Modeling Defeasible Reasoning with Answer Set Programming

Racquel Dennison
Department of Computer Science
University of Cape Town
dnnrac003@myuct.ac.za

Abstract
Defeasible reasoning aims to formalise aspects of human reasoning wherein prior conclusions are open to revision when new information emerges. While Classical propositional logic allows for natural language to be encoded in symbolic constructs, it lacks the ability to express nonmonotonicity. Given this limitation, much work has been done in the field of nonmonotonic reasoning to develop frameworks that incorporate nonmonotonicity. The approaches to defining defeasible consequences take an approach to ranking formulas in a knowledge base. This has shown to be a viable way of modeling defeasible reasoning; however, as the knowledge base grows, the complexity of computing the entailment grows exponentially requiring more resources and computing time. Many applications that model defeasible reasoning have been developed using imperative languages, while little work has been done with the use of declarative languages. In this review, we will provide a brief history and motivation for defeasible reasoning. Subsequently, we will explore a form of defeasible reasoning that will be modeled in the language, Answer Set Programming which is a declarative language that follows similar semantics to logic programming.

1 INTRODUCTION

At its core, knowledge representation entails the conveyance of information about an environment through a linguistic medium [10]. The process of manipulating such information, with the objective of drawing conclusions or deriving new insights, is commonly referred to as reasoning. While humans demonstrate a capacity for reasoning effortlessly, the same cannot be said for artificial intelligence systems. Artificial systems require rigorous and structured methodologies that provide a clearly defined framework for the deduction of conclusions from a given set of information. To allow for this process, knowledge is commonly encoded in a symbolic form. With the use of mathematical constructs, these symbols undergo manipulation which allows for the generation of new symbolic representations. In natural language, this is when conclusions are drawn from statements about some environment[15].

A simple yet powerful method for achieving this is through Classical Propositional Logic. This is widely employed in knowledge representation theory. Propositions, basic statements about the world, are assigned truth values and are the basic building blocks of the symbolic language. Using defined operators, more complex sentences can be constructed to encode the world more accurately [1]. Knowledge bases are collections of information that enable systems to reason effectively [11].

Propositional logic has several useful characteristics. It is simple to express and can express statements about the world without attaching biased meaning to it. However, it lacks the ability to express typicality. This is where a particular implication usually holds however, there may be some special cases where it does not. For example, *birds can fly*, *a penguin is a bird*, however, it is known that a penguin can not fly. It is typically the case that birds can fly, however, the penguin is an atypical creature.

Moreover, Propositional logic encounters limitations in handling nonmonotonicity. In nonmonotonic reasoning, previous conclusions may be retracted due to the introduction of new information into a knowledge base. In logic systems, entailment typically allows for drawing reasonable conclusions from a knowledge base. However, entailment in propositional logic is monotonic, meaning that once conclusions are drawn, they remain unaffected by subsequent information. This restricts the expressive capabilities of Propositional Logic.

Human reasoning, on the other hand, often is a nonmonotonic process. Therefore, a framework accommodating nonmonotonicity is essential to model systems capable of such reasoning. Defeasible reasoning systems emerge as a nonmonotonic approach to reasoning.

Numerous approaches to defeasible reasoning have been developed [11]. This review will concentrate on a specific form of defeasible reasoning embedded within the KLM(Kraus, Lehmann and Magidor) framework, initially proposed by Kraus, Lehmann, and Magidor [13].

The KLM framework extends propositional logic by incorporating a preferential consequence relation into its semantics at a meta-level. Within this framework, entailment and satisfaction are defined using preferential interpretations [11].

This consequence relation is presented by a set of postulates crafted to indicate that if a pair (α, β) is within the relation [11], it denotes "From α, I am inclined to infer β unless I possess information that contradicts this inference". This formulation mirrors the way humans tend to reason. To illustrate this, consider the following scenario: In the absence of light shining on a car, I deduce that the color of the car is grey. However, when light is shining on the car, revealing the car to be red, I retract my initial conclusion due to the presence of new information.

In this review, we will center our attention on a formalisation of deductive reasoning known as Rational Closure. Rational Closure serves as a prototypical way of drawing inferences. It operates on the principle of only deriving conclusions where typicality holds in most cases [3].

In their paper, Lehmann and Magidor [3] presented an algorithm for computing Rational Closure, which relies on ranking statements within a knowledge base based on their exceptionally. Statements deemed less exceptional are assigned lower rankings in this hierarchy. The review will further outline the semantics underlying the computation of the Rational Closure of a knowledge base as outlined by Lehmann and Magidor.

The algorithm enabling the computation of Rational Closure can be implemented in various programming languages. While traditionally presented imperatively there is merit in considering declarative ways of presenting it.

Answer Set Programming (ASP) emerged in the late 1990s as a paradigm for declarative modeling and solving NP-hard computational problems. This language has played a significant role in research areas about knowledge representation [18].

ASP has prominently featured in modeling nonmonotonic reasoning [5], yet there has been limited exploration within the KLM framework regarding the application of ASP for modeling Rational Closure.

Understanding the semantics and underlying architecture will aid in understanding how we will be able to implement Rational Closure in a declarative manner.

2 CLASSICAL PROPOSITIONAL LOGIC

Classical Propositional Logic formalizes reasoning through the utilisation of statements and connectives. These elements are used iteratively to construct a language [1]. Fundamentally, a language within propositional logic is dedicated to articulating factual assertions about the world using atoms and connectives. In this paper, we will denote the language as $\mathcal{L}$. The overarching objective of propositional logic is to formalise information representation which then facilitates the derivation of conclusions and inferences from a provided set of statements. Conceptually, this approach removes subjective interpretations, emphasising mathematical definitions that enable the encoding of truth.

2.1 Language

Propositional atoms serve as the fundamental constituents of knowledge representation. These entities are denoted by statements such as *"bird"* or *"fly"* or can be represented as letters such as p, q. The set $\mathcal{P}$ encompasses all atomic propositions, which may potentially be infinite. Despite our ability to attribute specific meanings to these atoms, their evaluation remains uninfluenced by such interpretations [11]. By employing connectives such as $\rightarrow, \wedge, \leftrightarrow, \vee$, atoms can be combined to construct well-formed formulas within the set $\mathcal{L}$ [1][11]. The negation operator $\neg$ enables the negation of a formula. Notably, the negation formula functions as a unary connective, whereas the remaining connectives are binary. Unary connectives are applied to one formula while binary connectives require two formulas. Formulas are typically denoted by Greek letters like α, β, γ, and they are defined recursively. Specifically, given any $\mathcal{L}$, it can be expressed as follows: for a given $p \in \mathcal{P}$ and $\tau, \alpha \in \mathcal{L}$, τ can take the form of p, τ, $\neg\tau$, $\tau \leftrightarrow \alpha$, $\tau \vee \alpha$, or $\tau \wedge \alpha$ [1]. The complete set of possible formulas constitutes the language of propositional logic, denoted as $\mathcal{L}$. With a comprehensive grasp of this foundational framework for knowledge representation, we can effectively utilise these formulas to facilitate inference drawing from a set of propositions [1]. When we model facts about the world that we want to conclude from, we have a knowledge base that acts as a repository.

Definition 2.1.1 A knowledge base, $\mathcal{K} \subseteq \mathcal{L}$, is a finite set of propositional formulas.

2.2 Semantics

Semantics in logic plays a crucial role in defining the concept of truth and facilitates language analysis [11]. By considering a set of propositional atoms, we can attribute truth values to them using an interpretation.

Definition 2.2.1: An interpretation is formally defined as a function ϕ that assigns truth values to propositional atoms. Mathematically, this function is represented as $\phi : \mathcal{P} \rightarrow \{ T, F \}$.

Given an interpretation, denoted as ϕ, and a set $\mathcal{P} = \{p, q\}$ of propositional atoms, we can then assign truth values to each atom. For instance, $\phi(p)$ = T and $\phi(q)$ = F. The collection of all possible interpretations is denoted as $\mathcal{U}$. For the set $\mathcal{P}$, the set $\mathcal{U}$ would be $\{pq, \overline{p}q, \overline{pq}, p\overline{q}\}$.

Definition 2.2.2: If a particular atom holds under a given interpretation, then that interpretation is said to satisfy the atom. This concept is symbolically represented as $\models$ [11].

Following from our previous example, $\phi \models p$, while $\phi \not\models q$. When working with formulas, we similarly assign truth values. Essentially breaking down the formula into its simplest atoms and then using the definitions of each connective used,s allows for the interpretation to be determined. Table 1 provides the definitions of each connective.

Table 1: Truth table of connectives

p	q	$\neg p$	$p \rightarrow q$	$p \wedge q$	$p \vee q$	$p \leftrightarrow q$
1	1	0	1	1	1	1
1	0	0	0	0	1	0
0	1	1	1	0	1	0
0	0	1	1	0	0	1

2.3 Satisfaction

All the interpretations that satisfy a knowledge base are referred to as models of that knowledge base. The set of models of the knowledge base $\mathcal{K}$ is denoted as $\mod(\mathcal{K})$ [11].

Definition 2.3.1: For any formula $\alpha \in \mathcal{L}$, let $\hat{\alpha}$ be defined as $\{ \phi \in U \mid \phi \models \alpha \}$. For any interpretation $\phi \in \hat{\alpha}$, ϕ is termed a model of α [11]. We will denote $\hat{\alpha}$ as the set of all models of α.

The concept of satisfaction can be extended to define a logical consequence, which occurs when one statement logically follows from another. For instance, given the statements "Every car is red" and "I drive a car," it logically follows that my car is red.

Definition 2.3.2: Given two formulas α and β, $\alpha \models \beta$, read as " β is a logical consequence of α" if for every interpretation $\phi \in \mathcal{U}$, if $\phi \models \alpha$, then $\phi \models \beta$ [1].

From this definition, it is evident that the models of α are a subset of the models of β, denoted as $\hat{\alpha} \subseteq \hat{\beta}$.

2.4 Entailment

Entailment signifies a pattern of reasoning that delineates which formulas logically follow from a given set of formulas, also known as a knowledge base. In natural language, it can be illustrated with an example such as: Given the statements "All mammals are warm-blooded" and "A dog is a mammal," we can logically deduce that "A dog is warm-blooded." A more formal definition is provided as follows:

Definition 2.4.1: Given a set of formulas $\mathcal{K}$ and a formula β, β is said to be entailed by $\mathcal{K}$, denoted as $K \models \beta$, if and only if for every interpretation $\phi \in \mathcal{U}$ such that $\phi \models \mathcal{K}$, it holds that $\phi \models \beta$.

In simpler terms, this implies that the set of models of $\mathcal{K}$ is a subset of the set of models of β, denoted as $\text{mod}(\mathcal{K}) \subseteq \hat{\beta}$. For example, consider the knowledge base $\mathcal{K} = \{ p \wedge r \}$, where the set of propositional atoms is $\mathcal{P} = \{ p, q, r \}$. We want to verify whether $\mathcal{K} \models p \vee r \rightarrow q$. Let $\mathcal{U}$ denote the valuations of the set $\mathcal{P}$. $\mathcal{U}$ $=\{ pqr, pq\overline{r}, p\overline{q}r, p\overline{qr}, \overline{p}qr, \overline{p}q\overline{r}, \overline{pq}r, \overline{pqr} \}$. We can see that from $\mathcal{U}$, the interpretations that satisfy our knowledge base $\mathcal{K}$ are $\{ pqr, p\overline{q}r \}$. Computing the models of $p \vee r \rightarrow q$, would result in $\{ pqr, pq\overline{r}, \overline{p}qr, \overline{p}q\overline{r}, \overline{pqr} \}$. From the definition, we can see that the models of $p \vee r \rightarrow q$ do not have the models of $\mathcal{K}$ as a subset so we can conclude that $\mathcal{K} \not\models p \vee r \rightarrow q$.

3 DEFEASIBALE REASONING

To motivate defeasible reasoning, let's consider a typical example. Suppose we have the following knowledge base $\mathcal{K} = \{b \rightarrow f, b \rightarrow w, p \rightarrow b, p \rightarrow \neg f\}$. Here, b represents *"bird"*, w represents *"wings"*, p represents *"penguin"*, and f represents *"fly"*. From this knowledge base, it's apparent that there are no models of $\mathcal{K}$ where *"penguin"* is true [11]. Indicating that *"penguin"* do not exist, however, this is not the case. They are simply atypical creatures when considering birds.

Classical Propositional Logic fails to accommodate exceptions in the knowledge base due to its monotonic nature. Nonmonotonic frameworks aim to address this limitation. One prominent framework we will focus on is the KLM framework.

3.1 KLM Framework and Defeasible Reasoning

In their paper [13], Kraus, Lehmann, and Magidor argued that any nonmonotonic knowledge base should express exceptionality as implying that "x typically follows from y." They introduced exceptionality by introducing a consequence relation denoted as $|\!\sim$. For formulas α and β from a language $\mathcal{L}$, the expression "β typically follows from α" is represented as $\beta \,|\!\sim \alpha$ [2]. Ranked interpretations are utilized in defining the semantics of $|\!\sim$ [3]. Additionally, it is noteworthy that Lehmann and Magidor [3] extended Propositional Logic. Importantly, this means that the semantics defined above still apply to classical statements. The KLM framework is a superset of Propositional Logic.

3.2 Ranked Interpretations

Ranked interpretations play a vital role in outlining how we can introduce exceptionality into a knowledge base, and how we are then able to draw inferences from an exceptional knowledge base.

Ranked interpretations are simply when the set of all interpretations has been assigned a rank. The lower ranks constitute the more typical worlds while higher ranks are more unlikely. Finite ranks are possible however infinite ranks are impossible. The ranking function is defined as follows.

Definition 3.2.1: Let $\mathscr{R}$ be a function which is known as a ranked interpretation function, $\mathscr{R} : \mathcal{U} \rightarrow \mathbb{N} \cup \{\infty\}$. If there exists some $i \in \mathbb{N}$ such that for some $u \in \mathcal{U}$, $\mathscr{R}(u) = i$, then there must be a $v \in \mathcal{U}$ such that $\mathrm{R}(v) = j$ with $0 \leq j < i$. $\mathcal{U}$ is the set of all possible interpretations [2].

Table 2 shows a possible ranked interpretation for the set of propositional atoms $\mathcal{P} = \{b, f, p\}$.

Table 2: Ranked Interpretation for $\mathcal{P} = \{ b, f, p \}$

∞	$p\overline{b}f$ $p\overline{b}\overline{f}$
2	pbf
1	$\overline{p}b\overline{f}$ $pb\overline{f}$
0	$\overline{pbf}$ $\overline{pb}f$ $\overline{p}bf$

From table 2, we can see that the interpretation $\overline{pbf}$ is more typical and desired to the interpretation $pb\overline{f}$, this is due to the fact that $\overline{pbf}$ is at a lower rank to $pb\overline{f}$.

3.3 Satisfaction

The ranked interpretation provides us with the relative typicality of worlds, through the use of a mathematical function that allows us to rank interpretations. When assessing whether a ranked interpretation satisfies a defeasible implication, we only need to ensure that it holds for the most typical worlds [11]. This simply means that for all the minimal interpretations in the ranking, they all are true for the defeasible implication that is being considered.

Definition 3.3.1: A world is minimal for an interpretation if, given a ranked interpretation $\mathscr{R}$ and any formula $\alpha \in \mathcal{L}$, it satisfies $u \in \alpha_R$ (the models of α in $\mathscr{R}$ is minimal) if and only if there is no $t \in \alpha_R$ such that $\mathscr{R}(t) < \mathscr{R}(u)$ [11].

From the definition above, we observe that for $\alpha \,|\!\sim\, \beta$ to satisfy $\mathscr{R}$, it only needs to satisfy $\alpha \rightarrow \beta$ in the most typical worlds. Since propositional formulas do not exhibit exceptionality, it is required that for every finite rank in $\mathscr{R}$, any propositional formula needs to be satisfied. If $\mathscr{R} \models \alpha |\!\sim\, \beta$, then we say that $\mathscr{R}$ is a *model* of $\alpha |\!\sim\, \beta$ [2].

3.4 Entailment

Defeasible entailment in a knowledge base is a crucial concept that requires careful definition. While there isn't a single standard approach to this, certain entailment relations have emerged over the years, demonstrating themselves to be more reasonable than others [13]. Building upon the foundational semantics outlined in Shoham's papers [20][21], Kraus, Lehmann, and Magidor (KLM) proposed seven consequence relations that they argued to be reasonable for modeling defeasible reasoning. These relations provide a framework for understanding how defeasible entailment can be defined within a knowledge base.

(1) (LLE) Left logical equivalence: $\frac{\mathcal{K}|\!\approx\alpha\leftrightarrow\beta,\mathcal{K}|\!\approx\alpha|\!\sim\gamma}{\mathcal{K}|\!\approx\beta|\!\sim\gamma.}$
(2) (RW) Right weakening: $\frac{\mathcal{K}|\!\approx\alpha\rightarrow\beta,\mathcal{K}|\!\approx\gamma|\!\sim\alpha}{\mathcal{K}|\!\approx\gamma|\!\sim\beta.}$
(3) (Ref) Reflexivity: $\mathcal{K}|\!\approx \alpha|\!\sim \alpha$
(4) And: $\frac{\mathcal{K}|\!\approx\alpha|\!\sim\beta,\mathcal{K}|\!\approx\alpha|\!\sim\gamma}{\mathcal{K}|\!\approx\alpha|\!\sim\beta\wedge\gamma}$
(5) Or: $\frac{\mathcal{K}|\!\approx\alpha|\!\sim\gamma,\mathcal{K}|\!\approx\beta|\!\sim\gamma}{\mathcal{K}|\!\approx\alpha\vee\beta|\!\sim\gamma}$
(6) (CM) Cautious Monotonicity: $\frac{\mathcal{K}|\!\approx\alpha|\!\sim\gamma,\mathcal{K}|\!\approx\alpha|\!\sim\beta}{\mathcal{K}|\!\approx\alpha\wedge\beta|\!\sim\gamma}$
(7) (RM) Rational Monotonicity: $\frac{\mathcal{K}|\!\approx\alpha|\!\sim\gamma,\mathcal{K}\not|\!\approx\alpha\not|\!\sim\beta}{\mathcal{K}|\!\approx\alpha\wedge\beta|\!\sim\gamma}$

KLM defined ranked entailment with ranked interpretations as follows.

Definition 3.4.1: Given a knowledge base $\mathcal{K}$ and a defeasible implication $\alpha|\!\sim\beta$, $\mathcal{K}|\!\approx_{\mathscr{R}} \alpha|\!\sim \beta$ is read as "$\mathcal{K}$ rank entails $\alpha|\!\sim\beta$," if for every ranked interpretation $\mathscr{R}$ such that $\mathscr{R} \vDash \mathcal{K}$, it holds true that $\mathscr{R} \vDash \alpha |\sim \beta$ [3].

This definition says that for every ranked interpretation that entails the knowledge base, it also needs to entail $\alpha|\sim\beta$ in order to be a logical consequence of the knowledge base.

While this method appears favored for defining entailment, it remains monotonic and fails to address the underlying issue [11]. Minimal ranked entailment emerged as a potential solution to this problem. The concept behind it involves defining a partial order denoted as $\preceq_{\mathcal{K}}$, which ranks all possible ranked interpretations of the knowledge base $\mathcal{K}$.

Definition 3.4.2: Given a knowledge base $\mathcal{K}$ and $\mathscr{R}^{\mathcal{K}}$ representing the set of all ranked interpretations of $\mathcal{K}$, it holds that for every $\mathscr{R}_1, \mathscr{R}_2 \in \mathscr{R}^{\mathcal{K}}$, $\mathscr{R}1 \preceq_{\mathcal{K}} \mathscr{R}_2$ if for every $u \in \mathcal{U}$, $\mathscr{R}_1(u) \leq \mathscr{R}_2(u)$.

What makes this partial order special is the fact that it possesses a unique minimal model, denoted as $\mathcal{R}^{\mathcal{K}}_{RC}$ [14]. From this minimal model, we can define a nonmonotonic entailment relation.

Definition 3.4.3: Given a defeasible knowledge base $\mathcal{K}$, the minimal ranked interpretation satisfying $\mathcal{K}$, denoted as $\mathcal{R}^{\mathcal{K}}_{RC}$, defines an entailment relation, $|\approx_{\text{RC}}$, called minimal ranked entailment. For any defeasible implication $\alpha|\sim\beta$, $\mathcal{K}|\approx \alpha|\sim\beta$ if $\mathcal{R}^{\mathcal{K}}_{RC} \vDash \alpha|\sim\beta$ [11].

The minimal ranked entailment is a way in which we can compute Rational Closure. It will not be explored further in this paper, however, it is worth noting that any form of rational defeasible entailment can be characterized by the minimal ranked entailment [2]. Defining Rational closure through BaseRank will be defined further on.

A very useful result concerning propositional statements is the ability to express them as defeasible implications. $\mathscr{R} \vDash \alpha$ iff $\mathscr{R} \vDash \neg\alpha|\sim\bot$. $\bot$ is a contradiction, referred to as bottom and seen as something always being false such as $(p \wedge \neg p)$. $\top$ is a tautology that is always true. $\mathscr{R} \vDash \neg\alpha|\sim\bot$ can be understood that in all the best worlds $\neg\alpha$ is false. For example, given a propositional statement $penguins \rightarrow \neg fly$, this can be expressed as $\neg(penguins \rightarrow \neg fly)|\sim\bot$ [11].

3.5 Rational Closure

Rational Closure was initially proposed as a potential solution for determining the content of a defeasible knowledge base [3]. The pattern of reasoning within this framework adheres to a prototypical approach, characterized by its inherent conservative nature. In [2], Casini et al. formalized Rational Closure as a conservative reasoning pattern present in certain nonmonotonic frameworks. Lehmann and Magidor proposed that approaches to reasoning that are more adventurous should extend the semantics described in Rational Closure.

There are two principal ways in which Rational Closure can be computed, minimal ranked entailment and ranking of statements from a knowledge base following an algorithmic approach. We will be focusing on the latter. Before we do, there are some definitions that we need to cover.

Definition 3.5.1: The material counterpart of a defeasible implication $\alpha|\sim\beta$ is the propositional formula $\alpha \rightarrow \beta$. Given a defeasible knowledge base $\mathcal{K}$, the $\overrightarrow{\mathcal{K}}$ material counterpart of $\mathcal{K}$, denoted $\overrightarrow{\mathcal{K}}$, is the set of material counterparts, $\alpha \rightarrow \beta$, for every defeasible implication $\alpha|\sim\beta$ [11]. The materialization of $\mathcal{K}$ is $\overrightarrow{\mathcal{K}}$.

Given a formula $\alpha \in \mathcal{L}$, α is said to be exceptional in a knowledge base $\mathcal{K}$ if in every interpretation satisfying $\mathcal{K}$, $\neg\alpha$ is typical in those interpretations. In all the best worlds α would be false, however, that does not mean that there does not exist a world where it is not true.

To consider typically of statements within a knowledge base, we will consider the notion of ranking statements through something which is known as a BaseRank. When encoding a BaseRank based on the statements of a knowledge base, we employ the notion of what is more specific. Statements of higher specificity are on higher ranks, while more generic statements are on lower ranks. The bulk of the work is done on the materialization of $\mathcal{K}$. The input into the BaseRank algorithm is still a knowledge base, not the materialization of $\mathcal{K}$. Materialization will be performed in the algorithm itself.

Algorithm 1 BaseRank

```
1:  Input: A knowledge base $\mathcal{K}$
2:  Output: An ordered tuple $(R_0, ..., R_{n-1}, R_\infty, n)$
3:  $i = 0$;
4:  $E_0 = \overrightarrow{\mathcal{K}}$;
5:  while $E_{i-1} \neq E_i$ do
6:      $E_{i+1} = \{ \alpha \rightarrow \beta | E_i \vDash \neg\alpha \}$ ;
7:      $R_i = E_i \setminus Ei + 1$;
8:      $i = i + 1$
9:  end while
10: $R_\infty = E_{i-1}$;
11: if $E_{i-1} =$ then
12:     $n = i - 1$;
13: else
14:     $n = i$;
15: end if
16: return $(R_0, ..., R_{n-1}, R_\infty, n)$;
```

Algorithm 1 shows how we can compute the BaseRank. Using the algorithm, we will consider an example. Consider the following knowledge base, $\mathcal{K} = \{ mammal |\sim fur, mammal |\sim liveyoung, \neg(platypus \rightarrow mammal)|\sim\bot, platypus|\sim\neg liveyoung\}$. The materialisation of $\mathcal{K}$ will be $\overrightarrow{\mathcal{K}} = \{ mammal \rightarrow fur, mammal \rightarrow liveyoung, platypus \rightarrow mammal, platypus \rightarrow \neg liveyoung\}$. We will ask if the query $platypus |\sim fur$ follows from $\mathcal{K}$. The materialization of this is $platypus \rightarrow fur$. The BaseRank is computed as follows :

- $\overrightarrow{\mathcal{K}} = E_0, E_0 \vDash \neg platypus. E_1 = \{ platypus \rightarrow mammal, platypus \rightarrow \neg liveyoung\}$.
- E_1 has no exceptional anticedents, so $E_2 = \varnothing$.
- In the next iteration, we will see that $E_2 == E_3$ and so the algorithm is complete.

The output of the BaseRank model will be the following tuple; $(\{mammals \rightarrow fur, mammals \rightarrow liveyoung\}, \{ platypus \rightarrow mammal, platypus \rightarrow \neg liveyoung\} \varnothing, 3)$. The rank formula table will look as follows :

Now that we have defined the semantics of what a BaseRank is, we will now look at the algorithm that allows us to query this

$\mathcal{R}_\infty$	$\emptyset$
$\mathcal{R}_1$	$platypus \to mammal, platypus \to \neg liveyoung$
$\mathcal{R}_0$	$mammals \to fur, mammals \to liveyoung$

Table 3: BaseRank of knowledge base

knowledge base. This is the RationalClosure algorithm, it will take in a BaseRank and a query $\alpha|\sim\beta$. The algorithm will then give an output of true or false, which tests whether the knowledge base $\mathcal{K}|\approx\alpha|\sim\beta$.

Algorithm 2 RationalClosure

```
1: Input: A knowledge base $\mathcal{K}$ and a defeasible implication $\alpha|\sim\beta$.
2: Output: true, if the query is entailed by the knowledge base,
   else false.
3: $(R_0,..., R_{n-1},R_\infty,n) = BaseRank(\mathcal{K})$;
4: $i = 0$;
5: $\mathcal{R} = \bigcup_{i=0}^{j<n} \mathcal{R}_j$
6: while $\mathcal{R}_\infty \cup \mathcal{R} \models \neg\alpha$ and $\mathcal{R} \neq \emptyset$ do
7:     $\mathcal{R} = \mathcal{R} \backslash \mathcal{R}_i$;
8:     $i = i + 1$;
9: end while
10: return $\mathcal{R}_\infty \bigcup \mathcal{R} \models \alpha \to \beta$;
```

Using the query $platypus|\sim fur$, we will use the BaseRank in Table 4 to answer this. In the first step, we see that $\mathcal{R} = \bigcup_{i=0}^{j<n} \mathcal{R}_j \models \neg platypus$. Thus, we will enter a second iteration of the algorithm, but this time, the lowest rank, $\mathcal{R}_0$ is not considered. We are left with the following table:

$\mathcal{R}_\infty$	$\emptyset$
$\mathcal{R}_1$	$platypus \to mammal, platypus \to \neg liveyoung$

Table 4: BaseRank of knowledge base without rank 0

We do the same step again and we see that $\mathcal{R} = \bigcup_{i=0}^{j<n} \mathcal{R}_j \not\models \neg platypus$. This rank will now be used to check if the entailment holds. We can see that it is not true that platypus typically have fur because there is no evidence to support this in the knowledge base.

From the example, we can see that as the knowledge base grows, the amount of computational steps required to compute Rational Closure increases drastically. Our project aims to test whether modeling Rational Closure in a declarative language will lead to a more efficient process.

4 ANSWER SET PROGRAMMING

A variant of declarative programming that proves partially effective in addressing NP-hard problems is Answer Set Programming (ASP), which has demonstrated significant utility in knowledge-intensive domains. ASP facilitates knowledge representation through the use of rules, facts, and constraints. The general idea of ASP is taking statements within our knowledge base, and then encoding them as facts, rules, or constraints. What ASP will then do, through the use of its underlying architecture, produce all the interpretations that can be drawn from the knowledge base. The interpretations produced are answer sets. Solvers are utilized in formulating answer sets to a given set of rules. [4]. This section aims to outline the fundamental semantics employed in ASP. Additionally, an exploration into the concept of stable models will be undertaken.

4.1 Introduction

ASP employs semantics similar to those found in Prolog style rules [17]. The rules typically look like this :

$$p :- q.$$
$$q :- not\ s.$$

Where $p\,q$ are statements, $:-$ is an implication and *not* is the negation. The statement $p : - q.$ would be read as "*if q, then p*", meaning if q holds to be true, it must hold that p is true. A collection of rules often has a unique stable model [4]. A stable model can be considered as a set of atoms that make all the rules true. Michael Gelfond and Vladimir Lifschitz first described the semantics of answer set programming in their paper [9]. Their paper outlined how rules expressed in the form below have associated models that enforce that the models are provably true [12].

$$\alpha : -\beta$$

α is the head of the rule and β is the body of the rule. A set of atoms is a model of a rule $head\ : -body$ if either the set of atoms does not make the body true, or at least one of the atoms in the head is true. The head of a rule will always be joined by *or* connectives. The formal structure in which rules are expressed is $p_1 V ... V p_n \leftarrow q_1, ...q_m, notr_1, ..., notr_k$ [4]. Consider the following for example :

$$p.$$
$$r : - p,\ not\ q.$$

p is a fact. This is something that is known to be true. The model of this example will be $\{\ r, p\}$ [4].

Let's extend the program and add another rule.

$$t.$$
$$t : - q.$$

This states that t is true and when t is true, q must follow from t. The new model would be $\{ p, q, t \}$. We can see that r falls away from the model because having q in the model makes r false.

4.2 Methodology of ASP

ASP follows the methodology of generate, define, and test [17].
Definition 4.2.1: The signature of an ASP program is as follows : $< \mathcal{P}, \mathcal{V}, \mathcal{C}, \mathcal{F} >$ where

- $\mathcal{P}$ is a set of predicate symbols.
- $\mathcal{V}$ are the variables.
- $\mathcal{C}$ are the constants.
- $\mathcal{F}$ are the functions.

Morden ASP systems will perform their computation by first grounding a set of rules. The grounder will ensure that all rules in the program has their constants replaced by the variables in the signature. None of the rules change, there are just no constants present. The output of the grounding stage is the input that will be used for the solver [12]. Figure 1 shows the process that an ASP program goes through to derive its answer sets.

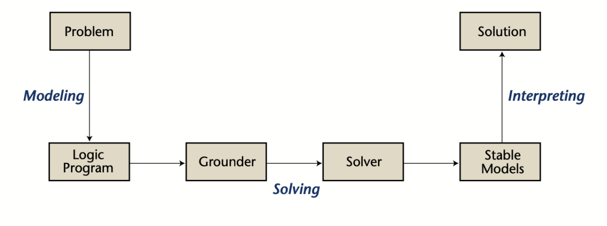

Figure 1: The workflow of Answer set programming. [12]

Newly advanced ASP solvers rely heavily upon advanced conflict-driven search procedures. Conflicts are analyzed and decisions are taken given any conflict scores that may arise. The general outline of ASP mimics the same semantics that are found in SAT, however, ASP has a more elaborate underlying approach to it. The stable model semantics ensure that the atoms are verifiable and able to be proven true. [12].

The system that we will be employing in this project will be Clasp. This is becoming one of the most widely used and state-of-the-art systems. Gringo is the grounder in the system. This grounds all instances of the clauses in the program. Clingo is the pipeline used to flow the input from the grounder to the solver, clasp.

4.2.1 Grounder. Solvers, which we will discuss later on in this review, rely heavily on preprocessing to be effective and efficient. The preprossor that we will be looking at is Gringo. Gringo combines effectively and the techniques from two other grounders used for ASP, *lparse* and *dlv's* [8].
Four processes are involved in the grounding process. First, the parser will ensure that the right syntax is used in the input of the program, it will then produce an internal representation of the input program[8]. The checker will verify that the program is λ-restricted. This ensures that a finite equivalent of the ground instantiation is guaranteed. The instantiator will compute the ground instances of the rules, which will then be processed by the evaluator that identifies newly derived ground instances [8]. Figure 2 illustrates the architecture discussed above.

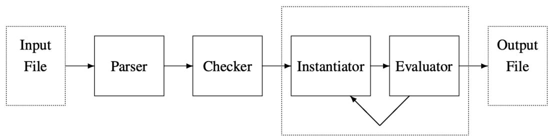

Figure 2: The GrinGo architecture. [8]

4.2.2 Solvers. Solvers in ASP will take the ground instantiations that were produced by the grounder and then will produce answer sets based on the rules and constraints outlined in the program. Clasp is an ASP solver that brings together both the high-level modeling capabilities of ASP and the modern techniques employed in the area of Boolean constraint solving. The main algorithm used in clasp relies on conflict-driven nogood learning [7].

4.3 Stable Model

The stable model semantic is used in ASP to produce answer sets. To bring it back to the definitions we used above, it is the interpretations that make all the stated rules true. Consider the rules of an answer set to be the formulas in a knowledge base. Stable models have been studied for years. The early work on stable models was due to the advancement of work done on formal nonmonotonic reasoning. Stable models were also used to look at the studies of the relationship between auto epistemic logic and the semantics of negation in logic programming [16]. Every program will have a different set of models. The programs that we are studying make use of the following semantics:

$$p \leftarrow q_1, ...q_m$$

where p is the atom and $q_1, ...q_m$ are the literals. The choice atoms to assume are true should make all the rules true[9]. When working with answer sets, we can introduce sets that have no negation and sets that have negation. When a set contains no negation, then the stable model is the minimal set of interpretations that make all the atoms true[16]. When a set of rules has negation then we need to consider the *reduct.*
Definition 4.3.1: The *reduct* of a formula, $\mathcal{F}^X$ relative to a set of atoms, is the formula obtained by replacing each maximal subformula that is not satisfied by X as T (falsity). X is then a stable model of the formula $\mathcal{F}$ is X is minimal among the sets that satisfy $\mathcal{F}$ [6].

5 CONCLUSION

In this review, it was demonstrated that monotonicity is not a barrier that has prevented us from representing reasoning similar to the way that humans do. Many defeasible reasoning frameworks allow us to model reasoning for systems. Rational Closure was shown to be a conservative pattern in reasoning, which should be the basis of frameworks that aim to extend it. With the use of an example, we were able to show how the Rational Closure framework included the notion of typicality and also allowed for nonmonotonic reasoning. When new information was added to the knowledge base, previous inferences were withdrawn and new ones were concluded. Rational Closure has a lot of real-world applications, such as deciding if someone is guilty or not in a court case.

Analysing the algorithm to compute Rational Closure, it is evident that it will grow in complexity exponentially as the knowledge base increases. Answer set programming could be a potential answer to address the time needed for computation. ASP has a multi-threading architecture, which could lead to a faster computational time of Rational Closure [19].

The challenge of the project lies in transforming the Rational Closure algorithm from an imperative form, to a declarative form of programming. ASP is structured differently to the languages that we have used before. However, there are promising possibilities that it could reduce the complexity of computing Rational Closure.

REFERENCES

[1] Mordechai Ben-Ari. 2012. *Mathematical logic for computer science* (3 ed.). Springer, London. 4–47 pages. https://doi.org/10.1007/978-1-4471-4129-7
[2] Meyer T. Casini, G. and I. 2018 Varzinczak. 27-29 October 2018. Defeasible Entailment: from Rational Closure to Lexicographic Closure and Beyond. In *17th International Workshop on Non-Monotonic Reasoning (NMR) 2018.* Arizona, USA, pp. 109–118.

[3] Menachem Magidor Daniel Lehmann. May 1992. What does a conditional knowledge base entail? *Journal of Artificial Intelligence* Vol. 55 no.1 (May 1992), 1–60.

[4] IBM Watson Group Erik T.Mueller and IBM Research. 2015. *Commonsense Reasoning (Second Edition)*. Morgan Kaufmann. 249–269 pages. https://doi.org/10.1016/C2014-0-00192-X

[5] Wolfgang Faber. 2020. An Introduction to Answer Set Programming and Some of Its Extensions. *Reasoning Web. Declarative Artificial Intelligence (pp.149-185)* (2020). 10.1007/978-3-030-60067-9_6

[6] P. Ferraris. 2005. Answer sets for propositional theories.. In *In Proceedings of International Conference on Logic Pro- gramming and Nonmonotonic Reasoning (LPNMR), 119– 131.*

[7] Kaufmann B. Neumann A. Schaub T. Gebser, M. 2007. clasp: A Conflict-Driven Answer Set Solver. *Baral, C., Brewka, G., Schlipf, J. (eds) Logic Programming and Nonmonotonic Reasoning. LPNMR 2007. Lecture Notes in Computer Science(), vol 4483. Springer, Berlin, Heidelberg.* (2007).

[8] Schaub T. Thiele S. Gebser, M. 2007. GrinGo: A New Grounder for Answer Set Programming. *In: Baral, C., Brewka, G., Schlipf, J. (eds) Logic Programming and Nonmonotonic Reasoning. Lecture Notes in Computer Science(), vol 4483. Springer, Berlin, Heidelberg.* (2007).

[9] Lifschitz Vladimir Gelfond, Michael. 1988. The Stable Model Semantics For Logic Programming. *Proceedings of International Logic Pro- gramming Conference and Symposium,1070–1080,MIT Press.* (1988).

[10] Crina Grosan and Ajith Abraham. 2011. Knowledge Representation and Reasoning. *Springer Berlin Heidelberg,Berlin,Heidelberg,131–147.* (2011). https://doi.org/10.1007/978-3-642-21004-4_6

[11] Adam Kaliski. 2020. *An Overview of KLM-Style Defeasible Entailment. Master's thesis.* Ph.D. Dissertation. Faculty of Science, University of Cape Town, Rondebosch, Cape Town, 7700.

[12] Leone N. Perri S. Schaub T. Kaufmann, B. 2016. Grounding and Solving in Answer Set Programming. *AI Magazine, 37(3), 25-32.* (2016).

[13] Sarit Kraus, Daniel Lehmann, and Menachem Magidor. 1990. Nonmonotonic Reasoning, Preferential Models and Cumulative Logics. *Journal of Artificial Intelligence* 44 (1990), 167–207.

[14] N. Olivetti L. Giordano, V. Gliozzi and G.L. Pozzato. 2015. Semantic characterization of rational closure: From propositional logic to description logics. *Artificial Intelligence 226 (2015)* (2015), 1–33. https://doi.org/10.1016/j.artint.2015.05.001

[15] Hector J. Levesque. 1986. Knowledge Representation and Reasoning. *Annual Review of Computer Science1,1(1986),255–287* (1986). https://doi.org/10.1146/annurev.cs.01.060186.001351arXiv:https://doi.org/10.1146/annurev.cs.01.060186.001351

[16] Vladimir Lifschitz. 2008. Twelve Definitions of a Stable Model. *Logic Programming. ICLP 2008. Lecture Notes in Computer Science, vol 5366. Springer, Berlin, Heidelberg* (2008), 37–51.

[17] Vladimir Lifschitz. 2019. *Answer Set Programming*. Springer Cham. https://doi.org/10.1007/978-3-030-24658-7

[18] Markus Stumptner Anna Ryabokon Claire Bagley Katharina Wolter Lothar Hotz, Alexander Felfernig. 2014, Pages 41-72. Chapter 6 - Configuration Knowledge Representation and Reasoning. *Knowledge-Based Configuration From Research to Business Cases* (2014, Pages 41-72). https://doi.org/10.1016/B978-0-12-415817-7.00006-2

[19] Benjamin Kaufmann Martin Gebser, Roland Kaminski and Torsten Schaub. 2012. Answer Set Solving in Practice. *Synthesis Lectures on Artificial Intelligence and Machine Learning* (2012). doi:10.2200/S00457ED1V01Y201211AIM019

[20] Yoav Shoham. 1987. Nonmonotonic Logics: Meaning and Utility. *In Proceedings of the 10th International Joint Conference on Artificial Intelligence - Volume 1 (Milan, Italy) (IJCAI'87). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 388–393.* (1987).

[21] Yoav Shoham. 1987. A Semantical Approach to Nonmonotonic Logics. *MorganKaufmann Publishers Inc., San Francisco, CA, USA, 227–250* (1987).