SACJ 32(2) December 2020
Research Article

Algorithmic definitions for KLM-style defeasible disjunctive Datalog

Matthew Morris[a, b], Tala Ross[a], Thomas Meyer[a, b]

[a] Department of Computer Science, University of Cape Town, South Africa
[b] Centre for Artificial Intelligence Research, 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 for extending the KLM properties and a well-known form of defeasible entailment: Rational Closure. We conclude by exploring Datalog extensions of less conservative forms of defeasible entailment: Relevant and Lexicographic Closure. We provide algorithmic definitions for these forms of defeasible entailment and prove that the definitions are LM-rational.

Keywords: knowledge representation and reasoning, defeasible reasoning, KLM approach, Rational Closure, Relevant Closure, Datalog

Categories: • **Theory of computation ∼ Automated reasoning • Theory of computation ∼ Logic and databases • Computing methodologies ∼ Nonmonotonic, default reasoning and belief revision**

Email:
Matthew Morris matthewthemorris@gmail.com (CORRESPONDING),
Tala Ross rsstal002@myuct.ac.za,
Thomas Meyer tmeyer@cs.uct.ac.za

Article history:
Received: 30 May 2020
Accepted: 26 Aug 2020
Available online: 08 December 2020

1 INTRODUCTION

The KLM approach, proposed by Kraus et al. (1990), is a well-known framework for defeasible reasoning. The KLM properties can be used to determine the rationality of different forms of defeasible entailment. While rationality can often be understood intuitively, a common definition is required in order to avoid confusion between different conflicting or assumed definitions. Lehmann and Magidor's definition of rationality (1992) is used extensively in the literature and we use it throughout the paper. The framework has been discussed at length in the literature for propositional logic (Kraus et al., 1990; Lehmann, 1995; Lehmann & Magidor, 1992) and description logics (Casini et al., 2014; Casini et al., 2013; Moodley, 2015; Straccia

& Casini, 2013). We present what we believe to be the first theoretical approach for extending the KLM framework to Datalog. We consider an extended form of Datalog, Disjunctive Datalog, which allows for disjunction in the head of Datalog clauses. We do not consider a semantic characterisation for the Datalog case. Instead, we provide algorithmic definitions of defeasible entailment.

There are two well-known forms of defeasible entailment that satisfy the KLM properties: Rational Closure (RC) (Lehmann & Magidor, 1992) and Lexicographic Closure (LC) (Lehmann, 1995). Both are rational (Casini et al., 2019), with RC being the most conservative form of rational defeasible entailment among the popular proposals, and LC a more permissive form. Another form of defeasible entailment, Relevant Closure (RelC) (Casini et al., 2014), has been proposed for description logics. It intuitively seems rational but does not satisfy all of the KLM properties. We provide algorithmic definitions of RC, LC and RelC, showing that RC and LC are still rational when converted to Datalog and that RelC is not.

Our results demonstrate that the KLM approach can be extended to operate in Datalog and that rational definitions of defeasible entailment exist within the language. This shows promise for the extension of KLM-style defeasible entailment to other more expressive logics, as well as providing evidence for the possibility of a semantic definition of defeasible entailment in Datalog.

In the next section we provide the relevant background material, after which we present our work on KLM-style defeasible entailment for the Datalog case. We conclude with a discussion of related work and suggestions for future work. This work is an extension of that done by Morris et al. (2019).

2 BACKGROUND

2.1 Propositional Logic

Propositional logic Ben-Ari, 2012 is a simple logic which is built up from a finite set $\mathcal{P}$ of propositional *atoms*, denoted by meta-variables $p, q, \ldots$. The language $\mathcal{L}$ of propositional logic is the set of all formulas, denoted by $\alpha, \beta, \ldots$, which are recursively defined as usual: $\alpha ::= \top \mid \bot \mid p \mid \neg\alpha \mid \alpha \wedge \alpha \mid \alpha \vee \alpha \mid \alpha \rightarrow \alpha \mid \alpha \leftrightarrow \alpha$.

An *interpretation* is a function $I : \mathcal{P} \rightarrow \{T, F\}$ which assigns a single truth value to each atom. A formula $\alpha \in \mathcal{L}$ is satisfied by an interpretation I, denoted $I \Vdash \alpha$, if it can be evaluated to true by I in the usual recursive truth-functional way. We define the models of a finite set of formulas X to be $[\![X]\!] = \{I : I \Vdash \alpha, \alpha \in X\}$. We say that a set of formulas X entails a formula α, denoted by $X \models \alpha$, if $[\![X]\!] \subseteq [\![\{\alpha\}]\!]$.

2.2 The Failures of Monotonicity

Classical reasoning systems, such as propositional logic, 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 the following statements are made:

Example 1

1. *Adults are people ($a \to p$)*
2. *Students are people ($s \to p$)*
3. *People pay taxes ($p \to t$)*

From this, we can conclude that "*students 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, people pay taxes*". Then, when we add the extra information that students do not pay taxes, we want the system to retract its conclusion that "*students pay taxes*". However, a monotonic, classical reasoning system cannot change previous conclusions, and knowing that "*students do not pay taxes*" and "*students pay taxes*", it must then conclude that no students 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 (Casini et al., 2013).

2.3 KLM-style Defeasible Entailment

The *KLM approach* (Kraus et al., 1990) is based on the concept of plausible inference, which is represented by defeasible implication operators of the form $\alpha \mid\sim \beta$. This is read as "typically, if α, then β".

Let a *knowledge base* $\mathcal{K}$ be a finite set of defeasible implications. The KLM framework answers the question: "What does it mean for a defeasible implication $\alpha \mid\sim \beta$ to be entailed by a knowledge base $\mathcal{K}$?". This is referred to as defeasible entailment, and denoted by $\mathcal{K} \models\!\!\approx \alpha \mid\sim \beta$.

Unlike classical entailment, it is well-accepted that defeasible entailment is not unique. There are multiple formalizations of defeasible entailment, such as *Rational Closure* (Lehmann & Magidor, 1992), *Lexicographic Closure* (Lehmann, 1995), and *Relevant Closure* (Casini et al., 2014). Lehmann and Magidor (Lehmann & Magidor, 1992) proposed a set of rationality properties known as the *KLM properties*. They argue that if a defeasible entailment algorithm satisfies all the properties it is believed to be an acceptable form of defeasible entailment. We adopt this approach and refer to these forms of defeasible entailment as *LM-rational*. The KLM properties for propositional logic are stated below:

(Ref) $\mathcal{K} \models\!\!\approx \alpha \mid\sim \alpha$

(LLE) $\dfrac{\alpha \equiv \beta,\ \mathcal{K} \models\!\!\approx \alpha \mid\sim \gamma}{\mathcal{K} \models\!\!\approx \beta \mid\sim \gamma}$

(RW) $\dfrac{\mathcal{K} \models\!\!\approx \alpha \mid\sim \beta,\ \beta \models \gamma}{\mathcal{K} \models\!\!\approx \alpha \mid\sim \gamma}$

(And) $\dfrac{\mathcal{K} \models\!\!\approx \alpha \mid\sim \beta,\ \mathcal{K} \models\!\!\approx \alpha \mid\sim \gamma}{\mathcal{K} \models\!\!\approx \alpha \mid\sim \beta \wedge \gamma}$

(Or) $\dfrac{\mathcal{K} \models\!\!\approx \alpha \mid\sim \gamma,\ \mathcal{K} \models\!\!\approx \beta \mid\sim \gamma}{\mathcal{K} \models\!\!\approx \alpha \vee \beta \mid\sim \gamma}$

(CM) $\dfrac{\mathcal{K} \models\!\!\approx \alpha \mid\sim \beta,\ \mathcal{K} \models\!\!\approx \alpha \mid\sim \gamma}{\mathcal{K} \models\!\!\approx \alpha \wedge \beta \mid\sim \gamma}$

(RM) $\dfrac{\mathcal{K} \models\!\!\approx \alpha \mid\sim \gamma,\ \mathcal{K} \not\models\!\!\approx \alpha \mid\sim \neg\beta}{\mathcal{K} \models\!\!\approx \alpha \wedge \beta \mid\sim \gamma}$

All of these properties have a fairly intuitive meaning. Consider the two defeasible implications: *"typically, tutors are employees"* ($t \mid\!\sim e$) and *"typically, tutors are students"* ($t \mid\!\sim s$). It seems rational to conclude that *"typically, tutors are employees and students"* ($t \mid\!\sim e \wedge s$). This is exactly what the *And* property enforces. Kraus et al. (Kraus et al., 1990) provide detailed descriptions of the intended meaning of each property.

2.4 Rational Closure

Rational closure is the most conservative form of defeasible entailment. This means that anything entailed by rational closure will also be entailed by the other common forms of defeasible entailment. We use the algorithmic definition (Freund, 1998), which we refer to as the *Rational Closure Algorithm*, as the sole definition of Rational Closure.

Algorithm 1: `RationalClosureProp`

```
Input: A defeasible propositional knowledge base $\mathcal{K}$ and a defeasible implication
       $\alpha \mid\!\sim \beta$
Output: true, if $\mathcal{K} \mid\!\approx \alpha \mid\!\sim \beta$, and false, otherwise
1 $(R_0, \ldots, R_{n-1}, R_\infty, n) :=$ BaseRankProp($\mathcal{K}$);
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 algorithm is split into two distinct sub-algorithms, proposed by Casini et al. (Casini et al., 2019). The `BaseRankProp` algorithm, Algorithm 2, is used to construct a ranking of the classical versions ($\overrightarrow{\mathcal{K}}$) of the statements in the defeasible knowledge base ($\mathcal{K}$). For example, the classical version of defeasible statement $\alpha \mid\!\sim \beta$ would be $\alpha \rightarrow \beta$. Intuitively, the more "typical" statements are placed further down in the ranking, corresponding to the notion that some typical implications are *more typical* than others.

The `RationalClosureProp` algorithm, Algorithm 1, is used to compute whether a defeasible implication is entailed by the knowledge base and uses the `BaseRankProp` algorithm. Also note that the `RationalClosureProp` algorithm just reduces to a sequence of classical entailment checks.

We can express any classical sentence α as a defeasible implication $\neg\alpha \mid\!\sim \bot$ (Casini et al., 2019). For example, the rank of the statement $\beta \rightarrow \gamma$ corresponds to the rank of the statement $\beta \wedge \neg\gamma \mid\!\sim \bot$. This means that we can use the `BaseRankProp` algorithm to rank knowledge bases which include classical sentences and the `RationalClosureProp` algorithm can be used to check classical queries as well. Note that this also means that the `BaseRankProp` algorithm will put all classical sentences on the bottom (infinite) level.

Algorithm 2: `BaseRankProp`

```
Input: A defeasible propositional knowledge base $\mathcal{K}$
Output: An ordered tuple $(R_0, \ldots, R_{n-1}, R_\infty, n)$
1  $i := 0$;
2  $E_0 := \overrightarrow{\mathcal{K}} := \{\alpha \to \beta \mid \alpha \mid\!\sim \beta \in \mathcal{K}\}$;
3  repeat
4      $E_{i+1} := \{\alpha \to \beta \in E_i \mid E_i \models \neg\alpha\}$;
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 $(R_0, \ldots, R_{n-1}, R_\infty, n)$
```

To illustrate how the algorithm works, consider the following example knowledge base $\mathcal{K}$.

Example 2

1. *Adults are people* ($a \to p$)
2. *Students are people* ($s \to p$)
3. *Typically, people pay taxes* ($p \mid\!\sim t$)
4. *Typically, students do not pay taxes* ($s \mid\!\sim \neg t$)

Figure 1 shows the ranking of $\mathcal{K}$ according to the `BaseRankProp` algorithm. Throughout the paper, we illustrate the ranking of a knowledge base with all sentences in their original form for ease of understanding.

Suppose we asked; *"Do students typically pay taxes?"*, corresponding to the query $s \mid\!\sim t$. Then at the start of the algorithm when $i = 0$, $R_\infty \cup R \models \neg s$, since we have $s \to p \to t$ and $s \to \neg t$. So the top level (R_0) is thrown away. Now when $i = 1$, $R_\infty \cup R \not\models \neg s$. Also, $R_\infty \cup R \not\models s \to t$, so the algorithm returns **false** for the query. This makes sense, and the algorithm has computed the intuitively correct result.

0	$p \mathrel{|\!\sim} t$
1	$s \mathrel{|\!\sim} \neg t$
∞	$a \to p \quad s \to p$

Figure 1: Ranking of the Knowledge Base $\mathcal{K}$

2.5 Disjunctive Datalog

Datalog (Ceri et al., 1989) is a more expressive logic than propositional logic and a popular query language for deductive databases Pasarella and Lobo, 2017; Shkapsky et al., 2016. Datalog is a simplified version of general logic programming. 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 represent the statements in Example 3, whereas in propositional logic we can only represent tutors in general.

Example 3

1. *For all X, X is a tutor.*
2. *For all X, if X is an under-graduate, then X is a student.*
3. *Tyler is a tutor.*

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. Disjunctive Datalog (Datalog$^{\vee}$), as defined in this section, allows us to make the same statements as standard Datalog as well as statements involving disjunction and negation, such as the following:

Example 4

4. *For all students X, X is an undergraduate <u>or</u> a postgraduate.*
5. *Tyler is not a lecturer.*

The language of Disjunctive Datalog is made up of function-free *Horn clauses* (Ceri et al., 1989), which are formulas with the general structure: $l_0 \wedge l_1 \wedge \cdots \wedge l_m \to l_{m+1} \vee l_{m+2} \vee \cdots \vee l_n$. Each literal l_i is either $\perp$ or is a positive atom of the form $p_i(t_0, \ldots, t_{k_i})$, where p_i is a *predicate symbol* and $t_0, \ldots, t_{k_i}$ are *terms*. A term is either a *constant* or a *variable*. In our version of Datalog, the left-hand side of the clause is referred to as the *body* and the right-hand side as the *head*. Horn clauses with a body are called *rules* and those without a body are called *facts*.

We can represent the statements from Example 3 and 4, using variable X, constant Tyler and predicates s, t, u, p and l which represent students, tutors, under-graduates, post-graduates and lecturers respectively:

1. $t(X)$
2. $u(X) \to s(X)$
3. $t(\text{Tyler})$
4. $s(X) \to u(X) \vee p(X)$
5. $l(\text{Tyler}) \to \bot$

We say that clauses, such as $t(\text{Tyler})$, are *ground* since they do not contain any variables. A *Herbrand Base* B^P is the set of all ground facts constructible from the symbols in a Datalog program P. For example; Tyler is the only constant in the program P, defined in Example 3 and 4, so the set of all possible ground facts that we can form is:

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

A *Herbrand interpretation* τ is simply a subset of a Herbrand Base: $\tau \subseteq B^P$. For any Herbrand interpretation τ, we define that $\bot$ is not in τ. So, following our example, a possible Herbrand interpretation is:

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

A rule $l_0 \wedge l_1 \wedge \cdots \wedge l_m \to l_{m+1} \vee l_{m+2} \vee \cdots \vee l_n$ is true for Herbrand interpretation τ if and only if, for each substitution θ which replaces variables by constants, if $l_0\theta \in \tau, l_1\theta \in \tau, \ldots, l_m\theta \in \tau$ then at least one of $l_{m+1}\theta \in \tau, l_{m+2}\theta \in \tau, \ldots, l_n\theta \in \tau$ holds. For example; $t(X) \wedge p(X) \to l(X)$ is not true for τ, since $t(\text{Tyler}) \in \tau$ and $p(\text{Tyler}) \in \tau$ but $l(\text{Tyler}) \notin \tau$. A fact $l_0 \wedge l_1 \wedge \cdots \wedge l_m$ is true for Herbrand interpretation τ if and only if, for each substitution θ which replaces variables by constants, $l_0\theta \in \tau, l_1\theta \in \tau, \ldots, l_m\theta \in \tau$ all hold. For example; $s(X)$ is true for τ but $u(X)$ is not. A Herbrand interpretation τ is a *Herbrand model* of a set of Horn clauses X if and only if every clause in X is true for τ. For example; τ is a Herbrand model of P.

2.5.1 Entailment of Horn Clauses

Entailment is defined in the standard way: a set of Horn clauses X entails Horn clause α, denoted by $X \models \alpha$, if and only if each Herbrand model of X is also a model of α.

2.5.2 Molecules as Combinations of Literals

We introduce the idea of *molecules* as a shorthand for a combination of literals. A *disjunctive molecule*, denoted $\alpha^\vee$, is a combination of literals of the form: $l_1 \vee l_2 \vee \cdots \vee l_n$. A *conjunctive molecule*, denoted $\alpha^\wedge$, is a combination of literals of the form: $l_1 \wedge l_2 \wedge \cdots \wedge l_n$. A *molecule*, denoted α, is either a disjunctive molecule or a conjunctive molecule. Now a Disjunctive Datalog rule can be written as $\alpha^\wedge \to \beta^\vee$.

3 DEFEASIBLE DISJUNCTIVE DATALOG

3.1 KLM-style Defeasible Rules

We represent plausible inference in Disjunctive Datalog using defeasible rules of the form: $b_1 \wedge \cdots \wedge b_m \mid\!\sim h_1 \vee \cdots \vee h_n$, where each b_i, h_i is a literal. This is read as "typically, if all of $b_1, \ldots, b_m$ are true, then at least one of $h_1, \ldots, h_n$ is true". We do not consider a semantic definition of defeasible rules. We will instead define defeasible rules by adapting rational defeasible entailment algorithms for Disjunctive Datalog.

3.2 Defeasible Entailment

Let knowledge base $\mathcal{K}$ be a finite set of defeasible rules. The main question of this paper is to algorithmically analyse *defeasible entailment* $\mathcal{K} \mid\!\approx \alpha^{\wedge} \mid\!\sim \beta^{\vee}$. That is, how do we answer the question: "*Can we conclude* $\alpha^{\wedge} \mid\!\sim \beta^{\vee}$ *from a defeasible knowledge base* K*?*". When analysing different defeasible entailment algorithms, Lehmann and Magidor (Lehmann & Magidor, 1992) advocate that the KLM properties be used to assess the rationality of these algorithms. We adopt this approach for Datalog and provide an extension of the KLM properties for Disjunctive Datalog.

3.2.1 A Motivation for Extending Disjunctive Datalog

We find that, due to the restrictive nature of Datalog's syntax, none of the KLM properties can be expressed using Disjunctive Datalog without violating the syntax. However, we need to ensure that LM-rational forms of defeasible entailment satisfy all of 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 Disjunctive Datalog.

Let us consider an example where we can come to a conclusion that cannot be expressed in Datalog's syntax. Even though we cannot express that conclusion, we still want the algorithm to be able to compute it, otherwise the algorithm would not be *rational*. For example, if tutor$(X) \mid\!\sim$ student(X) and tutor$(X) \mid\!\sim$ employee(X) both hold, then we would want to be able to conclude tutor$(X) \mid\!\sim$ student$(X) \wedge$ employee(X) holds as well.

As another example, if tutor$(X) \mid\!\sim$ teacher(X) and lecturer$(X) \mid\!\sim$ teacher(X) both hold, then we would want to be able to conclude tutor$(X) \ \vee$ lecturer$(X) \mid\!\sim$ teacher(X) holds as well. These examples illustrate the KLM properties of *And* and *Or* respectively. Note that in the conclusions of these examples, the syntax of Datalog, restricting disjunction to only be allowed in the head and conjunction in the body, is violated.

3.2.2 Datalog+

Our proposed extension to Datalog, Datalog +, introduces the idea of compounds. Compounds, denoted by $A, B, \ldots$, are recursively defined from base literals l as follows: $A ::= l \mid \neg A \mid A \wedge A \mid A \vee A$. In Datalog + a fact is a compound A and rules have the form $A \rightarrow B$.

Let τ be a Herbrand interpretation and consider some substitution θ which replaces variables by constants. We say that compound A is in τ under θ, 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 $l\theta \in \tau$
- $A = \neg B$ and $B\theta \notin \tau$
- $A = B \wedge \Gamma$, $B\theta \in \tau$ and $\Gamma\theta \in \tau$
- $A = B \vee \Gamma$ and $B\theta \in \tau$ or $\Gamma\theta \in \tau$

Herbrand interpretation τ is a model of fact A if and only if $A\theta \in \tau$ for every possible θ. Herbrand interpretation τ is a model of rule $A \to B$ if and only if, whenever $A\theta \in \tau$ for some θ, then $B\theta \in \tau$ for the same θ. A knowledge base $\mathcal{K}$ entails Datalog+ Horn clause (rule or fact) α, denoted by $\mathcal{K} \models \alpha$, if and only if each Herbrand model of $\mathcal{K}$ is also a model of α.

3.2.3 The KLM Properties Expressed in Datalog+

We state the KLM properties (in Defeasible Datalog+) for Datalog below, where compounds α, β, γ are used as a shorthand. Note that the defeasible implication operator $\mid\!\sim$ has been added to the language of Datalog+. These properties are used to define the rationality of a defeasible entailment algorithm for Datalog. Only an algorithm which satisfies all of these properties is considered rational and acceptable.

$$\text{(Ref)}\ \mathcal{K} \approx\!\!\!\!| \ \alpha \mid\!\sim \alpha \qquad \text{(RW)}\ \frac{\mathcal{K} \models\!\!\!\approx \alpha \mid\!\sim \beta,\ \models \beta \to \gamma}{\mathcal{K} \models\!\!\!\approx \alpha \mid\!\sim \gamma} \qquad \text{(And)}\ \frac{\mathcal{K} \models\!\!\!\approx \alpha \mid\!\sim \beta,\ \mathcal{K} \models\!\!\!\approx \alpha \mid\!\sim \gamma}{\mathcal{K} \models\!\!\!\approx \alpha \mid\!\sim \beta \wedge \gamma}$$

$$\text{(Or)}\ \frac{\mathcal{K} \models\!\!\!\approx \alpha \mid\!\sim \gamma,\ \mathcal{K} \models\!\!\!\approx \beta \mid\!\sim \gamma}{\mathcal{K} \models\!\!\!\approx \alpha \vee \beta \mid\!\sim \gamma} \qquad \text{(CM)}\ \frac{\mathcal{K} \models\!\!\!\approx \alpha \mid\!\sim \beta,\ \mathcal{K} \models\!\!\!\approx \alpha \mid\!\sim \gamma}{\mathcal{K} \models\!\!\!\approx \alpha \wedge \beta \mid\!\sim \gamma} \qquad \text{(RM)}\ \frac{\mathcal{K} \models\!\!\!\approx \alpha \mid\!\sim \gamma,\ \mathcal{K} \not\models\!\!\!\approx \alpha \mid\!\sim \neg\beta}{\mathcal{K} \models\!\!\!\approx \alpha \wedge \beta \mid\!\sim \gamma}$$

$$\text{(LLE)}\ \frac{\models \alpha \to \beta, \models \beta \to \alpha,\ \mathcal{K} \models\!\!\!\approx \alpha \mid\!\sim \gamma}{\mathcal{K} \models\!\!\!\approx \beta \mid\!\sim \gamma}$$

4 RATIONAL CLOSURE FOR DATALOG

In this section we propose a simple adaptation to the Rational Closure algorithms for the Disjunctive Datalog case.

4.1 Base Rank Algorithm

In the propositional case, we can rewrite a classical statement α as the defeasible statement $\neg\alpha \mid\!\sim \bot$ and, hence, we can assume that all of the statements in our knowledge base are defeasible. It is not possible to rewrite classical clauses as defeasible rules for the Datalog case. Instead, the adapted version of the `BaseRankProp` algorithm, `BaseRankDatalog`, 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 forms a ranking using only the defeasible statements by setting $E_0 := \overrightarrow{D}$ on *line 2*. Then, since the classical statements are all definite, it adds them to the most typical level (the infinite level).

In the propositional case, a statement α is *exceptional* with respect to a set of statements X if $X \models \neg\alpha$. Datalog$^{\vee}$'s syntax does not include the negation connective $\neg$, so we use the $\bot$ literal to define a notion of falsehood, and hence exceptionality.

Proposition 1 *Let τ be a Herbrand interpretation. Then, τ is a model of $\neg\alpha$ under Datalog+ semantics iff τ is a model of $\alpha \to \bot$ under Datalog$^{\vee}$ semantics.*

The exceptionality of molecule α is now assessed using the entailment check $E_i \cup C \models \alpha \to \bot$ on *line 4*. Finally, when all the defeasible rules are ranked, `BaseRankDatalog` adds the classical clauses to the infinite level by setting $R_\infty := E_{i-1} \cup C$ on *line 8*.

Algorithm 3: `BaseRankDatalog`

```
Input: A defeasible Datalog knowledge base $D$ and classical Datalog knowledge base $C$
Output: An ordered tuple $(R_0, \ldots, R_{n-1}, R_\infty, n)$
1  $i := 0$;
2  $E_0 := \overrightarrow{D}$;
3  repeat
4      $E_{i+1} := \{\alpha \to \beta \in E_i \mid E_i \cup C \models \alpha \to \bot\}$;
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 $(R_0, \ldots, R_{n-1}, R_\infty, n)$
```

4.2 Rational Closure Algorithm

As with the `BaseRankDatalog` algorithm, we choose to represent falsehood using the $\bot$ literal. The `RationalClosureDatalog` algorithm now uses the entailment check $R_\infty \cup R \models \alpha \to \bot$

on *line 4*. Under the assumption that we can compute classical entailment for Datalog$^{\vee}$, this adapted version of the `RationalClosureProp` algorithm can be used to check whether a rule $\alpha \mid\!\sim \beta$ is defeasibly entailed by the knowledge base $\mathcal{K} = D \cup C$.

Algorithm 4: `RationalClosureDatalog`

```
Input: A defeasible Datalog knowledge base $D$, a classical Datalog knowledge base $C$
       and a defeasible rule $\alpha \mid\!\sim \beta$
Output: true, if $\mathcal{K} \mid\!\approx \alpha \mid\!\sim \beta$, and false, otherwise
1 $(R_0, \ldots, R_{n-1}, R_\infty, n) :=$ BaseRankDatalog$(D, C)$;
2 $i := 0$;
3 $R := \bigcup_{i=0}^{j<n} R_j$;
4 while $R_\infty \cup R \models \alpha \to \bot$ and $R \neq \emptyset$ do
5     $R := R \setminus R_i$;
6     $i := i + 1$;
7 return $R_\infty \cup R \models \alpha \to \beta$;
```

Proposition 2 *The adapted* `RationalClosureDatalog` *algorithm is LM-rational.*

The proof of LM-rationality is provided in Appendix A.

4.3 Interactions Between Variables and Constants

While it is certainly important that `RationalClosureDatalog` is LM-rational, the KLM properties do not capture the full expressivity of Datalog. In particular, they do not express the interactions between constants and variables in rules. While a set of properties that fully express how defeasibilty should work with respect to this is certainly useful, it is beyond the scope of this paper. Instead, we provide an example that illustrates the basics of how `RationalClosureDatalog` fails to deal correctly with rules involving constants and variables.

Example 5 *Consider the following knowledge base $\mathcal{K}$:*

1. *Birds typically fly :* $b(X) \mid\!\sim f(X)$
2. *Tweety is a non-flying bird :* $b(t) \wedge \neg f(t)$
3. *Chirpy is a bird :* $b(c)$

When ranked according to the `BaseRankDatalog` algorithm, we would want the statements to appear as shown in Figure 2. Classical statements should have infinite rank and the defeasible statement should have a finite rank. However, instead of this, all statements get ranked on the same level, as this is in fact an inconsistent knowledge base. To see this, notice that the classical version of $b(X) \mid\!\sim f(X)$ is $b(X) \rightarrow f(X)$. We also know $b(t)$ is true, so we can use this to conclude that $f(t)$ must be in any Herbrand interpretation satisfying our knowledge base. But we know that $\neg f(t)$ must also be true, so $f(t)$ is not in any such Herbrand interpretation, a contradiction.

0	$b(X) \mid\!\sim f(X)$
∞	$b(t) \wedge \neg f(t) \quad b(c)$

Figure 2: Desired Ranking of the Knowledge Base $\mathcal{K}$

5 RELEVANT CLOSURE FOR DATALOG

It seems unnecessary for the Rational Closure algorithm to throw away an entire level of statements when there is a conflict. While it is true that a statement within the level is causing the conflict, there are other statements in the level that may have no effect on the conflict occurring. *Relevant closure* takes a finer-grained approach to removing statements, only removing the "relevant" statements in a level. In this section we give the definition for Relevant Closure as provided by Casini et al. (Casini et al., 2014). The algorithm is based on `RationalClosureProp`, with some slight changes.

5.1 Motivation for Relevant Closure

In this subsection we argue, by means of an example, that not all statements in a level are responsible for being able to prove $R_\infty \cup R \models \neg\alpha$, given the query $\alpha \mid\!\sim \beta$.

Example 6

1. *Adults are people ($a \rightarrow p$)*
2. *Students are people ($s \rightarrow p$)*
3. *Typically, people watch movies ($p \mid\!\sim m$)*
4. *Typically, people are adults ($p \mid\!\sim a$)*
5. *Typically, people pay taxes ($p \mid\!\sim t$)*
6. *Typically, students do not pay taxes ($s \mid\!\sim \neg t$)*

Figure 3 shows the ranking of these statements according to the `BaseRankProp` algorithm.

Let us now consider what happens when we ask the question; *"Do students typically watch movies?"*, corresponding to the query $s \mid\!\sim m$. In the algorithm, when $i = 0$, we can conclude that $s \rightarrow t$ and $s \rightarrow \neg t$. That is, students both pay and don't pay taxes, leading us to conclude that there are no students. Thus, $R_\infty \cup R \models \neg s$, so we throw away the entire top level of the ranking and check again.

0	$p \mid\!\sim m \quad p \mid\!\sim a \quad p \mid\!\sim t$
1	$s \mid\!\sim \neg t$
∞	$a \rightarrow p \quad s \rightarrow p$

Figure 3: Ranking of the Knowledge Base $\mathcal{K}$

Now, $R_\infty \cup R \not\models \neg s$, so we check if $s \rightarrow m$ holds. It does not, since the statement $p \rightarrow m$ was thrown away in the previous iteration. Thus, the algorithm returns **false**. This intuitively feels wrong, since the conclusion *"Students typically watch movies"* seems like a very reasonable one to make from the given information. The issue arises from throwing away the statement $p \rightarrow m$ in the previous iteration, even though it had nothing to do with us being able to conclude that there were no students.

We used the following statements to conclude $\neg s$;

$$s \rightarrow \neg t \qquad\qquad s \rightarrow p \qquad\qquad p \rightarrow t$$

One could argue that since the statement $p \rightarrow m$ was not relevant to us concluding $\neg s$, it should not have been thrown away with the top level. This is the idea behind Relevant Closure.

5.2 Algorithmic Definition

Algorithm 5: `RelevantClosureProp`

```
Input: A defeasible propositional knowledge base $\mathcal{K}$, a defeasible implication $\alpha \mid\!\sim \beta$,
       and a partition $< R,\ R^- >$ of $\mathcal{K}$
Output: true, if $\mathcal{K} \mid\!\approx \alpha \mid\!\sim \beta$, and false, otherwise
1 $(R_0, \ldots, R_{n-1}, R_\infty, n) :=$ BaseRankProp($\mathcal{K}$);
2 $i := 0$;
3 $R' := R$;
4 while $R_\infty \cup R^- \cup R' \models \neg\alpha$ and $R' \neq \emptyset$ do
5     $R' := R' \setminus \{R_i \cap R\}$;
6     $i := i + 1$;
7 return $R_\infty \cup R^- \cup R' \models \alpha \rightarrow \beta$;
```

The algorithm for Relevant Closure, provided by Casini et al. (Casini et al., 2014), is defined in terms of $\mathcal{ALC}$, a description logic. To make the algorithm easier to understand and convert to Datalog, we will first express it in terms of propositional logic.

In the partition $< R,\ R^- >$ of $\mathcal{K}$, R represents all statements *relevant* to the query $\alpha \mid\!\sim \beta$. When throwing away statements from a level, the algorithm only considers these statements in R as eligible for removal. We say that a statement $\alpha \mid\!\sim \beta$ is in the Relevant Closure of $\mathcal{K}$ if and only if the `RelevantClosureProp` algorithm returns **true** when given $\alpha \mid\!\sim \beta$ and $\mathcal{K}$.

5.3 Defining Relevance

Now that the algorithm has been defined, the only work remaining is to define how to calculate the partition $< R, R^- >$ for a given query $\alpha \mid\!\sim \beta$. Based on the ideas explored by Casini et al. (Casini et al., 2014), we would want R to contain exactly all the statements used to prove $\neg\alpha$. To formalize this, we present a sequence of definitions to gradually build up the idea of relevance.

Definition 1 *α is said to be* exceptional *for $\mathcal{K}$ if $\mathcal{K} \models \neg\alpha$.*

Definition 2 *Let $\mathcal{K}$ be a knowledge base, $\mathcal{J} \subseteq \mathcal{K}$ such that $\mathcal{J}$ only contains defeasible implications, and α a propositional sentence. Then $\mathcal{J}$ is said to be an α-*justification *w.r.t. $\mathcal{K}$ if α is exceptional for $\mathcal{J}$ and for any $\mathcal{J}' \subset \mathcal{J}$, α is not exceptional for $\mathcal{J}'$.*

Definition 3 *For a sentence α and knowledge base $\mathcal{K}$, let $\mathcal{J}^{\mathcal{K}}(\alpha) = \{\mathcal{J} \mid \mathcal{J}$ is an α-justification w.r.t. $\mathcal{K}\}$. Then $\alpha \mid\!\sim \beta$ is said to be in the* Basic Relevant Closure *of $\mathcal{K}$ if it is in the Relevant Closure of $\mathcal{K}$ w.r.t. $\bigcup \mathcal{J}^{\mathcal{K}}(\alpha)$.*

5.4 Minimal Relevant Closure

It could be argued that for Basic Relevant Closure, we are still considering too many statements as relevant to the query. This is because we consider *all* the statements in *all* α-justifications as relevant to proving that α is exceptional. However, we could instead consider only the statements of minimal rank from each α-justification as relevant, and still fix the exceptionality of α.

Definition 4 *For some set of justifications $\mathcal{J} \subseteq \mathcal{K}$, let $\mathcal{J}^{\mathcal{K}}_{min} = \{\alpha \mid\!\sim \beta \mid r_{\mathcal{K}}(\alpha) \leq r_{\mathcal{K}}(\gamma)$ for every $\gamma \mid\!\sim \lambda \in \mathcal{J}\}$.*

For a sentence α, let $\mathcal{J}^{\mathcal{K}}_{min}(\alpha) = \bigcup_{\mathcal{J} \in \mathcal{J}^{\mathcal{K}}(\alpha)} \mathcal{J}^{\mathcal{K}}_{min}$.

Then $\alpha \mid\!\sim \beta$ is said to be in the Minimal Relevant Closure *of $\mathcal{K}$ if it is in the Relevant Closure of $\mathcal{K}$ w.r.t. $\bigcup \mathcal{J}^{\mathcal{K}}_{min}(\alpha)$.*

5.5 Relevant Closure for Datalog

In terms of adapting the `RelevantClosureProp` algorithm for Datalog, no further work needs to be done beyond what has already been said for Rational Closure. To define a molecule α being *exceptional*, we simply need to be able to check entailment of negated molecules, which is something we already know how to do. The remainder of the definitions for both Basic and Minimal Relevant Closure only entail manipulating sets and checking the rankings of statements.

Algorithm 6: `RelevantClosureDatalog`

```
Input: A defeasible Datalog knowledge base $\mathcal{K}$, a defeasible Datalog rule $\alpha \mid\!\sim \beta$, and a
       partition $< R,\ R^{-} >$ of $\mathcal{K}$
Output: true, if $\mathcal{K} \approx \alpha \mid\!\sim \beta$, and false, otherwise
1 $(R_0, \ldots, R_{n-1}, R_\infty, n)$ := BaseRankDatalog($\mathcal{K}$);
2 $i := 0$;
3 $R' := R$;
4 while $R_\infty \cup R^{-} \cup R' \models \neg\alpha$ and $R' \neq \emptyset$ do
5     $R' := R' \setminus \{R_i \cap R\}$;
6     $i := i + 1$;
7 return $R_\infty \cup R^{-} \cup R' \models \alpha \rightarrow \beta$;
```

5.6 LM-Rationality

For this, we will use Minimal Relevant Closure as the definition for Relevant Closure. As shown by Casini et al. (Casini et al., 2014), Relevant Closure for propositional logic satisfies the properties *Ref*, *LLE*, *And*, and *RW*, and does not satisfy *Or*, *CM*, or *RM*. We will show that the same holds true for Relevant Closure for Datalog.

Let us consider the proofs that show that Rational Closure fulfills the KLM properties of *Ref*, *RW*, and *And*. The only difference `RelevantClosureDatalog` has from `RationalClosureDatalog` is the inclusion of the "relevance partition". Thus, the proofs can be re-used without editing, provided that the relevance partition is the same throughout the various queries.

The relevance partition is fully determined by the antecedent of the query (e.g. α in $\alpha \mid\!\sim \beta$), as can be seen in the definition of Minimal Relevant Closure. In the aforementioned properties, the antecedent is the same in all queries made to the algorithm. Hence, the proofs can be directly re-used to show that Relevant Closure fulfills the KLM properties of *Ref*, *RW*, and *And*.

The proof for satisfaction of the property *LLE* and the counter-examples for satisfaction of the properties *Or*, *CM*, and *RM* can be found in Appendix C. The counter-examples were adapted from the $\mathcal{ALC}$ case (Casini et al., 2014).

6 LEXICOGRAPHIC CLOSURE

In Section 5 we argue that it is unnecessary for the Rational Closure algorithm to throw away an entire level of statements when there is a conflict, since some of the statements may have no effect on the conflict occurring. Unfortunately the *Relevant closure* definition of defeasible entailment is not *LM-rational*. Instead of attempting to define "relevant" statements, *Lexicographic closure* (Lehmann, 1995) considers all possible subsets of worst-ranked statements and removes the smallest possible subset such that there is no longer a conflict. The semantic and algorithmic definitions of Lexicographic Closure for propositional logic are known and have been shown to be LM-rational (Lehmann, 1995). In this section we provide an extension of Lexicographic Closure to the Datalog$^{\vee}$ case.

6.1 Lexicographic Closure for Propositional Logic

We adapt the definition of Lexicographic Closure for propositional logic provided by Casini et al.(Casini et al., 2019). The new definition, in terms of the sub-algorithms `SubsetRankProp` and `LexicographicClosureProp`, can easily be adapted for Datalog$^{\vee}$.

The `SubsetRankProp` algorithm, Algorithm 7, constructs a new ranking of statements by using the base ranks $\mathcal{R}_0, \ldots, \mathcal{R}_{n-1}, \mathcal{R}_\infty$ computed by the `BaseRankProp` algorithm. It adds new rank levels D_{i,n_i-1}, D_{i,n_i-2},..., $D_{i,1}$ in between each existing rank level $\mathcal{R}_i$ and $\mathcal{R}_{i+1}$. Each level $D_{i,j}$ represents all the different ways of removing $|\mathcal{R}_i| - j$ statements from $\mathcal{R}_i$. The `Subsets`(X, k) function finds all possible subsets of size $k < n$ of a set X of size n.

Algorithm 7: `SubsetRankProp`

```
Input: A defeasible propositional knowledge base $\mathcal{K}$
Output: An ordered tuple $(\mathcal{R}_0, \ldots, \mathcal{R}_k, \mathcal{R}_\infty, k+1)$
1  $(B_0, \ldots, B_{m-1}, B_\infty, m) :=$ BaseRankProp$(\mathcal{K})$;
2  $i := 0$; $k := 0$;
3  repeat
4      for $j := |B_i|$ to $1$ do
5          $S_{i,j} :=$ Subsets$(B_i, j)$;
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 $\mathcal{R}_\infty := B_\infty$;
12 return $(\mathcal{R}_0, \ldots, \mathcal{R}_k, \mathcal{R}_\infty, k+1)$
```

As seen in Section 5, when the knowledge base in Example 6 is ranked according to the `BaseRankProp` algorithm, the statements appear in the ranking shown in Figure 4.

B_0	$p \to m \quad p \to a \quad p \to t$
B_1	$s \to \neg t$
B_∞	$a \to p \quad s \to p$

Figure 4: Base Ranking of the Knowledge Base $\mathcal{K}$

Let us look at B_0. Finding all the different ways of removing 1 statement from B_0, is the equivalent of finding all the subsets of B_0 of size 2 using $S_{0,2} = \texttt{Subsets}(B_0, 2) = \{\{p \rightarrow m, p \rightarrow a\}, \{p \rightarrow m, p \rightarrow t\}, \{p \rightarrow a, p \rightarrow t\}\}$. Now, using $\{p \rightarrow m, p \rightarrow a\}$ is equivalent to using the single statement $(p \rightarrow m) \wedge (p \rightarrow a)$. And, using at least one of the statements $(p \rightarrow m) \wedge (p \rightarrow a)$, $(p \rightarrow m) \wedge (p \rightarrow t)$ or $(p \rightarrow a) \wedge (p \rightarrow t)$ is equivalent to using the single statement $((p \rightarrow m) \wedge (p \rightarrow a)) \vee ((p \rightarrow m) \wedge (p \rightarrow t)) \vee ((p \rightarrow a) \wedge (p \rightarrow t))$. So, all the different ways of removing 1 statement from B_0 can be represented by this one statement. If we follow the algorithm for all possible subset sizes (from $|B_i|$ to 1) for each B_i, then we get the following ranking:

$R_0 = D_{0,3}$	$(p \rightarrow m) \wedge (p \rightarrow a) \wedge (p \rightarrow t)$
$R_1 = D_{0,2}$	$((p \rightarrow m) \wedge (p \rightarrow a)) \vee ((p \rightarrow m) \wedge (p \rightarrow t)) \vee ((p \rightarrow a) \wedge (p \rightarrow t))$
$R_2 = D_{0,1}$	$(p \rightarrow m) \vee (p \rightarrow a) \vee (p \rightarrow t)$
$R_3 = D_{1,1}$	$s \rightarrow \neg t$
B_∞	$a \rightarrow p \quad s \rightarrow p$

Figure 5: SubsetRanking of the Knowledge Base $\mathcal{K}$

The `LexicographicClosureProp` algorithm ranks the statements in the input knowledge base $\mathcal{K}$ using the `SubsetRankProp` algorithm. It then checks whether the defeasible implication $\alpha \mid\!\sim \beta$ is defeasibly entailed by $\mathcal{K}$ in a manner equivalent to that used by the `RationalClosureProp` algorithm (`LexicographicClosureProp` is the same as `RationalClosureProp`, barring the use of `SubsetRankProp` instead of `BaseRankProp`.)

Let us again consider what happens when we ask the question; *"Do students typically watch movies?"* ($s \mid\!\sim m$). In the algorithm, when $i = 0$, we can conclude that $s \rightarrow t$ and $s \rightarrow \neg t$ so we throw away R_0 and check again. Now, for $i = 1$, $\boldsymbol{R}_\infty \cup \boldsymbol{R} \not\models \neg s$, so we check if $s \rightarrow m$ holds and it does. Thus, the algorithm returns **true**.

6.2 Lexicographic Closure for Datalog

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

6.2.1 Rephrasing SubsetRank for Datalog

The definition of Lexicographic Closure for the propositional case cannot directly be applied to the Datalog case. The statement $D_{i,j}$ is formed by combining statements from subset $S_{i,j}$ using $\wedge$ and $\vee$ connectives. It will violate Datalog$^\vee$'s syntax if $S_{i,j}$ contains multiple rules or multiple subsets of facts. However, the statement $D_{i,j}$ can be transformed into Conjunctive

Normal Form (CNF) $D_{i,j} := D_1 \wedge D_2 \wedge \ldots \wedge D_n$, where:

$$\begin{aligned} D_i &:= \neg a_{i,1} \vee \ldots \vee \neg a_{i,r_i} \vee b_{i,1} \vee \ldots \vee b_{i,s_i} \\ &:= \neg(a_{i,1} \wedge \ldots \wedge a_{i,r_i}) \vee (b_{i,1} \vee \ldots \vee b_{i,s_i}) \\ &:= a_{i,1} \wedge \ldots \wedge a_{i,r_i} \rightarrow b_{i,1} \vee \ldots \vee \vee b_{i,s_i} \\ &:= \alpha_i^{\wedge} \rightarrow \beta_i^{\vee} \end{aligned}$$

Thus, $D_{i,j}$ can be rewritten as a conjunction of Disjunctive Datalog rules. Checking entailment from a conjunction of rules is equivalent to checking entailment from a set of the same rules. Hence, we can replace each statement $D_{i,j}$ with a set of Datalog$^{\vee}$ rules. On *line 7* of the `SubsetRankProp` algorithm, Algorithm 7, we now set $R_k :=$ `RNF`$(D_{i,j})$. The Rule Normal Form function `RNF`(Γ) takes an "extended" Disjunctive Datalog statement Γ as input and does the following:

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

Algorithm 8: `SubsetRankDatalog`

```
Input: A defeasible Datalog knowledge base $D$ and classical Datalog knowledge base $C$
Output: An ordered tuple $(R_0, \ldots, R_k, R_\infty, k+1)$
1  $(B_0, \ldots, B_{m-1}, B_\infty, m) :=$ BaseRankDatalog$(D, C)$;
2  $i := 0$; $k := 0$;
3  repeat
4      for $j := |B_i|$ to $1$ do
5          $S_{i,j} :=$ Subsets$(B_i, j)$;
6          $D_{i,j} := \bigvee_{X \in S_{i,j}} \bigwedge_{x \in X} x$;
7          $R_k :=$ RNF$(D_{i,j})$;
8          $k := k + 1$;
9      $i := i + 1$;
10 until $i := m$;
11 $R_\infty := B_\infty$;
12 return $(R_0, \ldots, R_k, R_\infty, k+1)$
```

6.2.2 Rephrasing Lexicographic Closure for Datalog

`LexicographicClosureDatalog` is the same as `RationalClosureDatalog`, with the exception that the adapted `SubsetRankDatalog` algorithm is used to rank statements on *line 1* instead of the `BaseRankDatalog` algorithm.

Proposition 3 *The adapted `LexicographicClosureDatalog` algorithm is LM-rational.*

Proof of Proposition 3: Notice that the proofs for the satisfaction of each KLM property by the `RationalClosureDatalog` procedure, in Appendix A, are independent of the ranking produced by the `BaseRankDatalog` procedure. Furthermore, notice that the only difference between the `LexicographicClosureDatalog` procedure and the `RationalClosureDatalog` procedure is the use of the `SubsetRankDatalog` procedure to rank statements instead of the `BaseRankDatalog` procedure. Thus, the proofs for the satisfaction of each KLM property in Appendix A can be used to prove the satisfaction of each KLM property by the `LexicographicClosureDatalo`
procedure.

7 CONCLUSIONS & FUTURE WORK

The main focus of this paper was to provide versions of defeasible reasoning for Disjunctive Datalog. To be able to express the KLM properties and the algorithm in Datalog, we motivated for extensions that would have to be made to the syntax and semantics of Datalog. We proved that Rational Closure for Datalog was LM-rational (i.e. it conforms to the KLM properties), but showed by example that it does not seem to capture the relationship between variables and constants in Datalog.

We introduced Relevant Closure and Lexicographic Closure as alternatives for computing defeasible entailment and adapted both of the algorithms for Datalog. We found that Lexicographic Closure is still LM-rational, but that Relevant Closure does not satisfy some of the KLM properties.

This paper is a first step towards defining defeasible entailment for more expressive logics. Future work on this topic would most definitely include finding a semantic definition of Rational Closure for Datalog, based on minimal models. Other future work could include an attempted adaptation of the Relevant Closure method for computing defeasible entailment, done in such a way that it satisfies the KLM properties, while still maintaining the basic ideas of Relevant Closure.

As another option, Casini et al. (Casini et al., 2019) showed that LM-rationality is necessary but not sufficient to define acceptability of defeasible entailment forms. The additional properties for Basic Defeasible Entailment proposed by Casini et al. (Casini et al., 2019) can be extended to Datalog. Furthermore, other properties that are specific to defeasible entailment for Datalog should be explored. These properties would likely lead to a revision of our proposed algorithm, as it does not fully capture how variables and constants should interact in defeasible Datalog.

References

Ben-Ari, M. (2012). *Mathematical logic for computer science*. Springer Science & Business Media. 10.1007/978-1-4471-4129-7

Casini, G., Meyer, T., Moodley, K. & Nortje, R. (2014). Relevant closure: A new form of defeasible reasoning for description logics. *JELIA 2014: Logics in Artificial Intelligence*, 92–106. 10.1007/978-3-319-11558-0_7

Casini, G., Meyer, T., Moodley, K. & Varzinczak, I. (2013). Towards practical defeasible reasoning for description logics. *Proceedings of the 26th International Workshop on Description Logics*, 587–599. http://researchspace.csir.co.za/dspace/handle/10204/7039

Casini, G., Meyer, T. & Varzinczak, I. (2019). Taking defeasible entailment beyond rational closure. *JELIA 2019: Logics in Artificial Intelligence*, 182–197. 10.1007/978-3-030-19570-0_12

Ceri, S., Gottlob, G. & Tanca, L. (1989). What you always wanted to know about datalog (and never dared to ask). *IEEE Transactions on Knowledge and Data Engineering*, *1*, 146–166. 10.1109/69.43410

Freund, M. (1998). Preferential reasoning in the perspective of Poole default logic. *Artificial Intelligence*, *98*, 209–235. 10.1016/S0004-3702(97)00053-2

Kraus, S., Lehmann, D. & Magidor, M. (1990). Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence*, *44*, 167–207. 10.1016/0004-3702(90)90101-5

Lehmann, D. (1995). Another perspective on default reasoning. *Annals of Mathematics and Artificial Intelligence*, *15*, 61–82. 10.1007/BF01535841

Lehmann, D. & Magidor, M. (1992). What does a conditional knowledge base entail? *Artificial Intelligence*, *55*, 1–60. 10.1016/0004-3702(92)90041-U

Moodley, K. (2015). *Practical Reasoning for Defeasible Description Logics* (Doctoral dissertation). University of KwaZulu-Natal. 10.31237/OSF.IO/DW5P2

Morris, M., Ross, T. & Meyer, T. (2019). Defeasible disjunctive datalog. *Proc. of the South African Forum for Artificial Intelligence Research FAIR*, 208–219. http://ceur-ws.org/Vol-2540/FAIR2019_paper_38.pdf

Pasarella, E. & Lobo, J. (2017). A Datalog Framework for Modeling Relationship-based Access Control Policies. *Proceedings of the 22nd ACM on Symposium on Access Control Models and Technologies - SACMAT '17 Abstracts*, 91–102. 10.1145/3078861.3078871

Shkapsky, A., Yang, M., Interlandi, M., Chiu, H., Condie, T. & Zaniolo, C. (2016). Big Data Analytics with Datalog Queries on Spark. *Proceedings of the 2016 International Conference on Management of Data - SIGMOD '16*, 1135–1149. 10.1145/2882903.2915229

Straccia, U. & Casini, G. (2013). Defeasible inheritance-based description logics. *Journal of Artificial Intelligence Research*, *48*, 415–473. 10.1613/jair.4062

APPENDICES

A 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 `RationalClosureDatalog` procedure. That is, we prove that `RationalClosureDatalog` is *LM-rational*. We start by showing that while checking $\mathcal{K} \approx\models \alpha \mid\sim \beta$, if it is always the case that $R_\infty \cup R \models \neg\alpha$, then the algorithm returns **true**.

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

Proof of Lemma 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**.■

A.1 Ref

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

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

Proof of Lemma 2: Let τ be any Herbrand interpretation and θ a substitution which replaces variables by constants. In the proofs, we refer to such a substitution as a *grounding substitution*. 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.■

Let τ be a Herbrand interpretation of $\mathcal{K}$ and θ a grounding substitution. We now consider 2 cases below:

Case 1: At some point (when $i \in [0, n]$) in the $\mathcal{K} \approx\models \alpha \mid\sim \alpha$ 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 $\mathcal{K} \approx\models \alpha \mid\sim \alpha$ checking that $R_\infty \cup R \models \neg\alpha$. Then, the algorithm returns **true**, by Lemma 1.■

A.2 LLE

Suppose $\models \alpha \rightarrow \beta$, $\models \beta \rightarrow \alpha$ and $\mathcal{K} \approx\models \alpha \mid\sim \gamma$. We want to show that $\mathcal{K} \approx\models \beta \mid\sim \gamma$. We will make use of Lemma 3 to do so.

Lemma 3 *Let τ be a Herbrand interpretation and θ a grounding substitution. Then, $\models \alpha \to \beta$ and $\models \beta \to \alpha$ iff $\alpha\theta \in \tau$ and $\beta\theta \in \tau$, or, $\alpha\theta \notin \tau$ and $\beta\theta \notin \tau$.*

Proof of Lemma 3: Let τ be some Herbrand interpretation and θ some grounding substitution. Suppose that $\alpha\theta \in \tau$. Since $\models \alpha \to \beta$ we must have that τ satisfies $\alpha \to \beta$ and so $\beta\theta \in \tau$. Now suppose that $\alpha\theta \notin \tau$. We know that τ satisfies $\beta \to \alpha$ since $\models \beta \to \alpha$. So we must have $\beta\theta \notin \tau$. Similar arguments hold for when $\beta\theta \in \tau$ and $\beta\theta \notin \tau$.■

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 grounding substitution. So τ is a model of $\neg\alpha$ and, hence, $\alpha\theta \notin \tau$. Thus, by Lemma 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 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 $R_\infty \cup R \not\models \neg\alpha$.■

We now consider 2 cases below:

Case 1: At some point (when $i \in [0, n]$) in the $\mathcal{K} \approx\models \alpha \mid\sim \gamma$ checking, $R_\infty \cup R \not\models \neg\alpha$ for the first time. Then, at point i, since $\mathcal{K} \approx\models \alpha \mid\sim \gamma$, $R_\infty \cup R \models \alpha \to \gamma$. 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 $R_\infty \cup R \models \beta \to \gamma$. Let τ be a model of $R_\infty \cup R$ and θ a grounding substitution. Suppose $\beta\theta \in \tau$ then, by Lemma 3, $\alpha\theta \in \tau$ too. And, since $R_\infty \cup R \models \alpha \to \gamma$, we must have $\gamma\theta \in \tau$. So $R_\infty \cup R \models \beta \to \gamma$ and the algorithm returns **true**.

Case 2: It is always the case in the $\mathcal{K} \approx\models \alpha \mid\sim \gamma$ checking that $R_\infty \cup R \models \neg\alpha$. Then, in the $\mathcal{K} \approx\models \beta \mid\sim \gamma$ 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 1.■

A.3 RW

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

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

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

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

A.4 And

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

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

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

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

A.5 Or

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

Case 1: It is always the case (for all $i \in [0, n]$) that in the $\mathcal{K} \approx \alpha \mid\sim \gamma$ checking, $R_\infty \cup R \models \neg\alpha$ and, in the $\mathcal{K} \approx \beta \mid\sim \gamma$ 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 grounding substitution. 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 $\mathcal{K} \approx \alpha \vee \beta \mid\sim \gamma$ checking, it is always the case that $R_\infty \cup R \models \neg(\alpha \vee \beta)$ so the algorithm returns **true**, by Lemma 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 $\mathcal{K} \approx \alpha \mid\sim \gamma$, at point i we must have that $R \models \alpha \to \gamma$.

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 $\mathcal{K} \approx \alpha \vee \beta \mid\sim \gamma$ 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 grounding substitution. 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 $\alpha \vee \beta \to \gamma$ 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 $\mathcal{K} \approx\models \beta \mid\sim \gamma$, 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.**■

A.6 CM

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

Lemma 4 *Suppose $\mathcal{K} \approx\models \alpha \mid\sim \beta$ and $\mathcal{K} \approx\models \alpha \mid\sim \gamma$ 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 `RationalClosureDatalog` 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 `RationalClosureDatalog` algorithm, then $R_\infty \cup R \not\models \neg(\alpha \wedge \beta)$, also for the first time, at that point i.*

Proof of Lemma 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 grounding substitution. Then $\alpha\theta \notin \tau$ so $(\alpha \wedge \beta)\theta \notin \tau$ and, hence, $(\neg(\alpha \wedge \beta))\theta \in \tau$. Hence, $R_\infty \cup R \models \neg(\alpha \wedge \beta)$.■

ii Suppose that, at point i, $R_\infty \cup R \not\models \neg\alpha$ for the first time. Then, since $\mathcal{K} \approx\models \alpha \mid\sim \beta$, 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 $R_\infty \cup R \not\models \neg(\alpha \wedge \beta)$.■

Now we consider 2 cases below:

Case 1: At some point ($i \in [0, n]$) in the $\mathcal{K} \approx\models \alpha \mid\sim \beta$ checking, $R_\infty \cup R \not\models \neg\alpha$ for the first time. Then, at the same point i, in the $\mathcal{K} \approx\models \alpha \mid\sim \gamma$ 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 4, at point i in the $\mathcal{K} \approx \alpha \wedge \beta \mid\!\sim \gamma$ 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 grounding substitution. 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 $\mathcal{K} \approx \alpha \mid\!\sim \beta$ checking that $R_\infty \cup R \models \neg\alpha$. Then, by Lemma 4, in the $\mathcal{K} \approx \alpha \wedge \beta \mid\!\sim \gamma$ checking, it is always the case that $R_\infty \cup R \models \neg(\alpha \wedge \beta)$ and so the algorithm returns **true**, by Lemma 1.■

A.7 RM

Suppose that $\mathcal{K} \approx \alpha \mid\!\sim \gamma$ and $\mathcal{K} \not\approx \alpha \mid\!\sim \neg\beta$. We want to show that $\mathcal{K} \approx \alpha \wedge \beta \mid\!\sim \gamma$. Consider the 2 cases below:

Case 1: At some point ($i \in [0, n]$) in the $\mathcal{K} \approx \alpha \wedge \beta \mid\!\sim \gamma$ 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 grounding substitution. 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 $\mathcal{K} \not\approx \alpha \mid\!\sim \neg\beta$, 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.■

So, since $R_\infty \cup R \not\models \neg\alpha$ at point i (and not before) and $\mathcal{K} \approx \alpha \mid\!\sim \gamma$, 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 $\mathcal{K} \approx \alpha \wedge \beta \mid\!\sim \gamma$ checking that $R_\infty \cup R \models \neg(\alpha \wedge \beta)$. Then, the algorithm returns **true**, by Lemma 1.■

B LM-RATIONALITY OF LEXICOGRAPHIC CLOSURE

In this section, for completeness, we provide proofs for the satisfaction of each KLM property by the `LexicographicClosureDatalog` procedure. That is, we prove that `LexicographicClosureDatalog` is *LM-rational*.

Notice that the proofs for the satisfaction of each KLM property by the `RationalClosureDatalog` procedure, in Appendix A, are independent of the ranking produced by the `BaseRankDatalog` procedure. Furthermore, notice that the only difference between the `LexicographicClosureDatalog` procedure and the `RationalClosureDatalog` procedure is the use of the `SubsetRankDatalog` procedure to rank statements instead of the `BaseRankDatalog` procedure.

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

C LM-RATIONALITY OF RELEVANT CLOSURE

Here we provide a proof that Minimal Relevant Closure satisfies the KLM property of *LLE*. We also provide counter-examples to show that it does not satisfy the properties of *Or*, *CM*, and *RM*.

C.1 LLE

Let us start by examining the KLM property of *LLE*:

$$\text{(LLE)}\ \frac{\models \alpha \rightarrow \beta, \models \beta \rightarrow \alpha,\ \mathcal{K} \approx\!\!\!\!|\ \alpha \mid\!\sim \gamma}{\mathcal{K} \approx\!\!\!\!|\ \beta \mid\!\sim \gamma}$$

The proof (in Appendix A.2) that Rational Closure satisfies *LLE* does not directly translate to a proof for Relevant Closure, since the relevance partitions in the two queries are different. The two queries in question are $\mathcal{K} \approx\!\!\!\!|\ \alpha \mid\!\sim \gamma$ and $\mathcal{K} \approx\!\!\!\!|\ \beta \mid\!\sim \gamma$. The relevance partitions for these queries are fully determined by α and β respectively, since the $\mathcal{K}$ in both instances is the same $\mathcal{K}$.

Thus, to allow the proof to translate, we just have to prove that $\bigcup \mathcal{J}^{\mathcal{K}}_{min}(\alpha) = \bigcup \mathcal{J}^{\mathcal{K}}_{min}(\beta)$ (i.e. the relevance partitions for the two queries are the same). To do this, we start by showing that $\mathcal{J}_{\mathcal{K}}(\alpha) = \mathcal{J}_{\mathcal{K}}(\beta)$. We first prove that $\mathcal{J}_{\mathcal{K}}(\alpha) \subseteq \mathcal{J}_{\mathcal{K}}(\beta)$.

Let $J \in \mathcal{J}_{\mathcal{K}}(\alpha)$; then $\mathcal{J}$ is an α-justification. This means that $\mathcal{J} \models \neg\alpha$. Now since $\models \beta \rightarrow \alpha$, $\mathcal{J} \models \neg\beta$.

Assume to the contrary that there is some $\mathcal{J}' \subset \mathcal{J}$ such that $\mathcal{J}' \models \neg\beta$. Then since $\models \alpha \rightarrow \beta$, $\mathcal{J}' \models \neg\alpha$, contradicting $\mathcal{J}$ being an α-justification. So $\mathcal{J}$ is also a β-justification. This means that $\mathcal{J} \in \mathcal{J}_{\mathcal{K}}(\beta)$ as required, proving that $\mathcal{J}_{\mathcal{K}}(\alpha) \subseteq \mathcal{J}_{\mathcal{K}}(\beta)$. The proof that $\mathcal{J}_{\mathcal{K}}(\beta) \subseteq \mathcal{J}_{\mathcal{K}}(\alpha)$ is very similar.

Thus, we have that $\mathcal{J}_{\mathcal{K}}(\alpha) = \mathcal{J}_{\mathcal{K}}(\beta)$. Since these are exactly equal, it must also be the case then that $\bigcup \mathcal{J}^{\mathcal{K}}_{min}(\alpha) = \bigcup \mathcal{J}^{\mathcal{K}}_{min}(\beta)$. The proof (in Appendix A.2) that Rational Closure satisfies the property of *LLE* can now be used directly to show that Relevant Closure also satisfies the property.

C.2 Or

For the upcoming three counter-examples, we will use the symbol a to represent the Datalog molecule a(Tyler), b to represent b(Tyler), and so on. In this case, *Tyler* is just some constant in our Datalog program. This way, truth can be assigned to these molecules in the same way that it is for propositional logic.

We need to find a knowledge base $\mathcal{K}$ and molecules a, g, e such that $\mathcal{K} \approx a \mid\!\sim e$ and $\mathcal{K} \approx g \mid\!\sim e$, but $\mathcal{K} \not\approx a \vee g \mid\!\sim e$. Define:

$$\mathcal{K} = \{a \mid\!\sim b,\ b \mid\!\sim c,\ a \mid\!\sim \neg c,\ a \mid\!\sim d,\\ g \mid\!\sim d,\ d \mid\!\sim e,\ g \mid\!\sim h,\ h \mid\!\sim \neg e,\ g \mid\!\sim e\}$$

This can be represented neatly by a lattice, where $a \mid\!\sim b$ would be represented by a line going upwards from a to b and $a \mid\!\sim \neg b$ by a dashed line.

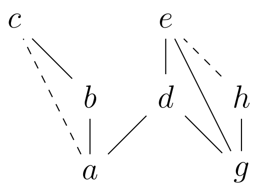

Figure 6: Lattice Representing $\mathcal{K}$

When ranked according to the base rank algorithm, the statements would appear as follows. Note that there is no "infinite rank", since there are no classical statements in the knowledge base.

0	$b \mid\!\sim c \quad d \mid\!\sim e \quad h \mid\!\sim \neg e$
1	$a \mid\!\sim b \quad a \mid\!\sim \neg c \quad a \mid\!\sim d \quad g \mid\!\sim d \quad g \mid\!\sim h \quad g \mid\!\sim e$

Figure 7: Ranking of the Knowledge Base $\mathcal{K}$

To compute Relevant Closure, we first need to compute the justification sets for each of the queries we will be making:

- $\mathcal{J}_{\mathcal{K}}(a) = \{a \mid\!\sim b,\ b \mid\!\sim c,\ a \mid\!\sim \neg c\}$, so $\mathcal{J}^{\mathcal{K}}_{min}(a) = \{b \mid\!\sim c\}$

- $\mathcal{J}_\mathcal{K}(g) = \{\{g \mid\sim e,\ g \mid\sim h,\ h \mid\sim \neg e\},\ \{g \mid\sim d,\ d \mid\sim e,\ g \mid\sim h,\ h \mid\sim \neg e\}\}$, so $\mathcal{J}^\mathcal{K}_{min}(g) = \{h \mid\sim \neg e,\ d \mid\sim e\}$
- $\mathcal{J}_\mathcal{K}(a \vee g) = \mathcal{J}_\mathcal{K}(\neg(\neg a \wedge \neg g)) = \{\{a \mid\sim b,\ b \mid\sim c,\ a \mid\sim \neg c,\ g \mid\sim e,\ g \mid\sim h,\ h \mid\sim \neg e\},\ \{a \mid\sim b,\ b \mid\sim c,\ a \mid\sim \neg c,\ g \mid\sim d,\ d \mid\sim e,\ g \mid\sim h,\ h \mid\sim \neg e\}\}$, so $\mathcal{J}^\mathcal{K}_{min}(a \vee g) = \{b \mid\sim c,\ h \mid\sim \neg e,\ d \mid\sim e\}$

From this, it can be clearly seen that $\mathcal{K} \models\!\approx a \mid\sim e$ and $\mathcal{K} \models\!\approx g \mid\sim e$. We now consider what happens when the algorithm is passed the query $a \vee g \mid\sim e$. When $i = 0$, $R^- \cup R' = \mathcal{K} \models \neg(a \vee g)$, so the algorithm proceeds to the next iteration. When $i = 1$, $R^- \cup R' \not\models \neg(a \vee g)$, so we check if $R^- \cup R' \models a \vee g \to e$. At this point:

$$R^- \cup R' = \{a \mid\sim b,\ a \mid\sim \neg c,\ a \mid\sim d,\ g \mid\sim d,\ g \mid\sim h,\ g \mid\sim e\}$$

Consider a Herbrand interpretation τ such that $a, b, d, h \in \tau$ and $c, g, e \notin \tau$. Then $\tau \Vdash R^- \cup R'$, but $\tau \not\Vdash a \vee g \to e$. Thus $R^- \cup R' \not\models a \vee g \to e$, and the algorithm returns **false**. So $\mathcal{K} \not\models\!\approx a \vee g \mid\sim e$ as required.

C.3 CM

We now need to find $\mathcal{K}$, c, d, and e such that $\mathcal{K} \models\!\approx c \mid\sim d$, $\mathcal{K} \models\!\approx c \mid\sim e$, but $\mathcal{K} \not\models\!\approx c \wedge d \mid\sim e$. Define:

$$\mathcal{K} = \{e \mid\sim \neg g,\ h \mid\sim e,\ b \mid\sim \neg d,\ c \mid\sim d,\\ c \mid\sim b,\ c \wedge d \mid\sim g,\ c \mid\sim h,\ c \wedge d \mid\sim h\}$$

This can be represented by a lattice, where $a \mid\sim b$ would be represented by a line going upwards from a to b, $a \mid\sim \neg b$ by a dashed line, and $a \wedge b$ by thinly dotted lines from a, b to $a \wedge b$.

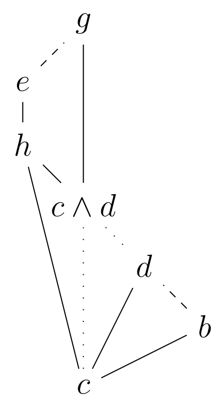

Figure 8: Lattice Representing $\mathcal{K}$

When ranked according to the base rank algorithm, the statements would appear as shown in Figure 9.

We now compute the justification sets for each of the queries we will be making:

0	$e \mid\!\sim \neg g \quad h \mid\!\sim e \quad b \mid\!\sim \neg d$
1	$c \mid\!\sim d \quad c \mid\!\sim b \quad c \wedge d \mid\!\sim g \quad c \mid\!\sim h \quad c \wedge d \mid\!\sim h$

Figure 9: Ranking of the Knowledge Base $\mathcal{K}$

- $\mathcal{J}_{\mathcal{K}}(c) = \{b \mid\!\sim \neg d,\ c \mid\!\sim d,\ c \mid\!\sim b\}$, so $\mathcal{J}^{\mathcal{K}}_{min}(c) = \{b \mid\!\sim \neg d\}$

Thus $\mathcal{K} \mathrel{|\!\approx} c \mid\!\sim d$ and $\mathcal{K} \mathrel{|\!\approx} c \mid\!\sim e$. Now we consider the justification set for $c \wedge d$, which consists of three different $c \wedge d$-justifications:

- $\{b \mid\!\sim \neg d,\ c \mid\!\sim d,\ c \mid\!\sim b\}$
- $\{c \wedge d \mid\!\sim g,\ c \wedge d \mid\!\sim h,\ h \mid\!\sim e,\ e \mid\!\sim \neg g\}$
- $\{c \wedge d \mid\!\sim g,\ c \mid\!\sim h,\ h \mid\!\sim e,\ e \mid\!\sim \neg g\}$

Thus $\mathcal{J}^{\mathcal{K}}_{min}(c \wedge d) = \{b \mid\!\sim \neg d,\ e \mid\!\sim \neg g,\ h \mid\!\sim e\}$. Thus, when processing the query $c \wedge d \mid\!\sim e$, the 0th rank is entirely thrown away, leaving only the 1st rank in $R^- \cup R'$. So when $i = 1$, $R^- \cup R' \not\models \neg(c \wedge d)$, so we check if $R^- \cup R' \models c \wedge d \to e$.

Consider a Herbrand interpretation τ such that $c, b, d, h, g \in \tau$ and $e \notin \tau$. Then $\tau \Vdash R^- \cup R'$, but $\tau \nVdash c \wedge d \to e$. Thus $R^- \cup R' \not\models c \wedge d \to e$, and the algorithm returns **false**. So $\mathcal{K} \mathrel{|\!\not\approx} c \wedge d \mid\!\sim e$ as required.

C.4 RM

We now need to find $\mathcal{K}$, c, d, and e such that $\mathcal{K} \mathrel{|\!\not\approx} c \mid\!\sim \neg d$, $\mathcal{K} \mathrel{|\!\approx} c \mid\!\sim e$, but $\mathcal{K} \mathrel{|\!\not\approx} c \wedge d \mid\!\sim e$. Consider the same counter-example as above for *CM*. Since $\mathcal{K} \mathrel{|\!\approx} c \mid\!\sim d$, it is also the case that $\mathcal{K} \mathrel{|\!\not\approx} c \mid\!\sim \neg d$.

To see this, assume to the contrary that $\mathcal{K} \mathrel{|\!\approx} c \mid\!\sim \neg d$. Then at some point i when $R^- \cup R' \not\models \neg c$, $R^- \cup R' \models c \to \neg d$. However, this stopping point i is the same stopping point for the query $c \mid\!\sim d$, so $R^- \cup R' \models c \to d$. But then $R^- \cup R' \models c \to \neg d$ and $R^- \cup R' \models c \to d$, so $R^- \cup R' \models \neg c$, a contradiction.

Thus, we have that $\mathcal{K} \mathrel{|\!\not\approx} c \mid\!\sim \neg d$, $\mathcal{K} \mathrel{|\!\approx} c \mid\!\sim e$, but $\mathcal{K} \mathrel{|\!\not\approx} c \wedge d \mid\!\sim e$ as before. So the previous counter-example is also a counter-example for *RM*.

D OTHER PROOFS

D.1 Proposition 1

Let τ be a Herbrand interpretation. Then, τ is a model of $\neg\alpha$ under Datalog+ semantics iff τ is a model of $\alpha \to \bot$ under Datalog$^\vee$ semantics.

Proof of Proposition 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 $\alpha \rightarrow \bot$ under Datalog$^\vee$ 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$^\vee$ semantics. So, $\alpha \rightarrow \bot$ is true under τ for θ. Hence, τ is a model of $\alpha \rightarrow \bot$ under Datalog$^\vee$ semantics.

Suppose τ is a model of $\alpha \rightarrow \bot$ under Datalog$^\vee$ semantics. We claim that $\alpha\theta \notin \tau$ under Datalog$^\vee$ semantics. Suppose, to the contrary, that $\alpha\theta \in \tau$. Notice that it is always the case that $\bot\theta \notin \tau$. Thus, $\alpha \rightarrow \bot$ is not true under τ for θ, contradicting the assumption that τ is a model of $\alpha \rightarrow \bot$. Thus, our claim holds - $\alpha\theta \notin \tau$ under Datalog$^\vee$ 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.■