1.1: Motivation

The intuition behind defeasible reasoning is to allow the propagation of having exceptions in our current knowledge base. In our basic monotonic reasoning (classical logic) we cannot argue the implication reasoning. Moreover, there is a very bleak chance that we can argue our own set of propositions when we have formalised it. 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.

The process of allowing exceptions in our knowledge base motivates around the concept of defeasible logic. Moreover, consider an example of a flight system operating in the airport, for a current departure time the system does not explicitly have to state that these current set of flights are not operating (using negation for each flight) unless the system can specify a weak rule which could be defeated by a rule with stronger premises. Defeasible laws enable us to express what is really known to our current knowledge base, rather than creating an exhaustive list of exceptions that might not be the case for our current knowledge base. Moreover, the relationship between the premises and the conclusion is a tentative one i.e. it can be defeated by additional information. 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.

1.2: Background

Defeasible reasoning could be understood as non-monotonic reasoning. Moreover, if a set of premises logically entails a conclusion, then any superset will also entail that same conclusion. However, in defeasible reasoning, the consequence is non-monotonic. A conclusion follows nonmonotonically from a set of premises as these premises are nearly true in all the models that verify the premises.

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. (1) Johnson says that the tiger beside him looks pink. (2) Johnson's colour vision becomes unreliable in presence of pink tigers. Ordinarily, the belief (1) would support the conclusion that the tiger is pink, but this conclusion undercuts the argument in support of the belief (2). 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

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.

2 Design and Implementation

2.1.1: 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

$$\text{human}(x) \rightarrow \text{mammal}(x)$$

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,

$$\text{mammal}(x) \Rightarrow \neg \text{flies}(x)$$

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.

$$\text{heavy}(x) \rightsquigarrow \neg \text{flies}(x)$$

The defeaters only want to prevent the conclusion flies.
Superiority Relation between the rules
The priority of one rule over the another 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 certain rule providing the conclusion of some literal and the another set of rules providing the negation of 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 showcase '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 that John which has a broken leg, wouldn't be able to walk. Overrides the main conclusion of 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 conflict between two opposite literals, there is no point in using superiority relation where it is not needed in proof theory.

2.1.2: 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 1.1. A rule r : A(r) ,→ C(r) consists of its unique label r, its antecedent A(r) (A(r) may be omitted if it is an empty set) which is a finite set of literals, an arrow ,→ (which is a placeholder to be introduced later), and its head (or consequence) C(r) which is a literal [1].
Three different types of rule in defeasible logic is 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 set of all strict and defeasible rule is $\mathbf{R_{sd}}$; and name R[q] with set of all rules R with head q.

Definition 1.2. 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 1.3. 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].

Definition 1.4. Given a defeasible theory D = (F, R, >). A rule $r_a \in R$ with antecedent A is considered to be more specific than another rule $r_b \in R$ with antecedent B if we can derive all of B from A using only the rules in R, but not vice versa.

2.1.3: Proof theory in Defeasible logic

The proof in defeasible logic uses following notations, based on the concept of a proof, the derivation in D = (F,R,>) is a finite sequence of P = (P(1),…….P(n)) of tagged literals. A tagged literal consists of the sign '+', which denotes provability, "- "denotes failure to prove. The tags are used to indicate the strength of indicated tag literals. For the care of simplicity, we will focus on two specific tags in general: Δ indicates definite provability based on monotonic proofs, and ∂ denotes defeasible provability based on non-monotonic proofs.

+Δq indicates that the literal q is definitely provable in D (using only facts and strict rules)
- Δq indicates that the literal q is definitely rejected in D.
+ ∂q indicates that the literal is defeasibly provable in D.
- ∂q indicates that the literal is defeasibly rejected in D.

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 the strict rule 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 derivation give us the 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 strict rule 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
$(2)(2.1) \exists r \in R_{sd}[q] \forall a \in A(r), +\partial a \in P(1..n)$, and
$(2.2) - \Delta \sim q \in P(1..n)$, and
$(2.3) \forall s \in R[\sim q]$ either
$(2.3.1) \exists a \in A(s), -\partial a \in P(1..n)$; 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 belongs to ~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 be 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 belongs to ∼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).**

Implementation

The defeasible logic can be extended with the domain of languages and inference engines. RuleML is 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 in 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 [3].

2.1: Deimos and Delores

Maher et al. [6] presented the above two inference engines, which basically runs through 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 component that links 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 [2].

2.1: DR DEVICE

Dr-Device [6] is a defeasible logic reasoner that is implemented using the CLIPS (C language Integrated product system)