UNIVERSITY OF CAPE TOWN

DEPARTMENT OF COMPUTER SCIENCE

CS/IT Honours Final Paper 2019

Title: Defeasible Disjunctive Datalog

Author: Matthew Morris

Project Abbreviation: DDLV

Supervisor(s): Professor 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 (*this section allowed only with motivation letter from supervisor*)	*0*	*10*	0
Total marks		**80**	

Defeasible Disjunctive Datalog

Matthew Morris
matthewthemorris@gmail.com
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 for extending the KLM properties and Rational Closure; a well-known form of defeasible entailment. We conclude by exploring a Datalog extension of Relevant Closure; a less conservative form of defeasible entailment.

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, Relevant Closure, Datalog

1 INTRODUCTION

The Knowledge Representation and Reasoning approach to Artificial Intelligence uses logics to represent knowledge. Automated reasoning methods are used 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 previously draw. 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 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. 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 [10]; a well-supported approach to non-monotonic reasoning. In Section 2, 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 6. The work in Sections 1 - 5 was done jointly with Ross.

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 a 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 the more "common sense" non-monotonic reasoning for propositional logic. The *KLM approach* proposed by Kraus et al. [10] is an axiomatic approach based on the concept of plausible inference, and is currently one of the most preferred methods [4, 13]. In this approach, plausible inference is represented by a defeasible implication operator of the form $\alpha \mid\sim \beta$, 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:

$$s \mid\sim \neg x$$

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

2.3 The KLM Properties

Unlike classical entailment, defeasible entailment is not unique. There exist multiple formalizations of defeasible entailment, such as *Rational Closure* [12], *Relevant Closure* [4], and *Lexicographic Closure* [11]. The KLM framework provides a list of rationality properties, which Lehmann and Magidor [12] argued must be satisfied by defeasible entailment methods. This provides a way of differentiating between acceptable and non-acceptable methods of entailment. Any method that satisfies these properties is referred to as being *LM-rational*.

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 [8], 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 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.

Algorithm 1: `BaseRank`

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

$i := 0$;
$E_0 := \overrightarrow{\mathcal{K}}$;
repeat
  $E_{i+1} := \{\alpha \rightarrow \beta \in E_i \mid E_i \models \neg\alpha\}$;
  $R_i := E_i \setminus E_{i+1}$;
  $i := i + 1$;
until $E_{i-1} = E_i$;
$R_\infty := E_{i-1}$;
if $E_{i-1} = \emptyset$ **then**
  $n := i - 1$;
else
  $n := i$;
return $(R_0, \ldots, R_{n-1}, R_\infty, n)$

Intuitively, the `BaseRank` algorithm starts with all the statements from $\mathcal{K}$ in E_i, converted to their classical forms. That is, every $\alpha \mathrel{|\!\sim} \beta$ would become $\alpha \rightarrow \beta$. Then, to get to E_{i+1}, we keep all the defeasible statements such that the left hand side can be "proven false" by our defeasible statements in the previous iteration and by all classical statements.

When the algorithm stops, what will eventually be left in E_{i+1} is all classical statements. Thus, R_∞ (the infinite level), represents all certain information in the knowledge base. It is important to note that any classical sentence α can be expressed as a defeasible implication: α if and only if $\neg\alpha \mathrel{|\!\sim} \bot$. So some statements that appear defeasible may in fact be classical, and will appear on the level of R_∞.

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 ($t \mathrel{|\!\sim} x$)
(4) Students typically do not pay taxes ($s \mathrel{|\!\sim} \neg x$)
(5) Students typically drink coffee ($s \mathrel{|\!\sim} c$)

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

0	Students typically do not pay taxes ($s \mathrel{|\!\sim} \neg x$) Students typically drink coffee ($s \mathrel{|\!\sim} c$)
1	Tutors typically pay taxes ($t \mathrel{|\!\sim} x$)
∞	Tutors are students ($t \rightarrow s$) First years are students ($f \rightarrow s$)

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

Intuitively, more general statements will appear higher up 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 in the ranking.

Algorithm 2: `RationalClosure`

Input: A knowledge base $\mathcal{K}$ and a defeasible implication $\alpha \mathrel{|\!\sim} \beta$
Output: **true**, if $\mathcal{K} \mid\!\sim \alpha \mathrel{|\!\sim} \beta$, and **false**, otherwise

$(R_0, \ldots, R_{n-1}, R_\infty, n) :=$ `BaseRank`$(\mathcal{K})$;
$i := 0$;
$R := \bigcup_{i=0}^{j<n} R_j$;
while $R_\infty \cup R \models \neg\alpha$ **and** $R \neq \emptyset$ **do**
  $R := R \setminus R_i$;
  $i := i + 1$;
return $R_\infty \cup R \models \alpha \rightarrow \beta$;

The `RationalClosure` algorithm starts with the ranking provided by `BaseRank`. It is also asked a question, whether $\alpha \mathrel{|\!\sim} \beta$. 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.

The algorithm then goes on, and in each iteration, 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 $\alpha \mathrel{|\!\sim} \beta$ 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 $t \mathrel{|\!\sim} x$, 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	~~Students do not pay taxes. Students drink coffee.~~
1	Tutors pay taxes.
∞	Tutors are students. First years are students.

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 now check whether $\alpha \rightarrow \beta$ holds in the classical case, and return that as our result. Thus, 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. [7] 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 form[1]:

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

Each literal l_i has the form $p_i(t_1, \ldots, t_{k_i})$, where p_i is a *predicate symbol* and $t_1, \ldots, t_{k_i}$ 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:

$$l_0$$

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 so too does the head hold true" 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:

$$\{f(Tyler), s(Tyler), t(Tyler), p(Tyler), a(Tyler)\}.$$

So a possible Herbrand interpretation is:

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

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; $t(Tyler) \in \tau$ so $t(Tyler)$ is true 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. For example; $t(Tyler) \in \tau$ and $p(Tyler) \in \tau$ so the requirements are all true but $a(Tyler) \notin \tau$ so the conclusion is not true. Thus, the statement $t(X) \wedge p(X) \rightarrow a(X)$ must not be true for τ. This is a consequence of the implicit universal quantifier in the rule (*for all X*).

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 statement below cannot be represented using standard Datalog:

Example 3.2.

(4) Student, X, is an undergraduate or a postgraduate

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. However, Datalog does not allow us to represent falsehood either. For example; the statement below cannot be represented using standard Datalog:

Example 3.3.

(5) X is never a student and an employee

We now propose a slightly extended version of Datalog: Datalog$^{\vee}$. We introduce the literal $\bot$. Intuitively, we mean that the literal $\bot$ is such that it is never true. Additionally, we 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:

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

Now we can represent the statement from Examples 3.2 and 3.3, using predicates s, u, p and e to represent students, undergraduates, postgraduates, and employees respectively:

(4) $s(X) \rightarrow u(X) \vee p(X)$
(5) $s(X) \wedge e(X) \rightarrow \bot$

We now need to define the semantics for our extended logic. We consider $\bot$ to be a ground literal. For any Herbrand interpretation τ, we define that $\bot$ is never in τ. We say that a rule $b_1 \wedge \cdots \wedge b_m \rightarrow h_1 \vee \cdots \vee h_n$ 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 literals in the head is in τ. Intuitively, this now means that whenever we can make the "requirements" of the rule true, then at least one of the "conclusions" is true.

[1]Datalog Horn clauses usually have the form $l_0 \leftarrow l_1 \wedge \cdots \wedge l_m$. However, we choose to use a syntax that mirrors that of propositional logic for the ease of the reader. The semantics defined below are equivalent to the semantics defined for clauses of the original form by Ceri et al. [7]

For example; if we combine the statements in Examples 3.1 and 3.2, τ defined below is a Herbrand interpretation. Now, $s(Tyler) \in \tau$ so the requirement is true and $p(Tyler) \in \tau$ 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.

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

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 3.4.

(1) Typically, if X is a tutor, X pays tax.

As discussed above, it is often useful to be able to represent such statements. The KLM approach [10] for propositional logic introduces defeasible implications of the form $\alpha \mathrel{|\sim} \beta$ whose semantics are given by *ranked interpretations* [12]. We want to allow for similar defeasible statements to be represented by Disjunctive Datalog. So, we introduce defeasible rules of the form:

$$b_1 \wedge \cdots \wedge b_m \mathrel{|\sim} h_1 \vee \cdots \vee h_n$$

We intend for the logical connective $\mathrel{|\sim}$ to be the defeasible form of the logical connective $\rightarrow$ in rules. The rule $b_1 \wedge \cdots \wedge b_m \mathrel{|\sim} h_1 \vee \cdots \vee h_n$ is intended to intuitively mean that "typically, if all of $b_1, \ldots, b_m$ are true, then at least one of $h_1, \ldots, h_n$ is true". For example; the statement in Example 3.4 can be represented in Defeasible Disjunctive Datalog as shown below, where x represents taxes.

(1) $t(X) \mathrel{|\sim} x(X)$

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.

Notice that in the defeasible entailment algorithm, `RationalClosure`, defined above, the definition of entailment of complex formulas such as $\alpha \rightarrow \beta$ is required (see 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 define 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 [9]. 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 focus of this paper is to algorithmically define *defeasible entailment* $\mathcal{K} \mathrel{|\!\approx} \alpha \mathrel{|\sim} \beta$. That is, how do we answer the question: "*Can we typically conclude $\alpha \mathrel{|\sim} \beta$ from a defeasible knowledge base K?*". We want to extend algorithms for answering this question in the propositional case to the Datalog case. We also want to ensure that our adapted algorithms remain "reasonable".

Lehmann and Magidor [12] recommend that different forms of defeasible entailment satisfy a set of *rationality properties*, referred to as the KLM properties. If a defeasible entailment algorithm satisfies all the properties, then it is believed to be an acceptable means of computing defeasible entailment and is called *LM-rational*. For completeness, the KLM properties for propositional logic are stated below:

(Ref) $\mathcal{K} \mathrel{|\!\approx} \alpha \mathrel{|\sim} \alpha$

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

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

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

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

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

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

We adopt this approach to analysing the acceptability of our extended defeasible entailment algorithms. 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 connectives $\wedge$ and $\vee$. That is, the defeasible rules which we consider take the following restricted form:

$$b \mathrel{|\sim} h$$

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

(Ref) $\mathcal{K} \mathrel{|\!\approx} l \mathrel{|\sim} l$ (CM) $\dfrac{\mathcal{K} \mathrel{|\!\approx} l \mathrel{|\sim} m,\ \mathcal{K} \mathrel{|\!\approx} l \mathrel{|\sim} n}{\mathcal{K} \mathrel{|\!\approx} l \wedge m \mathrel{|\sim} n}$

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 in terms of the entailment $m \models n$.

$$\text{(RW)}\ \frac{\mathcal{K} \mid\!\sim l \mid\sim m,\ \models m \rightarrow n}{\mathcal{K} \mid\!\sim l \mid\sim n}$$

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 C.3, is proved in Appendix C.

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 in terms of the equivalence $l \equiv m$.

$$\text{(LLE)}\ \frac{\models l \rightarrow m, \models m \rightarrow l,\ \mathcal{K} \mid\!\sim l \mid\sim n}{\mathcal{K} \mid\!\sim m \mid\sim n}$$

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:

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

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

$$\text{(And)}\ \frac{\mathcal{K} \mid\!\sim l \mid\sim m,\ \mathcal{K} \mid\!\sim l \mid\sim n}{\mathcal{K} \mid\!\sim l \mid\sim m \wedge n}$$

Now, consider the naive extensions of the *Or* and *RM* properties. In the *Or* property, the rule $l \vee m \mid\sim n$ 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 $l \mid\sim \neg m$ in the *RM* property cannot be stated.

$$\text{(Or)}\ \frac{\mathcal{K} \mid\!\sim l \mid\sim n,\ \mathcal{K} \mid\!\sim m \mid\sim n}{\mathcal{K} \mid\!\sim l \vee m \mid\sim n} \qquad \text{(RM)}\ \frac{\mathcal{K} \mid\!\sim l \mid\sim n,\ \mathcal{K} \not\mid\!\sim l \mid\sim \neg m}{\mathcal{K} \mid\!\sim l \wedge m \mid\sim n}$$

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 a *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 $\alpha^{\vee}$, to be a combination of literals using $\vee$ connectives in the form:

$$l_1 \vee l_2 \vee \cdots \vee l_n$$

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

$$l_1 \wedge l_2 \wedge \cdots \wedge l_n$$

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, we notice that the current syntax of Datalog is too restrictive to state these properties. We reiterate that the current syntax of Disjunctive Datalog only allows for rules of the form:

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

Notice that molecule α occurs in both the head and body of the rule $\alpha \mid\sim \alpha$ in the naive general extension of the *Ref* property below. 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, $\alpha \mid\sim \alpha$ violates the structure of Datalog rules.

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

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.

$$\text{(RW)}\ \frac{\mathcal{K} \mid\!\sim \alpha^{\wedge} \mid\sim \beta,\ \models \beta \rightarrow \gamma^{\vee}}{\mathcal{K} \mid\!\sim \alpha^{\wedge} \mid\sim \gamma^{\vee}} \qquad \text{(CM)}\ \frac{\mathcal{K} \mid\!\sim \alpha^{\wedge} \mid\sim \beta,\ \mathcal{K} \mid\!\sim \alpha^{\wedge} \mid\sim \gamma^{\vee}}{\mathcal{K} \mid\!\sim \alpha^{\wedge} \wedge \beta \mid\sim \gamma^{\vee}}$$

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

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

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:

If X is a tutor, then X is typically a student. $\quad t(X) \mid\sim s(X)$

If X is a tutor, then X is typically an employee. $\quad t(X) \mid\sim e(X)$

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

If X is a tutor, then X is typically a student and an employee.
$$t(X) \mid\sim s(X) \wedge e(X)$$

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 [9], 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 [9] 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:

$$A \rightarrow B \qquad A \mid\sim B$$

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 shorthand.

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

$$\text{(Ref)}\ \mathcal{K} \mid\!\sim \alpha \mid\sim \alpha \qquad \text{(RW)}\ \frac{\mathcal{K} \mid\!\sim \alpha \mid\sim \beta,\ \models \beta \rightarrow \gamma}{\mathcal{K} \mid\!\sim \alpha \mid\sim \gamma}$$

$$\text{(And)}\ \frac{\mathcal{K} \mid\!\sim \alpha \mid\sim \beta,\ \mathcal{K} \mid\!\sim \alpha \mid\sim \gamma}{\mathcal{K} \mid\!\sim \alpha \mid\sim \beta \wedge \gamma} \qquad \text{(Or)}\ \frac{\mathcal{K} \mid\!\sim \alpha \mid\sim \gamma,\ \mathcal{K} \mid\!\sim \beta \mid\sim \gamma}{\mathcal{K} \mid\!\sim \alpha \vee \beta \mid\sim \gamma}$$

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

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 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 "proven false" 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 $\bot$. We will use $\bot$ 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 $\bot$ is always false. So, whenever α is true, the rule $\alpha \rightarrow \bot$ is false. Thus, we can rewrite $\neg\alpha$ as $\alpha \rightarrow \bot$. This is formally stated in Proposition 5.1 below, the proof of which is found in Appendix D.

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 $\alpha \rightarrow \bot$ under Datalog$^\vee$ 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 $\neg\alpha \mid\sim \bot$. 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 $E_0 := \overrightarrow{D}$ on *line 2*. It now assesses the exceptionality of molecule α by using the entailment check $E_i \cup C \models \alpha \rightarrow \bot$ 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 4 in Appendix B.

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 prove molecule α false 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 $R_\infty \cup R \models \alpha \rightarrow \bot$ on *line 4* instead of the original $R_\infty \cup R \models \neg\alpha$ check.

Under the assumption that there is an algorithm to compute classical entailment for Datalog$^\vee$, this adapted version of the RationalClosure algorithm can now be used to check whether a rule $\alpha \mid\sim \beta$ is defeasibly entailed by the knowledge base $\mathcal{K}$. For clarity, the adapted RationalClosure algorithm for the Datalog case is fully expressed in Algorithm 5 in Appendix B.

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 C. 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.

$$\text{(And)}\ \frac{\mathcal{K} \mid\approx \alpha \mid\sim \beta,\ \mathcal{K} \mid\approx \alpha \mid\sim \gamma}{\mathcal{K} \mid\approx \alpha \mid\sim \beta \wedge \gamma}$$

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

(1) When passed the query $\alpha \mid\sim \beta$, the algorithm returns **true**
(2) When passed the query $\alpha \mid\sim \gamma$, the algorithm returns **true**

Then when passed the query $\alpha \mid\sim \beta \wedge \gamma$, 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 and that the the ranking returned by the BaseRank algorithm will be the same for all of them.

The first case is one where during the $\mathcal{K} \mid\approx \alpha \mid\sim \beta$ 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 $\mathcal{K} \mid\approx \alpha \mid\sim \beta \wedge \gamma$ 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 $R_\infty \cup R \models \alpha \rightarrow \beta \wedge \gamma$ 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 $\mathcal{K} \mid\approx \alpha \mid\sim \beta$, $\mathcal{K} \mid\approx \alpha \mid\sim \gamma$, 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:

$\alpha \rightarrow \beta$ (tutors are students) $\qquad$ $\alpha \rightarrow \gamma$ (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 RELEVANT CLOSURE FOR DATALOG

6.1 Motivation for Relevant Closure

Rational Closure appears to be a suitable definition for computing defeasible entailment. It is simple and it satisfies the KLM properties. However, there are other forms of defeasible entailment as well. In her paper, Ross discusses another approach, Lexicographic Closure, that is more complex but that does still satisfy the KLM properties [11]. Here, we will introduce the definition for Relevant Closure as provided by Casini et al. [4] and will start by motivating for its existence with an example. Consider the following knowledge base $\mathcal{K}$:

Example 6.1.

(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 pay taxes ($p \mid\sim t$)
(5) Typically, students do not pay taxes ($s \mid\sim \neg t$)

When ranked according to the `BaseRank` algorithm, the statements would appear as follows:

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

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

For now, we'll use the algorithm for `RationalClosure` to compute the results for queries. Suppose we asked; *"Do adults typically watch movies?"*, corresponding to the query $a \mid\sim m$. Then at the start of the algorithm when $i = 0$, $R_\infty \cup R \not\models \neg a$. Also, knowing $a \rightarrow p$ and $p \rightarrow m$ allows us to conclude $a \rightarrow m$, so the algorithm returns **true** for the query. This makes sense, and the algorithm appears to have correctly computed the result.

However, 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.

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 $\neg s$;

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

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.

6.2 Algorithmic Definition

The algorithm for Relevant Closure provided by Casini et al. [4] is defined in terms of ALC, which is a type of description logic. To make the algorithm easier to understand and to convert to Datalog, we will first express it in terms of propositional logic. The algorithm converts almost directly, but there is one aspect of it that does not. We will explain the translation at a very high level, as formally introducing the syntax and semantics of ALC is beyond the scope of this project.

In ALC, statements take the form $A \sqsubseteq B$, which intuitively says "all objects of type A are also objects of type B". However, due to how entailment is defined, it is impossible to simply add the defeasible statements to the knowledge base in the same way that we do for propositional logic. Thus, instead of checking something like $\mathcal{K} \models \neg\alpha$ in propositional logic, we instead check something like $T \models D \sqsubseteq \neg A$. In this case, T represents all of the classical information, and D represents all of the defeasible information, bundled together in a special way (again see Casini et al. [4] for more information).

This is made possible by the fact that checking if $A \sqsubseteq B$ holds true is the same as checking if $\neg A \sqcup B$ always holds true. The closest translation of this idea for propositional logic is just to add our defeasible information to the knowledge base like we did for `RationalClosure`. With that in mind, here follows the adapted definition of Relevant Closure for propositional logic:

Algorithm 3: `RelevantClosure`

Input: A 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

```
(R_0, ..., R_{n-1}, R_∞, n) := BaseRank(K);
i := 0;
R' := R;
while R_∞ ∪ R^- ∪ R' ⊨ ¬α and R' ≠ ∅ do
    R' := R' \ {R_i ∩ R};
    i := i + 1;
return R_∞ ∪ R^- ∪ R' ⊨ α → β;
```

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. Besides from this, the algorithm operates exactly as `RationalClosure` does. We say that a statement $\alpha \mid\sim \beta$ is in the Relevant Closure of $\mathcal{K}$ if and only if the `RelevantClosure` algorithm returns **true** when given $\alpha \mid\sim \beta$ and $\mathcal{K}$.

6.3 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 $\alpha \mid\sim \beta$ (i.e. how to distinguish between relevant and non-relevant statements). Based on the ideas explored in example 6.1, 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 6.2. α is said to be *exceptional* for $\mathcal{K}$ if $\mathcal{K} \models \neg\alpha$.

Intuitively, something is exceptional if we can prove it doesn't exist using our current information, which corresponds to it always being false. For example, if $\mathcal{K}$ contains the statements

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

then we can conclude $\neg s$ (there are no students) like we did before. So s is exceptional for $\mathcal{K}$.

Definition 6.3. 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 $\mathcal{J}'$.

This may appear complex, but actually states something very simple. The aim of this is to try capture the idea of throwing away as few statements as possible. $\mathcal{J}$ is a group of statements we can use to prove that α is exceptional, such that no subset of $\mathcal{J}$ can also be used to do the same. For example, our previous collection of statements for $\mathcal{K}$ can be used to prove that s is exceptional, but it is not an s-justification. That is because there is a subset of those statements that can be used to prove s exceptional:

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

These statements form an s-justification, as no subset of them can prove s exceptional.

Definition 6.4. 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. $\bigcup \mathcal{J}^{\mathcal{K}}(\alpha)$.

Firstly, notice that there is not necessarily a single α-justification for a given α and $\mathcal{K}$. To see this, consider an extension of Example 6.1, where the following statements are added:

Example 6.5.

(1) Tutors are students ($u \rightarrow s$)
(2) Typically, students take breaks ($s \mid\sim b$)
(3) Typically, tutors do not take breaks ($u \mid\sim \neg b$)

When ranked according to the `BaseRank` algorithm, the statements would now appear as follows:

0	$p \mid\sim m \quad p \mid\sim t$
1	$s \mid\sim b \quad s \mid\sim \neg t$
2	$u \mid\sim \neg b$
∞	$u \rightarrow s \quad a \rightarrow p \quad s \rightarrow p$

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

In this example, there ends up being two different u-justifications for $\mathcal{K}$:

- $\mathcal{J}_1 = \{s \mid\sim \neg t,\ p \mid\sim t\}$, since $u \rightarrow s \rightarrow \neg t$ and $u \rightarrow s \rightarrow p \rightarrow t$ allow us to conclude $\neg u$
- $\mathcal{J}_2 = \{s \mid\sim b,\ u \mid\sim \neg b\}$, since $u \rightarrow \neg b$ and $u \rightarrow s \rightarrow b$ allow us to conclude $\neg u$

To get $\mathcal{J}^{\mathcal{K}}(u)$, simply collect all the sets of such u-justifications. This collection is then turned into the collection of statements from those u-justifications, and passed to the `RelevantClosure` algorithm as the set of all "relevant statements". In the algorithm, this corresponds to the set R in the partition $< R,\ R^{-} >$. So for Example 6.5, $R = \{s \mid\sim \neg t,\ p \mid\sim t,\ s \mid\sim b,\ u \mid\sim \neg b\}$.

We now consider the same query that failed in Example 6.1, and check if $s \mid\sim m$ is in the Basic Relevant Closure of $\mathcal{K}$. For the query, $R = \{s \mid\sim \neg t,\ p \mid\sim t\}$. So $p \mid\sim m$ is in R^{-} and it is not thrown away as it was before when the algorithm processes. Thus, the algorithm returns **true**, $s \mid\sim m$ is in the Basic Relevant Closure of $\mathcal{K}$, and we have solved the highlighted issue of Rational Closure.

6.4 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 simply consider only *one* statement from each α-justification as relevant, and still fix the exceptionality of α. This is the idea behind Minimal Relevant Closure, although it may sometimes consider multiple statements as relevant.

Definition 6.6. 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. $\bigcup \mathcal{J}^{\mathcal{K}}_{min}(\alpha)$.

This definition captures the idea of considering only the statements furthest up in the ranking (with minimal rank number) from each α-justification as relevant. This corresponds with the intuition that statements with lower rankings should be removed first. Let us now consider Example 6.5, where previously we calculated $\mathcal{J}_1$ and $\mathcal{J}_2$. We can now calculate $\mathcal{J}_1^{min}$ and $\mathcal{J}_2^{min}$ by looking at where the statements are in the ranking of $\mathcal{K}$.

- $\mathcal{J}_1^{min} = \{p \mid\sim t\}$
- $\mathcal{J}_2^{min} = \{s \mid\sim b\}$

Thus, the statements considered as relevant by Minimal Relevant Closure are $p \mid\sim t$ and $s \mid\sim b$. To further motivate for the use of Minimal Relevant Closure over Basic Relevant Closure, let us now consider the query $u \mid\sim \neg t$, corresponding to the question *"Do tutors typically not pay taxes?"*. In this case, tutors are students and students typically do not pay taxes, so it seems like the answer to this question should be yes.

For Minimal Relevant Closure, $R = \{p \mid\sim t,\ s \mid\sim b\}$. Thus, the statement $s \mid\sim \neg t$ is retained while processing, allowing us to conclude that $u \rightarrow s \rightarrow \neg t$. So the algorithm returns **true** like we expect it to. However, for Basic Relevant Closure and the query $u \mid\sim \neg t$, $R = \{s \mid\sim \neg t,\ p \mid\sim t,\ s \mid\sim b,\ u \mid\sim \neg b\}$, meaning the information $s \mid\sim \neg t$ is thrown away while processing. So we can no longer make the same conclusion we did before and the algorithm returns **false**. This violates our intuition as to what conclusion we expect to be able to come to.

6.5 Relevant Closure for Datalog

In terms of adapting the `RelevantClosure` algorithm for Datalog, no further work needs to be done past what has already been said for Rational Closure. The only change the algorithm has experienced is that it now removes fewer statements per level, based on the partition given to it. In addition, the exact same `BaseRank` algorithm has been used. The only new machinery we need is being able to determine whether or not a statement is relevant to a query.

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. To define an *α-justification*, we only need to be able to check for subsets. The remainder of the definitions for both Basic and Minimal Relevant Closure only entail manipulating sets and checking the rankings of statements. Thus, both Basic and Minimal Relevant Closure translate directly to Datalog.

6.6 LM-Rationality

From here on, we will use Minimal Relevant Closure as the definition for Relevant Closure. Also, we use the $\not\approx$ symbol to represent entailment by Relevant Closure. As shown by Casini et al. [4], Relevant Closure for propositional logic satisfies the properties *Ref*, *LLE*, *And*, and *RW*, and does not satisfy *Or*, *CM*, and *RM*. We will show that the same holds true for Relevant Closure for Datalog.

Let us consider the proofs in Appendix C that show that Rational Closure fulfills the KLM properties of *Ref*, *RW*, and *And*. The only difference `RelevantClosure` has from `RationalClosure` 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 $\alpha \not\sim \beta$), as can be seen in the definition of Minimal Relevant Closure. In the aforementioned proofs, 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 A. The counter-examples were adapted from the ALC case.

7 RELATED WORK

Kraus, Lehmann and Magidor (KLM) [10] introduced preferential reasoning, KLM-style defeasible implications and the KLM properties. Lehmann and Magidor [11] presented the concept of Rational Closure for propositional logic and provided an algorithm to compute it. It was also shown that Rational Closure satisfies all the KLM properties and that it must be the most conservative form of defeasible entailment with respect to subset inclusion.

Britz et al. [3] provided an extension of the KLM properties for description logics and presented an extension of Rational Closure for description logics. Casini et al. [4] introduced Relevant Closure for description logics; another method for computing defeasible entailment which is strictly less conservative than Rational Closure. It was shown that this does not satisfy all the KLM properties.

8 CONCLUSIONS

The main focus of this paper was to provide a version of defeasible reasoning for Disjunctive Datalog. We started by discussing the KLM approach to defeasible reasoning, where the KLM properties are used to differentiate between acceptable and non-acceptable ways of defining entailment. We then introduced an algorithm for computing defeasible entailment, Rational Closure, expressed in terms of propositional logic.

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 eventually settled on what we call Datalog+ to express the properties. Once we had adapted both the algorithm and properties, we proved that Rational Closure for Datalog was LM-rational (i.e. it conforms to the KLM properties). This is useful, as it means that any system currently using Datalog can add defeasible reasoning and still have the system compute entailment in a reasonable manner.

Finally, we introduced Relevant Closure as an alternative for computing defeasible entailment. We considered two different forms of Relevant Closure; Basic Relevant Closure and Minimal Relevant Closure. Next, we adapted Relevant Closure for Datalog and showed that it conforms to some of the KLM properties, but not to others. This means that it is not an acceptable way of computing defeasible entailment, according to the properties provided by Lehmann and Magidor [12].

9 FUTURE WORK

Future work on this topic would most likely include finding a semantic definition of Rational Closure for Datalog+, based on minimal models. This would be similar to the work done by Lehmann and Magidor [12]. We would have to use the same ideas and motivations to come up with an equivalent semantic definition for Datalog, and then attempt to prove it equivalent to an algorithmic definition that can be used to compute the entailment. The hope is that the equivalent algorithmic definition is the one defined in this paper, as we have argued that it is the adaptation from propositional logic that makes the most sense.

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. If this turns out to be impossible, another potential route is to add extra assumptions to the KLM properties, in a way that Relevant Closure will now satisfy them. If this is done, motivation will have to be given as to why the original KLM properties are inaccurate, and do not fully capture how we expect defeasible reasoning to operate.

REFERENCES

[1] Chitta Baral and Michael Gelfond. 1994. Logic programming and knowledge representation. *The Journal of Logic Programming* 19-20 (may 1994), 73–148. https://doi.org/10.1016/0743-1066(94)90025-6

[2] Mordechai Ben-Ari. 2012. *Mathematical Logic for Computer Science* (3 ed.). Springer Science & Business Media, Rehovot, Israel. https://books.google.co.za/books?hl=en

[3] Katarina Britz, Giovanni Casini, Thomas Meyer, Kody Moodley, Uli Sattler, and Ivan Varzinczak. 2017. *Rational Defeasible Reasoning for Description Logics.* Technical Report. University of Cape Town, South Africa. https://core.ac.uk/download/pdf/151756088.pdf

[4] Giovanni Casini, Thomas Meyer, Kodylan Moodley, and Riku Nortjé. 2014. Relevant Closure: A New Form of Defeasible Reasoning for Description Logics. In *JELIA 2014: Logics in Artificial Intelligence.* Springer, Cham, Funchal, Madeira, Portugal, 92–106. https://doi.org/10.1007/978-3-319-11558-0_7

[5] G Casini, T Meyer, K Moodley, and I Varzinczak. 2013. *Towards practical defeasible reasoning for description logics.* Technical Report. Centre for Artificial Intelligence Research. http://researchspace.csir.co.za/dspace/handle/10204/7039

[6] Giovanni Casini, Thomas Meyer, and Ivan Varzinczak. 2019. Taking Defeasible Entailment beyond Rational Closure. (2019), 18 pages.

[7] S. Ceri, G. Gottlob, and L. Tanca. 1989. What you always wanted to know about Datalog (and never dared to ask). *IEEE Transactions on Knowledge and Data Engineering* 1, 1 (mar 1989), 146–166. https://doi.org/10.1109/69.43410

[8] Michael Freund. 1998. Preferential reasoning in the perspective of Poole default logic. *Artificial Intelligence* 98, 1-2 (jan 1998), 209–235. https://doi.org/10.1016/S0004-3702(97)00053-2

[9] Michael Genesereth and Kao Eric. 2013. *Introduction to Logic* (second ed.). Morgan & Claypool, Stanford, California. https://doi.org/10.2200/S00518ED2V01Y201306CSL006

[10] S Kraus, D Lehmann, and M Magidor. 1990. Nomonotonic Reasoning, Preferential Method and Cumulative Logics. *Artificial Intelligence* 44 (1990), 167–207. arXiv:arXiv:cs/0202021v1

[11] Daniel Lehmann. 1995. Another perspective on default reasoning. *Annals of Mathematics and Artificial Intelligence* 15, 1 (1995), 61–82. https://doi.org/10.1007/BF01535841

[12] D Lehmann and M Magidor. 1994. What does a conditional knowledge base entail ? *Artificial Intelligence* 55, 1 (1994), 1–60.

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

APPENDICES

A 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*.

A.1 LLE

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

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

The proof (in Appendix C.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 $\mathcal{K} \mid\!\approx \alpha \mid\sim \gamma$ and $\mathcal{K} \mid\!\approx \beta \mid\sim \gamma$. 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 C.2) that Rational Closure satisfies the property of *LLE* can now be used directly to show that Relevant Closure also satisfies the property.

A.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 $\mathcal{K} \mid\!\approx a \mid\sim e$ and $\mathcal{K} \mid\!\approx g \mid\sim e$, but $\mathcal{K} \not\mid\!\approx a \vee g \mid\sim e$. 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.

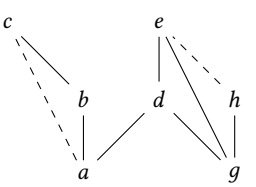

Figure 6: Lattice Representing $\mathcal{K}$

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

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 $\mathcal{K} \mid\!\approx a \mid\sim e$ and $\mathcal{K} \mid\!\approx g \mid\sim e$. 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 $R^{-} \cup R' \models a \vee g \rightarrow e$. 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 $\tau \nVdash a \vee g \rightarrow e$. Thus $R^{-} \cup R' \not\models a \vee g \rightarrow e$, and the algorithm returns **false**. So $\mathcal{K} \not\mid\!\approx a \vee g \mid\sim e$ as required.

A.3 CM

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

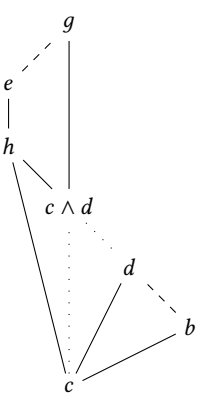

Figure 8: Lattice Representing $\mathcal{K}$

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 $\mathcal{K}$

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:

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

Thus $\mathcal{K} \mid\approx c \mid\sim d$ and $\mathcal{K} \mid\approx c \mid\sim e$. 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 $R^{-} \cup R' \models c \wedge d \rightarrow e$.

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 $\tau \nVdash c \wedge d \rightarrow e$. Thus $R^{-} \cup R' \not\models c \wedge d \rightarrow e$, and the algorithm returns **false**. So $\mathcal{K} \not\mid\approx c \wedge d \mid\sim e$ as required.

A.4 RM

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

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

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

B RATIONAL CLOSURE ALGORITHMS FOR DATALOG

Algorithm 4: `BaseRank`

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

$i := 0$;
$E_0 := \overrightarrow{D}$;
repeat
  $E_{i+1} := \{\alpha \rightarrow \beta \in E_i \mid E_i \cup C \models \alpha \rightarrow \bot\}$;
  $R_i := E_i \setminus E_{i+1}$;
  $i := i + 1$;
until $E_{i-1} = E_i$;
$R_\infty := E_{i-1} \cup C$;
if $E_{i-1} = \emptyset$ **then**
  $n := i - 1$;
else
  $n := i$;
return $(R_0, \ldots, R_{n-1}, R_\infty, n)$

Algorithm 5: `RationalClosure`

Input: A defeasible knowledge base D, a classical 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

$(R_0, \ldots, R_{n-1}, R_\infty, n) :=$ `BaseRank`(D, C);
$i := 0$;
$R := \bigcup_{i=0}^{j<n} R_j$;
while $R_\infty \cup R \models \alpha \rightarrow \bot$ **and** $R \neq \emptyset$ **do**
  $R := R \setminus R_i$;
  $i := i + 1$;
return $R_\infty \cup R \models \alpha \rightarrow \beta$;

C 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 $\mathcal{K} \mid\!\approx \alpha \mid\!\sim \beta$, if it is always the case that $R_\infty \cup R \models \neg\alpha$, then the algorithm returns **true**.

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

Proof of Lemma C.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**. □

C.1 Ref

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

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

Proof of Lemma C.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. □

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 $\mathcal{K} \mid\!\approx \alpha \mid\!\sim \alpha$ 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 $\mathcal{K} \mid\!\approx \alpha \mid\!\sim \alpha$ checking that $R_\infty \cup R \models \neg\alpha$. Then, the algorithm returns **true**, by Lemma C.1. □

C.2 LLE

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

Lemma C.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 C.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 $\beta\theta \notin \tau$. □

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 C.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 C.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 $\mathcal{K} \mid\!\approx \alpha \mid\!\sim \gamma$ checking, $R_\infty \cup R \not\models \neg\alpha$ for the first time. Then, at point i, since $\mathcal{K} \mid\!\approx \alpha \mid\!\sim \gamma$, $R_\infty \cup R \models \alpha \rightarrow \gamma$. 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 C.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 $\mathcal{K} \mid\!\approx \alpha \mid\!\sim \gamma$ checking that $R_\infty \cup R \models \neg\alpha$. Then, in the $\mathcal{K} \mid\!\approx \beta \mid\!\sim \gamma$ 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 C.1. □

C.3 RW

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

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

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

C.4 And

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

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

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

C.5 Or

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

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

C.6 CM

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

Lemma C.4. *Suppose $\mathcal{K} \mid\!\approx \alpha \mid\!\sim \beta$ and $\mathcal{K} \mid\!\approx \alpha \mid\!\sim \gamma$ 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 `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 C.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, $R_\infty \cup R \models \neg(\alpha \wedge \beta)$. □

ii Suppose that, at point i, $R_\infty \cup R \not\models \neg\alpha$ for the first time. Then, since $\mathcal{K} \mid\!\approx \alpha \mid\!\sim \beta$, 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 $\mathcal{K} \mid\!\approx \alpha \mid\!\sim \beta$ checking, $R_\infty \cup R \not\models \neg\alpha$ for the first time. Then, at the same point i, in the $\mathcal{K} \mid\!\approx \alpha \mid\!\sim \gamma$ 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 C.4, at point i in the $\mathcal{K} \mid\!\approx \alpha \wedge \beta \mid\!\sim \gamma$ 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 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 $\mathcal{K} \mid\!\approx \alpha \mid\!\sim \beta$ checking that $R_\infty \cup R \models \neg\alpha$. Then, by Lemma C.4, in the $\mathcal{K} \mid\!\approx \alpha \wedge \beta \mid\!\sim \gamma$ checking, it is always the case that $R_\infty \cup R \models \neg(\alpha \wedge \beta)$ and so the algorithm returns **true**, by Lemma C.1. □

C.7 RM

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

Case 1: At some point ($i \in [0, n]$) in the $\mathcal{K} \mid\!\approx \alpha \wedge \beta \mid\!\sim \gamma$ 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 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 $\mathcal{K} \not\mid\!\approx \alpha \mid\!\sim \neg\beta$, 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 $\mathcal{K} \not\approx \alpha \not\sim \gamma$, 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 $\mathcal{K} \not\approx \alpha \wedge \beta \not\sim \gamma$ checking that $R_\infty \cup R \models \neg(\alpha \wedge \beta)$. Then, the algorithm returns **true**, by Lemma C.1. □

D OTHER PROOFS

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 $\alpha \rightarrow \bot$ 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. □