Allowing exceptions in knowledge base : Defeasible Reasoning

Sankhya Singh (u6737668)

Comp 4560 - Advance computing Project
Supervised by - Mr Dirk Pattinson
Research school of Computer science and Engineering
Australian National University, Acton 2601

Abstract. Classical logic is not suited for formalizing exceptions in knowledge base. We introduce the concept of defeasible reasoning and motivate its applicability. Next, we demonstrate the proof theory with suitable reasoner. We apply defeasible logic to various legal texts and argue whether it is suitable for allowing exceptions. Moreover, there were few domains in which applicability of this reasoning was not sound and scalable. Further work can be done in extending defeasible logic with deontic logic and understanding ambiguity propagation in defeasible logic.

Table of contents

Acknowledgments

I would like to thank my supervisor, Mr Dirk Pattinson, for supervising me throughout this project. I would also like to thank my family and friends for supporting me throughout the semester. Finally, I would like to thank Professor Zhenchang Xing for allowing me to enrol in this course.

1 Introduction

Defeasible reasoning is a special form of non-monotonic logic. In monotonic logic, everything that is concluded before a clause is added can still have the same conclusion after the clause is added. Furthermore, adding more clauses in our knowledge base does not necessarily reduce the number of propositions derived. However, in non-monotonic logic conclusions can be refuted by adding extra knowledge. The logic of definite clause with negation as failure in non-monotonic.
Non-monotonic logic aligns with the assumption of default logic, a default is a rule that can be used unless it is overridden by exceptions. The kind of reasoning that follows the concept of inertia: the presumption that given condition will not change unless there is *an external force* (or in this case an exception) and that would change by actual events or dynamic laws.
Formally we can describe Monotonic logic as

(M) If $A \vdash C$, then $A \& B \vdash C$

For reasoning is non-monotonic, the inference relation (M) sometimes fails. Furthermore, it can be thought of as common sense reasoning with a process to allow exceptions. A more prolific way of thinking non-monotonic is of belief revision i.e. one is inclined to non-monotonically infer B from A if adding A to one's stock of belief results in B being believed [6].

The intuition behind defeasible reasoning is to allow the propagation of having exceptions in our current knowledge base. Consider the example that person A might infer that he does not have a brother since if he did, he would certainly know it, and if he does not, in fact, know that he has a brother. Such an inference is, of course, defeasible, since if he subsequently learns that he has a brother, after all, the basis for the original inference is nullified. Another example that will motivate the reader towards defeasible reasoning, if Smith is crossing the street, then it would normally be the case that Smith will succeed in crossing the street. However, this inference is clearly defeasible: Smith might get hit by a truck and never complete the crossing of the street. Hence, our reasoning can be defeated from time to time according to the rules in a modified knowledge base.

Classical logic is not suitable for allowing exceptions in the domain of knowledge base. We can observe this by an example, suppose we want to formalize an example such as, "Birds normally fly".Now if we exhibit this statement in classical logic then it would have an incessant list of exceptions that can not be acknowledged.

$$(\forall x) \cdot \mathrm{Bird}(x) \& \neg \mathrm{Penguin}(x) \& \neg \mathrm{Emu}(x) \& \neg \mathrm{Dead}(x) \& \ldots \supset \mathrm{Fly}(x)$$

Penguin, emu's and dead birds are a couple of exceptions that can not fly. The second crucial problem is that, even if we enumerate all such exceptions, we still can not derive the antecedent of the problem i.e. Bird(x) and without antecedent, we can not derive the consequent.

The definition of a belief in defeasible reasoning has been defined by pollock[3]. Let **p** be a

defeater for **q** as a prima facie reason for **r.** Pollock defines that there are two kinds of defeaters: rebutting defeaters, which are themselves the prima facie reason for believing the negation of the conclusion, and undercutting defeaters, which provide a reason for doubting that q provides any support for the actual circumstance in **r.** An interesting example to support the hypothesis is that Johnson's colour vision becomes unreliable in the presence of pink tigers. Johnson says that the tiger beside him looks pink. Johnson's colour vision becomes unreliable in the presence of pink tigers. Ordinarily, the belief would support the conclusion that the tiger is pink, but this conclusion undercuts the argument in support of the belief. Thus, the argument that the tiger is pink is self-defeating.

The continuation of defeasible belief would now introduce us to the special connective ⇒, with p⇒q meaning that p would be the prima facie reason for q, then undercutting defeaters could be represented by means of negating the conditional. The special connective is also known as a defeasible implication because it can be defeated by any other connective which has a higher superiority rule defined on the same rule.

A more useful property known as a generalization often decides which rules hold more superiority than the other. Consider an example that most A's (set) are B's, and the most AC's are not B's, then one should, upon learning individual B's for both A and C, give priority to the AC generalization over A's. The key understanding is to incorporate the rule of ***specificity***, it gives priority to the rule whose antecedents are more specific than any other rule. Furthermore, if an antecedent of one rule is directly linked to the antecedent of the second rule, then automatically the first rule gains priority over the second rule. Consider an example, if Quakers are typically pacifists, then, when reasoning about a Quaker pacifist, rules pertaining to Quakers would override rules pertaining to the pacifists.

A lot of work has been done on defeasible logic and its application. However, we will sharply focus on applying defeasible logic in real-world scenario problems. G. Governatori et al [1], have focused on representing the defeasible logic proof theory. A lot of the representation has helped informally deriving the proofs in our reasoner. The representation results have also clearly worked on ambiguity propagation and cyclic superiority in defeasible logic.

Ho-pun lam et al [2] have illustrated how to use defeasible logic with deontic extensions in order to formulate the legal contract. The deontic extensions are a gift to defeasible logic. However, the results are only focused on the problems which are of specific legal domains. We will ensure that our reasoning propagates through simple as well as deontic extension in order to support multiple problems from different domains. Francesco el al [4] have described the resource allocation of a rational agent using substructural defeasible logic. The method of proof theory and subsequent rational allocation of applied rules helped in arguing the proof theory in our reasoner. The choice of framework is SPINDLE [5] which is developed by Ho-pun lam et al.

The whole defeasible reasoning is applied to subdomains of real-world scenario problems. Whether we applied to subdivisions of formal legislative laws and then to some university timetabling related problems. The main task was to reason that is defeasible reasoning scalable to a large domain of problems or Is the reasoning suitable when we have no underlying principle of hierarchical regulations. We tested examples from real laws in order to check for inconsistency and ambiguousness in their domain. Moreover, by allowing the process of exceptions we think of the various domains that can have conclusive evidence supporting one argument and it can be refuted with another strong evidence. In this paper, we formalize real-world domain problems with defeasible logic and test their scalability and accuracy on various scenarios. A sample problem has been explained rigorously through the proof theory of defeasible logic. The paper is organised as

follows in section 2 we introduce the concept of defeasible logic (both formally and informally) and we also look at the proof theory of defeasible logic. In section 3 we move to the implementation part, more specifically we examine various reasons for defeasible logic (section 3.1,3.2 and 3.3) then we explain a real-world example through SPINDLE reasoner in section 3.4. Section 3.5 examines the implementation of defeasible logic on different domain problems and discusses why the formalization is sometimes efficient or not. Section 3.6 briefly discusses the implementation regarding section 3.5.

2 Defeasible Logic

2.1 An informal introduction to defeasible logic

A defeasible theory consists of five different aspects for its knowledge base: facts, strict rules, defeasible rule, defeaters and a superiority relation. The defeasible theory can be represented as (F, R,>) where F is set of facts and R is a set of rules while > is an acyclic superiority relation between the set of rules. Acyclic superiority relation helps in determining which rule is superior to another without a continuous recursion between the rules. The language consists of a finite set of literals as with the monotonic logic, where a literal is either an atomic proposition or it's negation. The literal m, ~m it's negation (or it's a compliment). If m = p, then ~m = -p.
Facts are indisputable statements, whose provability are not required. They are thought to be the actions that are performed and will always be true. For example, "Australia is a country" and can be represented as a country(Australia).
A rule R, on the other hand, describes the relationship between the set of literals (the antecedent *A(r))* and the literal (the consequent *C(r))*. The rules can be of three different types namely: Strict, Defeasible and Defeaters.
Strict rules are the good old classical logic connectives: whenever the premise is indisputable so is the conclusion. For example

$$human(x) \rightarrow mammal(x)$$

Which states that every human is a mammal
Defeasible rules are the rules that can be defeated with a shred of contrary evidence. For example,

$$mammal(x) \Rightarrow \neg flies(x)$$

The main synopsis behind this statement is that if we know that X is a mammal then we can defeasibly imply that it cannot fly **unless** there is an exception that defeats this rule (e.g. Bats can fly). Notice that we can have an empty antecedent in our defeasible system i.e. the underlying notion of *presumption*.
Defeaters are the rules whose only purpose is to defeat the specific conclusion.

$$heavy(x) \rightsquigarrow \neg flies(x)$$

The defeaters only want to prevent the conclusion flies.
Superiority Relation between the rules

The priority of one rule over the other depends on multiple factors. A rule whose antecedent is more specific than the other is prone to have more priority over the other. Superiority rule comes to play when we have a certain rule providing the conclusion of some literal and the other set of rules providing the negation of the same literal. In this theory we are mainly focused on *acyclic* superiority relation. Acyclic superiority can be explained from a small example,

r: human(X) ⇒ walk(X)
r': brokenleg(X) ⇒ ¬walk(X)

The example which showcases 'John' is a human and he has a broken leg. If the specification states that r'> r, then with more preciseness we can state that John, who has a broken leg, wouldn't be able to walk. Overrides the main conclusion that humans can walk. It does not make sense to argue with both r > r', and r' > r (cyclic relation). For more details on cyclic superiority in defeasible logic please refer [1]. Moreover, in all the cases superiority relations are used in order to resolve the conflict between two opposite literals, there is no point in using superiority relation where it is not needed in proof theory.

2.2 Formal definition of defeasible logic

The rules of defeasible logic are defined over a language Σ, the set of propositions and literals defined over the language

Definition 2.1. Let Lab be a set of arbitrary labels. Rules have form "r : A(r) ↬ C(r)":
1. r ∈ Lab is a unique name.
2. A(r) is the antecedent, or body, of the rule. A(r) can either have the form A(r) = a_1, .,a_n to denote a multi-set, or A(r) = a_1, .,a_n to denote a sequence.
3. ↬∈ {→,⇒, ⇝} denotes a strict rule, a defeasible rule, and a defeater, respectively.
4. C(r) is the consequent, or head, of the rule. For the head, we consider three options: (a) The head is a single literal 'p', (b) The head is a multi-set 'p_1, . . ., p_m', and (c) The head is a sequence 'p_1;. . .; p_m'.
Three different types of rule in defeasible logic are as follows, strict rules use →, defeasible rules use ⇒, while defeaters use ⇝.
The rule notation for these different kinds of rules can be denoted as follows, $\mathbf{R_s}$ denotes all the strict rules, the set of all defeasible rule is $\mathbf{R_d}$, while the set of all strict and defeasible rule is $\mathbf{R_{sd}}$; and name R[q] with a set of all rules R with head q.

Definition 2.2. A superiority relation on R is a relation > on R. Where r1 > r2, then r1 is superior to r2, and r2 is inferior to r1. Intuitively, r1 > r2 expresses that r1 overrules r2, should both rules be applicable [1]. Throughout the reasoning of this report, we will only look for acyclic superiority rules.

Definition 2.3. A defeasible theory D is a triple (F, R, >) where F is a finite set of literals (called facts), R a finite set of rules, and $> \subseteq R \times R$ an acyclic superiority on R [1]

2.3 Proof theory in defeasible logic

A proof p in standard defeasible logic is a finite vector P(1),....., P(n) of tagged literals, which can

be of the type $\pm\Delta p, \pm\partial p$. Throughout the domain of this report, we would have pre-assumed that facts are simultaneously true at the starting of the computation. The tagged literal $\pm\Delta p$ means that p is strictly proved or refuted in D (Defeasible logic), similarly $\pm\partial p$ means that p is defeasibly proved or refuted in D.
The set of positive and negative conclusions is called extensions. For a given set of facts, a set of rules, and a set of superiority relations, the given extension is unique. Depending upon the order of the rules applied, we can get rather different extensions. The extensions are obtained by examining the notion of rules being applied or discarded at a point. The correct extension which supports our own intuition of reasoning would be unique because intuitively a rule is applicable when every literal in the antecedent has been proven in previous steps.
The proof tag of literals is unique and true for a rule that is sequence applicable,i.e. when the derivation order reflects the order in which the literals appear in the antecedent. A rule sequence is disproven when there exists a literal in antecedent which has already been disproven.

Let us derive the proof conditions for the following tags. Hence, these conditions would form the basis for us to derive the important results from our defeasible reasoner SPINDLE [3].

$+\Delta$: If $P(i+1) = +\Delta q$ then either

$q \in F$ or
$\exists r \in R_s[q] \forall a \in A(r) : +\Delta a \in P(1..i)$

The above representation gives us the condition to prove the tag literal +Δq. Let us understand that what all these lines mean, P(i+1) = +Δq means that if a literal q is proved in i+1[th] layer then it must belong to facts (q ∈ F) or there must exist a strict rule with head q and for every such rule it's antecedent is applicable in the previous layers of the proof (P(1..i)). Here $R_s[q]$ denotes a set of strict rules with head q.

$-\Delta$: If $P(i+1) = -\Delta q$ then
$q \notin F$ and
$\forall r \in R_s[q] \exists a \in A(r) : -\Delta a \in P(1..i)$

The above set of derivations gives us proof of tagged literal -Δq. For the literal to be definite provable in (i+1)[th] layer, it must not belong to facts and for every strict rule with head q, there must exist an antecedent which is not applicable in previous proof layers ($R_s[q]$ denotes set of strict rules with head q).
The above set of proofs are monotonic proofs. Furthermore, we will explore the defeasible proofs, which are indeed non-monotonic.

$+\partial$) If $P(n+1) = +\partial q$ then either
$(1) + \Delta p \in P(1..n)$; or
$(2)(2.1) \exists r \subset R_{sd}[q] \forall a \subset A(r), \ \partial a \subset P(1..n)$, and
$(2.2) -\Delta \sim q \in P(1..n)$, and
$(2.3) \forall s \in R[\sim q]$ either
$(2.3.1) \exists a \subset A(s), \ \partial a \subset P(1..n)$; or
$(2.3.2) \exists t \in R_{sd}[q]$ such that
$\forall a \in A(t), +\partial a \in P(1..n)$ and $t > s$

In order to prove the tagged literal $+\partial q$ in the proof lines, the above set of conditions must be satisfied. The explanation of the rules will be from top to bottom, analyzing each aspect. If a literal q is defeasibly proved in (n+1)[th] line, then (1) states that it must be definitely proved in previous layer i.e. P(1..n) or the

following set of conditions must be satisfied **(2.1)** ∩ **(2.2)** ∩ **(2.3).** **(2.1)** states that there must exist a rule (either strict or defeasible) with head q, and for every such rule r it's antecedent must be applicable in previous proof lines. *($R_{sd}[q]$ denotes either rule belonging to strict or defeasible with head q and A(r) denotes antecedent of that rule)*.

In order to prove the defeasibility of q, we need to consider the rules that belong to $\tilde{q}$. Moreover, we need to consider possible "attack", i.e., reasoning chains in support of ~q. To be more specific: to prove q defeasibly provable we must show that ~q is not definitely provable **(2.2)**. Now, **(2.3)** states that we need to consider the rules which are not known to be inapplicable and have head ~q. Essentially, each such rule s attacks the conclusion q **(2.3.1)**. For q to be provable, each such rule s must be counterattacked by a rule t with head q with the following properties: (i) t must be applicable at this point, and (ii) t must be stronger than s **(2.3.2)**. Thus, each attack on the conclusion q must be counterattacked by a stronger rule. If all the conditions i.e. **(2.1), (2.2) and (2.3)** is satisfied then we can defeasibly prove the literal q.

$-\partial$: If $P(n+1) = -\partial q$ then either
$(1) - \Delta q \in P(1..n)$ or
(2) $(2.1) \forall r \in R_{sd}[q] \exists a \in A(r) : -\partial a \in P(1..n)$ and
$(2.2) + \Delta \sim q \in P(1..n)$ and
$(2.3) \forall s \in R[\sim q]$ either
$(2.3.1) \forall a \in A(s) : +\partial a \in P(1..n)$ or
$(2.3.2) \forall t \in R_{sd}[q]$ such that
$\exists a \in A(t) : -\partial a \in P(1..n)$ or not $t > s$

In order to prove that the literal is defeasibly rejected in D i.e. $-\partial q$, the above set of conditions must be satisfied. If a literal q is defeasibly rejected in (n+1)th line, then (1) states that it must not be definitely proved in previous layer i.e. P(1..n) or the following set of conditions must be satisfied **(2.1)** ∩ **(2.2)** ∩ **(2.3).).** **(2.1)** states that there must exist a rule (either strict or defeasible) with head q, and for every such rule r, it's antecedent must not be applicable in previous proof lines. *($R_{sd}[q]$ denotes either rule belonging to strict or defeasible with head q and A(r) denotes antecedent of that rule)*.

In order to prove the q is defeasibly rejected, we need to consider the rules that belong to $\tilde{q}$. Moreover, we need to consider possible "attack", i.e., reasoning chains in support of ~q. To be more specific: to prove q is defeasibly rejected, we must show that ~q is definitely provable **(2.2)**. Consider all the rules with head ~q then there must be an applicable rule s with head ~q such that no possibly applicable rule t with head q is superior to s **(2.3.1) and (2.3.2).**

3 Implementation

The defeasible logic can be extended with the domain of languages and inference engines. RuleML is a rule-based extensive markup language that supports the declaration of facts, superiority rules and defeaters. The inference engine works on the transformation of defeasible theory into the regular one. The regular theory consists of no superiority rules and defeaters, they have been transformed into a bunch of defeasible rules. The section first focuses on basic reasoning engines used in defeasible logic, then we move on to our main reasoner i.e. SPINDLE [5].

3.1 Deimos and Delores

Maher et al. [7] presented the above two inference engines, which basically runs through a query-answering algorithm (for a single query for the reasoner) while the other works in deriving all the conclusions. The query answering system consists of components that link to backward chaining in Defeasible logic-based directly on the inference rules. The algorithm was implemented in Haskell and the proof of the conclusion is achieved by depth-first search with memorization of already proved conclusions and loop checking.
The algorithm begins by deriving conclusions that can immediately be established: all facts are provable, and all literals with no rules supporting them are unprovable. Then the algorithm proceeds by modifying the rules in the theory. When inferring with positive consequences, the algorithm proceeds similarly to unit resolution for definite clauses in classical logic: when an atom is proved, it can be eliminated from the bodies of all other definite clauses. That is, when a literal is established defeasibly, it can be removed from the body of all rules. Similarly, when it is established that a literal cannot be established defeasibly, then those rules which contain that literal in their body cannot be used to prove the head, and therefore they can be removed from the theory. However, when inferring a positive conclusion ($+\partial$) for a literal p, defeasible provability is complicated, in comparison to definite clauses, by the need to consider rules for ~p .

3.2 Dr- Device

Dr-Device [7] is a defeasible logic reasoner that is implemented using the CLIPS (C Language Integrated product system)
The Dr- Device runs on the defeasible logic rule base or the document which is processed in RuleML. A little bit of insight in RuleML, it is an XML file where you have explicitly explained all the strict rules and facts under different nodes in the XML tree. The inference process involves many steps such as 1) The user inputs the defeasible logic rules as a RuleML format (basically an XML format of the rules discussed later) 2) The reasoner then extracts the strict rules in the RDF format (plain XML format with no precedence of rules) 3) The reasoner then produces namespaces for all the different kind of rules (if not provided the reasoner readily assumes the presumed namespaces) which are also known as N-triples. 4) Then the reasoner sends the namespaces to CLIPS syntax generator and the language generates the

production rules. A more detailed examination of the reasoner can be found by N.Bassiiades et al [8]. Then the reasoner outputs the conclusion in RDF format which can be converted into RuleML format. (see figure 1)

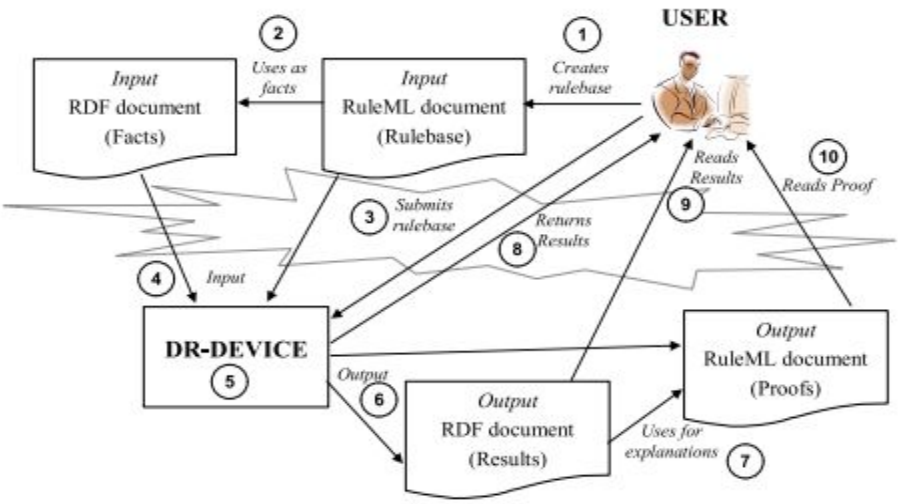

Fig 1 Illustrates the process of the inference engine in Dr-Device

A little detail on the corresponding CLIPS syntax is defined as follows
For example John is looking for a suitable apartment, let the name of apartment be x, hence the rule r1 states that
R1 : => acceptable(x)
But john wants to have pets allowed in the apartment so the next rule would be
R2: ¬pets(x) => ¬acceptable(x)
Then the superiority would be defined as R2>R1, the corresponding CLIPS format would be

```
(defeasiblerule r1
        (John:apartment
                (John:name ?x))
 => (acceptable
                (apartment ?x)))
```

The other rule CLIPS format would be as follows

```
(defeasiblerule r2
                (declare (superior r1))
        (John:apartment
                (John:name ?x)
                (John:pets "no"))
 =>
                (not
        (acceptable       (apartment ?x))))
```

Where the x denotes the name of the apartment.
The main drawback of Dr-Device is that it outputs the conclusion in RuleML format, which makes our task difficult to understand.

3.3 SPINDLE

The spindle is the default reasoner for this project. The main advantage of using Spindle then any of the other reasoners is the ability to read conclusions in a natural way. Moreover, reasoners such as Dr-Devise and Deimos were unable to provide the conclusions in the form of tagged literals. The spindle is built in accordance with the general philosophy of logic programming, it is written in Java and it computes the total conclusion of the defeasible theory.

- It supports all rule types of defeasible logic, namely: fact, strict rules, defeasible rules, defeaters and superiority relations.
- It supports the negation and conflicting heads of two mutually exclusive rules.
- For each type of literals, the spindle spits out the conclusion that is either definitely provable or defeasibly provable based on the strength type and user interference on such rule types.
- The spindle can also be extended to support various forms of defeasible theory such as ambiguity blocking, ambiguity propagation. Interested readers can refer to Representation in defeasible logic by Guido Governatori et al [1].

The inference process in whole defeasible logic is divided into two phases (1) theory transformations phase (2) conclusions generation phase. The theory transformation phase acts as a pre-processing step for our defeasible reasoning, it converts the theory into regularized theory with no superiority rules and defeaters. The conclusion generation phase first asserts that the literal is provable and the strength of its derivation then on the other phase it progressively reduces and simplifies the theory. The above inference process can be understood by the diagram below (see Fig 2):-

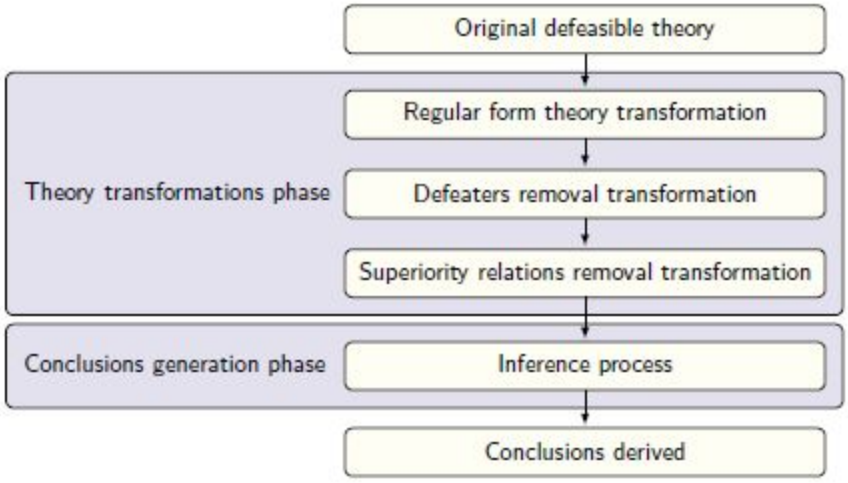

Fig 2 Illustrates the theory transformation phase in Spindle

After loading the defeasible theory into the Spindle, it will check for any constraints that are not honoured by the inference engine (i.e. whether in the normalized theory we have strict rules or defeaters) and will throw an exception to the user.
The algorithm for the inference engine will derive conclusions and send them to users by the following set of the process: -

1. After asserting each fact as an atom (and derive it as a conclusion), then remove the atom from the rules, where atom occurs 'positively' in the body and deactivate or remove the rule where it occurs negatively in the body.
2. Furthermore, scan the list for empty heads in the body. Moreover, take the head element as an atom and search for rules with conflicting heads. If there are no rules that have conflicting heads, then those heads are removed and appended in the list of facts. Otherwise, the atom will be appended to the pending conclusion list until all rules with conflicting heads are proved negatively.
3. Repeat the first step.
4. The algorithm stops when one of the lists of facts is empty or one of the two steps fails. On termination, the reasoner outputs the set of conclusions.

3.4 Example Explanation of defeasible logic using Spindle

The main idea behind showcasing the proof theory of an example is to motivate readers behind the intuition of defeasible reasoning. Furthermore, the results obtained by the inference engine matches the intuition of one's understanding regarding the knowledge base. The goal of the formalization of real-world problems is to help in understanding the basic mechanism behind the reasoner results. One of the pragmatic questions in reasoning with non-monotonic logic is to get around with exceptions and allow the intuition results to match with the inference results.
The following example is of an overseas student. The overseas student has to generally pay the student fees (FPOS) **unless** they come from an exchange background. Now we are allowing the rules that can accommodate the need of having an exception. Moreover, we want to make sure that the exception in our knowledge base is justified by our reasoner and our theoretical proofs match with the intuition results. The below formalization helps us in understanding the goal of theoretical defeasible logic and its implications.

– r1: student(X), overseas(X) ⇒ payFPOS(X)

– r2: student(X), overseas(X), exchange(X) ⇒ ~ payFPOS(X)

– r3: student(X) ⇒ payHECS(X)

– r4: student(X), payFPOS(X) ⇒ ~payHECS(X)

– r4 > r3

Here the predicates student(x) or overseas(x) denotes the literal in the defeasible logic. Consider an overseas student (not an exchange student) must pay the FPOS fees (which rule r1 states). Now we allow the exception rule that is R2, that if the overseas student is an exchange student then it does not have to

pay the FPOS. The rule r3 states that every student must pay the HECS fee **unless** if the student pays the FPOS he/she does not need to pay HECS (rule R4). Rule r4 is more specific to rule r3 because of the principle of *specificity* in non- monotonic logic **(see Intro).** Now let us argue this formalization with a student named Sofia who is an overseas student but not an exchange student has to pay the HECS fees or not. The main intuition behind this example is to see that an exchange student does not have to pay the HECS fees.

Now in order to provide the reasoner with the above set of rules, we need to write them in XML format or RuleML format.

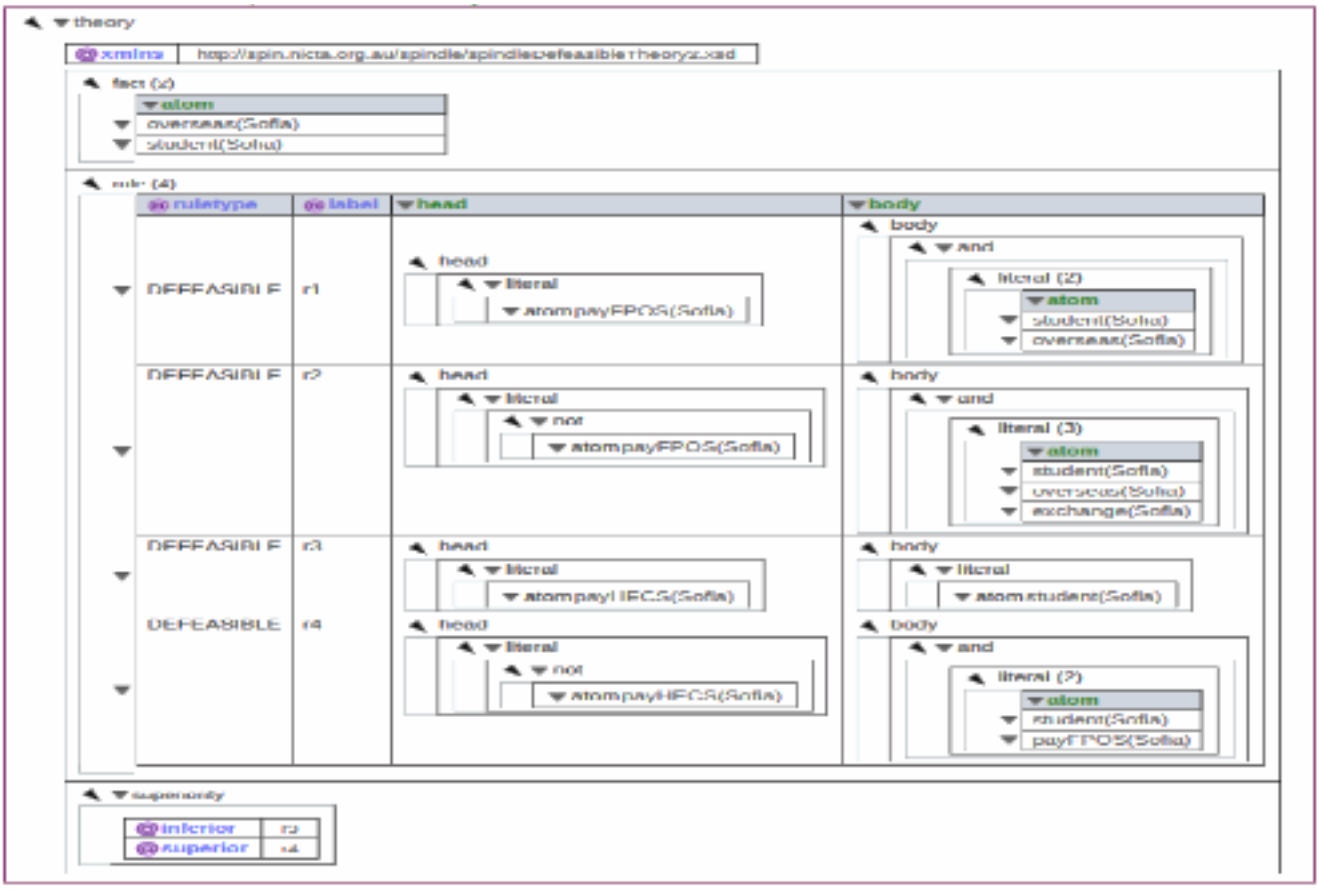

Fig 3 Illustrates the XML schema for our current example

The above figure illustrates the tree structure of the RuleML file that we will send to the Spindle inference engine. The literal overseas(Sofia) and student(Sofia) are facts and therefore their provability is undisputable. Furthermore, we have a defeasible set of rules as defined above.

For the subsequent proofs, we denote P(i) as a tagged literal proved in line i. While P(1...i) corresponds to the tagged literal in the previous layer(with one inclusive)

Let us see the reasoner result from the spindle: -

Facts
-student(Sofia)
-overseas(Sofia)

1) +D overseas(Sofia)

$$+\Delta : \text{If } P(1) = +\Delta overseas(Sofia) \text{ then either}$$

$$overseas(Sofia) \in F \text{ or}$$
$$\exists r \in R_s[overseas(Sofia)] \forall a \in A(r) : +\Delta a \in P(\forall i \in (0,..n) i \leqslant 1)$$

The literal +D overseas(Sofia) is proved in line 1, i.e. P (1), then its provability depends on two factors

1. 1.The literal must belong to the set of facts i.e. overseas(Sofia)$\in$ (*Facts*)
2. or 2. There must exist a strict rule with head overseas(Sofia) and for that such rule, we need to show that its antecedent is definitely provable in previous layers. Here (R_s denotes the strict rules) and A(r) denote the antecedent of each such rule.
3. From the above set of conditions, we can conclude the first condition is satisfied. Hence the literal overseas(Sofia) is definitely provable.

2) +D student(Sofia)

$$+\Delta : \text{If } P(1) = +\Delta student(Sofia) \text{ then either}$$

$$student(Sofia) \in F \text{ or}$$
$$\exists r \in R_s[student(Sofia)] \forall a \in A(r) : +\Delta a \in P(\forall i \in (0,..n) i \leqslant 2)$$

The literal +D student(Sofia) is proved in line 2, i.e. P (2), then its provability depends on two factors

1. 1.The literal must belong to the set of facts i.e. student(Sofia)$\in$ (*Facts*)
2. or 2. There must exist a strict rule with the head student(Sofia) and for that such rule, we need to show that its antecedent is definitely provable in previous layers. Here (R_s denotes the strict rules) and A(r) denote the antecedent of each such rule.
3. From the above set of conditions, we can conclude the first condition is satisfied. Hence the literal student(Sofia) is definitely provable.

3) -D exchange(Sofia)

$-\Delta:$ If $P(2+1) = -\Delta exchange(sofia)$ then
$exchange(sofia) \notin F$ and
$\forall r \in R_x[q] \exists a \in A(r) : -\Delta a \in P(1 \ldots 2)$

The literal -D student(Sofia) is proved in line 3, i.e. P (2), then its provability depends on two factors

1. 1.The literal must **not** belong to the set of facts and 2. For every rule belonging to the strict rule (here $R_s[q]$ denotes strict rule with head exchange(Sofia)) with head exchange(Sofia) is known to be inapplicable. Thus, for every such rule r, there must be at least one antecedent exchange(Sofia) which is not definitely provable in previous layers i.e. P (1…2)
2. From the above set of conditions, we can conclude that both conditions are satisfied. Hence the literal exchange(Sofia) is not definitely provable.

Similarly, the following literals are also not definitely provable by the above scenarios with the same applicable rules as defined above.

4) -D payFPOS(Sofia)

5) -D -payFPOS(Sofia)

6) -D payHECS(Sofia)

7) -D -payHECS(Sofia)

Moving on to the proof of non-monotonic literals, the defeasibility conditions of the following literals have been discussed in the proof theory in detail.

8) +d overseas(Sofia)

Since form P (1…7) proves lines, we can observe that the literal overseas(Sofia) is definitely provable, and the literal is a fact. Hence the literal is defeasibly provable.

9) +d payFPOS(Sofia)

$+\partial:$ If $P(8+1) = +\partial payFPOS(sofia)$ then either
$(1) + \Delta payFPOS(sofia) \in P(1..8)$ or
(2) $(2.1) \exists r \in R_{sd}[payFPOS(sofia)] \forall a \in A(r) : +\partial a \in P(1..8)$ and
$(2.2) - \Delta \sim payFPOS(sofia) \in P(1 \ldots 8)$ and
$(2.3) \forall s \in R[\sim payFPOS(sofia)]$ either
$(2.3.1) \exists a \in A(s) : -\partial a \in P(1..8)$ or
$(2.3.2) \exists t \in R_{sd}[payFPOS(sofia)]$ such that
$\forall a \in A(t) : +\partial a \in P(1 \ldots 8)$ and $t > s$

So, let us prove the defeasibility of the above literal by considering all the steps, so a literal is defeasibly provable if it is definitely provable (1) and then we have a **or set of conditions**. Since our literal payFPOS(Sofia) is not definitely provable (as explained above) then we move on to the set of conditions. The (2.1) states that there must exist a strict or defeasible rule with the head as payFPOS(Sofia), so we have a defeasible rule i.e. rule r1 with head payFPOS(Sofia) (here the R_{sd}[payFPOS(Sofia)] states that there must exist a rule (either strict or defeasible) with head payFPOS(Sofia))

R1: student(Sofia), overseas(Sofia) ⇒ payFPOS(Sofia)

Now moving on to the next set of conditions that must be true, so (**2.1 and 2.2 and 2.3**), then we can prove its defeasibility. But now we need to consider attacks by the negation of this literal i.e. -payFPOS(SOFIA), so **2.2** states that in order to prove payFPOS(Sofia) we need to ensure that -payFPOS(SOFIA) is not definitely provable in the prove lines(1 to 8). Since we have already seen that -D of (-payFPOS(Sofia)) is definitely provable, then this condition is satisfied. Now **2.3** states that we must consider the set of all rules which are not known to be inapplicable and which have head -payFPOS(Sofia). i.e.

R2: student(Sofia),overseas(Sofia),exchange(Sofia) => -payFPOS(Sofia)

Essentially each such rule s (here R2) attacks the conclusion q. For q to be provable, each such rule s must be counterattacked by a rule t with head q with the following properties: (i) t must be applicable at this point, and (ii) t must be stronger than s. Thus, each attack on the conclusion q must be counterattacked by a stronger rule. Now we know that rule R1 with the head (payFPOS) is applicable and the rule R1 is stronger than the rule R2 because it's head payFPOS is obtained by two facts while -payFPOS is obtained by two facts and one non-fact.

Now according to the semantic inheritance network in the defeasible logic, if the antecedent of one rule is defeasibly linked with the antecedent of the second rule, then automatically the first rule gains priority of the second rule. Here in our case rule r1 antecedent is defeasibly linked to rule r2 so rule r1 gains priority.

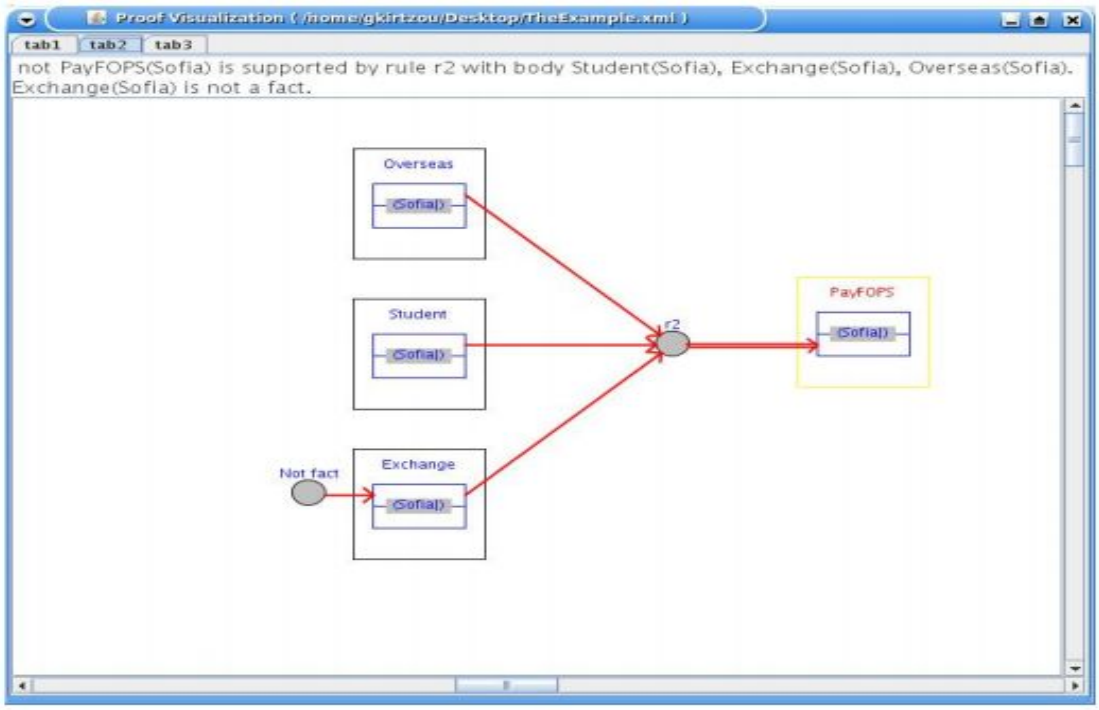

Fig 4 Illustrates the proof visualization of payFPOS(Sofia)

We can also visualize this using block diagram (see Fig 4)

10) **+d** **-payHECS(Sofia)**

$+\partial$: If $P(9+1) = +\partial - payHECS(sofia)$ then either
(1) $+\Delta - payHECS(sofia) \in P(1..9)$ or
(2) (2.1)$\exists r \in R_{sd}[-payHECS(sofia)]\forall a \in A(r) : +\partial a \in P(1..9)$ and
(2.2) $-\Delta payHECS(sofia) \in P(1 \ldots 9)$ and
(2.3)$\forall s \in R[payHECS(sofia)]$ either
(2.3.1)$\exists a \in A(s) : -\partial a \in P(1..9)$ or
(2.3.2)$\exists t \in R_{sd}[-payHECS(sofia)]$ such that
$\forall a \in A(t) : +\partial a \in P(1 \ldots 9)$ and $t > s$

Let us prove the defeasibility of the above literal by considering all the steps, so a literal is defeasibly provable if it is definitely provable (1) and then we have a **or set of conditions**. Since our literal -payHECS(Sofia) is not definitely provable (as explained above) then we move on to the set of conditions. The (2.1) states that there must exist a strict or defeasible rule with the head as -payHECS(Sofia), so we have a defeasible rule i.e. rule r4 with head -payHECS(Sofia) (here the R_{sd}[-payHECS(Sofia)] states that there must exist a rule (either strict or defeasible) with head -payHECS(Sofia) and it is applicable in previous prove lines).
R4: student(Sofia), payFPOS(Sofia) ⇒ -payHECS(Sofia)

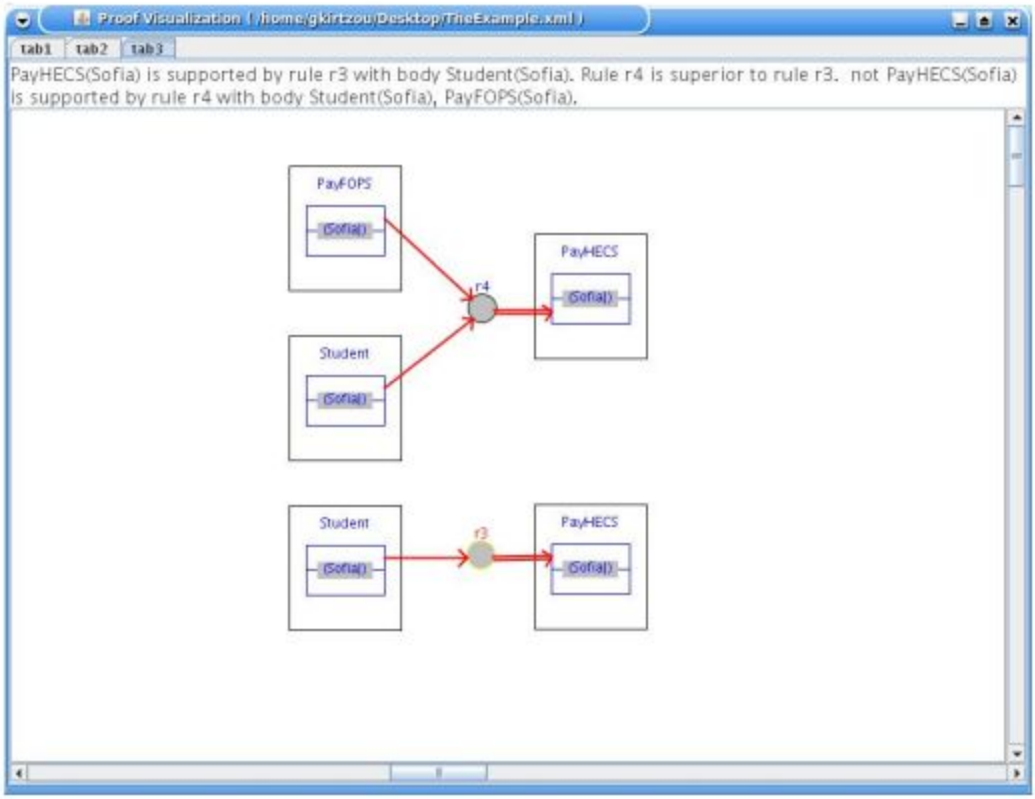

Fig 5 Illustrates the proof visualization of rule r4 and r3

Now moving on to the next set of conditions that must be true, so (**2.1 and 2.2 and 2.3**), then we can prove its defeasibility. But now we need to consider attacks by the negation of this literal i.e. ¬(¬payHECS(Sofia)) = payHECS(Sofia), so **2.2** states that in order to prove ¬payHECS(Sofia) we need to ensure that payHECS(Sofia) is not definitely provable in the prove lines(1 to 9). Since we have already seen that (payHECS(Sofia)) is not definitely provable, then this condition is satisfied. Now **2.3** states that we must consider the set of all rules which are not known to be inapplicable and which have head payHECS(Sofia). i.e.

R3: student(Sofia) ⇒ payHECS(Sofia)

Essentially each such rule S (here R3) attacks the conclusion -payHECS(Sofia). For -payHECS(Sofia) to be provable, each such rule S must be counterattacked by a rule T (here Rule R4) with head -payHECS(Sofia) with the following properties: (i) T (here R4) must be applicable at this point, and (ii) T must be stronger than S. Thus, each attack on the conclusion must be counterattacked by a stronger rule. Now we know that rule R4 with the head (-payHECS) is applicable and the rule R4 is stronger than the rule R3 because it is already being given to our knowledge base. So, the literal -payHECS(Sofia) is defeasibly proved.

This proof can also be visualized using a diagram (see Fig 5)

11)+d overseas(Sofia)

The overseas(Sofia) is a fact, and we know that in the provability conditions of defeasible tagged literal that if a tagged literal is definitely provable then we know that it must be defeasibly proved.

12)+d student(Sofia)

The tagged literal student(Sofia) is a fact, and according to the rules if a literal is definite provable then it must be the case that it is defeasibly provable.

13) -d exchange(Sofia)

The tagged literal exchange(Sofia) is not defeasibly provable, in order to show this first we need to show that it is not definitely provable, which is already proved in P(3). Furthermore, we must establish that it cannot be proven using the defeasible part of the theory. There are three possibilities to achieve this: either we have established that none of the (strict and defeasible) rules with head exchange(Sofia) can be applied. Since there are no rules with head exchange(Sofia) that can be applied. Furthermore, there should be no rule with head ¬exchange(Sofia) that is superior to the rule with head exchange(Sofia) and there is already no rule with head ¬exchange(Sofia), which proves it's defeasibility.

14) -d ¬payFPOS(Sofia)

The tagged literal ¬payFPOS(Sofia) is not defeasibly provable, in order to show this first we need to show that it is not definitely provable, which is already proved in P(5). Furthermore, we must establish that it cannot be proven using the defeasible part of the theory. There are three possibilities to achieve this: either we have established that none of the (strict and defeasible) rules with head ¬payFPOS(Sofia) can be applied or payFPOS(Sofia) is definitely provable (2.2); or there must be an applicable rule s with head payFPOS(Sofia) such that no possibly applicable rule t with head q is superior to s (2.3). The condition (2.3) is satisfied because we have the rule (i.e. rule r2 with head ¬payFPOS(Sofia)) is inferior to rule r1 with head payFPOS(Sofia)

15) -d payHECS(Sofia)

The derivation of this tagged literal is almost the same as the derivation of tagged literal in line 14 because the rule r4 with head -payHECS(Sofia) is stronger than rule r3 with head payHECS(Sofia).

3.5 Application of defeasible logic in real-world scenarios

The ability to allow the exceptions in our knowledge base helps in understanding the applications of all these formalizations in real-world scenarios. This section will primarily focus on applying the laws of defeasible logic on various subsections of legislative acts and we will briefly touch on why this kind of reasoning is ideal for some scenarios, while some may not oblige to our intuition.

3.5.1 Australian capital territory health act 1993

We will be primarily focusing on a subdivision of part 5 in the act known as abortions. The goal of the formalization is to capture any inconsistency in the set of laws, and we know that one can capture those inconsistencies by allowing them in our current knowledge base and argue it's provability by processing them in our inference engine.
Let us get acquainted with some of the formal terms in this subdivision: -
Abortifacient means a medicine, drug or other substance that causes a pregnancy to end prematurely.
Abortion means a medical abortion or surgical abortion.
We first look at the legal text and then we attempt to formalize the given laws in defeasible logic
(1)A person commits an offence if—

(a) The person supplies or administers an abortifacient to another person; and
(b) The abortifacient is supplied or administered by the person for the purpose of ending a pregnancy; and
(c) The person is not a doctor.

(2)Subsection (1) does not apply to—

(a) A pharmacist supplying an abortifacient in accordance with a prescription; or
(b) A person assisting a pharmacist in supplying an abortifacient in accordance with a prescription.

The formalization is as follows:-
r1: personsupply, purposeendingpregnancy, -doctor => abortion(offence)
r2: assistingpharmacist=> -abortion(offence)
r3: pharmacistsupply => -abortion(offence)
r4: abortion(offence) => guilty
r5: =>-guilty

The first rule states that if a person supplies an abortifacient to another person and he/she has the sole purpose of ending a pregnancy and the person is not a doctor then we can defeasibly imply that he/she has committed an offence. Since the rule r1 can have exceptions, we need to use defeasible implication. The second rule is the exemptions from the first rule i.e. If the pharmacist is supplying the abortifacient (in accordance with prescription) or a person is assisting the pharmacist then he/she has not done the abortion offence(r2 and r3).
The above set of rules (i.e. r1,r2 and r3) can be presumed as a set of evidence indicating certain offences and both of them are reliable (i.e. facts). Henceforth, the literals such as, the person who is supplying,

pharmacist(supply) and a person who is assisting the pharmacist are all facts and their provability is not questionable. The rule r4 states that if someone has committed an offence then it is guilty. Furthermore, the rule r5 states that according to the underlying legal system a defendant is presumed innocent (i.e., not guilty) unless responsibility has been proved (without any reasonable doubt).
The above example of legal text has been formalized in a simple manner. However, when there are abundant rules then it is somewhat difficult to construct the formalization.
Now in order to check our knowledge base, let us check the above set of rules with an exception rule. In natural language the rule is described as follows, If a person is a doctor and he/she has the sole purpose of ending the pregnancy then he/she should be proven guilty and he/she must have committed an offence.

r5: doctor, purposeendingpregnancy => abortion(offence)

After processing these rules in our inference engine, we got the derivation of tagged literals. The main highlight of the results was that we derived the +d *-abortion(offence)* and +d *-guilty(x)*, which showcase that we can defeasibily prove that abortion offence was not committed and the presumption of innocence is proved.
The above situation illustrates that there are two sets of evidence which supports the conclusion of the abortion offence (rule r1 and rule r2) and after introducing the inconsistency rule in the knowledge base there is another set of evidence (i.e. rule) that supports some specific conclusion. Furthermore, we have not specified any superiority rule in this whole knowledge base, hence the problem is formed as *ambiguity propagation* in defeasible logic. The situation arises when we have a set of evidence concluding an argument and then another set of evidence concluding the negation of the same argument. The ambiguity could be solved but it is outside the scope of this current project.
Hence the main conclusion obtained with this exercise is that even after allowing an exception in the domain of our formalization, if we don't really know which set of pieces of evidence (or rules) are more specific than other, then we can obtain ambiguous results. Henceforth it is important to note that the manifestation of the defeasible rule to any formal system must be done by carefully supervising the analogy of those underlying laws to defeasible theory.

3.5.2 Stronger futures in northern territory act 2012

In this section, we will primarily focus on the licensing regime of community stores established by the northern territory state. The licensing regime for community stores applies to stores in all the prescribed areas of the Northern territory. Mainly we will cover the stores which play an important role in assisting the food security of Aboriginal people. The determination of license on these community stores is an important step in safeguarding the people who inhabit these lands for decades. Hence, we will primarily focus on the rules that play a key role in the determination of these licenses.
First, we will look at the subject matter as a process of a natural language, then we will formalize the laws. The secretary of the state determines which store is balanced enough to hold a community store license. The conditions that are required for determination is 1) the store should be financially stable 2) It should be a rich source of food and grocery to the nearby community. Furthermore, the license can not be

obtained if the owner and manager of the store have not given written notice to the secretary underlying the reason for the license. The written notice must have a reason for obtaining the license, and a short summary from both owner and manager. The written notice must be submitted in a definite period of time after lodging the application unless the secretary has agreed for a longer period of time. The license would not be issued if the documents provided by the owner or manager are forged. Now if we formulate the above rules in defeasible theory then we get,

R1: financialstability(x) => determination(license)
R2:sourceoffood(x),sourceofgrocery(x) => determination(license)
R3: Notice(owner),Notice(manager)=> determination(license)
R4:reason(x),writtensummary(x)=>Notice(owner)
R5:reason(x),writtensummary(x)=> Notice(manager)

Here the predicate financialstability(x) or sourceoffood(x) denotes the literal and x can be replaced by any ground atom i.e. name of any aboriginal store. The above set of rules seems reasonable for defeasible rules perspective, but these rules exhibit obligation and reparation. Moreover, we cannot use simple defeasible logic here. For example, if the notice is not submitted within a given time frame then the one cannot get the license. Similarly, if all these conditions are not fulfilled then as a result the license would not be obtained.
The legal text of these original rules was spanned across three pages with four divisions. Furthermore, there were a lot of rules which could not be formalized because they did not have a direct link with license determination.
There is no obligation or reparation in the rules provided above, hence we need some different type of notation and reasoning. The defeasible logic can be extended with modal operators. Usually, modal logics are extensions of classical propositional logic with some intentional operators. Thus, any modal logic should account for two components: (1) the underlying logical structure of the propositional base; and (2) the logical behaviour of the modal operators. The classical logic is not well suited for real-life scenarios. We will not go in detail of modal extension with defeasible logic, interested readers can refer [8]. Just a small example to clarify the operator use,

$$\mathrm{AdvertisedPrice}(X) \Rightarrow_{O_{\mathrm{purchaser}}} \mathrm{Pay}(X)$$

Here the above example illustrates that" The purchaser shall follow the Supplier price lists". The primitive symbol O is defined as 'it is obligatory that'. Now we can reformulate the above laws as an extension with modal logic

R1: $O_{s,b}$*financialstability(x),* $O_{s,b}$*sourceoffood(x),* $O_{s,b}$*sourceofgrocery(x) => determination(License)*

R2: $O_{s,b}$*Notice(owner),* $O_{s,b}$*Notice(manager) => determination(License)*

Here $\mathrm{O}_{\mathrm{s,b}}\mathrm{A}$ states that A is obligatory such that s is subject of obligation and b is the beneficiary. Moreover, the subject of obligation, in this case, is determination of license and store owner and manager are the beneficiary.
The above rules can be summed up to one rule as shown below

R3: determination(license) => $\otimes_{i=1}^{n} O_{s,b}B_i \otimes \neg$determination(license)

The symbol ⊗ denotes reparation, the above equation demonstrates that the determination of license is subjected to the above set of obligations and if anyone of them is not succeeded then the reparation is the failure to obtain the license. The formulation demonstrated that the simple defeasible logic is not always appropriate for a given formalization. Moreover, we could have an inconsistency in this formal system such as what if the manager and owner were the same people and they forged the documents while handling to the secretary but these type of inconsistency is hard to formalize and it won't interplay with our current system of rules, so it is necessary to realize that what kind of our problem objective is, Is it a simple DL or it needs extension?. The above example also showcased that with large legal texts, we have numerous rules with exceptions. However, those exceptions were not directly linked with the determination of license. Hence, the involvement of those rules in the knowledge base would be ambiguous. The above example was to showcase the fact that sometimes it is laborious to formalize large legal texts when the exceptions do not exhibit related to the main cause.
.

3.5.3 Final exam timetabling of a university

The final example will help in understanding how one can mould superiority relations in order to support your own intuition answers. Moreover, depending upon the assumption that one supports during initial stages greatly affects the results.
The example will exploit the nature of the final exam and the related coursework period in the university. This example can showcase that for a simple defeasible rule we can have a multitude of exceptions and the superiority between these exceptions can be hierarchical [9]. The example will showcase an easy modification of natural language into defeasible logic. Moreover, the results exploited by the defeasible theory reasoner can be sustained by balancing some exceptions than the other.
Let us describe the natural language of all these rules:-

1) The students who are preparing for the final exam in the university usually have one-week preparation time unless the final exam content is not a big part of their coursework. Moreover, there should be at least a week gap between the final exam and the end of course content.
2) The one-week duration of the final exam should be in week 13 followed by the commencement of the final exam by end of week 13 and going on till week 14 and week 15. Now the courses which do not have a final exam (i.e. have continuous assessment scheme) do not have to ordinarily follow the guidelines in the subsection(1). Moreover, for those courses, the teaching period can be extended by week 13.
3) If any of the course lecturers seems that the teaching period is extended beyond week 13 (because of the constraint regarding the public holidays) then it should be the case that the revision week must be shifted accordingly and then the whole final exam timetabling schedule.

The above formulation of the rules can be done using simple defeasible logic without any modal or deontic extensions. Then we will discuss how the rule 1 and rule 3 seems to have a little bit of redundancy in the formulation which is also known as anomaly detection.

R1: finalcontent(x),oneweekduration => commencementexam(x)

R2: finalexampercentagelessthan40 => ¬oneweekduration
R3: oneweekduration => week13
R4: finalexampercentagemorethan40 => oneweekduration
R5: continousassesment(x) => ¬commencementexam(x)
R6: oneweekduration, continousassesment(x) => ¬week13
R7: lectureragreesonextension(x) => ¬commencementexam(x)

The rule R7 is superior then rule R1 because it has a hierarchical advantage over these rules. It is clearer in introspection than any other rule. Furthermore, let us simply examine the above set of rules with a course (example comp 4450) this course is a continuous assessment.
The rule R5 would be as follows
continousassesment(comp4450) => ¬commencement(comp4450) and
The rule R6 is as follows
Oneweekduration, continousassesment(comp4450) => ¬week12
This is where the hierarchy of the superiority rule takes place. If we process these rules in SPINDLE with R5 > R1 we would obtain the result as *+d ¬commencementexam(comp4450)*. The implicit constraints about the course comp 4450 would be that it is indeed a continuous assessment and it is a fact. Hence, provability is indisputable. Furthermore, these implicit constraints can sometimes help us in facilitating the superiority between the rules. Moreover, if we had another example course such as comp 3620 which is not a continuous assessment course and which is indeed a fact, then the rule R1 would override other rules such as R5, and derive the proof tagged literal as *+d commencementexam(comp3620).*

3.6 Brief discussion regarding the implementation

The formal representation of the above examples can be a process of drafting specific rules or regulation according to the user's specific choice. In the first example, we formed the rules for a subset of laws in the health care act and the result was the process of ambiguity propagation in defeasible logic. The process can be seen as *Anomaly detection* i.e. before formalizing the problem we did not know that there would be any circular dependency or ambiguous results. Hence, this process could be understood as checking any formal system for any inconsistencies and since we are not specifying any superiority rules, we let the reasoner lay out the proof. The result would have been very different if we allowed priority of inconsistency rule in defeasible theory. The second example can be seen as the matter of *hypothetical testing* since we formalized the set of rules which had a rather large domain. Moreover, the formalization was done through modal extension, we hypothesized over an inconsistency that what would be the case if the owner and manager are the same people and they have forged the documents for the license. Hence, we could not get the answers because there was ambiguity on how to allow those rules in deontic logic. The third example focused on the aspect of *debugging* i.e. in many of those cases we know that what actual answer of the query would be, as the course comp 4450 is continuous assessment course, it would have greater precedence over the other rules(because of the hierarchical structure). Hence if we change

the current set of the regulation (superiority relation) it would give us a different result. Debugging suggests that changes to the regulations will have an effect on the desired outcome.

4 Conclusions

The following findings were obtained during the whole project

- Classical logic is not suitable for allowing exceptions in the knowledge base, defeasible reasoning is the appropriate norm.
- The selection of appropriate reasoner for defeasible reasoning was dependent on various factors and Spindle stood out because the extension of proof tags was readable in a simple format.
- The example implementation showcased how the proof theory works in defeasible logic. Furthermore, it aligns with the idea that intuition results should match with the inference result.
- Defeasible theory can be tested on legal texts, especially to the laws which can have exceptions. However, they must align with the informal definition of the theory i.e. (F, R,>), as the conclusion we had obtained on abortion (law) was ambiguous.
- Simple defeasible logic can not be implemented to the legal documents which have obligation and reparation. It must be extended with deontic operators. Moreover, the logic is scalable to a large domain of problem with exceptions unless the exception has a prominent role to play in defeasibility of rules.
- The regulation of superiority rules and the assumption of certain facts at the beginning of reasoning can mould the reasoner result in a certain manner.

In the future, we want to prevent ambiguous results by understanding the domain of problem we are applying to in defeasible logic. Furthermore, a general retrospection could be done on the problems which have heaps of exceptions in the knowledge base.

References

[1]Antoniou, Grigoris, et al. "Representation results for defeasible logic." ACM Transactions on Computational Logic (TOCL) 2.2 (2001): 255-287.

[2]Lam, Ho-Pun, and Mustafa Hashmi. "Enabling reasoning with LegalRuleML." Theory and Practice of Logic Programming 19.1 (2019): 1-26.

[3]Pollock, John L. "Defeasible reasoning." Cognitive science 11.4 (1987): 481-518.

[4]Olivieri, Francesco, et al. "Applications of Linear Defeasible Logic: combining resource consumption and exceptions to energy management and business processes." arXiv preprint arXiv:1908.05737 (2019).

[5]Lam, Ho-Pun, and Guido Governatori. "The making of SPINdle." International Workshop on Rules and Rule Markup Languages for the Semantic Web. Springer, Berlin, Heidelberg, 2009.

[6]Hunter, Daniel. "Non-monotonic reasoning and the reversibility of belief change." Uncertainty Proceedings 1991. Morgan Kaufmann, 1991. 159-164.

[7]Kontopoulos, Efstratios, Nick Bassiliades, and Grigoris Antoniou. "Deploying defeasible logic rule bases for the semantic web." Data & Knowledge Engineering 66.1 (2008): 116-146.

[8]Bassiliades, Nick, Grigoris Antoniou, and Guido Governatori. "Proof explanation in the DR-DEVICE system." International Conference on Web Reasoning and Rule Systems. Springer, Berlin, Heidelberg, 2007.

[9]Koons, Robert, "Defeasible Reasoning", The Stanford Encyclopedia of Philosophy (Winter 2017 Edition), Edward N. Zalta (ed.), URL = <https://plato.stanford.edu/archives/win2017/entries/reasoning-defeasible/>

[10]Strasser, Christian and G. Aldo Antonelli, "Non-monotonic Logic", The Stanford Encyclopedia of Philosophy (Summer 2019 Edition), Edward N. Zalta (ed.), URL = <https://plato.stanford.edu/archives/sum2019/entries/logic-nonmonotonic/>