



UNIVERSITY OF CAPE TOWN

DEPARTMENT OF COMPUTER SCIENCE



CS/IT Honours Final Paper 2019

Title: Defeasible Disjunctive Datalog

Author: Tala Ross

Project Abbreviation: DDLV

Supervisor(s): Tommie Meyer

Category	Min	Max	Chosen
Requirement Analysis and Design	0	20	0
Theoretical Analysis	0	25	25
Experiment Design and Execution	0	20	0
System Development and Implementation	0	20	0
Results, Findings and Conclusion	10	20	20
Aim Formulation and Background Work	10	15	15
Quality of Paper Writing and Presentation	10		10
Quality of Deliverables	10		10
Overall General Project Evaluation (<i>this section allowed only with motivation letter from supervisor</i>)	0	10	
Total marks	80		

Defeasible Disjunctive Datalog

Tala Ross

rsstal002@myuct.ac.za

University of Cape Town

Cape Town, South Africa

ABSTRACT

Datalog is a declarative logic programming language that uses classical logical reasoning as its basic form of reasoning. Defeasible reasoning is a form of non-classical reasoning that is able to deal with exceptions to general assertions in a formal manner. The KLM approach to defeasible reasoning is an axiomatic approach based on the concept of plausible inference. Since Datalog uses classical reasoning, it is currently not able to handle defeasible implications and exceptions. We aim to extend the expressivity of Datalog by incorporating KLM-style defeasible reasoning into classical Datalog. We present a systematic approach to extending the KLM properties and two well-known forms of rational defeasible entailment, Rational Closure and Lexicographic Closure, to Datalog.

CCS CONCEPTS

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

KEYWORDS

knowledge representation and reasoning, defeasible reasoning, KLM approach, Rational Closure, Lexicographic Closure, Datalog

1 INTRODUCTION

The Knowledge Representation and Reasoning approach to Artificial Intelligence uses logics to represent knowledge and automated reasoning methods to draw new conclusions from that knowledge. Classical reasoning systems are monotonic. This means that all information is certain and adding new information does not change the conclusions that you could draw before. This form of reasoning can be too weak to model certain systems. To illustrate this, consider an example where these statements are made:

Example 1.1.

- (1) Students do not pay taxes.
- (2) First years are students.
- (3) Tutors are students.

From this, we can conclude that “*tutors do not pay taxes*”, which may in fact be incorrect. However, each of these statements is perfectly reasonable from a human perspective. What we actually meant was “*typically, students do not pay taxes*”. Then, when we add the extra information that tutors typically pay taxes, we want the system to retract its conclusion that “*tutors do not pay taxes*”. However, a monotonic, classical reasoning system cannot change previous assumptions, and knowing that “*tutors pay taxes*” and “*tutors do not pay taxes*”, it must then conclude that no tutors can exist, otherwise we would get a contradiction. In non-monotonic systems, defeasible statements of the form “*typically, something is the case*” are permitted. This allows for a more “common sense” approach to reasoning than in the approach of classical reasoning[5].

In this paper, we discuss the KLM approach[12], a well supported approach to non-monotonic reasoning. In Sections 2 and 6 we discuss the existing algorithmic definition of this approach for propositional logic. In Section 3 we discuss Datalog, a more expressive logic. The central focus of this paper is to extend the KLM approach to the Datalog case. We discuss this extension in Sections 4, 5 and 7. The work in Sections 1 - 5 was done jointly with Morris.

2 BACKGROUND

2.1 Propositional Logic

Propositional logic [2] is a simple logic which uses classical reasoning. A propositional *atom*, denoted by p or q , is a statement that can be assigned a *truth value* (true or false), which cannot be decomposed into smaller such statements. We build up a language \mathcal{L} of propositional logic, by recursively combining statements using Boolean operators, for example: $\alpha \rightarrow \beta$ (implies), $\alpha \wedge \beta$ (and), $\alpha \vee \beta$ (or), and $\neg\alpha$ (not).

We can represent the statements from Example 1.1 in propositional logic, using atoms s , x , f and t to represent students, taxes, first years and tutors respectively:

- (1) $s \rightarrow \neg x$
- (2) $f \rightarrow s$
- (3) $t \rightarrow s$

Intuitively, $\alpha \rightarrow \beta$ means that if α is true then β is true, $\alpha \wedge \beta$ means that both α and β are true, $\alpha \vee \beta$ means that at least one of α or β are true, and $\neg\alpha$ means that α is not true.

Reasoning in propositional logic involves answering questions such as “*Can we logically conclude α from a knowledge base K ?*”. This is referred to as entailment, and is denoted by $\mathcal{K} \models \alpha$. For example; we can logically conclude that “*first years do not pay taxes*” from the statements in Example 1.1 and we denote this by $\mathcal{K} \models f \rightarrow \neg x$.

2.2 The KLM Approach

There are many different formalizations of non-monotonic reasoning for propositional logic. The *KLM approach* proposed by Kraus et al [12] is an axiomatic approach based on the concept of plausible inference. It is currently a well supported approach[4, 15]. In this approach, plausible inference is represented by a defeasible implication operator of the form $\alpha \sim \beta$, which intuitively means that α is typically a good enough reason to believe β . In this case, α and β are just classical statements. For example; we can now represent the statement “*typically, students do not pay taxes*” as follows:

$$s \sim \neg x$$

We now want to answer questions such as “*Can we typically conclude $\alpha \sim \beta$ from a defeasible knowledge base K (one including defeasible implications)?*”. This is referred to as defeasible entailment, and denoted by $\mathcal{K} \approx \alpha \sim \beta$.

2.3 The KLM Properties

Unlike classical entailment, defeasible entailment is not unique. There exist multiple formalizations of defeasible entailment, such as *Rational Closure* [14], *Relevant Closure* [4] and *Lexicographic Closure* [13]. The KLM framework provides a list of *rationality properties*, which Lehmann and Magidor [14] argued must be satisfied by defeasible entailment methods. This provides a way of differentiating between acceptable and non-acceptable methods of entailment. If a defeasible entailment algorithm satisfies all the properties it is believed to be an acceptable form of defeasible entailment and is called *LM-rational*. The KLM properties for propositional logic are stated below:

$$\begin{array}{ll}
(\text{Ref}) \mathcal{K} \approx \alpha \vdash \alpha & (\text{LLE}) \frac{\alpha \equiv \beta, \mathcal{K} \approx \alpha \vdash \gamma}{\mathcal{K} \approx \beta \vdash \gamma} \\
(\text{RW}) \frac{\mathcal{K} \approx \alpha \vdash \beta, \beta \models \gamma}{\mathcal{K} \approx \alpha \vdash \gamma} & (\text{And}) \frac{\mathcal{K} \approx \alpha \vdash \beta, \mathcal{K} \approx \alpha \vdash \gamma}{\mathcal{K} \approx \alpha \vdash \beta \wedge \gamma} \\
(\text{Or}) \frac{\mathcal{K} \approx \alpha \vdash \gamma, \mathcal{K} \approx \beta \vdash \gamma}{\mathcal{K} \approx \alpha \vee \beta \vdash \gamma} & (\text{CM}) \frac{\mathcal{K} \approx \alpha \vdash \beta, \mathcal{K} \approx \alpha \vdash \gamma}{\mathcal{K} \approx \alpha \wedge \beta \vdash \gamma} \\
(\text{RM}) \frac{\mathcal{K} \approx \alpha \vdash \gamma, \mathcal{K} \approx \alpha \vdash \neg \beta}{\mathcal{K} \approx \alpha \wedge \beta \vdash \gamma} &
\end{array}$$

All of these properties have a fairly intuitive meaning. For example; consider the following two defeasible implications which can be represented using propositional logic:

$$\begin{array}{ll}
\textit{Typically, tutors are students.} & t \vdash s \\
\textit{Typically, tutors are employees.} & t \vdash e
\end{array}$$

It seems rational to conclude the statement below. This is exactly what the *And* property enforces. Kraus et al [12] provide detailed descriptions of the intended meaning of each property.

$$\textit{Typically, tutors are students and employees.} \quad t \vdash s \wedge e$$

2.4 Rational Closure

Rational closure is in many ways the most simple and intuitive way of defining defeasible entailment. There is a semantic definition, based on underlying structures we call *Ranked Interpretations*. There is also an equivalent algorithmic definition [10], which we refer to as the *Rational Closure Algorithm*. For our purposes, we will use the algorithmic definition as the sole definition of Rational Closure.

This algorithm is split into two distinct sub-algorithms, proposed by Casini et al [6]. The BaseRank algorithm is used to construct a ranking of the statements in the defeasible knowledge base \mathcal{K} . The RationalClosure algorithm is used to compute whether a defeasible rule is entailed by the knowledge base, and uses the BaseRank algorithm in its definition.

The BaseRank algorithm takes a defeasible knowledge base \mathcal{K} as input. It is important to note that we can assume that \mathcal{K} is defeasible because any classical sentence α can be expressed as a defeasible implication $\neg \alpha \vdash \perp$.

Intuitively, the BaseRank algorithm starts by putting all of the statements from \mathcal{K} , converted to their classical forms, into E_0 . That is, every $\alpha \vdash \beta$ becomes $\alpha \rightarrow \beta$. Then, to get E_{i+1} from E_i , we keep all the statements such that the left hand side can be “disproved” by the statements in E_i .

When the algorithm stops, what will eventually be left in E_{i+1} is all classical statements. Thus, the infinite level R_∞ , represents all certain information in the knowledge base.

Algorithm 1: BaseRank

Input: A knowledge base \mathcal{K}
Output: An ordered tuple $(R_0, \dots, R_{n-1}, R_\infty, n)$

```

1  $i := 0;$ 
2  $E_0 := \overrightarrow{\mathcal{K}};$ 
3 repeat
4    $E_{i+1} := \{\alpha \rightarrow \beta \in E_i \mid E_i \models \neg \alpha\};$ 
5    $R_i := E_i \setminus E_{i+1};$ 
6    $i := i + 1;$ 
7 until  $E_{i-1} = E_i;$ 
8  $R_\infty := E_{i-1};$ 
9 if  $E_{i-1} = \emptyset$  then
10   $n := i - 1;$ 
11 else
12   $n := i;$ 
13 return  $(R_0, \dots, R_{n-1}, R_\infty, n)$ 

```

Let us consider the following example knowledge base \mathcal{K} , made up of information which we know to be true:

Example 2.1.

- (1) Tutors are students ($t \rightarrow s$)
- (2) First years are students ($f \rightarrow s$)
- (3) Tutors typically pay taxes ($t \vdash x$)
- (4) Students typically do not pay taxes ($s \vdash \neg x$)
- (5) Students typically drink coffee ($s \vdash c$)

Following the BaseRank algorithm, we find the following ranking of the statements from Example 2.1:

0	$s \vdash \neg x$ $s \vdash c$
1	$t \vdash x$
∞	$t \rightarrow s$ $f \rightarrow s$

Figure 1: Base Ranking of Statements in \mathcal{K} in Example 2.1

Intuitively, more general statements will appear higher up in the ranking. For example, since “Tutors are students”, *Students* are a more general concept than *Tutors*, so the statements with *Students* on the left hand side will appear higher up in the ranking.

Algorithm 2: RationalClosure

Input: A knowledge base \mathcal{K} and a defeasible implication $\alpha \vdash \beta$
Output: **true**, if $\mathcal{K} \approx \alpha \vdash \beta$, and **false**, otherwise

```

1  $(R_0, \dots, R_{n-1}, R_\infty, n) := \text{BaseRank}(\mathcal{K});$ 
2  $i := 0;$ 
3  $R := \bigcup_{i=0}^{j < n} R_j;$ 
4 while  $R_\infty \cup R \models \neg \alpha$  and  $R \neq \emptyset$  do
5    $R := R \setminus R_i;$ 
6    $i := i + 1;$ 
7 return  $R_\infty \cup R \models \alpha \rightarrow \beta;$ 

```

The RationalClosure algorithm starts with the ranking provided by BaseRank. It is also asked whether we can conclude $\alpha \vdash \beta$ from \mathcal{K} . Note again that, while this query must be expressed in terms of the defeasible implication operator, we can

express any classical sentence as a defeasible implication. Thus, the algorithm can be used to check classical queries as well.

In each iteration, the algorithm checks if we can conclude $\neg\alpha$ from our current set of information. If we can, then there is no point in even trying to consider if $\alpha \sim \beta$ is the case, since α is never true. Thus, we throw away the top level of the ranking (the least typical statements) and proceed to the next iteration.

In Example 2.1, consider \mathcal{K} being presented with the query $t \sim x$, which represents asking “Do tutors typically pay taxes?”. The algorithm finds that $\neg t$ is a logical consequence of all of the statements. That is, we can conclude that there are no tutors. Thus, we throw away the top level, as shown in Figure 2, and check again if there are any tutors with the remaining statements.

0	$s \sim x \sim s \sim e$
1	$t \sim x$
∞	$t \rightarrow s \quad f \rightarrow s$

Figure 2: Top level of \mathcal{K} in Example 2.1 is thrown away

Eventually, when we reach a point at which we can no longer conclude $\neg\alpha$, we can then check whether $\alpha \rightarrow \beta$ holds in the classical case, and return that as our result. So the RationalClosure algorithm just reduces to a sequence of classical entailment checks.

3 DATALOG

3.1 Standard Datalog

Datalog is a more expressive logic than propositional logic. It allows us to represent statements about specific individuals as well as generic concepts, which can be associated with many individuals. For example; in Datalog we can say that “For all X , X is a tutor” or “Tyler is a tutor” but in propositional logic we can only talk about tutors in general.

Consider the following example, which includes some statements that can be represented using Datalog:

Example 3.1.

- (1) For all X , if X is a first year, then X is a student.
- (2) If Tyler is a tutor, then Tyler is a student.
- (3) For all X , if X is a tutor and post-graduate, then X is a teaching assistant.

In this section we briefly discuss the syntax and semantics of Datalog. Complete formalizations of Logic programming and Datalog are provided by Baral et al [1] and Ceri et al [8] respectively.

The language of Datalog is made up of *facts* and *rules*. Facts provide information about the world and rules allow us to deduce facts from other facts. Rules are expressed as *Horn clauses* with the general form¹:

$$l_1 \wedge l_2 \wedge \dots \wedge l_m \rightarrow l_0$$

Each literal l_i (of arity k_i) has the form $p_i(t_1, \dots, t_{k_i})$, where p_i is a *predicate symbol* and t_1, \dots, t_{k_i} are *terms*. A term is either a *constant* or a *variable*. The left-hand side of the clause is referred to as the *body* and the right-hand side as the *head*.

A fact is expressed as a Horn clause with no body:

$$l_0$$

¹Datalog Horn clauses usually have the form $l_0 \leftarrow l_1 \wedge \dots \wedge l_m$. However, we choose to use a syntax that mirrors that of propositional logic for the ease of the reader. The semantics defined in this paper are equivalent to the semantics defined for clauses of the original form by Ceri et al [8]

We can represent the statements from Example 3.1, using variable X , constant Tyler and predicates f, s, t, p and a which represent first years, students, tutors, post-graduates and teaching assistants respectively:

- (1) $f(X) \rightarrow s(X)$
- (2) $t(\text{Tyler}) \rightarrow s(\text{Tyler})$
- (3) $t(X) \wedge p(X) \rightarrow a(X)$

We say that a clause, such as $t(\text{Tyler}) \rightarrow s(\text{Tyler})$, is *ground* since it does not contain any variables.

Intuitively, a rule says that “If everything in the body holds true, then the head holds true too” and a fact says that “The head is always true”. We formally assign meaning to statements using *Herbrand interpretations*. A *Herbrand interpretation* is a subset τ of all ground facts that can be formed using the predicates and constants expressed in a Datalog program. For example; Tyler is the only constant above so the set of all possible ground facts that we can form is:

$$B^P = \{f(\text{Tyler}), s(\text{Tyler}), t(\text{Tyler}), p(\text{Tyler}), a(\text{Tyler})\}.$$

So a possible Herbrand interpretation is:

$$\tau = \{s(\text{Tyler}), t(\text{Tyler}), p(\text{Tyler})\} \subseteq B^P.$$

To assign truth values to ground facts we check whether they are in the set τ . That is, a ground fact is true for an interpretation τ if and only if it is in τ . For example; $t(\text{Tyler}) \in \tau$ so $t(\text{Tyler})$ is true under τ and $a(\text{Tyler}) \notin \tau$ so $a(\text{Tyler})$ is false under τ .

We say that a rule is true under τ if and only if whenever we can replace variables in the rule by constants and all the literals in the body are in τ , then the head is also in τ . Intuitively this means that whenever we can make the “requirements” of the rule true then the “conclusion” of the rule is also true. More formally, this means that the rules are universally quantified. For example; $t(\text{Tyler}) \in \tau$ and $p(\text{Tyler}) \in \tau$ so the requirements are all true but $a(\text{Tyler}) \notin \tau$ so the conclusion is not true. Thus, the statement $t(X) \wedge p(X) \rightarrow a(X)$ must be false for τ .

If every clause (fact or rule) in a knowledge base is true in τ then we call τ a *Herbrand model*. We say that a ground fact α is entailed by \mathcal{K} , denoted $\mathcal{K} \models \alpha$, if and only if α is in each Herbrand model of \mathcal{K} . Intuitively, this means that whenever our current statements are all true, the new fact is also true.

3.2 Disjunctive Datalog

Datalog can be seen as more expressive than propositional logic in the sense that it allows us to represent statements about individuals. However, it restricts the type of statements that we can make about these individuals. For instance, the statements below cannot be represented using standard Datalog:

Example 3.2.

- (1) For all X , if X is a student, then X is an undergraduate or a postgraduate.
- (2) X is never a student and an employee.

It is often useful to be able to represent statements that involve the disjunction “or”, since these type of statements allow us to model incomplete knowledge. It is also useful to represent statements about falsehood so that you can say what is certainly false as well as what is certainly true.

We now propose an extended version of Datalog, Datalog[∨]. We introduce the literal \perp . Intuitively, we mean that the literal \perp is never true.

We also extend the syntax of rules to allow for disjunction \vee (or) in the head of rules. That is, rules now have the following form, where each b_i and h_j is a literal:

$$b_1 \wedge \dots \wedge b_m \rightarrow h_1 \vee \dots \vee h_n$$

Now we can represent the statement from Example 3.2, using predicates s , u , p and e to represent students, undergraduates, postgraduates and employees respectively:

- (1) $s(X) \rightarrow u(X) \vee p(X)$
- (2) $s(X) \wedge e(X) \rightarrow \perp$

We now need to define the semantics for our extended logic. We consider \perp to be a ground literal. For any Herbrand interpretation τ , we define that \perp is never in τ . We say that a rule $b_1 \wedge \dots \wedge b_m \rightarrow h_1 \vee \dots \vee h_n$ is true for Herbrand interpretation τ if and only if, whenever we can replace variables in the rule by constants and all the literals in the body are in τ , then at least one of the literals in the head is in τ . Intuitively, this now means that whenever we can make the “requirements” of the rule true then the at least one of the “conclusions” is true.

For example; τ defined below is a Herbrand interpretation for the statements in Example 3.1. Now, $s(\text{Tyler}) \in \tau$ so the requirement is true and $p(\text{Tyler}) \in \tau$ so one of the conclusions is true. Thus, since Tyler is the only constant we can replace X with, $s(X) \rightarrow u(X) \vee p(X)$ is true for τ .

$$\tau = \{s(\text{Tyler}), t(\text{Tyler}), p(\text{Tyler})\}.$$

3.3 Defeasible Disjunctive Datalog

Since Disjunctive Datalog uses classical reasoning, it cannot be used to represent defeasible statements such as the one below:

Example 3.3.

- (1) Typically, for all X , if X is a tutor, then X pays tax.

As discussed above, it is often useful to be able to represent such statements. The KLM approach [12] for propositional logic introduces defeasible implications of the form $\alpha \sim \beta$ whose semantics are given by *ranked interpretations* [14]. We want to allow for similar defeasible statements to be represented by Disjunctive Datalog. So, we introduce defeasible rules of the form:

$$b_1 \wedge \dots \wedge b_m \sim h_1 \vee \dots \vee h_n$$

We intend for the logical connective \sim to be the defeasible form of the logical connective \rightarrow in rules. The rule $b_1 \wedge \dots \wedge b_m \sim h_1 \vee \dots \vee h_n$ is intended to intuitively mean that “typically, if all of b_1, \dots, b_m are true, then at least one of h_1, \dots, h_n is true”. For example; the statement in Example 3.3 can be represented in Defeasible Disjunctive Datalog as shown below. In this paper we will not consider a semantic definition of defeasible rules. We will instead define defeasible rules by adapting rational defeasible entailment algorithms for Disjunctive Datalog.

- (1) $t(X) \sim x(X)$

Notice that in the RationalClosure algorithm, defined in Section 2.4, the entailment of complex formulas such as $\alpha \rightarrow \beta$ is required on *line 7*. It stands to reason that such definitions of entailment will also be required when adapting these algorithms for Datalog. However, the semantics of Datalog only defines entailment of ground facts. So, we want to extend the semantics of Datalog to allow for classical entailment of non-ground facts and rules too.

Since Datalog can be seen as a subset of first-order logic, we extend the definition of classical entailment under Herbrand semantics for Datalog to match the definition of entailment under

Herbrand semantics for first-order logic[11]. We define entailment of a Horn clause (rule or fact) as follows: a knowledge base \mathcal{K} entails Horn clause α , denoted by $\mathcal{K} \models \alpha$, if and only if each Herbrand model of \mathcal{K} is also a model of α . Intuitively, this means that whenever our current statements are always true, the clause is also always true.

4 ADAPTED KLM PROPERTIES

Let knowledge base \mathcal{K} be a finite set of defeasible rules. The main question of this paper is to algorithmically analyse *defeasible entailment* $\mathcal{K} \models \alpha \sim \beta$. That is, how do we answer the question: “Can we typically conclude $\alpha \sim \beta$ from a defeasible knowledge base \mathcal{K} ?”. We want to extend the algorithms for answering this question in the propositional case to the Datalog case. We also want to ensure that our adapted algorithms remain “reasonable”. To do so, we adopt Lehmann and Magidor’s approach [14] of analysing the rationality of defeasible entailment algorithms using the KLM properties. In this section we will adapt the KLM properties for Datalog.

4.1 Basic KLM Properties for Datalog

Initially, we attempt to state basic versions of the KLM properties for Datalog. We state the properties in terms of single literals in the head and body of Datalog rules without the use of \wedge and \vee connectives. That is, the defeasible rules which we consider take the following restricted form:

$$b \sim h$$

Let l, m, n be Datalog literals of any arity. The properties below are a simple extension of the KLM properties for propositional logic:

$$\text{(Ref)} \mathcal{K} \models l \sim l \quad \text{(CM)} \frac{\mathcal{K} \models l \sim m, \mathcal{K} \models l \sim n}{\mathcal{K} \models l \wedge m \sim n}$$

We notice that the intuitive “meaning” of entailment $m \models n$ in propositional logic is different to that for Datalog. This is due to the introduction of variables into the logic of Datalog. To understand why, we first need to realise that a Datalog rule $m(X) \rightarrow n(X)$ is equivalent to a first-order logic statement of the form:

$$\forall X, m(X) \rightarrow n(X)$$

For propositional logic, $m \models n$ intuitively means “Whenever m is true, then n is true”. However, $m(X) \models n(X)$ in Datalog is equivalent to the following first-order logic entailment:

$$\forall X, m(X) \models \forall X, n(X)$$

This intuitively means “Whenever $m(X)$ is true for every X , then $n(X)$ is true for every X ”. The problem is that this actually does not say that the X ’s are the same for m and n . So we could have some constant, say *Tyler*, that replaces X for m but not for n . We want to link the X ’s so that what we are actually saying is “For every X , whenever $m(X)$ is true, then $n(X)$ is true”. In other words, we want to say that $\forall X, m(X) \rightarrow n(X)$ is always true. We say that $\forall X, m(X) \rightarrow n(X)$ is a tautology and denote this by $\models \forall X, m(X) \rightarrow n(X)$. That is, in Datalog we write $\models m \rightarrow n$. This intuitive description of $\models m \rightarrow n$ is described formally by Proposition 4.1 below. The proof of Proposition 4.1 is trivial.

PROPOSITION 4.1. *Let τ be a Herbrand interpretation and θ any substitution which replaces variables by constants. Then, $\models m \rightarrow n$ iff $m\theta \in \tau$ implies that $n\theta \in \tau$.*

We can now state the *RW* property in terms of the tautology $\models m \rightarrow n$ so that it has the same meaning as the *RW* property for propositional logic stated in terms of the entailment $m \models n$.

$$(RW) \frac{\mathcal{K} \models l \vdash m, \models m \rightarrow n}{\mathcal{K} \models l \vdash n}$$

Furthermore, we notice that the intuitive “meaning” of equivalence $l \equiv m$ in propositional logic is different to that for Datalog. For propositional logic, $l \equiv m$ intuitively means “*m is true if and only if l is true*”. That is, “*Whenever l is true, then m is true*” and “*Whenever m is true, then l is true*”. However, $l(X) \equiv m(X)$ in Datalog is equivalent to the following first-order logic statement:

$$\forall X, l(X) \equiv \forall X, m(X)$$

This intuitively means “*l(X) is true for every X if and only if m(X) is true for every X*”. We again find that this does not actually say that the X ’s are the same for m and l . This does not correspond to our intuitive understanding of the word “equivalence”. We want to link the X ’s so that what we are actually saying is “*For every X, l(X) is true if and only if m(X) is true*”. In other words, we want to say that $\forall X, m(X) \equiv l(X)$ is always true. That is, in Datalog we want to say that $m \equiv n$ is a statement which is always true.

The problem is that the syntax of Datalog does not include the equivalence relation \equiv so we cannot make the statement $m \equiv n$ in Datalog. However, we can rewrite $m \equiv n$ as $\models l \rightarrow m$ and $\models m \rightarrow l$. Intuitively, this is because “*For every X, l(X) is true if and only if m(X) is true*” means the same thing as “*For every X:*

- (1) *if l(X) is true, then m(X) is true, and,*
- (2) *if m(X) is true, then l(X) is true”*

This intuitive description of why we can rewrite $m \equiv n$ as $\models l \rightarrow m$ and $\models m \rightarrow l$ is described formally by Proposition 4.2 below. A generalized version of Proposition 4.2, Lemma B.3, is proved in Appendix B.

PROPOSITION 4.2. *Let τ be a Herbrand interpretation and θ any substitution which replaces variables by constants. Then, $\models l \rightarrow m$ and $\models m \rightarrow l$ iff $l\theta \in \tau$ and $m\theta \in \tau$, or, $l\theta \notin \tau$ and $m\theta \notin \tau$.*

We can now state the *LLE* property in terms of the tautologies $\models l \rightarrow m$ and $\models m \rightarrow l$ so that it has the same meaning as the *LLE* property for propositional logic stated in terms of the equivalence $l \equiv m$.

$$(LLE) \frac{\models l \rightarrow m, \models m \rightarrow l, \mathcal{K} \models l \vdash n}{\mathcal{K} \models m \vdash n}$$

4.2 Basic KLM Properties which we cannot State in Datalog

The *And*, *Or* and *RM* properties, at first glance, also seem to be simple extensions of the KLM properties for propositional logic. However, we notice that the current syntax of Datalog is too restrictive to state these properties. Recall that the current syntax of Disjunctive Datalog only allows for rules of the form:

$$b_1 \wedge \dots \wedge b_m \rightarrow h_1 \vee \dots \vee h_n$$

Consider the naive extension of the *And* property below. The rule $l \vdash m \wedge n$ has a \wedge connective in its head. However, the current version of Datalog only allows for \vee connectives in the head of a rule.

$$(And) \frac{\mathcal{K} \models l \vdash m, \mathcal{K} \models l \vdash n}{\mathcal{K} \models l \vdash m \wedge n}$$

Now, consider the naive extensions of the *Or* and *RM* properties. In the *Or* property, the rule $l \vee m \vdash n$ has a \vee connective in its body, but the current version of Datalog only allows for \wedge connectives in the body of a rule. Furthermore, the current Datalog syntax does not allow for negation \neg . Hence, the rule $l \vdash \neg m$ in the *RM* property cannot be stated.

$$(Or) \frac{\mathcal{K} \models l \vdash n, \mathcal{K} \models m \vdash n}{\mathcal{K} \models l \vee m \vdash n} \quad (RM) \frac{\mathcal{K} \models l \vdash n, \mathcal{K} \models l \vdash \neg m}{\mathcal{K} \models l \wedge m \vdash n}$$

Thus, the extension of the *And*, *Or* and *RM* properties as stated above cannot be used for the current version of Datalog, since they all violate its syntax.

4.3 Molecules as Combinations of Literals

We introduce the idea of *molecules* as a shorthand for a combination of literals. This shorthand will be used to define more general KLM properties.

We define a *disjunctive molecule*, denoted α^\vee , to be a combination of literals using \vee connectives in the form:

$$l_1 \vee l_2 \vee \dots \vee l_n$$

We define a *conjunctive molecule*, denoted α^\wedge , to be a combination of literals using \wedge connectives in the form:

$$l_1 \wedge l_2 \wedge \dots \wedge l_n$$

We say that a *molecule*, denoted α is either a disjunctive molecule or a conjunctive molecule. We remark that since molecules are just a shorthand, they have no impact on the semantics of Datalog.

4.4 Generalized KLM Properties for Datalog

The basic KLM properties stated in Section 4.1 are stated only in terms of single literals in the head and body of rules. However, in general, Datalog rules may have multiple literals in both the head and body of rules. Thus, since the basic version of the KLM properties limits the structure of rules, it does not fully assess the acceptability of defeasible entailment for Datalog. In this section we analyse generalized versions of the KLM properties for Datalog. We find that, due to the restrictive nature of Datalog’s syntax, none of the properties can be expressed in a general manner without violating Datalog’s syntax.

Firstly, it is clear that the *And*, *Or* and *RM* properties cannot be expressed in general form, since they already cannot be expressed in basic form using the current version of Datalog.

The general versions of the *Ref*, *LLE*, *RW* and *CM* properties, at first glance, all seem to be simple extensions of the properties defined in Section 4.1. However, it turns out that the current syntax of Datalog is too restrictive to state these properties.

$$(Ref) \mathcal{K} \models \alpha \vdash \alpha$$

Notice that molecule α occurs in both the head and body of the rule $\alpha \vdash \alpha$ in the naive general extension of the *Ref* property above. So if α is disjunctive then there will be a \vee connective in the body of the rule, and, if α is conjunctive then there will be a \wedge connective in the head of the rule. Thus, $\alpha \vdash \alpha$ violates the structure of Datalog rules.

A similar discussion can be had about the molecule β , which occurs in both the head and body of different rules in the naive general extensions of the *RW* and *CM* properties below.

$$(RW) \frac{\mathcal{K} \models \alpha^\wedge \vdash \beta, \models \beta \rightarrow \gamma^\vee}{\mathcal{K} \models \alpha^\wedge \vdash \gamma^\vee} \quad (CM) \frac{\mathcal{K} \models \alpha^\wedge \vdash \beta, \mathcal{K} \models \alpha^\wedge \vdash \gamma^\vee}{\mathcal{K} \models \alpha^\wedge \wedge \beta \vdash \gamma^\vee}$$

In the naive general extension of the *LLE* property below, the molecules α^\wedge and β^\wedge both occur in the head of rules. Thus, the \wedge connective occurs in the head of rules, violating the current syntax of Datalog.

$$(LLE) \frac{\models \alpha^\wedge \rightarrow \beta^\wedge, \models \beta^\wedge \rightarrow \alpha^\wedge, \mathcal{K} \models \alpha^\wedge \vdash \gamma^\vee}{\mathcal{K} \models \beta^\wedge \vdash \gamma^\vee}$$

Hence, the extension of the *Ref*, *LLE*, *RW* and *CM* properties in general form cannot be used for the current version of Datalog since they all violate its syntax.

4.5 Motivation for Extended Datalog

We have found that all of the KLM properties cannot be expressed in a general manner and some of them cannot be expressed even in a basic manner. This is due to the restrictive nature of Datalog’s syntax.

However, we need to ensure that LM-rational forms of defeasible entailment satisfy all the KLM properties. We argue that this is necessary even though the reasoning described by some of these properties will never be computed by defeasible entailment algorithms for Datalog. We illustrate why by means of an example. Consider the following two defeasible rules which can be represented using Datalog:

For all X, if X is a tutor, then X is typically a student. $t(X) \vdash s(X)$
For all X, if X is a tutor, then X is typically an employee. $t(X) \vdash e(X)$

It seems rational to conclude the statement below. However, we cannot represent this statement using the current version of Datalog.

For all X, if X is a tutor, then X is typically a student and an employee. $t(X) \vdash s(X) \wedge e(X)$

Thus, to ensure that a form of defeasible entailment is rational, we need to ensure that it will make this conclusion, even though we cannot actually represent the conclusion using Datalog.

We argue that the restrictive nature of Datalog’s syntax is only in place to limit the computational complexity of reasoning about Datalog rules. In fact, by looking at the Herbrand semantics for first-order logic [11], we notice that the Herbrand interpretation semantics allow us to express much more in both the head and body of Datalog rules. We propose that a Datalog extension be used to fully express generalized versions of all of the KLM properties. This way we can analyse the rationality of defeasible entailment using the extended syntax. However, when we actually compute defeasible entailment, we will only ever use the non-extended version of Datalog.

4.6 Datalog+

Our proposed extension to Datalog, Datalog+, introduces the idea of compounds. We again make use of the approaches of first-order logic [11] to define the syntax and semantics of this extended logic.

We recursively define a compound in Datalog+, denoted by A, B . If l is a literal and A and B are compounds, then the following are all compounds:

- l
- $\neg A$
- $A \wedge B$
- $A \vee B$

We define a fact in Datalog+ to be a compound A . We define rules and defeasible rules in Datalog+ to have the following forms respectively:

$$A \rightarrow B \quad A \vdash B$$

Let τ be a Herbrand interpretation and consider some replacement θ of variables by constants. We say that compound A is in τ under the replacement, denoted $A\theta \in \tau$, if and only if one of the following conditions holds, where B, Γ are compounds and l is a literal:

- $A = l$ and after the replacement l is in τ ($l\theta \in \tau$)
- $A = \neg B$ and after the replacement B is not in τ ($B\theta \notin \tau$)
- $A = B \wedge \Gamma$ and after the replacement both B and Γ are in τ ($B\theta \in \tau$ and $\Gamma\theta \in \tau$)
- $A = B \vee \Gamma$ and after the replacement at least one of B or Γ are in τ ($B\theta \in \tau$ or $\Gamma\theta \in \tau$)

We say that fact A is true under Herbrand interpretation τ if and only if A is in τ under every possible replacement of variables by constants. We say that rule $A \rightarrow B$ is true under Herbrand interpretation τ if and only if, whenever A is in τ under some replacement of variables by constants, B is also in τ under the same replacement. If a Horn clause (rule or fact) α is true under τ we say that τ is a model of α .

We define entailment of a Horn clause (rule or fact) as we did before. That is, a knowledge base \mathcal{K} entails Datalog+ Horn clause α , denoted by $\mathcal{K} \models \alpha$, if and only if each Herbrand model of \mathcal{K} is also a model of α .

Notice that any Horn clause expressed in Datalog can be expressed in Datalog+ so Datalog+ is simply an extension of Datalog.

4.7 The KLM Properties Expressed in Datalog+

We state the KLM properties (in Datalog+) for Datalog below, where molecules α, β, γ are used as a shorthand.

$$(LLE) \frac{\models \alpha \rightarrow \beta, \models \beta \rightarrow \alpha, \mathcal{K} \models \alpha \vdash \gamma}{\mathcal{K} \models \beta \vdash \gamma}$$

$$(Ref) \mathcal{K} \models \alpha \vdash \alpha \quad (RW) \frac{\mathcal{K} \models \alpha \vdash \beta, \models \beta \rightarrow \gamma}{\mathcal{K} \models \alpha \vdash \gamma}$$

$$(And) \frac{\mathcal{K} \models \alpha \vdash \beta, \mathcal{K} \models \alpha \vdash \gamma}{\mathcal{K} \models \alpha \vdash \beta \wedge \gamma} \quad (Or) \frac{\mathcal{K} \models \alpha \vdash \gamma, \mathcal{K} \models \beta \vdash \gamma}{\mathcal{K} \models \alpha \vee \beta \vdash \gamma}$$

$$(CM) \frac{\mathcal{K} \models \alpha \vdash \beta, \mathcal{K} \models \alpha \vdash \gamma}{\mathcal{K} \models \alpha \wedge \beta \vdash \gamma} \quad (RM) \frac{\mathcal{K} \models \alpha \vdash \gamma, \mathcal{K} \models \alpha \vdash \neg \beta}{\mathcal{K} \models \alpha \wedge \beta \vdash \gamma}$$

5 RATIONAL CLOSURE FOR DATALOG

In this section we propose a simple adaptation to the BaseRank and RationalClosure algorithms so that they can be used for Datalog. We make use of molecules α, β, γ , as described in Section 4.3, as a shorthand, when describing the algorithms.

5.1 Base Rank Algorithm

The idea of the exceptionality of a statement is central to the BaseRank algorithm. A statement is exceptional with respect to a set of statements if it can be “disproved” by those statements. In the propositional case, we express the notion of falsehood using the negation connective \neg , which intuitively means “not”. Disjunctive Datalog does not allow us to use of the negation connective \neg , but it does allow us to use \perp . We will use \perp to define a notion of falsehood for Datalog.

Notice that, intuitively, $\neg\alpha$ means that α is never true. That is, if α is true, then $\neg\alpha$ is false. Recall \perp is always false. So, whenever α is true, the rule $\alpha \rightarrow \perp$ is false. Thus, we can rewrite $\neg\alpha$ as $\alpha \rightarrow \perp$. This is formally stated in Proposition 5.1 below, the proof of which is found in Appendix C.

PROPOSITION 5.1. *Let τ be a Herbrand interpretation. Then, τ is a model of $\neg\alpha$ under Datalog⁺ semantics iff τ is a model of $\alpha \rightarrow \perp$ under Datalog^v semantics.*

In the propositional case, we assume that all of the statements in our knowledge base are defeasible. We can do this because we can rewrite a classical statement α as the defeasible statement $\neg\alpha \vdash \perp$. However, we cannot rewrite classical Datalog clauses in this manner, since we cannot use \neg . In fact, there is no way to rewrite classical clauses as defeasible rules for the Datalog case. Instead, we form a ranking of only the defeasible statements. Then, since the classical statements are all definite, we add them to the the most typical level, the infinite level.

We can now adapt the BaseRank algorithm, Algorithm 1, for the Datalog case. The adapted version ranks the statements in a knowledge base $\mathcal{K} := D \cup C$, where D is the set of defeasible rules and C the set of classical clauses. It sets out to rank the defeasible rules by setting $E_0 := \overline{D}$ on *line 2*. It now assesses the exceptionality of molecule α by using the entailment check $E_i \cup C \models \alpha \rightarrow \perp$ on *line 4*. Finally, when all the defeasible rules are ranked, it adds the classical clauses to the infinite level by setting $R_\infty := E_{i-1} \cup C$ on *line 8*. For clarity, the adapted BaseRank algorithm for the Datalog case is fully expressed in Algorithm 5 in Appendix A.

5.2 Rational Closure Algorithm

In the RationalClosure algorithm, Algorithm 2, we loop through the statements, level by level, checking for a level where we cannot “disprove” molecule α with the statements remaining. Thus, we again need a notion of falsehood. As with the BaseRank algorithm, we choose to adapt the RationalClosure algorithm by using the entailment check $R_\infty \cup R \models \alpha \rightarrow \perp$ on *line 4* instead of the original $R_\infty \cup R \models \neg\alpha$ check.

Under the assumption that we can compute classical entailment for Datalog^v, this adapted version the RationalClosure algorithm can now be used to check whether a rule $\alpha \vdash \beta$ is defeasibly entailed by the knowledge base \mathcal{K} . For clarity, the adapted RationalClosure algorithm for the Datalog case is fully expressed in Algorithm 6 in Appendix A.

5.3 LM-Rationality

PROPOSITION 5.2. *The adapted RationalClosure algorithm is LM-rational. That is, it satisfies each KLM property.*

Full proofs for the satisfaction of each KLM property by the RationalClosure procedure are provided in Appendix B. We provide a high-level overview for the proof of the *And* property; to illustrate the principles used in the proof. To start, let us take a look at what the *And* property is actually stating.

$$\text{(And)} \quad \frac{\mathcal{K} \models \alpha \vdash \beta, \mathcal{K} \models \alpha \vdash \gamma}{\mathcal{K} \models \alpha \vdash \beta \wedge \gamma}$$

This says; if we operate on a fixed knowledge base \mathcal{K} such that

- (1) when given query $\alpha \vdash \beta$, the algorithm returns **true**, and,
- (2) when given query $\alpha \vdash \gamma$, the algorithm returns **true**.

Then, when passed the query $\alpha \vdash \beta \wedge \gamma$, the algorithm will also return **true**.

To see what this actually entails, we need to take a closer look at the algorithm, and consider 2 cases. Note first that all 3 queries have the same symbol α on the left hand side of the defeasible implication. So the ranking returned by the BaseRank algorithm will be the same for all of them.

The first case is one where during the $\mathcal{K} \models \alpha \vdash \beta$ checking, $R_\infty \cup R \models \neg\alpha$ the entire time. Then, since the ranking is the same, it is also the case in the $\mathcal{K} \models \alpha \vdash \beta \wedge \gamma$ checking, $R_\infty \cup R \models \neg\alpha$ the entire time. So the algorithm reaches the following line:

return $R_\infty \cup R \models \alpha \rightarrow \beta \wedge \gamma$;

But at this point, still $R_\infty \cup R \models \neg\alpha$, so $\alpha \rightarrow \beta \wedge \gamma$ is vacuously true and $R_\infty \cup R \models \alpha \rightarrow \beta \wedge \gamma$ will return **true**. The reason for this is best seen by example. Suppose it was known that there are no apples (α), which corresponds to the statement $\neg\alpha$. We then claim that all apples (α) are bananas (β) and grapes (γ), which corresponds to the statement $\alpha \rightarrow \beta \wedge \gamma$. Since there are no apples, this statement is technically true. Thus, the algorithm will return **true** in the first case.

The second case is where $R_\infty \cup R \not\models \neg\alpha$ for the first time at some point i . Again, since the ranking is the same for all queries, this will be the exact same point i in all 3 of the queries.

0	$R_\infty \cup R \models \neg\alpha$
...	$R_\infty \cup R \models \neg\alpha$
i	$R_\infty \cup R \not\models \neg\alpha$
...	...

Figure 3: At some point i , $R_\infty \cup R \not\models \neg\alpha$

Then since $\mathcal{K} \models \alpha \vdash \beta$, $\mathcal{K} \models \alpha \vdash \gamma$, we know that at point i , $R_\infty \cup R \models \alpha \rightarrow \beta$ and $R_\infty \cup R \models \alpha \rightarrow \gamma$. If we let α represent tutors, β represent students and γ represent employees, then we know:

$$\alpha \rightarrow \beta \text{ (tutors are students)} \quad \alpha \rightarrow \gamma \text{ (tutors are employees)}$$

Then we can conclude that $\alpha \rightarrow \beta \wedge \gamma$, which corresponds to tutors being both students and employees. Thus $R_\infty \cup R \models \alpha \rightarrow \beta \wedge \gamma$ is true at point i , so the algorithm returns **true**.

6 LEXICOGRAPHIC CLOSURE

The Rational Closure approach is simple and intuitive. However, it seems unnecessary to throw away an entire level of statements when we can “disprove” α . While it is true that a statement within the level is likely causing the conflict, there are other statements in the level that have no effect on the conflict occurring. To address this, the Lexicographic Closure algorithm takes a finer grained approach to removing statements when a conflict is found.

6.1 An Intuitive Description

Unlike in the Rational Closure algorithm, if we can “disprove” α using the remaining ranked statements, we do not remove all statements with the worst rank. We try to only remove one of the n statements of worst rank. However, the statements all have equal rank so we cannot simply choose any one of them. We need to remove the statement which is causing the conflict. We do this by first considering the ranking under all possible ways of removing one statement from the worst rank: all subsets of

worst-ranked statements of cardinality $n - 1$. We again try to “disprove” α , each time using one of these subsets as our new worst-ranked level of statements.

Recall, in Example 2.1, when \mathcal{K} is presented with the query “Do tutors pay taxes?” ($t \sim x$), we can conclude that there are no tutors ($\neg t$). Following the Lexicographic Closure procedure, as shown in Figure 4, we now try to remove the single statement $s \sim \neg x$ that “Students do not pay taxes” from the top level of statements. Then, we check whether there are still no tutors. We find that this is, in fact, the case. So we repeat the process, this time removing the single statement $s \sim c$ that “Students drink coffee”.

0	$s \sim \neg x$ $s \sim c$
1	$t \sim x$
∞	$t \rightarrow s$ $f \rightarrow s$

Figure 4: Single statement in the top level of \mathcal{K} in Example 2.1 is thrown away

If we can “disprove” α for all of the subsets of size $n - 1$, we then try to remove only two of the worst-ranked n statements. We now consider all subsets of worst-ranked statements of cardinality $n - 2$ and try to “disprove” α . We continue in this manner until we find that we cannot “disprove” α .

If we find that for all subsets of statements of cardinality 1 we can still “disprove” α , then all the statements in the worst rank are causing the conflict. We throw away the whole worst-ranked level and repeat the previous process with the remaining ranked statements.

For instance, in Example 2.1, when we remove the statement “Students drink coffee”, there are still no tutors. Thus, we throw away the whole top level, as shown in Figure 2.

As in the Rational Closure algorithm, we stop this process when we find that we cannot “disprove” α or when we reach the infinite rank. We conclude by checking whether we can logically conclude $\alpha \rightarrow \beta$ from the remaining statements at this point.

6.2 Rephrasing the Intuitive Description

The translation of the intuitive description of Lexicographic Closure into an algorithm results in an algorithm which does not satisfy the *And* property. This is undesirable since we want the algorithm to be LM-rational.

Suppose we are at some level i and point $j \in [1, n_i - 1]$, where j is the number of worst ranked statements we want to remove from the n_i worst-ranked statements. In the intuitive description, we want to check whether we can “disprove” α using one of the subsets of statements, S_1, S_2, \dots, S_n , of cardinality $n_i - j$ as our new worst-rank level of statements. That is, we want to check whether we can “disprove” α using S_1 or S_2 or, ..., or S_n as our new worst-rank level of statements.

We are using “or” on a meta level, a level where we are making logical statements about logical statements. The use of this meta statement weakens our entailment checking in a manner that violates the *And* property.

In the propositional case, we address this issue by rephrasing the algorithm without meta level statements. We do so by rewriting the intuitive description using \vee connectives in a single statement instead of “or”s between multiple statements. This results in an algorithm which is LM-rational. We will systematically describe this rephrasing in the rest of this section.

We begin by considering the following subset of statements:

$$S_i = \{x_1, x_2, \dots, x_m\}$$

Suppose we want to check whether we can logically conclude $p \rightarrow q$ from S_i ($S_i \models p \rightarrow q$). So, we want to check that whenever “ x_1 is true, and, x_2 is true, and, ..., and, x_m is true”, then it is also the case that “ $p \rightarrow q$ is true”. This is intuitively the same as saying that whenever “ x_1 and x_2 and ... and x_m are true”, then it is also the case that “ $p \rightarrow q$ is true”. That is, we want to check whether we can logically conclude $p \rightarrow q$ from the single statement:

$$s_i := \bigwedge_{x \in S} x = x_1 \wedge x_2 \wedge \dots \wedge x_m$$

Consider the set of n subsets of ranking R_i , each of size m :

$$S = \{S_1, S_2, \dots, S_n\}$$

Suppose we want to check whether $S_1 \models p \rightarrow q$, or, $S_2 \models p \rightarrow q$, or, ..., or, $S_n \models p \rightarrow q$. This is the same as checking whether $S_1 \models p \rightarrow q$, or, $S_2 \models p \rightarrow q$, or, ..., or, $S_n \models p \rightarrow q$. That is, we want to check that “whenever s_1 is true then $p \rightarrow q$ is true” or “whenever s_2 is true then $p \rightarrow q$ is true” or ... or “whenever s_n is true then $p \rightarrow q$ is true”.

Notice that if the check for “whenever s_1 or s_2 or ... or s_n is true then $p \rightarrow q$ is true” holds then the above check must also hold. So this is a stronger form of entailment check. Furthermore, this is the same as checking that “whenever $s_1 \vee s_2 \vee \dots \vee s_n$ is true then $p \rightarrow q$ is true”. So we can restate the previous list of entailment checks in terms of a single stronger form of entailment check:

$$s_1 \vee s_2 \vee \dots \vee s_n \models p \rightarrow q$$

Using this intuition, we can replace the list of entailment checks, using S_1 or S_2 or, ..., or S_n as the worst-ranked level of statements, with a single entailment check using $s_1 \vee s_2 \vee \dots \vee s_n$ as a single statement on the worst rank level.

6.3 Algorithm for Propositional Logic

We can now define a Lexicographic Closure algorithm in terms of the sub-algorithms `SubsetRank` and `LexicographicClosure`. This form of Lexicographic Closure for the propositional case has been shown to be LM-rational[13].

The `SubsetRank` algorithm, Algorithm 3, constructs a new ranking of statements by using the base ranks $R_0, \dots, R_{n-1}, R_\infty$ computed by the `BaseRank` algorithm. It adds new rank levels $D_{i, n_i-1}, D_{i, n_i-2}, \dots, D_{i, 1}$ in between each existing rank level R_i and R_{i+1} .

Algorithm 3: SubsetRank

Input: A knowledge base \mathcal{K}

Output: An ordered tuple $(R_0, \dots, R_k, R_\infty, k + 1)$

1 $(B_0, \dots, B_{m-1}, B_\infty, m) := \text{BaseRank}(\mathcal{K});$

2 $i := 0; k := 0;$

3 **repeat**

4 **for** $j := |B_i|$ **to** 1 **do**

5 $S_{i,j} := \text{Subsets}(R_i, j);$

6 $D_{i,j} := \bigvee_{X \in S_{i,j}} \bigwedge_{x \in X} x;$

7 $R_k := D_{i,j};$

8 $k := k + 1;$

9 $i := i + 1;$

10 **until** $i := m;$

11 $R_\infty := B_\infty;$

12 **return** $(R_0, \dots, R_k, R_\infty, k + 1)$

The function $\text{Subsets}(X, k)$ finds all possible subsets of size $k < n$ of a set X of size n .

Following the SubsetRank algorithm, we find the following ranking of the statements from Example 2.1:

R_0	$s \vdash \neg x \quad s \vdash c$
$D_{0,1}$	$(s \vdash \neg x) \vee (s \vdash c)$
R_1	$t \vdash x$
R_∞	$t \rightarrow s \quad f \rightarrow s$

Figure 5: Subset Ranking of Statements in \mathcal{K} in Example 2.1

The $\text{LexicographicClosure}$ algorithm, Algorithm 4, uses the ranking produced by the SubsetRank algorithm to compute $\text{Lexicographic Closure}$ in a manner equivalent to that used by the RationalClosure algorithm.

Algorithm 4: LexicographicClosure

Input: A knowledge base \mathcal{K} and a defeasible rule $\alpha \sim \beta$

Output: true, if $\mathcal{K} \approx \alpha \sim \beta$, and false, otherwise

- 1 $(R_0, \dots, R_k, R_\infty, k+1) := \text{SubsetRank}(\mathcal{K})$;
 - 2 $i := 0$;
 - 3 $R := \bigcup_{i=0}^{j \leq k} R_j$;
 - 4 **while** $R_\infty \cup R \models \neg \alpha$ **and** $R \neq \emptyset$ **do**
 - 5 $R := R \setminus R_i$;
 - 6 $i := i + 1$;
 - 7 **return** $R_\infty \cup R \models \alpha \rightarrow \beta$;
-

Intuitively, each set of statements $D_{i,j}$, in between base rank levels R_i and R_{i+1} , is a weaker form of the set of statements R_i on level i . In fact, the set of models of R_i must be a subset of the set of models of $D_{i,j}$ ($\llbracket R_i \rrbracket \subseteq \llbracket D_{i,j} \rrbracket$). This means that we can only logically conclude $\alpha \rightarrow \beta$ from $R_i \cup D_{i,j}$ if we can logically conclude $\alpha \rightarrow \beta$ from R_i . So, adding these extra levels has no effect on the entailment checks on *line 4* and *line 7* of Algorithm 4, when we are on a base rank level R_i .

However, if we can “disprove” α when we are on a base rank level R_i , then Algorithm 4 will remove level R_i . The algorithm will now check whether we can “disprove” α using subset ranking D_{i,n_i-1} as the worst level. If we can, it will then remove subset ranking D_{i,n_i-1} and check whether we can “disprove” α using subset ranking D_{i,n_i-2} as the worst level. It will proceed in this manner until we find that either we cannot “disprove” α or we have reached level R_{i+1} .

Notice that, as described in the previous section, checking entailment with subset ranking $D_{i,j}$ as the worst level is the same as the list of entailment checks, using each of the subsets of R_i of size j as the worst-rank level. Thus, the algorithmic definition of $\text{Lexicographic Closure}$ is equivalent to the intuitive definition.

7 LEXICOGRAPHIC CLOSURE FOR DATALOG

In this section we extend the $\text{Lexicographic Closure}$ algorithm for the propositional case to the Datalog case. We conclude the section by showing that our extended algorithm is LM-rational.

7.1 Rephrasing the Intuitive Description for Datalog

Unfortunately, the rephrasing of the intuitive definition for the propositional case cannot directly be used for the Datalog case. This is due to the fact that some of the statements in each subset S_i may be rules. Clearly combining multiple rules using \wedge and \vee connectives will violate the current syntax of Datalog, which only allows for clauses of the following form:

$$b_1 \wedge \dots \wedge b_m \rightarrow h_1 \vee \dots \vee h_n$$

For example; consider the set of statements on some level i :

$$\{a(X) \rightarrow b(X), c(X) \rightarrow d(X)\}$$

The statement representing $S_{i,1}$ is shown below. Clearly this violates the current syntax of Datalog.

$$(a(X) \rightarrow b(X)) \vee (c(X) \rightarrow d(X))$$

Notice that if $a(X) \rightarrow b(X)$ is true for some replacement of X by constants, then either $a(X)$ is not true or $b(X)$ is true. So we can rewrite $a(X) \rightarrow b(X)$ as $\neg a(X) \vee b(X)$. Recall that the clause $\neg a(X) \vee b(X)$ is the same as the following universally quantified first-order logic clause:

$$\forall X, \neg a(X) \vee b(X)$$

It can be shown that the first-order statement formed by combining universally quantified statements with \wedge and \vee connectives can be transformed into a universally quantified Conjunctive Normal Form (CNF):

$$\forall X, D_1 \wedge D_2 \wedge \dots \wedge D_n, \text{ where,}$$

$$D_i := \neg a_{i,1}(X) \vee \dots \vee \neg a_{i,r_i}(X) \vee b_{i,1}(X) \vee \dots \vee b_{i,s_i}(X)$$

For example; the statement $\forall X, (a(X) \rightarrow b(X)) \vee (c(X) \rightarrow d(X))$ can be written as $\forall X, (\neg a(X) \vee b(X)) \vee (\neg c(X) \vee d(X))$ which can be rewritten as $\forall X, (\neg a(X) \vee \neg c(X) \vee b(X) \vee d(X))$.

We notice that statements of the form $\neg a_{i,1}(X) \vee \dots \vee \neg a_{i,r_i}(X)$ can be written as $\neg(a_{i,1}(X) \wedge \dots \wedge a_{i,r_i}(X))$. So we find that each D_i in Conjunctive Normal Form can be rewritten in the following form:

$$D_i := \neg(a_{i,1}(X) \wedge \dots \wedge a_{i,r_i}(X)) \vee (b_{i,1}(X) \vee \dots \vee b_{i,s_i}(X))$$

Now, using our previous intuition for rewriting rules, we can rewrite each D_i as follows:

$$D_i := a_{i,1}(X) \wedge \dots \wedge a_{i,r_i}(X) \rightarrow b_{i,1}(X) \vee \dots \vee b_{i,s_i}(X)$$

That is, each D_i can be written as a rule of the following form:

$$D_i := \alpha^\wedge \rightarrow \beta^\vee$$

Thus, we can now replace each set $S_{i,j} = \{S_1, S_2, \dots, S_j\}$ in the SubsetRank algorithm, with a single statement of the form:

$$(\alpha_1^\wedge \rightarrow \beta_1^\vee) \wedge (\alpha_2^\wedge \rightarrow \beta_2^\vee) \wedge \dots \wedge (\alpha_k^\wedge \rightarrow \beta_k^\vee)$$

This statement still violates the syntax of Datalog. However, as discussed previously, checking whether $(\alpha_1^\wedge \rightarrow \beta_1^\vee) \wedge (\alpha_2^\wedge \rightarrow \beta_2^\vee) \wedge \dots \wedge (\alpha_k^\wedge \rightarrow \beta_k^\vee) \models \alpha \rightarrow \beta$ is the same as checking whether $\{\alpha_1^\wedge \rightarrow \beta_1^\vee, \alpha_2^\wedge \rightarrow \beta_2^\vee, \dots, \alpha_k^\wedge \rightarrow \beta_k^\vee\} \models \alpha \rightarrow \beta$.

So, we can now replace each set $S_{i,j} = \{S_1, S_2, \dots, S_j\}$ in the SubsetRank algorithm, with the set of statements of the form:

$$S_{i,j} := \{\alpha_1^\wedge \rightarrow \beta_1^\vee, \alpha_2^\wedge \rightarrow \beta_2^\vee, \dots, \alpha_k^\wedge \rightarrow \beta_k^\vee\}$$

7.2 Algorithm for Datalog

We can now formally define the adjustments to the SubsetRank algorithm which are required for the Datalog case. On *line 7* of Algorithm 3 for the propositional case, the algorithm sets ranking R_k to $D_{i,j}$. For the Datalog case, the algorithm should now set R_k to $\text{RNF}(D_{i,j})$, the Rule Normal Form of $D_{i,j}$.

Given the “extended” Datalog statement Γ , the Rule Normal Form function $\text{RNF}(\Gamma)$ does the following:

- (1) Computes the Conjunctive Normal Form $\text{CNF}(\Gamma)$.
- (2) Converts $\text{CNF}(\Gamma)$ into a conjunction of clauses of the form $(\alpha_1^\wedge \rightarrow \beta_1^\vee) \wedge (\alpha_2^\wedge \rightarrow \beta_2^\vee) \wedge \dots \wedge (\alpha_k^\wedge \rightarrow \beta_k^\vee)$.
- (3) Converts the conjunction of clauses into a set of clauses $\{\alpha_1^\wedge \rightarrow \beta_1^\vee, \alpha_2^\wedge \rightarrow \beta_2^\vee, \dots, \alpha_k^\wedge \rightarrow \beta_k^\vee\}$.
- (4) Returns the set of clauses.

We need to adapt our definition of the notion of exceptionally in the `LexicographicClosure` algorithm, Algorithm 4. As with the `RationalClosure` algorithm for Datalog, we choose to adapt the `LexicographicClosure` algorithm by using the entailment check $R_\infty \cup R \models \alpha \rightarrow \perp$ on *line 4* instead of the original $R_\infty \cup R \models \neg\alpha$ check.

Under the assumption that we can compute classical entailment for Datalog^\vee , we can define `Lexicographic Closure` for Datalog, using the adapted `SubsetRank` and `LexicographicClosure` algorithms. For clarity, the adapted algorithms for the Datalog case are fully expressed in Appendix C.

7.3 LM-Rationality of the Algorithm

The `LexicographicClosure` algorithm is exactly the same as the `RationalClosure` algorithm, barring the use of the `SubsetRank` ranking instead of the `BaseRank` ranking. Furthermore, by examining the proofs for LM-rationality of the `RationalClosure` algorithm in Appendix B, we find that none of the proofs are dependant on the type of ranking produced by the `BaseRank` algorithm. Thus, the LM-rationality of the `LexicographicClosure` algorithm follows directly from the proof of LM-rationality of the `RationalClosure` algorithm. This is discussed further in Appendix D.

PROPOSITION 7.1. *The adapted `LexicographicClosure` algorithm is LM-rational. That is, it satisfies each KLM property.*

8 RELATED WORK

Kraus, Lehmann and Magidor (KLM) [12] introduced preferential reasoning, KLM-style defeasible implications and the KLM properties. Lehmann and Magidor [13] presented the concept of Rational Closure for propositional logic, provided an algorithm to compute it, and, showed that it is LM-rational. Britz et al[3] provided an extension of the KLM properties for description logics and presented an extension of Rational Closure for description logics.

Lehmann [13] introduced `Lexicographic Closure` for propositional logic, showed that it was strictly less conservative than Rational Closure, and proved that it is also LM-rational. Casini et al[6] presented a systematic approach for enriching propositional logic with a defeasible implication connective, and described algorithms for Rational Closure and `Lexicographic Closure` within this framework. Casini et al[7] presented an algorithm for `Lexicographic Closure` for description logics.

9 CONCLUSIONS

The central focus of this paper is to determine what defeasible entailment means for Datalog enriched with a defeasible rule connective. We define this systematically by extending the KLM approach[12] for propositional logic to the Datalog case. Due to the differences between the Herbrand semantics of Datalog and the semantics of propositional logic, the KLM properties cannot be directly extended to the Datalog case. However, the

properties can be rephrased to have the same intended meaning in a slightly less restricted version of Datalog. The Rational Closure algorithm is easily extended to the Datalog case and remains LM-rational. The less conservative form of defeasible entailment, Lexicographic Closure, remains LM-rational when extended to the Datalog case. However, it requires some adjustments for the Datalog case, due to the restrictive nature of Datalog’s syntax.

10 FUTURE WORK

There are at least three lines of research to which the work in this paper can lead. First is an implementation and optimization of the defeasible entailment algorithms for Disjunctive Datalog. The logic of Datalog^\vee can be seen as a subset of the logic of DLV[9], a disjunctive logic programming system. Hence, the defeasible entailment algorithms can be used by the DLV system for restricted cases where negation-as-failure and weak constraints are not used.

Secondly, we only define defeasible entailment for Datalog algorithmically. However, the semantic definition in terms of minimal models should be explored for both Rational Closure and `Lexicographic Closure` for Datalog.

Finally, Casini et al[6] showed that LM-rationality is necessary but not sufficient. The additional properties for Basic Defeasible Entailment proposed by Casini et al[6] can be extended to Datalog. Furthermore, other properties that are specific to defeasible entailment for Datalog should be explored.

REFERENCES

- [1] C. Baral and M. Gelfond. 1994. Logic programming and knowledge representation. *The Journal of Logic Programming* 19-20 (1994), 73–148.
- [2] M. Ben-Ari. 2012. *Mathematical Logic for Computer Science*. Springer Science & Business Media, Rehovot, Israel.
- [3] K. Britz, G. Casini, T. Meyer, K. Moodley, U. Sattler, and I. Varzinczak. 2015. *Rational Defeasible Reasoning for Description Logics*. Technical Report. CSIR Meraka Institute, UKZN, Stellenbosch University, UCT.
- [4] G. Casini, T. Meyer, K. Moodley, and R. Nortje. 2014. Relevant Closure: A New Form of Defeasible Reasoning for Description Logics. In *JELIA 2014: Logics in Artificial Intelligence*. Springer, Funchal, Madeira, Portugal, 92–106.
- [5] G. Casini, T. Meyer, K. Moodley, and I. Varzinczak. 2013. Towards Practical Defeasible Reasoning for Description Logics. In *Proceedings of the 26th International Workshop on Description Logics*. CEUR Workshop Proceedings, Ulm, Germany, 587–599.
- [6] G. Casini, T. Meyer, and I. Varzinczak. 2019. Taking Defeasible Entailment beyond Rational Closure. In *JELIA 2019: Logics in Artificial Intelligence*. Springer, Rende, Italy, 182–197.
- [7] G. Casini and U. Straccia. 2012. Lexicographic closure for Defeasible Description Logics. In *Proceedings of the 8th Australasian Ontology Workshop*. CEUR Workshop Proceedings, Sydney, Australia, 28–39.
- [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 (1989), 146–166.
- [9] T. Eiter, W. Faber, C. Koch, N. Leone, and G. Pfeifer. 2000. DLV - A System for Declarative Problem Solving. In *Proceedings of the 8th International Workshop on Non-Monotonic Reasoning*. Breckenridge, Colorado, USA, 28–39.
- [10] M. Freund. 1998. Preferential reasoning in the perspective of Poole default logic. *Artificial Intelligence* 98 (1998), 209–235.
- [11] M. Genesereth and K. Eric. 2013. *Introduction to Logic*. Morgan & Claypool, Stanford, California.
- [12] S. Kraus, D. Lehmann, and M. Magidor. 1990. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence* 44 (1990), 167–207.
- [13] D. Lehmann. 1995. Another perspective on default reasoning. *Annals of Mathematics and Artificial Intelligence* 15, 1 (1995), 61–82.
- [14] D. Lehmann and M. Magidor. 1992. What does a conditional knowledge base entail? *Artificial Intelligence* 55 (1992), 1–60.
- [15] U. Straccia and G. Casini. 2013. Defeasible Inheritance-Based Description Logics. *Journal of Artificial Intelligence Research* 48 (2013), 415–473.

APPENDICES

A RATIONAL CLOSURE ALGORITHMS FOR DATALOG

Algorithm 5: BaseRank

Input: A defeasible knowledge base D and classical knowledge base C
Output: An ordered tuple $(R_0, \dots, R_{n-1}, R_\infty, n)$

```

1  $i := 0;$ 
2  $E_0 := \vec{D};$ 
3 repeat
4    $E_{i+1} := \{\alpha \rightarrow \beta \in E_i \mid E_i \cup C \models \alpha \rightarrow \perp\};$ 
5    $R_i := E_i \setminus E_{i+1};$ 
6    $i := i + 1;$ 
7 until  $E_{i-1} = E_i;$ 
8  $R_\infty := E_{i-1} \cup C;$ 
9 if  $E_{i-1} = \emptyset$  then
10   $n := i - 1;$ 
11 else
12   $n := i;$ 
13 return  $(R_0, \dots, R_{n-1}, R_\infty, n)$ 

```

Algorithm 6: RationalClosure

Input: A defeasible knowledge base D , a classical knowledge base C and a defeasible rule $\alpha \vdash \beta$
Output: **true**, if $\mathcal{K} \approx \alpha \vdash \beta$, and **false**, otherwise

```

1  $(R_0, \dots, R_{n-1}, R_\infty, n) := \text{BaseRank}(D, C);$ 
2  $i := 0;$ 
3  $R := \bigcup_{i=0}^{j < n} R_j;$ 
4 while  $R_\infty \cup R \models \alpha \rightarrow \perp$  and  $R \neq \emptyset$  do
5    $R := R \setminus R_i;$ 
6    $i := i + 1;$ 
7 return  $R_\infty \cup R \models \alpha \rightarrow \beta;$ 

```

B LM-RATIONALITY OF RATIONAL CLOSURE

Let $\mathcal{K} := D \cup C$ be a Datalog knowledge base, where D is a set of defeasible rules and C is a set of classical clauses. Let α, β, γ be molecules. We provide proofs below for the satisfaction of each KLM property by the RationalClosure procedure. That is, we prove that RationalClosure is *LM-rational*. We start by showing that while checking $\mathcal{K} \approx \alpha \vdash \beta$, if it is always the case that $R_\infty \cup R \models \neg\alpha$, then the algorithm returns **true**.

LEMMA B.1. *Let \mathcal{K} be a knowledge base and α, β molecules such that when checking $\mathcal{K} \approx \alpha \vdash \beta$, it is always the case that $R_\infty \cup R \models \neg\alpha$. Then, the RationalClosure algorithm returns **true**.*

Proof of Lemma B.1: Since, in the checking, it is always the case that $R_\infty \cup R \models \neg\alpha$, the *while* loop on line 4 will keep looping, until $R = \emptyset$. Then the algorithm will jump to line 7, and return $R_\infty \cup R \models \alpha \rightarrow \beta$.

But, since $R_\infty \cup R \models \neg\alpha$, we know that $\alpha\theta \notin \tau$ for every substitution θ and model τ (of $R_\infty \cup R$). Thus, $\alpha \rightarrow \beta$ is true under every substitution θ and model τ . Hence, the query $R_\infty \cup R \models \alpha \rightarrow \beta$ must return **true**. So the algorithm itself returns **true**. \square

B.1 Ref

We want to show that $\mathcal{K} \approx \alpha \vdash \alpha$. We will make use of Lemma B.2 to do so.

LEMMA B.2. *The defeasible rule $\alpha \rightarrow \alpha$ is a tautology.*

Proof of Lemma B.2: Let τ be any Herbrand interpretation and θ a substitution which replaces variables by constants. If $\alpha\theta \in \tau$ then $\alpha\theta \in \tau$. So τ is a model of $\alpha \rightarrow \alpha$. Hence, $\alpha \rightarrow \alpha$ is a tautology. \square

Let τ be a Herbrand interpretation of \mathcal{K} and θ a substitution which replaces variables by constants. We now consider 2 cases below:

Case 1: At some point (when $i \in [0, n]$) in the $\mathcal{K} \approx \alpha \vdash \alpha$ checking, $R_\infty \cup R \not\models \neg\alpha$ for the first time. Then, since $\alpha \rightarrow \alpha$ is a tautology, any model of $R_\infty \cup R$ must satisfy $\alpha \rightarrow \alpha$ so $R_\infty \cup R \models \alpha \rightarrow \alpha$. Thus, the algorithm returns **true**.

Case 2: It is always the case in the $\mathcal{K} \approx \alpha \vdash \alpha$ checking that $R_\infty \cup R \models \neg\alpha$. Then, the algorithm returns **true**, by Lemma B.1. \square

B.2 LLE

Suppose $\models \alpha \rightarrow \beta$, $\models \beta \rightarrow \alpha$ and $\mathcal{K} \approx \alpha \vdash \gamma$. We want to show that $\mathcal{K} \approx \beta \vdash \gamma$. We will make use of Lemma B.3, a generalized version of Proposition 4.2, to do so.

LEMMA B.3. *Let τ be a Herbrand interpretation and θ a substitution which replaces variables by constants. Then, $\models \alpha \rightarrow \beta$ and $\models \beta \rightarrow \alpha$ iff $\alpha\theta \in \tau$ and $\beta\theta \in \tau$, or, $\alpha\theta \notin \tau$ and $\beta\theta \notin \tau$.*

Proof of Lemma B.3: Let τ be some Herbrand interpretation and θ some substitution which replaces variables by constants. Suppose that $\alpha\theta \in \tau$. Since $\models \alpha \rightarrow \beta$ we must have that τ satisfies $\alpha \rightarrow \beta$ and so $\beta\theta \in \tau$. Now suppose that $\alpha\theta \notin \tau$. We know that τ satisfies $\beta \rightarrow \alpha$ since $\models \beta \rightarrow \alpha$. So we must have $\beta\theta \notin \tau$. Similar arguments hold for when $\beta\theta \in \tau$ and $\beta\theta \notin \tau$. \square

Claim: At each level, $R_\infty \cup R \models \neg\beta$ iff $R_\infty \cup R \models \neg\alpha$.

Proof of Claim: Suppose that, at some point $i \in [0, n]$, $R_\infty \cup R \models \neg\alpha$. Let τ be a model of $R_\infty \cup R$ and θ some substitution which replaces variables by constants. So τ is a model of $\neg\alpha$ and, hence, $\alpha\theta \notin \tau$. Thus, by Lemma B.3, $\beta\theta \notin \tau$ so τ is a model of $\neg\beta$. Hence, $R_\infty \cup R \models \neg\beta$. Similarly, we can show that if at some point $i \in [0, n]$, $R_\infty \cup R \models \neg\beta$, then $R_\infty \cup R \models \neg\alpha$.

Now suppose that, at some point $i \in [0, n]$, $R_\infty \cup R \not\models \neg\alpha$. Then, there is some model τ of $R_\infty \cup R$ such that τ is not a model of $\neg\alpha$. So there must be some substitution θ such that $\alpha\theta \in \tau$. Hence, by Lemma B.3, $\beta\theta \in \tau$ so τ is not a model of $\neg\beta$. Thus, $R_\infty \cup R \not\models \neg\beta$. Similarly, we can show that if at some point $i \in [0, n]$, $R_\infty \cup R \not\models \neg\beta$, then $R_\infty \cup R \not\models \neg\alpha$. \square

We now consider 2 cases below:

Case 1: At some point (when $i \in [0, n]$) in the $\mathcal{K} \approx \alpha \vdash \gamma$ checking, $R_\infty \cup R \not\models \neg\alpha$ for the first time. Then, at point i , since $\mathcal{K} \approx \alpha \vdash \gamma$, $R_\infty \cup R \models \alpha \rightarrow \gamma$. As shown above, at the same point i , $R_\infty \cup R \not\models \neg\beta$ for the first time. The algorithm now checks that $R_\infty \cup R \models \beta \rightarrow \gamma$. Let τ be a model of $R_\infty \cup R$ and θ a substitution which replaces variables by constants. Suppose $\beta\theta \in \tau$ then, by Lemma B.3, $\alpha\theta \in \tau$ too. And, since $R_\infty \cup R \models \alpha \rightarrow \gamma$, we must have $\gamma\theta \in \tau$. So $R_\infty \cup R \models \beta \rightarrow \gamma$ and the algorithm returns **true**.

Case 2: It is always the case in the $\mathcal{K} \approx \alpha \vdash \gamma$ checking that $R_\infty \cup R \models \neg\alpha$. Then, in the $\mathcal{K} \approx \beta \vdash \gamma$ checking, as shown above, it is also always the case that $R_\infty \cup R \models \neg\beta$. So the algorithm returns **true**, by Lemma B.1. \square

B.3 RW

Suppose $\models \beta \rightarrow \gamma$ and $\mathcal{K} \models \alpha \vdash \beta$. We want to show that $\mathcal{K} \models \alpha \vdash \gamma$. Consider the 2 cases below:

Case 1: At some point ($i \in [0, n]$) in the $\mathcal{K} \models \alpha \vdash \beta$ checking, $R_\infty \cup R \not\models \neg\alpha$ for the first time. Then, at that point i , since $\mathcal{K} \models \alpha \vdash \beta$, we have that $R_\infty \cup R \models \alpha \rightarrow \beta$. When checking $\mathcal{K} \models \alpha \vdash \gamma$, the algorithm reaches that same point i , where $R_\infty \cup R \not\models \neg\alpha$ for the first time and then checks whether $R_\infty \cup R \models \alpha \rightarrow \gamma$.

Let τ be a model of $R_\infty \cup R$ and θ a substitution which replaces variables by constants. Suppose $\alpha\theta \in \tau$ then, since $R_\infty \cup R \models \alpha \rightarrow \beta$, we have that $\beta\theta \in \tau$. Since $\beta \rightarrow \gamma$ is a tautology, we must also have that $\gamma\theta \in \tau$. So $R_\infty \cup R \models \alpha \rightarrow \gamma$ and the algorithm returns **true**.

Case 2: It is always the case in the $\mathcal{K} \models \alpha \vdash \beta$ checking that $R_\infty \cup R \models \neg\alpha$. Then, in the $\mathcal{K} \models \alpha \vdash \gamma$ checking, it is also always the case that $R_\infty \cup R \models \neg\alpha$. So the algorithm returns **true**, by Lemma B.1. \square

B.4 And

Suppose $\mathcal{K} \models \alpha \vdash \beta$ and $\mathcal{K} \models \alpha \vdash \gamma$. We want to show that $\mathcal{K} \models \alpha \vdash \beta \wedge \gamma$. Consider the 2 cases below:

Case 1: At some point ($i \in [0, n]$) in the $\mathcal{K} \models \alpha \vdash \beta$ checking, $R_\infty \cup R \not\models \neg\alpha$ for the first time. Then, at the same point i in the $\mathcal{K} \models \alpha \vdash \gamma$ checking, $R_\infty \cup R \not\models \neg\alpha$ for the first time. Now, since $\mathcal{K} \models \alpha \vdash \beta$ and $\mathcal{K} \models \alpha \vdash \gamma$, at point i we have that $R_\infty \cup R \models \alpha \rightarrow \beta$ and $R_\infty \cup R \models \alpha \rightarrow \gamma$. So, at point i in the $\mathcal{K} \models \alpha \vdash \beta \wedge \gamma$ checking, $R_\infty \cup R \not\models \neg\alpha$ for the first time and the algorithm checks whether $R_\infty \cup R \models \alpha \rightarrow \beta \wedge \gamma$.

Let τ be a model of $R_\infty \cup R$ and θ a substitution which replaces variables by constants. Suppose $\alpha\theta \in \tau$ then, since $R_\infty \cup R \models \alpha \rightarrow \beta$ and $R_\infty \cup R \models \alpha \rightarrow \gamma$, we must have $\beta\theta \in \tau$ and $\gamma\theta \in \tau$. So $(\beta \wedge \gamma)\theta \in \tau$. Thus, $R_\infty \cup R \models \alpha \rightarrow \beta \wedge \gamma$ and the algorithm returns **true**.

Case 2: It is always the case in the $\mathcal{K} \models \alpha \vdash \beta$ checking that $R_\infty \cup R \models \neg\alpha$. Then, in the $\mathcal{K} \models \alpha \vdash \beta \wedge \gamma$ checking, it is also always the case that $R_\infty \cup R \models \neg\alpha$. So the algorithm returns **true**, by Lemma B.1. \square

B.5 Or

Suppose $\mathcal{K} \models \alpha \vdash \gamma$ and $\mathcal{K} \models \beta \vdash \gamma$. We want to show that $\mathcal{K} \models \alpha \vee \beta \vdash \gamma$. Consider the 2 cases below:

Case 1: It is always the case (for all $i \in [0, n]$) that in the $\mathcal{K} \models \alpha \vdash \gamma$ checking, $R_\infty \cup R \models \neg\alpha$ and, in the $\mathcal{K} \models \beta \vdash \gamma$ checking, $R_\infty \cup R \models \neg\beta$. Let τ be a model of $R_\infty \cup R$ at some point ($i \in [0, n]$) and θ a substitution which replaces variables by constants. Then, at point i , we must have that $\alpha\theta \notin \tau$ and $\beta\theta \notin \tau$ so $(\alpha \vee \beta)\theta \notin \tau$. Thus, $R_\infty \cup R \models \neg(\alpha \vee \beta)$ at point i . Hence, in the $\mathcal{K} \models \alpha \vee \beta \vdash \gamma$ checking, it is always the case that $R_\infty \cup R \models \neg(\alpha \vee \beta)$ so the algorithm returns **true**, by Lemma B.1.

Case 2: There is some point ($i \in [0, n]$) at which, without loss of generality, $R_\infty \cup R \not\models \neg\alpha$ for the first time and at each point before point i (for each $0 \leq j < i$), $R_\infty \cup R \models \neg\beta$. That is, $R_\infty \cup R \not\models \neg\alpha$ for the first time either at the same level or a higher level than the level at which $R_\infty \cup R \not\models \neg\beta$ for the first time. Since we know that $\mathcal{K} \models \alpha \vdash \gamma$, at point i we must have that $R \models \alpha \rightarrow \gamma$.

At point i , since $R_\infty \cup R \not\models \neg\alpha$, there is some model τ of $R_\infty \cup R$ which is not a model of $\neg\alpha$. Thus, there is some substitution θ such that $\alpha\theta \in \tau$. Thus, $(\alpha \vee \beta)\theta \in \tau$ so $(\neg(\alpha \vee \beta))\theta \notin \tau$. Hence, at point i in the $\mathcal{K} \models \alpha \vee \beta \vdash \gamma$ checking, $R_\infty \cup R \not\models \neg(\alpha \vee \beta)$.

Furthermore, at any point $j < i$, we have that $R_\infty \cup R \models \neg\alpha$ and $R_\infty \cup R \models \neg\beta$. Thus, as shown above in Case 1, we must have that $R_\infty \cup R \models \neg(\alpha \vee \beta)$ at point j . So point i is the first point at which $R_\infty \cup R \not\models \neg(\alpha \vee \beta)$.

We again let τ be a model of $R_\infty \cup R$ at point i and θ a substitution which replaces variables by constants. Now we consider 2 sub-cases below:

- i At point i , $R_\infty \cup R \models \neg\beta$. Then $\beta\theta \notin \tau$. Suppose that $\alpha\theta \notin \tau$. Then, $(\alpha \vee \beta)\theta \notin \tau$ so $\alpha \vee \beta \rightarrow \gamma$ is true under τ for substitution θ . Now suppose that $\alpha\theta \in \tau$. Then, $(\alpha \vee \beta)\theta \in \tau$ and, since $R \models \alpha \rightarrow \gamma$, $\gamma\theta \in \tau$. So, $\alpha \vee \beta \rightarrow \gamma$ is true under τ for substitution θ . Hence, $R \models \alpha \vee \beta \rightarrow \gamma$ and the algorithm returns **true**.
- ii At point i , $R_\infty \cup R \not\models \neg\beta$ (and this is not the case for any $j < i$, otherwise it would violate our assumption for case 2). So, since $\mathcal{K} \models \beta \vdash \gamma$, we have that $R_\infty \cup R \models \beta \rightarrow \gamma$. Suppose that $\alpha\theta \notin \tau$ and $\beta\theta \notin \tau$. Then, $(\alpha \vee \beta)\theta \notin \tau$ so $\alpha \vee \beta \rightarrow \gamma$ is true under τ for substitution θ . Now suppose that, without loss of generality (since both $R \models \alpha \rightarrow \gamma$ and $R \models \beta \rightarrow \gamma$), $\alpha\theta \in \tau$. Then, $(\alpha \vee \beta)\theta \in \tau$ and, since $R \models \alpha \rightarrow \gamma$, $\gamma\theta \in \tau$. So, $\alpha \vee \beta \rightarrow \gamma$ is true under τ for substitution θ . Hence, $R \models \alpha \vee \beta \rightarrow \gamma$ and the algorithm returns **true**. \square

B.6 CM

Suppose $\mathcal{K} \models \alpha \vdash \beta$ and $\mathcal{K} \models \alpha \vdash \gamma$. We want to show that $\mathcal{K} \models \alpha \wedge \beta \vdash \gamma$. We will make use of Lemma B.4 to do so.

LEMMA B.4. *Suppose $\mathcal{K} \models \alpha \vdash \beta$ and $\mathcal{K} \models \alpha \vdash \gamma$ for some knowledge base \mathcal{K} . Then, the following holds:*

- i If $R_\infty \cup R \models \neg\alpha$ at some point i in the RationalClosure algorithm, then $R_\infty \cup R \models \neg(\alpha \wedge \beta)$ at that point i .
- ii If $R_\infty \cup R \not\models \neg\alpha$ for the first time at some point i in the RationalClosure algorithm, then $R_\infty \cup R \not\models \neg(\alpha \wedge \beta)$, also for the first time, at that point i .

Proof of Lemma B.4:

- i Suppose that $R_\infty \cup R \models \neg\alpha$ at some point i . Let τ be a model of $R_\infty \cup R$ at point i and θ a substitution which replaces variables by constants. Then $\alpha\theta \notin \tau$ so $(\alpha \wedge \beta)\theta \notin \tau$ and, hence, $(\neg(\alpha \wedge \beta))\theta \in \tau$. Hence, $R_\infty \cup R \models \neg(\alpha \wedge \beta)$. \square
- ii Suppose that, at point i , $R_\infty \cup R \not\models \neg\alpha$ for the first time. Then, since $\mathcal{K} \models \alpha \vdash \beta$, we have that $R_\infty \cup R \models \alpha \rightarrow \beta$. And, since $R_\infty \cup R \not\models \neg\alpha$, there is some model τ of $R_\infty \cup R$ which is not a model of $\neg\alpha$. Thus, there is some substitution θ such that $\alpha\theta \in \tau$. Since $R_\infty \cup R \models \alpha \rightarrow \beta$, we must have that $\beta\theta \in \tau$ too. So $(\alpha \wedge \beta)\theta \in \tau$ and, thus, $(\neg(\alpha \wedge \beta))\theta \notin \tau$. Hence, at point i , $R_\infty \cup R \not\models \neg(\alpha \wedge \beta)$. Now, it remains to show that point i is the first point at which $R_\infty \cup R \not\models \neg(\alpha \wedge \beta)$. Assume, to the contrary, that at some point $j < i$, $R_\infty \cup R \not\models \neg(\alpha \wedge \beta)$. But, then at this point, we know $R_\infty \cup R \models \neg\alpha$, so $R_\infty \cup R \models \neg(\alpha \wedge \beta)$, which is a contradiction. Thus, point i is the first point at which $R_\infty \cup R \not\models \neg(\alpha \wedge \beta)$. \square

Now we consider 2 cases below:

Case 1: At some point ($i \in [0, n]$) in the $\mathcal{K} \models \alpha \vdash \beta$ checking, $R_\infty \cup R \not\models \neg\alpha$ for the first time. Then, at the same point i , in the $\mathcal{K} \models \alpha \vdash \gamma$ checking, $R_\infty \cup R \not\models \neg\alpha$ for the first time. Thus, at this point i we have that $R_\infty \cup R \models \alpha \rightarrow \beta$ and $R_\infty \cup R \models \alpha \rightarrow \gamma$. And, by Lemma B.4, at point i in the $\mathcal{K} \models \alpha \wedge \beta \vdash \gamma$ checking, $R_\infty \cup R \not\models \neg(\alpha \wedge \beta)$ for the first time.

Let τ be a model of $R_\infty \cup R$ at point i and θ a substitution which replaces variables by constants. Suppose that $\alpha\theta \notin \tau$. Then, $(\alpha \wedge \beta)\theta \notin \tau$ so $\alpha \wedge \beta \rightarrow \gamma$ is true under τ for substitution θ . Suppose now that $\alpha\theta \in \tau$ so, since $R_\infty \cup R \models \alpha \rightarrow \beta$ and $R_\infty \cup R \models \alpha \rightarrow \gamma$, we have that $\beta\theta \in \tau$ and $\gamma\theta \in \tau$. Thus, $(\alpha \wedge \beta)\theta \in \tau$ and $\gamma\theta \in \tau$ so $\alpha \wedge \beta \rightarrow \gamma$ is true under τ for substitution θ . Hence, $R_\infty \cup R \models \alpha \wedge \beta \rightarrow \gamma$ so the algorithm returns **true**.

Case 2: It is always the case in the $\mathcal{K} \approx \alpha \vdash \beta$ checking that $R_\infty \cup R \models \neg\alpha$. Then, by Lemma B.4, in the $\mathcal{K} \approx \alpha \wedge \beta \vdash \gamma$ checking, it is always the case that $R_\infty \cup R \models \neg(\alpha \wedge \beta)$ and so the algorithm returns **true**, by Lemma B.1. \square

B.7 RM

Suppose that $\mathcal{K} \approx \alpha \vdash \gamma$ and $\mathcal{K} \approx \alpha \vdash \neg\beta$. We want to show that $\mathcal{K} \approx \alpha \wedge \beta \vdash \gamma$. Consider the 2 cases below:

Case 1: At some point ($i \in [0, n]$) in the $\mathcal{K} \approx \alpha \wedge \beta \vdash \gamma$ checking, $R_\infty \cup R \not\models \neg(\alpha \wedge \beta)$. We claim that we must have that both $R_\infty \cup R \not\models \neg\alpha$ and $R_\infty \cup R \not\models \neg\beta$. Suppose, to the contrary, $R_\infty \cup R \models \neg\alpha$. Let τ be a model of $R_\infty \cup R$ at point i and θ a substitution which replaces variables by constants. Then $\alpha\theta \notin \tau$ so $(\alpha \wedge \beta)\theta \notin \tau$. Thus, $R_\infty \cup R \models \neg(\alpha \wedge \beta)$, a contradiction. Similarly, if $R_\infty \cup R \models \neg\beta$ then $R_\infty \cup R \models \neg(\alpha \wedge \beta)$, a contradiction.

Claim: Point i is the first point at which $R_\infty \cup R \not\models \neg\alpha$.

Proof of Claim: Assume to the contrary that there exists some $j < i$ such that $R_\infty \cup R \not\models \neg\alpha$, where j is minimal. Based on the assumptions of *Case 1*, we know that $R_\infty \cup R \models \neg(\alpha \wedge \beta)$ at point j . And, since $\mathcal{K} \approx \alpha \vdash \neg\beta$, we know that $R_\infty \cup R \not\models \alpha \rightarrow \neg\beta$ at point j . Let τ be a model of $R_\infty \cup R$ and θ a substitution that replaces variables with constants. Now, either $\alpha\theta \in \tau$ or $\alpha\theta \notin \tau$. We consider 2 sub-cases below:

- i If $\alpha\theta \notin \tau$. Then, $\alpha \rightarrow \neg\beta$ must be true under τ for θ .
- ii If $\alpha\theta \in \tau$. Then, we must have that $\beta\theta \notin \tau$. Otherwise, we would have $(\alpha \wedge \beta)\theta \in \tau$, and, hence, $R_\infty \cup R \not\models \neg(\alpha \wedge \beta)$, a contradiction. Thus, $\neg\beta\theta \in \tau$ and so $\alpha \rightarrow \neg\beta$ must be true under τ for θ .

Either way, $\alpha \rightarrow \neg\beta$ is true under τ for θ , so $R_\infty \cup R \models \alpha \rightarrow \neg\beta$, a contradiction. Thus, no such $j < i$ exists. \square

So, since $R_\infty \cup R \not\models \neg\alpha$ at point i (and not before) and $\mathcal{K} \approx \alpha \vdash \gamma$, we know that $R_\infty \cup R \models \alpha \rightarrow \gamma$ at this point. Suppose that at least one of $\alpha\theta \notin \tau$ or $\beta\theta \notin \tau$ holds. Then, $(\alpha \wedge \beta)\theta \notin \tau$ so $\alpha \wedge \beta \rightarrow \gamma$ is true under τ for substitution θ . Now suppose that both $\alpha\theta \in \tau$ and $\beta\theta \in \tau$. Then, $(\alpha \wedge \beta)\theta \in \tau$ and, since $R_\infty \cup R \models \alpha \rightarrow \gamma$, we know that $\gamma\theta \in \tau$ too. So $\alpha \wedge \beta \rightarrow \gamma$ is true under τ for substitution θ . Hence, $R_\infty \cup R \models \alpha \wedge \beta \rightarrow \gamma$ and the algorithm returns **true**.

Case 2: It is always the case in the $\mathcal{K} \approx \alpha \wedge \beta \vdash \gamma$ checking that $R_\infty \cup R \models \neg(\alpha \wedge \beta)$. Then, the algorithm returns **true**, by Lemma B.1. \square

C LEXICOGRAPHIC CLOSURE ALGORITHMS FOR DATALOG

Proof of Proposition 5.1: Let τ be a Herbrand interpretation and θ a substitution which replaces variables with constants. We want to show that τ is a model of $\neg\alpha$ under Datalog+ semantics iff τ is a model of $\alpha \rightarrow \perp$ under Datalog^V semantics.

Suppose τ is a model of $\neg\alpha$ under Datalog+ semantics. Then, $\alpha\theta \notin \tau$ under Datalog+ semantics. Clearly, we also have that $\alpha\theta \notin \tau$ under Datalog^V semantics. So, $\alpha \rightarrow \perp$ is true under τ for θ . Hence, τ is a model of $\alpha \rightarrow \perp$ under Datalog^V semantics.

Suppose τ is a model of $\alpha \rightarrow \perp$ under Datalog^V semantics. We claim that $\alpha\theta \notin \tau$ under Datalog^V semantics. Suppose, to the contrary, that $\alpha\theta \in \tau$. Notice that it is always the case that $\perp\theta \notin \tau$. Thus, $\alpha \rightarrow \perp$ is not true under τ for θ , contradicting the assumption that τ is a model of $\alpha \rightarrow \perp$. Thus, our claim holds - $\alpha\theta \notin \tau$ under Datalog^V semantics. Clearly, we also have that $\alpha\theta \notin \tau$ under Datalog+ semantics. Thus, $\neg\alpha$ is true under τ for θ . Hence, τ is a model of $\neg\alpha$ under Datalog+ semantics. \square

Algorithm 7: SubsetRank

Input: A knowledge base \mathcal{K}
Output: An ordered tuple $(R_0, \dots, R_k, R_\infty, k + 1)$

```

1  $(B_0, \dots, B_{m-1}, B_\infty, m) := \text{BaseRank}(\mathcal{K});$ 
2  $i := 0; k := 0;$ 
3 repeat
4   for  $j := |B_i|$  to 1 do
5      $S_{i,j} := \text{Subsets}(R_i, j);$ 
6      $D_{i,j} := \bigvee_{X \in S_{i,j}} \bigwedge_{x \in X} x;$ 
7      $R_k := \text{RNF}(D_{i,j});$ 
8      $k := k + 1;$ 
9    $i := i + 1;$ 
10 until  $i := m;$ 
11  $R_\infty := B_\infty;$ 
12 return  $(R_0, \dots, R_k, R_\infty, k + 1)$ 
```

Algorithm 8: LexicographicClosure

Input: A knowledge base \mathcal{K} and a defeasible rule $\alpha \sim \beta$
Output: **true**, if $\mathcal{K} \approx \alpha \vdash \beta$, and **false**, otherwise

```

1  $(R_0, \dots, R_k, R_\infty, k + 1) := \text{SubsetRank}(\mathcal{K});$ 
2  $i := 0;$ 
3  $R := \bigcup_{i=0}^k R_i;$ 
4 while  $R_\infty \cup R \models \alpha \rightarrow \perp$  and  $R \neq \emptyset$  do
5    $R := R \setminus R_i;$ 
6    $i := i + 1;$ 
7 return  $R_\infty \cup R \models \alpha \rightarrow \beta;$ 
```

D LM-RATIONALITY OF LEXICOGRAPHIC CLOSURE

In this section we provide proofs for the satisfaction of each KLM property by the LexicographicClosure procedure. That is, we prove Proposition 7.1, that LexicographicClosure is LM-rational.

Notice that the proofs for the satisfaction of each KLM property by the RationalClosure procedure, in Appendix B, are independent of the ranking produced by the BaseRank procedure. Furthermore, notice that the only difference between the LexicographicClosure procedure and the RationalClosure procedure is the use of the SubsetRank procedure to rank statements instead of the BaseRank procedure.

Thus, the proofs for the satisfaction of each KLM property in Appendix B can be used to prove for the satisfaction of each KLM property by the LexicographicClosure procedure.