
\Delta Q^-(a) \Rightarrow \delta Q^-(a)\\
\Delta P^+(a) \Rightarrow \delta P^+(a)\\
\Delta P^-(a) \Rightarrow \delta P^-(a)\\
\Delta R^+(a) \Rightarrow \delta R^+(a)\\
\Delta R^-(a) \Rightarrow \delta R^-(a)\\
\Delta S^+(a) \Rightarrow \delta S^+(a)\\
\Delta S^-(a) \Rightarrow \delta S^-(a)
\end{array}$$

$$\begin{array}{l}
\Delta P^+(a) \Rightarrow \delta_1 P^+(a)\\
\Delta P^-(a) \Rightarrow \delta_1 P^-(a)\\
\Delta P^+(a) \Rightarrow \delta_2 P^+(a)\\
\Delta P^-(a) \Rightarrow \delta_2 P^-(a)\\
\Delta Q^+(a) \Rightarrow \delta_1 Q^+(a)\\
\Delta Q^-(a) \Rightarrow \delta_1 Q^-(a)\\
\Delta Q^+(a) \Rightarrow \delta_2 Q^+(a)\\
\Delta Q^-(a) \Rightarrow \delta_2 Q^-(a)\\
\delta Q^+(a) \wedge \neg\delta_1 P^-(a) \Rightarrow \delta_1 P^+(a)\\
\delta R^+(a) \wedge \neg\delta_2 P^+(a) \Rightarrow \delta_2 P^-(a)\\
\delta P^+(a) \wedge \neg\delta_1 Q^-(a) \Rightarrow \delta_1 Q^+(a)\\
\delta S^+(a) \wedge \neg\delta_2 Q^+(a) \Rightarrow \delta_2 Q^-(a)\\
(\delta_1 P^+(a) \vee \delta_2 P^+(a)) \wedge (\neg\delta_1 P^-(a) \wedge \neg\delta_2 P^-(a)) \Rightarrow \delta P^+(a)\\
(\delta_1 P^-(a) \vee \delta_2 P^-(a)) \wedge (\neg\delta_1 P^+(a) \wedge \neg\delta_2 P^+(a)) \Rightarrow \delta P^-(a)\\
(\delta_1 Q^+(a) \vee \delta_2 Q^+(a)) \wedge (\neg\delta_1 Q^-(a) \wedge \neg\delta_2 Q^-(a)) \Rightarrow \delta Q^+(a)\\
(\delta_1 Q^-(a) \vee \delta_2 Q^-(a)) \wedge (\neg\delta_1 Q^+(a) \wedge \neg\delta_2 Q^+(a)) \Rightarrow \delta Q^-(a).
\end{array}$$

It is easily checked that the program Π is not stratified. We now compute the well-founded model of Π. We start with

$$\begin{array}{rcl}
\mathbf{S}_0 & = & \perp = \{\neg\Delta Q^+(a), \neg\Delta P^+(a), \neg\Delta R^+(a), \neg\Delta S^+(a), \neg\Delta Q^-(a), \neg\Delta P^-(a),\\
& & \neg\Delta R^-(a), \neg\Delta S^-(a), \neg\delta Q^+(a), \neg\delta P^+(a), \neg\delta R^+(a), \neg\delta S^+(a), \neg\delta Q^-(a),\\
& & \neg\delta P^-(a), \neg\delta R^-(a), \neg\delta S^-(a), \neg\delta_1 P^+(a), \neg\delta_1 P^-(a), \neg\delta_2 P^+(a), \neg\delta_2 P^-(a),\\
& & \neg\delta_1 Q^+(a), \neg\delta_1 Q^-(a), \neg\delta_2 Q^+(a), \neg\delta_2 Q^-(a)\}.
\end{array}$$

Applying 3-$\mathcal{CS}_\Pi$ operator, we obtain the following sequence of interpretations.

$$\begin{array}{rcl}
\mathbf{S}_1 & = & \{\Delta Q^+(a), \neg\Delta P^+(a), \neg\Delta R^+(a), \neg\Delta S^+(a), \neg\Delta Q^-(a), \neg\Delta P^-(a),\\
& & \neg\Delta R^-(a), \neg\Delta S^-(a), \delta Q^+(a), \delta P^+(a), \neg\delta R^+(a), \neg\delta S^+(a), \neg\delta Q^-(a),\\
& & \neg\delta P^-(a), \neg\delta R^-(a), \neg\delta S^-(a), \delta_1 P^+(a), \neg\delta_1 P^-(a), \neg\delta_2 P^+(a), \neg\delta_2 P^-(a),\\
& & \delta_1 Q^+(a), \neg\delta_1 Q^-(a), \delta_2 Q^+(a), \neg\delta_2 Q^-(a)\}.\\
\mathbf{S}_2 & = & \mathbf{S}_1.
\end{array}$$

Hence, $\mathbf{S}^* = \mathbf{S}_* = \mathbf{S}_1$ and thus $\mathbf{S}^*_*$ is given by

$$\begin{array}{l}
\{\Delta Q^+(a), \neg\Delta P^+(a), \neg\Delta R^+(a), \neg\Delta S^+(a), \neg\Delta Q^-(a), \neg\Delta P^-(a),\\
\neg\Delta R^-(a), \neg\Delta S^-(a), \delta Q^+(a), \delta P^+(a), \neg\delta R^+(a), \neg\delta S^+(a), \neg\delta Q^-(a),\\
\neg\delta P^-(a), \neg\delta R^-(a), \neg\delta S^-(a), \delta_1 P^+(a), \neg\delta_1 P^-(a), \neg\delta_2 P^+(a), \neg\delta_2 P^-(a),\\
\delta_1 Q^+(a), \neg\delta_1 Q^-(a), \delta_2 Q^+(a), \neg\delta_2 Q^-(a)\}.
\end{array}$$

Note that the above interpretation agrees with our intuitions.■

7. Conclusions

In this paper, we have provided a new version of defeasible logic which properly deals with cyclic defeasible rules. Like all logical formalisms expressible using the CAKE method, our formalism is tractable.

Although the well-founded semantics can be viewed as a natural generalization of the stratified semantics, we decided to present both of them. The reason is that stratified programs should be computed using the stratified semantics rather, because it is more efficient than the well-founded one.

Defeasible logic, and our variant of this formalism in particular, can be easily extended by allowing defeasible rules with strong premises. This would allow to express naturally occurring reasoning patterns which otherwise are difficult to formalize. An example of this kind of a rule is the rule stating: "If a person is known to be honest, then in the absence of evidence to the contrary she/he can be safely lent a large amount of money".

The prototype system of Defeasible Logic in CAKE based on stratified semantics has been implemented by our students at Warsaw University.

References

[1] S. Abiteboul, R. Hull and V. Vianu. *Foundations of Databases.* Addison-Wesley, 1996.

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

[3] K. R. Apt, H. A. Blair, A. Walker. Towards a Theory of Declarative Knowledge. *Foundations of Deductive Databases and Logic Programming*, J. Minker (ed.), Morgan Kaufmann Publishers, Palo Alto, CA, 89-148, 1988.

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

[5] A. K. Chandra, D. Harel. Horn Clause Queries and Generalizations. *Journal of Logic Programming*, 2(1), 1-15, 1985.

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

[8] V. Lifschitz. On the Declarative Semantics of Logic Programs with Negation. *Foundations of Deductive Databases and Logic Programming*, J. Minker (ed.), Morgan Kaufmann Publishers, Palo Alto, CA, 177-192, 1988.

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

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

[11] 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.

[12] T. Przymusiński. Well-founded Semantics Coincides with Three-valued Stable Semantics. *Fundamenta Informaticae*, IOS Press, XIII, 1990, 445-463.

[13] A. Van Gelder. Negation as Failure Using Tight Derivations for General Logic Programs. *IEEE Symposium on Logic Programming*, 127-139, 1986.