

# Defeasible DLV Literature Review

Matthew Morris  
matthewthemorris@gmail.com  
University of Cape Town  
Cape Town, South Africa

## ABSTRACT

Knowledge representation and reasoning is a way of achieving artificial intelligence (AI). In the system, a machine has some internal representation of the world (the knowledge base) from which it can draw conclusions by reasoning. This is often a preferred form of AI as the machine can reason in a manner similar to a human. However, humans are often prone to making general statements that do not always hold true, parts of which need to be later retracted. This aspect of human reasoning is not captured in classical reasoning, and so a model of non-monotonic reasoning such as defeasible reasoning is required. We aim to effectively implement defeasible reasoning in an answer set programming language, DLV.

## CCS CONCEPTS

• **Theory of computation** → **Automated reasoning**; • **Computing methodologies** → **Nonmonotonic, default reasoning and belief revision**; *Description logics*;

## KEYWORDS

artificial intelligence, knowledge representation and reasoning, defeasible reasoning, Datalog

## 1 INTRODUCTION

Currently, the two primary means for obtaining AI are machine learning and knowledge representation and reasoning [11]. While machine learning has demonstrated effective results across a variety of disciplines, a primary problem with the method is the inability of the machine to explain how it arrived at a conclusion. Such a property would require the machine to reason in a manner similar to a human. There are two main aspects to an AI that use knowledge representation and reasoning; the knowledge base representing what the machine knows about the world and the reasoning system to draw conclusions from the knowledge. Here, we will be focusing on the reasoning aspect.

The notion of reasoning can be poorly defined, so we will build up to it step by step. First, we will look at propositional logic as the most simplistic form. From there, we will introduce the notion of defeasible reasoning, define some properties we expect it to follow and introduce an algorithm that can be used to reason within the framework. Finally, we will introduce Datalog, a description logic language with limitations that make it computationally efficient.

The aim of the project is to first extend the KLM [12] properties to statements within the Datalog language. Then, we will also extend the rational closure algorithm [7] to Datalog, and prove that it conforms to the extended KLM properties. Finally, we will provide an implementation of this algorithm in DLV, an answer set programming language that is an extension of Datalog.

## 2 PROPOSITIONAL LOGIC

Propositional logic is a basic logical framework which provides a system with which to make basic statements about the world. We will be using the formalization provided by Ben-Ari [2] to introduce the system.

### 2.1 Motivation

Reasoning is something that humans do on a day to day basis, and the ability to formalize that reasoning is important. By providing a system with which to reason, we can find mistakes in our reasoning and construct algorithms that can compute conclusions. Propositional logic builds up *formulas* by combining *atomic propositions* with *Boolean operators*. An atomic proposition is simply a statement about the world that can be assigned a value of *true* or *false*, for example; 'it is raining outside'. There are 16 distinct Boolean operators. However, we will provide definitions for only 5 of them, as this is sufficient to model any statement without introducing too much notation.

An example of a Boolean operator is 'and'. Using that operator and the atomic propositions 'it is raining', 'I am indoors' and 'it is cold', we can construct a formula which states 'it is raining and I am indoors and it is cold'. If we know this formula to be true (i.e. what it asserts is correct), then we can come to the conclusion that 'it is raining'. This is a trivial example, but it illustrates how the system could be used to reason.

### 2.2 Formalization

**2.2.1 Constructing Formulas.** We have a base finite set  $\mathcal{P}$  of *atomic propositions*, such that  $\mathcal{P} = \{p, q, \dots\}$ , where each element of  $\mathcal{P}$  can be assigned a true or false value. We can then introduce *Boolean operators* to combine atoms together into more complex formulas.

Name	Meaning	Symbol
negation	not	$\neg$
conjunction	and	$\wedge$
disjunction	or	$\vee$
implication	if then	$\rightarrow$
equivalence	if and only if	$\leftrightarrow$

Figure 1: The Boolean operators used in propositional logic

The negation operator takes in a single operand and the other operators take 2 operands. The order of operator precedence from high to low goes from top to bottom in the table.

We can then recursively define the set of all propositional formulas  $\mathcal{L}$  from our atoms and operators; for every  $p \in \mathcal{P}, p \in \mathcal{L}$ , and if  $\alpha, \beta \in \mathcal{L}$ , then  $\neg\alpha, \alpha \wedge \beta, \alpha \vee \beta, \alpha \rightarrow \beta, \alpha \leftrightarrow \beta \in \mathcal{L}$ .

2.2.2 *Assigning Meaning to Formulas.* A formula  $\alpha \in \mathcal{L}$  also has a true or false value. If  $\alpha$  is simply a propositional atom, we can assign it the true/false value that the atom has. However, we need a way to define whether a formula is true or false, given the true/false values of the operands. One easy way to do this is with truth tables. Here follows the truth table for the above listed operators, where an entry provides the truth value of the formula in the column, given the truth values of the operands  $p$  and  $q$ .

$p$	$q$	$\neg p$	$p \wedge q$	$p \vee q$	$p \rightarrow q$	$p \leftrightarrow q$
T	T	F	T	T	T	T
T	F	-	F	T	F	F
F	T	T	F	T	T	F
F	F	-	F	F	T	T

Figure 2: A truth table for the Boolean operators  $\neg$ ,  $\wedge$ ,  $\vee$ ,  $\rightarrow$  and  $\leftrightarrow$

The other Boolean operators can be defined similarly. Using our definition for elements of  $\mathcal{L}$ , we can recursively evaluate the true/false value of any formula. We do these evaluations within the context of an *interpretation*. A *valuation* or *interpretation* is an assignment of true/false values to every atom in  $\mathcal{P}$ .

Formally defined, an *interpretation* for  $\mathcal{P}$  is a function  $I : \mathcal{P} \mapsto \{T, F\}$ . For any  $\alpha \in \mathcal{L}$ , the truth value of  $\alpha$  under  $I$  is notated as  $I(\alpha)$ . We can express an interpretation in terms of a sequence of atoms and barred atoms (e.g.  $p, q, \bar{r}$ ), where the atom represents the statement being true in the valuation and the barred atom represents the statement being false in the valuation.

An interpretation  $I$  is said to *satisfy* a formula  $\alpha$  if, when  $\alpha$  is evaluated using the true/false values for the atoms provided by  $I$ , the formula evaluates to true (i.e.  $I(\alpha) = T$ ). This is denoted by  $I \models \alpha$ . We then call  $I$  a *model* of  $\alpha$  and define  $\text{Mod}(\alpha)$  to be the set of all models of  $\alpha$ .

2.2.3 *Defining Entailment.* We now define the notion of entailment. It can be thought of as starting with some statement  $\alpha$  we know to be true and coming to the conclusion that  $\beta$  is also true. Formally, we say that a formula  $\alpha$  *entails*  $\beta$ , notated  $\alpha \models \beta$ , if and only if  $\text{Mod}(\alpha) \subseteq \text{Mod}(\beta)$ .

However, this is currently not a very useful definition, as we would like to be able to take multiple pieces of knowledge into account when drawing conclusions. Thus, we introduce the notion of a *knowledge base*, which represents every formula the system knows to be true. Formally, if  $\mathcal{K}$  is a finite set of formulas, then  $\mathcal{K}$  is a *knowledge base*. For example,  $\mathcal{K} = \{p, p \rightarrow q, q \vee \neg r\}$  is a knowledge base. We can now define entailment in terms of a knowledge base.

We first define  $\text{Mod}(\mathcal{K}) \equiv_{\text{def}} \bigcap \{\text{Mod}(\alpha) : \alpha \in \mathcal{K}\}$ . Thus  $\text{Mod}(\mathcal{K})$  is the set of all interpretations where every formula in  $\mathcal{K}$  is satisfied. Now we can say  $\mathcal{K}$  entails  $\alpha$ , notated  $\mathcal{K} \models \alpha$  if and only if  $\text{Mod}(\mathcal{K}) \subseteq \text{Mod}(\alpha)$ .

## 2.3 Semantics

This can all be a lot to take in, but the meaning behind all the notation is really quite simplistic. In the real world, a knowledge

base would represent what an AI knows to be true about the world. Let us take a simplistic example, where we know the following statements to be true:

- (1) This is a bird and this is an animal
- (2) Birds fly

We can identify 3 base statements here; '*this is a bird*', '*this is an animal*' and '*this flies*'. If we let  $b$ ,  $a$  and  $f$  represent those statements respectively, we now have 3 propositional atoms to work with. We can then use the notation of propositional logic to turn that sequence of statements into:

- (1)  $b \wedge a$
- (2)  $b \rightarrow f$

The above is a representation of a knowledge base. Now since  $b \wedge a$  is true,  $b$  is also true which implies that  $f$  is also true. In other words, from our initial sequence of statements, we have concluded that '*this flies*'.

Formally, we have that  $\mathcal{K} = \{b \wedge a, b \rightarrow f\} \models f$ . We can formally prove this by showing that  $\text{Mod}(\mathcal{K}) \subseteq \text{Mod}(f)$ . Intuitively, we would need to show that every world in which all statements in  $\mathcal{K}$  hold true, so too does  $f$  hold true.

## 3 DEFEASIBLE REASONING

### 3.1 Motivation

As discussed by Casini et al[5], reasoning without certainty is a major topic in AI. A *monotonic* system is one in which all information is certain. In such a system, adding new information to a knowledge base will mean you can still draw the same conclusion that you had before. For example, if it is known that '*birds fly*' and '*things that fly have wings*', we can conclude that '*birds have wings*'. If we add the extra knowledge of '*wings have feathers*', we can still conclude that '*birds have wings*'.

The opposite of this is a *non-monotonic system*, where statements of the form '*typically, something is the case*' are permitted in addition to the conventional statements. This is necessary as a monotonic system fails to capture a 'common sense' approach to reasoning [5]. Consider this example, where the following statements are made:

- (1) Birds fly
- (2) Robins are birds
- (3) Penguins are birds

From this, we can conclude that '*penguins fly*', which is completely nonsensical. However, each of the above statements is perfectly reasonable from a human perspective. What we actually meant was '*typically, birds fly*'. Then, when we add the extra information that penguins in fact do not fly, the system should be able to retract its conclusion that '*penguins fly*'.

There are many different formalizations of non-monotonic reasoning. One of the earliest approaches by Reiter [18] is known as *default reasoning*. However, as discussed by Casini and Straccia [20], *rational closure* within the *KLM approach* is currently one of the most preferred methods.

### 3.2 Expanded Notation

We now wish to be able to make general statements that may not always hold true. The previous syntax for expressing '*birds fly*' was  $b \rightarrow f$ . So we replace  $\rightarrow$  with  $\vdash$ , giving us the statement  $b \vdash f$  to

represent 'typically, birds fly'. Formally, to extend the language  $\mathcal{L}$ , for all  $\alpha, \beta \in \mathcal{L}$ , we add  $\alpha \sim \beta$ . Note; this means that no nesting of this defeasible implication operator is permitted.

To start defining entailment, we first need to introduce the notion of *ranked interpretations* [6]. Informally, a ranked interpretation is simply a ranking where every possible interpretation is assigned a level of 'normality'. Multiple interpretations can be considered to be 'equally normal' and would then have the same ranking in the ranked interpretation. Interpretations further down the ranking are considered 'more normal' whereas ones further up are 'less normal'. The top level of the ranking represents all 'impossible worlds'.

Formally, a ranked interpretation  $R$  is a function from  $U$  (the set of all interpretations) to  $\mathbb{N} \cup \{\infty\}$  such that

- (1)  $R(u) = 0$  for some  $u \in U$
- (2) For all  $i \in \mathbb{N}$ , if  $R(v) = i$  then for all  $j$  s.t.  $0 \leq j < i$ , there exists  $u \in U$  s.t.  $R(u) = j$ .

We then say that  $R$  satisfies  $\alpha$ , denoted  $R \models \alpha$  if  $\alpha$  is true in all non-impossible levels of  $R$ . We can then say that  $R \models \alpha \sim \beta$  if in the most 'normal' level where  $\alpha$  holds,  $\beta$  also holds. This satisfaction is analogous to satisfaction by an interpretation within classical logic.

Finally, we can define a partial order  $\leq_{\mathcal{K}}$  on all ranked models of a knowledge base  $\mathcal{K}$  as follows.  $R_1 \leq_{\mathcal{K}} R_2$  if for every  $v \in U$ ,  $R_1(v) \leq R_2(v)$ . Intuitively, ranked models lower down in the ordering are more normal. Giordano et al[10] showed that there is a unique minimal element with respect to  $\leq_{\mathcal{K}}$ .

As an example, figure 3 gives a ranked interpretation for  $\mathcal{P} = \{b, f, p\}$  satisfying  $\mathcal{K} = \{p \rightarrow b, b \sim f, p \sim \neg f\}$  [6]. For easier reading, we omit the valuations with rank  $\infty$  in our graphical representations of ranked interpretations.

2	pbf
1	$\bar{p}b\bar{f}$ $p\bar{b}\bar{f}$
0	$\bar{p}\bar{b}\bar{f}$ $\bar{p}b\bar{f}$ $p\bar{b}\bar{f}$

Figure 3: A ranked interpretation for  $\mathcal{P} = \{b, f, p\}$ .

### 3.3 KLM Properties

Unlike classical entailment, defeasible entailment is not unique. There exist multiple formalizations and processes for performing defining defeasible entailment. For example, Lehmann and Magidor [15] suggested *rational closure* for defining defeasible entailment, whereas Lehmann [14] proposed *lexicographic closure* as another definition.

Kraus, Lehmann and Magidor [12] introduced what is commonly known as the *KLM Properties* for defeasible entailment. In their paper, they identified several properties that one would expect any correct form of defeasible entailment to follow. Here, we will consider the extended KLM properties provided by Lehmann and Magidor [15].

$$\begin{array}{ll}
(\text{Ref}) \mathcal{K} \models \alpha \sim \alpha & (\text{LLE}) \frac{\alpha \equiv \beta, \mathcal{K} \models \alpha \sim \gamma}{\mathcal{K} \models \beta \sim \gamma} \\
(\text{RW}) \frac{\mathcal{K} \models \alpha \sim \beta, \beta \models \gamma}{\mathcal{K} \models \alpha \sim \gamma} & (\text{And}) \frac{\mathcal{K} \models \alpha \sim \beta, \mathcal{K} \models \alpha \sim \gamma}{\mathcal{K} \models \alpha \sim \beta \wedge \gamma} \\
(\text{Or}) \frac{\mathcal{K} \models \alpha \sim \gamma, \mathcal{K} \models \beta \sim \gamma}{\mathcal{K} \models \alpha \vee \beta \sim \gamma} & (\text{CM}) \frac{\mathcal{K} \models \alpha \sim \beta, \mathcal{K} \models \alpha \sim \gamma}{\mathcal{K} \models \alpha \wedge \beta \sim \gamma} \\
(\text{RM}) \frac{\mathcal{K} \models \alpha \sim \gamma, \mathcal{K} \not\models \alpha \sim \neg\beta}{\mathcal{K} \models \alpha \wedge \beta \sim \gamma} &
\end{array}$$

### 3.4 Rational Closure

The versions of defeasible entailment *rational closure* [15] and *lexicographic closure* [14] both satisfy the above listed KLM properties. We will be focusing on the rational closure definition and provide an overview of it here. Full details can be seen in the paper by Casini et al[6].

**3.4.1 Entailment in Terms of Ranked Interpretations.** The first way to check defeasible entailment is as follows. We know from Giordano et al[10] that there is a unique minimal ranked interpretation with respect to  $\leq_{\mathcal{K}}$ , call it  $R_{\mathcal{K}}$ . We say that  $\mathcal{K}$  defeasibly entails  $\alpha \sim \beta$ , notated  $\mathcal{K} \models \alpha \sim \beta$ , if  $R_{\mathcal{K}} \models \alpha \sim \beta$ .

**3.4.2 Entailment in Terms of the Base Rank Algorithm.** While the above definition is neat, it says nothing about how to go about finding the unique minimal ranked interpretation  $R_{\mathcal{K}}$ . An alternative definition, which we will refer to as the *Base Rank Algorithm*, provides another means for checking defeasible entailment that is equivalent to the above rational closure definition.

First, we construct a ranking  $\mathcal{K}_R$  of all the statements in  $\mathcal{K}$ . This is done as follows:

- Define  $C$  to be all classical statements in  $\mathcal{K}$ , and  $D$  to be all defeasible statements in  $\mathcal{K}$ . So we aim to rank  $C \cup D$
- Define a sequence of knowledge bases  $\Sigma_0, \Sigma_1, \dots, \Sigma_n$  as follows
- $\Sigma_0 = D$
- $\Sigma_1 = \{\alpha \sim \beta \in \Sigma_0 : C \cup \bar{\Sigma}_0 \models \neg\alpha\}$ . Intuitively, we keep all the defeasible statements such that the left hand side can be 'disproven' by our classical statements and all defeasible statements not in the previous iteration
- $\Sigma_2 = \{\alpha \sim \beta \in \Sigma_1 : C \cup \bar{\Sigma}_1 \models \neg\alpha\}$
- and so on...
- We stop when  $\Sigma_{i+1} = \Sigma_i$  or  $\Sigma_{i+1} = \emptyset$

We end up with a sequence such that  $\Sigma_n \subset \Sigma_{n-1} \subset \dots \subset \Sigma_0$ . We then assign  $\Sigma_{n-1} - \Sigma_n$  to the bottom rank,  $\Sigma_{n-2} - \Sigma_{n-1}$  to second from the bottom, and so on. We end up with a ranking of  $\mathcal{K}$  where the classical statements appear on the bottom; the 'infinite' level. The higher up in the ranking you go, the more 'general' the statements become. Note that some statements that appear defeasible are in fact classical, and appear on the bottom rank. This is due to the fact that  $R \models \neg\alpha \sim \perp$  if and only if  $R \models \alpha$ .

This ranking algorithm is easier shown by example. Consider the knowledge base  $\mathcal{K} = \{b \sim f, r \rightarrow b, p \rightarrow b, p \sim \neg f, b \sim w\}$ . Then  $C = \{r \rightarrow b, p \rightarrow b\}$ .

- $\Sigma_0 = \{b \vdash f, b \vdash w, p \vdash \neg f\}$
- $\Sigma_1 = \{p \vdash \neg f\}$
- $\Sigma_2 = \emptyset$

So we have a ranking of:

0	$b \vdash f, b \vdash w$
1	$p \vdash \neg f$
$\infty$	$r \rightarrow b, p \rightarrow b$

Now, to check entailment, perform the following steps:

- (1) Turn all defeasible statements into classical statements: define  $\vec{\mathcal{K}} = \{\alpha \rightarrow \beta : \alpha \vdash \beta \in D\} \cup \{\alpha \in \mathcal{K} : \alpha \in C\}$
- (2) If we are checking if  $\mathcal{K} \models \alpha$ , simply check if it is entailed by the classical statements (all the statements in the infinite rank)
- (3) If we are checking if  $\mathcal{K} \models \alpha \vdash \beta$ , first check if  $\vec{\mathcal{K}} \models \neg \alpha$ 
  - (a) If not, then return  $\vec{\mathcal{K}} \models \alpha \rightarrow \beta$
  - (b) If so, then we have a conflict. Throw away the highest level of the ranking and check again if  $\vec{\mathcal{K}} \models \neg \alpha$
- (4) If we end up with only the bottom level remaining, and it is still the case that  $\vec{\mathcal{K}} \models \neg \alpha$ , then we confirm  $\mathcal{K} \models \alpha \vdash \beta$  as true

Freund [9] showed that this algorithm returns true for  $\mathcal{K} \models \alpha \vdash \beta$  if and only if  $\mathcal{K} \models \alpha \vdash \beta$  by the ranked interpretations definition. Note also that the entire algorithm reduces down to a sequence of classical entailment checks from  $\vec{\mathcal{K}}$ .

## 4 DATALOG

Datalog is designed to be a database query language and is in many aspects a simplified version of logic programming [8]. It can be considered a description logic as well [13], albeit one which is less expressive than a standard description logic.

### 4.1 Motivation

Propositional logic is inherently less expressive than most description logics. As soon as one begins to make distinctions between individuals and classes of individuals, it becomes too difficult to usefully express the world with propositional logic any more.

In terms of its expressive power, Datalog has been shown to be strictly more expressive than positive relational algebra [8], due to its ability to express recursive queries. In terms of actual real-world applications, both Shkapsky et al[19] and Pasarella and Lobo [17] found success applying Datalog to solve relevant problems.

### 4.2 Formalization

A Datalog program consists of a finite set of *facts* and *rules*, where facts provide information about the world and rules allow us to deduce facts from other facts [8]. For example; 'Alice is Bob's sister' is a fact and 'If X is the parent of Y and X is the parent of Z, then Y and Z are siblings' is a rule. In Datalog, both facts and rules are expressed as *Horn clauses*.

A Datalog clause has the general form:

$$L_0 : -L_1, \dots, L_n$$

where each  $L_i$  has the form  $p_i(t_1, \dots, t_{k_i})$ . In this case,  $p_i$  is referred to as a *predicate symbol* and  $t_1, \dots, t_{k_i}$  the *terms*. A term is either a

*constant* or a *variable*. The right-hand side of the clause is referred to as the *body* and the left hand side as the *head*. Intuitively, we are saying 'if everything in the body holds true, the so too does the head hold true'.

If the body is empty, the clause represents a fact, otherwise it represents a rule. For example, 'Alice is Bob's sister' can be expressed as *sister(Alice, Bob)*. The rule, 'If X is the parent of Y and X is the parent of Z, then Y and Z are siblings' can be represented as

$$\text{siblings}(Y, Z) : \neg \text{parent}(X, Y), \text{parent}(X, Z)$$

This is analogous to the propositional logic formula of

$$pxy \wedge pxz \rightarrow syz$$

where  $pxy$  represents 'X is the parent of Y' and so on. From this, it is clear that Datalog is more expressive than propositional logic in at least some aspects, although it only has access to the  $\wedge$  and  $\rightarrow$  operators.

### 4.3 Limitations

The limitations within Datalog come from the limited pool of operators and strict required structure of the rules. Most importantly, the omission of the negation ( $\neg$ ) and disjunction ( $\vee$ ) operators lead to statements that are impossible to make, for example;  $p(X) \vee q(X) \rightarrow r(X)$ . We can extend Datalog to *Disjunctive Datalog*, which can then be further extended to *DLV*, to somewhat circumvent these limitations.

### 4.4 DLV

Leone et al[16] provide a full definition for language and kernel of DLV. Here, we will give a brief overview of the language.

We define a *disjunctive rule*, which is a formula of the form

$$a_1 \vee \dots \vee a_n : -b_1, \dots, b_k, -b_{k+1}, \dots, -b_m$$

where each  $a_i, b_i$  is a predicate as before. Semantically, this can be interpreted in the same way as classic Datalog. However, disjunction is now permitted in the head and negation is permitted in the body. For example;

$$\text{apple}(X) \vee \text{orange}(X) : -\text{fruit}(X), \text{round}(X), -\text{lemon}(X)$$

could represent the sentence 'Every round fruit that is not a lemon is either an apple or an orange'. We also define a *weak constraint*, which is a formula of the form

$$:\sim b_1, \dots, b_k, -b_{k+1}, \dots, -b_m [w : l]$$

where each  $b_i$  is a predicate.

The notion of a weak constraint is that the assertions on the right-hand side should be satisfied if possible, but violating them does not invalidate the models. This is somewhat analogous to defeasible implication. In the constraint,  $w$  and  $l$  are integer constants, and are referred to as the *weight* and *layer* respectively. Constraints with higher weights are considered more important, with the best models minimizing the sum of the weights of the violated weak constraints.

A *DLV Program* is simply a finite set of disjunctive rules and weak constraints. Given this set of rules and constraints, it will output the set of models that are consistent with the provided constraints. DLV is one of many answer set programming (ASP) systems [1]. However, it is one of the first reliable ASP systems, which has led to it being employed in many academic and industrial applications.

It has recently been updated with modern evaluation techniques and development platforms. This makes it very suitable for the incorporation of defeasible reasoning.

## 5 INCORPORATION OF DEFEASIBLE REASONING

The main aim of the project is to incorporate defeasible reasoning into DLV. While the inclusion of weak constraints adds a non-monotonic element to DLV, it appears that there currently does not exist an implementation of defeasible reasoning within DLV.

However, Britz et al[3] provided a means of computing rational closure within a description logic environment. They showed that computing defeasible entailment can be reduced to classical entailment checking, and performed tests that demonstrated the computational efficiency of the checking. The algorithm provided is comparable in nature to the base rank algorithm for propositional logic.

In their paper, Casini et al[4] highlighted some key failings of rational closure for defeasible reasoning within description logics. The example to demonstrate these failings is as follows.

Suppose we know that mammalian and avian red blood cells are vertebrate red blood cells, denoted  $MRBC \sqsubseteq VRBC$  and  $ARBC \sqsubseteq VRBC$ . We also know that vertebrate red blood cells normally have a cell membrane ( $VRBC \sqsubset \exists hasCM.T$ ), that vertebrate red blood cells normally have a nucleus ( $VRBC \sqsubset \exists hasN.T$ ), but that mammalian red blood cells normally do not ( $MRBC \sqsubset \neg \exists hasN.T$ ). Rational closure allows us to conclude that avian vertebrate red blood cells normally have a cell membrane ( $ARBC \sqsubset \exists hasCM.T$ ), but not so for mammalian red blood cells ( $MRBC \sqsubset \exists hasCM.T$ ).

They proposed two new forms of defeasible reasoning to solve this problem; *Basic Relevant Closure* and *Minimal Relevant Closure*. They converted the KLM properties be in terms of description logics, and then showed that neither of the new algorithms conform to all of the converted KLM properties for defeasible reasoning. However, due to the limited expressivity of DLV, this specific counter-example does not show why rational closure would fail in the DLV case.

## 6 CONCLUSIONS

Propositional logic, while neat and easy to understand, is not sufficient to model all natural human reasoning. The ability to reason without certainty is a desired property of some systems and a major topic in AI [5]. Such reasoning is referred to as *non-monotonic reasoning* and the chosen form thereof is called *defeasible reasoning*. However, defeasible entailment is not unique, with multiple definitions for computing entailment existing.

Kraus, Lehmann and Magidor [12] introduced the *KLM properties*, which specify rules for how one would expect any correct form of defeasible entailment to behave. Lehmann and Magidor [15] provided the *rational closure* definition, which has been shown to satisfy the the KLM properties. Furthermore, it was shown that *base rank* algorithm fully computes rational closure.

We highlighted the failings of the expressivity of propositional logic, and provided Datalog as an alternative. Datalog is a database query language and can also be considered as a description logic [13]. However, due to the limited pool of operators and strict rule structure, Datalog also has limitations.

We provided DLV as an extension to Datalog, and concluded that it is very suitable for the incorporation of defeasible reasoning. The work of Britz et al[3] and Casini et al[4] show the feasibility of incorporating defeasible reasoning into description logics. Much like with propositional logic, the computation for defeasible entailment can be reduced to a sequence of classical entailment checks. In many aspects, DLV can be considered a weaker form of a description logic. Thus, the techniques used by Casini et al[4], particularly with respect to the adaption of the KLM properties, can be applied to defeasible DLV.

## REFERENCES

- [1] Weronika T. Adrian, Mario Alviano, Francesco Calimeri, Bernardo Cuteri, Carmine Dodaro, Wolfgang Faber, Davide Fuscà, Nicola Leone, Marco Manna, Simona Perri, Francesco Ricca, Pierfrancesco Veltri, and Jessica Zangari. 2018. The ASP System DLV: Advancements and Applications. *KI - Künstliche Intelligenz* 32, 2-3 (aug 2018), 177–179. <https://doi.org/10.1007/s13218-018-0533-0>
- [2] Mordechai Ben-Ari. 2012. *Mathematical Logic for Computer Science* (3 ed.). Springer Science & Business Media, Rehovot, Israel. <https://books.google.co.za/books?hl=en>
- [3] Katarina Britz, Giovanni Casini, Thomas Meyer, Kody Moodley, Uli Sattler, and Ivan Varzinczak. 2017. *Rational Defeasible Reasoning for Description Logics*. Technical Report. University of Cape Town, South Africa. <https://core.ac.uk/download/pdf/151756088.pdf>
- [4] Giovanni Casini, Thomas Meyer, Kodylan Moodley, and Riku Nortjé. 2014. Relevant Closure: A New Form of Defeasible Reasoning for Description Logics. In *JELIA 2014: Logics in Artificial Intelligence*. Springer, Cham, Funchal, Madeira, Portugal, 92–106. [https://doi.org/10.1007/978-3-319-11558-0\\_7](https://doi.org/10.1007/978-3-319-11558-0_7)
- [5] G Casini, T Meyer, K Moodley, and I Varzinczak. 2013. *Towards practical defeasible reasoning for description logics*. Technical Report. Centre for Artificial Intelligence Research. <http://researchspace.csir.co.za/dspace/handle/10204/7039>
- [6] Giovanni Casini, Thomas Meyer, and Ivan Varzinczak. 2019. Taking Defeasible Entailment beyond Rational Closure. (2019), 18 pages.
- [7] Giovanni Casini and Umberto Straccia. 2010. Rational Closure for Defeasible Description Logics. *Lecture Notes in Computer Science* 6341 (2010), 77–90. [https://doi.org/10.1007/978-3-642-15675-5\\_9](https://doi.org/10.1007/978-3-642-15675-5_9)
- [8] S. Ceri, G. Gottlob, and L. Tanca. 1989. What you always wanted to know about Datalog (and never dared to ask). *IEEE Transactions on Knowledge and Data Engineering* 1, 1 (mar 1989), 146–166. <https://doi.org/10.1109/69.43410>
- [9] Michael Freund. 1998. Preferential reasoning in the perspective of Poole default logic. *Artificial Intelligence* 98, 1-2 (jan 1998), 209–235. [https://doi.org/10.1016/S0004-3702\(97\)00053-2](https://doi.org/10.1016/S0004-3702(97)00053-2)
- [10] L. Giordano, V. Gliozzi, N. Olivetti, and G.L. Pozzato. 2015. Semantic characterization of rational closure: From propositional logic to description logics. *Artificial Intelligence* 226 (sep 2015), 1–33. <https://doi.org/10.1016/J.ARTINT.2015.05.001>
- [11] David Gunning. 2017. *Explainable Artificial Intelligence (XAI)*. Technical Report. DARPA. <http://listverse.com/>
- [12] S Kraus, D Lehmann, and M Magidor. 1990. Nonmonotonic Reasoning, Preferential Method and Cumulative Logics. *Artificial Intelligence* 44 (1990), 167–207. [arXiv:arXiv:cs/0202021v1](https://arxiv.org/abs/cs/0202021v1)
- [13] Markus Krötzsch, Sebastian Rudolph, and Peter H. Schmitt. 2015. A closer look at the semantic relationship between Datalog and description logics. *Semantic Web* 6, 1 (jan 2015), 63–79. <https://doi.org/10.3233/SW-130126>
- [14] Daniel Lehmann. 1995. Another perspective on default reasoning. *Annals of Mathematics and Artificial Intelligence* 15, 1 (1995), 61–82. <https://doi.org/10.1007/BF01535841>
- [15] D Lehmann and M Magidor. 1994. What does a conditional knowledge base entail? *Artificial Intelligence* 55, 1 (1994), 1–60.
- [16] Nicola Leone, Gerald Pfeifer, Wolfgang Faber, Thomas Eiter, Georg Gottlob, Simona Perri, and Francesco Scarcello. 2006. The DLV system for knowledge representation and reasoning. *ACM Transactions on Computational Logic* 7, 3 (jul 2006), 499–562. <https://doi.org/10.1145/1149114.1149117>
- [17] Edelmira Pasarella and Jorge Lobo. 2017. A Datalog Framework for Modeling Relationship-based Access Control Policies. In *Proceedings of the 22nd ACM on Symposium on Access Control Models and Technologies - SACMAT '17 Abstracts*. ACM Press, New York, New York, USA, 91–102. <https://doi.org/10.1145/3078861.3078871>
- [18] R. Reiter. 1980. A logic for default reasoning. *Artificial Intelligence* 13, 1-2 (apr 1980), 81–132. [https://doi.org/10.1016/0004-3702\(80\)90014-4](https://doi.org/10.1016/0004-3702(80)90014-4)
- [19] Alexander Shkapsky, Mohan Yang, Matteo Interlandi, Hsuan Chiu, Tyson Condie, and Carlo Zaniolo. 2016. Big Data Analytics with Datalog Queries on Spark. In *Proceedings of the 2016 International Conference on Management of Data - SIGMOD '16*. ACM Press, New York, New York, USA, 1135–1149. <https://doi.org/10.1145/>

2882903.2915229

- [20] Umberto Straccia and Giovanni Casini. 2013. Defeasible Inheritance-Based Description Logics. *Journal of Artificial Intelligence Research* 48 (jun 2013), 415–473. <https://www.aaai.org/ocs/index.php/IJCAI/IJCAI11/paper/viewPaper/2924>