UNIVERSITY OF CAPE TOWN

DEPARTMENT OF COMPUTER SCIENCE

CS Honours Project
Final Paper 2024

Title: Extending Defeasible Reasoning Beyond Rational Closure

Author: Thabo Vincent Moloi

Project Abbreviation: EXTRC

Supervisor(s): Professor Thomas Meyer

Category	*Min*	*Max*	Chosen
Requirement Analysis and Design	*0*	*20*	*5*
Theoretical Analysis	*0*	*25*	*15*
Experiment Design and Execution	*0*	*20*	*0*
System Development and Implementation	*0*	*20*	*15*
Results, Findings and Conclusions	*10*	*20*	*10*
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*	
Total marks		**80**	

Extending Defeasible Reasoning Beyond Rational Closure

Thabo Vincent Moloi
University of Cape Town
Cape Town, South Africa
mlxtha036@myuct.ac.za

ABSTRACT

Logic plays a crucial role in artificial intelligence (AI) for representing and reasoning with knowledge. Classical reasoning relies on deductive logic to draw conclusions from a knowledge base, assuming that each statement is either true or false. In contrast, defeasible reasoning allows conclusions to change when new information or exceptions arise. The KLM framework is a widely recognised model for defeasible reasoning. This paper explores the theoretical foundations and current practical implementations of defeasible reasoning within the context of AI. The discussion covers Rational Closure and Lexicographic Closure, two significant extensions of the KLM framework. Additionally, we present a tool that implements these algorithms to help users better understand defeasible reasoning, which will be used as a debugging tool to address issues within knowledge bases and entailment algorithms.

CCS CONCEPTS

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

KEYWORDS

Artificial Intelligence, Knowledge Representation and Reasoning, Defeasible Reasoning, Rational Closure, Lexicographic Closure

1 INTRODUCTION

Knowledge Representation and Reasoning (KRR) forms a crucial domain within the field of AI, focusing on the symbolic representation of knowledge and its automated manipulation through reasoning programs [3]. A knowledge base is a collection of knowledge (real-world information), and it serves as a foundation for storing, managing, and retrieving knowledge. The knowledge-based system can use reasoning to draw conclusions from the knowledge base.

Reasoning can contain exceptions where conclusions are drawn from an uncertain or incomplete knowledge base. Using logic can help identify the exceptions and avoid contradictions within the knowledge base. This is where defeasible reasoning comes in. In defeasible reasoning, adding new information can lead to the retraction of previously inferred conclusions. The most widely recognised framework for defeasible reasoning is the KLM framework developed by Kraus, Lehmann and Magidor [5]. In the KLM approach, defeasible reasoning is modelled using non-monotonic logic, allowing conclusions to be drawn tentatively, subject to revision in light of new information. The key idea behind the KLM approach is preference orderings on rules to determine the strength of conclusions.

The paper is part of a broader project that includes contributions from Mohlerepe [19] and Sadiki [21]. Mohlerepe [19] focuses on optimizing rational closure, while Sadiki [21] focuses on generating defeasible knowledge bases.

This paper aims to explore different algorithms that extend the KLM approach, find ways to display the defeasible reasoning process and develop a web application tool that shows this process. This tool will be used as a debugging tool to address issues with knowledge bases and problems or differences in entailment algorithms.

Section 2 provides background information on propositional logic and entailment, setting the foundation for understanding defeasible reasoning. Section 3 delves into defeasible reasoning, focusing on the KLM approach, rational closure, and other related concepts. Section 4 discusses the design and implementation details of the web application tool. The final sections cover discussions, conclusions and potential future work.

2 BACKGROUND

2.1 Propositional Logic

Propositional logic is a formal system used to reason about knowledge or information, and it abstracts concepts from natural language into a formal language [14]. It deals with indivisible statements, known as atomic propositions [15], declarative statements that are either true or false but not both.

The language can typically be constructed from a set of propositional variables or atoms, which represent statements that can either be true or false and the boolean connectives (such as AND, OR, NOT) are used to combine these variables to form more complex formulas [10]. The statements represent real-world information, for example, *"humans are mammals"* or *"an ostrich can fly"*. Sometimes, a statement may have one or more other statements; an example would be "The sky is blue, and the sun is shining".

2.1.1 ***Syntax.*** Each propositional atom is associated with a Boolean value: true, denoted as T, or false, denoted as F. The lowercase letter $p, q, r, \ldots$ denotes the indivisible atoms. The set of all propositional atoms will be denoted as $\mathcal{P}$, which consists of all the statements such that $\mathcal{P} = \{p, q, r, \ldots\}$.

Atoms can be combined to form formulas using Boolean operators, also known as logical connectives, each with specific rules for their combinations [2]. The key connectives are negation, conjunction, disjunction, implication, and equivalence, shown in Table 1.

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

Table 1: Boolean operators in propositional logic

All operators in Table 1 are binary, except negation. This means that they take two operands while negation only takes one operand. The order of precedence for logical operators is as follows: negation has the highest precedence, followed by a conjunction, disjunction, and finally, implication and equivalence, which have the lowest precedence [11].

The propositional language $\mathcal{L}$ is formed by recursively defining propositional formulas using the binary operators and the negation operator $\neg$ over the set $\mathcal{P}$. The set of all propositional formulas $\mathcal{L}$ can be defined as $\forall p \in \mathcal{P}, p \in \mathcal{L}$ and if $\alpha, \beta \in \mathcal{L}$ then $\neg\alpha, \alpha \wedge \beta, \alpha \vee \beta, \alpha \rightarrow \beta, \alpha \leftrightarrow \beta \in \mathcal{L}$.

2.1.2 ***Semantics***. In propositional logic, the meaning of formulas is defined by assigning truth values to the atoms of a formula, similar to evaluating arithmetic expressions by assigning values to variables [2]. We use an interpretation to assign truth values to atoms.

Definition 1 (Interpretation)**.** *An interpretation $\mathcal{I}$ is a function that maps propositions $\mathcal{P}$ to truth values expressed as $\mathcal{I} : \mathcal{P} \rightarrow \{T, F\}$ where and T and F represent true and false respectively [14].*

For example, an interpretation such as $p\overline{q}r$ specifies that p is true, q is false, and r is true. Suppose we have a set of propositional atoms $\mathcal{P} = \{p, q\}$. The truth values of these propositions and the results of various logical operations involving them are summarised in the truth table shown in Table 2.

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

Table 2: Truth values for the logical operations on p and q.

Each row in the truth table represents a possible interpretation [15].

Definition 2 (Satisfaction)**.** *An interpretation $\mathcal{I}$ satisfies a formula α, written $\mathcal{I} \Vdash \alpha$, if (and only if) one of the following conditions hold:*

- *$\alpha \in \mathcal{P}$ and $\mathcal{I}(\alpha) = T$*
- *$\alpha = \neg\beta$ and $\mathcal{I}$ fails to satisfy β*
- *$\alpha = \beta \wedge \gamma$ and $\mathcal{I}$ satisfies both β and γ*
- *$\alpha = \beta \vee \gamma$ and $\mathcal{I}$ satisfies at least one of β and γ*
- *$\alpha = \beta \rightarrow \gamma$ and $\mathcal{I}$ satisfies at least one of $\neg\beta$ and γ*
- *$\alpha = \beta \leftrightarrow \gamma$ and $\mathcal{I}$ satisfies either both $\neg\beta$ and γ or else neither*

If an interpretation $\mathcal{I}$ satisfies a formula α, it is considered a model of α.

Definition 3 (Models)**.** *Given a formula α, the models of α, denoted as $Mod(\alpha)$, are the set of all interpretations that satisfy α [14]. $Mod(\alpha) \subseteq \mathcal{U}$, where $\mathcal{U}$ represents the universal set of all possible interpretations.*

Example 1. Suppose we have a set of propositional atoms $\mathcal{P} = \{p, q\}$ and the formula $\alpha = p \rightarrow q$. The universal set of interpretations for $\mathcal{P}$ is given by $\mathcal{U} = \{pq, p\overline{q}, \overline{p}q, \overline{pq}\}$, where each interpretation represents a combination of truth values for p and q. The formula $p \rightarrow q$ is false only when p is true, and q is false (i.e., $p\overline{q}$) as shown in Table 2. In all other cases, it is true. Therefore, the set of models for α is $Mod(\alpha) = \{pq, \overline{p}q, \overline{pq}\}$.

A formula is a tautology if it is true in all possible interpretations and a contradiction if it is false in all possible interpretations.

Definition 4 (Tautology)**.** *A tautology, denoted as $\top$, is a formula that is true in all possible interpretations, meaning that $Mod(\top) = \mathcal{U}$.*

Tautologies are always true irrespective of the individual components. For example, the formula $p \vee \neg p$ is always true because the disjunction is true for any truth value of p.

Definition 5 (Contradiction)**.** *A contradiction, denoted as $\bot$, is a formula that is false in all possible interpretations, meaning that $Mod(\bot) = \emptyset$.*

A contradiction, such as the formula $p \wedge \neg p$, can never be true because p and $\neg p$ can never be true simultaneously.

2.2 Entailment

Entailment is a logical relationship between statements where one statement, called the conclusion, necessarily follows from one or more other statements, known as the premises. It is a fundamental concept in reasoning that formally shows how certain states of the world logically result from others [22]. For example, given the premises *"All penguins are birds"* and *"Skipper is a penguin"*, the conclusion *"Skipper is a bird"* is entailed by the premises. Formally, entailment can be defined in terms of the relationship between the models of two statements.

Definition 6 (Entailment)**.** *Given propositional statements or formulas $\alpha, \beta \in \mathcal{L}$, α is said to entail β, or β is a logical consequence of α, denoted as $\alpha \models \beta$, if and only if the models of α are a subset of the models of β, written $Mod(\alpha) \subseteq Mod(\beta)$.*

The premises can be represented as a set of formulas.

Definition 7 (Knowledge Base)**.** *Let $\mathcal{K}$ be a set of formulas. If $\mathcal{K}$ is finite, it is called a knowledge base.*

The models of $\mathcal{K}$ are defined as $Mod(\mathcal{K}) := \bigcap \{Mod(\alpha) \mid \alpha \in \mathcal{K}\}$. This expression denotes the set of all models that satisfy every formula in the knowledge base $\mathcal{K}$. The knowledge base $\mathcal{K}$ entails α, written $\mathcal{K} \models \alpha$, if and only if $Mod(\mathcal{K}) \subseteq Mod(\alpha)$.

Example 2. Suppose we have the knowledge base $\mathcal{K}$ with the statements $b \rightarrow f$, $p \rightarrow b$, and $p \rightarrow \neg f$ where the atoms, b, f, and p represent *birds*, *fly*, and *penguins*, respectively. The statements mean *"birds fly"* ($b \rightarrow f$), *"penguins are birds"* ($p \rightarrow b$), and *"penguins don't fly"* ($p \rightarrow \neg f$). We want to check if the statement *"penguins fly"* (i.e., $p \rightarrow f$) follows from $\mathcal{K}$. So, $\mathcal{K} = \{b \rightarrow f, p \rightarrow b, p \rightarrow \neg f\}$.

Ne we have the set, $\mathcal{U} = \{bfp, bf\overline{p}, b\overline{f}p, b\overline{f}\overline{p}, \overline{b}fp, \overline{b}f\overline{p}, \overline{b}\overline{f}p, \overline{b}\overline{f}\overline{p}\}$. We now find $Mod(\alpha)$, $\forall\alpha \in \mathcal{K}$, $Mod(\mathcal{K})$, and $Mod(p \rightarrow f)$.

$$Mod(b \rightarrow f) = \{bfp, bf\overline{p}, \overline{b}fp, \overline{b}f\overline{p}, \overline{b}\overline{f}p, \overline{b}\overline{f}\overline{p}\}$$

$$Mod(p \rightarrow b) = \{bfp, bf\overline{p}, b\overline{f}p, b\overline{f}\overline{p}, \overline{b}\overline{f}p, \overline{b}\overline{f}\overline{p}\}$$

$$Mod(p \rightarrow \neg f) = \{bf\overline{p}, b\overline{f}p, b\overline{f}\overline{p}, \overline{b}fp, \overline{b}f\overline{p}, \overline{b}\overline{f}\overline{p}\}$$

$$Mod(\mathcal{K}) = \bigcap \{Mod(\alpha) \mid \alpha \in \mathcal{K}\} = \{bf\overline{p}, \overline{b}f\overline{p}, \overline{b}\overline{f}\overline{p}\}$$

$$Mod(p \rightarrow f) = \{bfp, bf\overline{p}, b\overline{f}\overline{p}, \overline{b}fp, \overline{b}f\overline{p}, \overline{b}\overline{f}\overline{p}\}$$

$\mathcal{K} \models p \rightarrow f$ and $\mathcal{K} \models p \rightarrow \neg f$ because $Mod(\mathcal{K}) \subseteq Mod(p \rightarrow f)$ and $Mod(\mathcal{K}) \subseteq Mod(p \rightarrow \neg f)$. However, this leads to a contradiction because $p \rightarrow f$ and $p \rightarrow \neg f$ cannot be true simultaneously

unless $\neg p$ is true. $Mod(\neg p) = \{bf\overline{p}, b\overline{f}p, \overline{b}f\overline{p}, \overline{bf}\overline{p}\}$, thus $\mathcal{K} \models \neg p$ since $Mod(\mathcal{K}) \subseteq Mod(\neg p)$. Therefore, p is always false, meaning penguins do not exist.

There is a contradiction because classical reasoning cannot handle exceptions. In this case, we have the statements $b \rightarrow f$, $p \rightarrow f$ and the exception $p \rightarrow \neg f$. The problem of exceptional information arises from classical reasoning's adherence to the property of *monotonicity*[14]. According to monotonicity, a valid conclusion remains so even when new information is introduced [9]. Therefore, the conclusion $p \rightarrow f$ will always be derived from $p \rightarrow b$ and $b \rightarrow f$ regardless of the statement $p \rightarrow \neg f$.

3 DEFEASIBLE REASONING

Defeasible reasoning is a form where conclusions are derived from incomplete or uncertain information, and these conclusions can be revised or overridden when new information becomes available. Defeasible reasoning differs from classical reasoning primarily in how it handles conclusions. In classical reasoning, once a conclusion is derived from a set of premises, it is considered definitive. It does not change unless a logical error or new information contradicts it. On the other hand, in defeasible reasoning, conclusions are tentative and can be overridden or revised based on new information or exceptions to the general rules. This flexibility allows defeasible reasoning to model real-world scenarios more accurately [1], where certainty is often limited and subject to change.

3.1 KLM Framework

The KLM approach introduced a conditional assertion $\alpha \vdash \beta$ if α and β are formulas. $\alpha \vdash \beta$ is read as "α typically implies β" [16, 18]. Unlike a strict logical implication, this statement allows for exceptions or cases where the association does not hold. Revisiting our knowledge base in Example 2, we can replace the symbol $\rightarrow$ with $\vdash$ in statements where we can have exceptions. So, the new knowledge base would be $\mathcal{K} = \{b \vdash f, p \rightarrow b, p \vdash \neg f\}$. From the new knowledge, it can be seen that *"birds typically fly"*, *"penguins are birds"*, and *"penguins typically don't fly"*. This allows us to conclude from uncertain or incomplete information.

KLM outlines criteria that an entailment relation must satisfy to be considered rational. Specifically, an entailment relation $\mathrel{|\!\sim}$ is deemed rational if it adheres to the following principles [13, 16, 18].

$$\text{(Reflexivity)} \qquad \mathcal{K} \mathrel{|\!\sim} \alpha \vdash \alpha$$

$$\text{(Left Logical Equivalence)} \qquad \frac{\mathcal{K} \mathrel{|\!\sim} \alpha \leftrightarrow \beta, \mathcal{K} \mathrel{|\!\sim} \alpha \vdash \gamma}{\mathcal{K} \mathrel{|\!\sim} \beta \vdash \gamma}$$

$$\text{(Right Weakening)} \qquad \frac{\mathcal{K} \mathrel{|\!\sim} \alpha \rightarrow \beta, \mathcal{K} \mathrel{|\!\sim} \gamma \vdash \alpha}{\mathcal{K} \mathrel{|\!\sim} \gamma \vdash \beta}$$

$$\text{(And)} \qquad \frac{\mathcal{K} \mathrel{|\!\sim} \alpha \vdash \beta, \mathcal{K} \mathrel{|\!\sim} \alpha \vdash \gamma}{\mathcal{K} \mathrel{|\!\sim} \alpha \vdash \beta \wedge \gamma}$$

$$\text{(Or)} \qquad \frac{\mathcal{K} \mathrel{|\!\sim} \alpha \vdash \gamma, \mathcal{K} \mathrel{|\!\sim} \beta \vdash \gamma}{\mathcal{K} \mathrel{|\!\sim} \alpha \vee \beta \vdash \gamma}$$

$$\text{(Continuous Monotonicity)} \qquad \frac{\mathcal{K} \mathrel{|\!\sim} \alpha \vdash \gamma, \mathcal{K} \mathrel{|\!\sim} \alpha \vdash \beta}{\mathcal{K} \mathrel{|\!\sim} \alpha \wedge \beta \vdash \gamma}$$

$$\text{(Rational Monotonicity)} \qquad \frac{\mathcal{K} \mathrel{|\!\sim} \alpha \vdash \gamma, \mathcal{K} \not\mathrel{|\!\sim} \alpha \not\vdash \beta}{\mathcal{K} \mathrel{|\!\sim} \alpha \wedge \beta \vdash \gamma}$$

Lehmann and Magidor suggest that defeasible entailment should adhere to all the KLM properties mentioned earlier [4]. This concept is referred to as LM-rationality.

3.1.1 ***Preferential Interpretations.*** Preferential semantics establishes an order among interpretations, indicating that if an agent favours one interpretation u over another v, the agent will likely prioritise u over v [14]. The concept of preference can be interpreted in various ways, with one interpretation being that more typical interpretations are favoured.

3.1.2 ***Ranked Interpretations.*** The concept of preferential interpretations is refined to create ranked interpretations by introducing conditions that restrict the partial order for the interpretation [18]. These restrictions lead to an ordering that establishes a series of non-empty levels, where interpretations on the lower levels are considered more "typical" or preferred.

Moreover, they extend the definition of a preferential consequence relation to define a rational consequence relation, which must satisfy all seven KLM properties. They demonstrate that ranked interpretations can define rational consequence relations, and conversely, every rational consequence relation can be defined by a ranked interpretation.

Definition 8 (Ranked Interpretation)**.** *a ranked interpretation is a function $\mathcal{R} : \mathcal{U} \rightarrow \mathcal{N} \cup \{\infty\}$ which must satisfy the following property: $\forall i \in \mathcal{N}, \exists u \in \mathcal{U}$ such that $\mathcal{R}(u) = i$, then there must be a $v \in \mathcal{U}$ such that $\mathcal{R}(v) = j$ where $0 \leq j < i$ [14].*

Defeasibility is addressed by providing semantics for the operator $\vdash$ using ranked interpretations. In a ranked interpretation $\mathcal{R}$, a defeasible implication $\alpha \vdash \beta$ holds if, for all the most typical interpretations in which α is true (i.e., the models of α on the lowest level containing an interpretation that satisfies α), β is also true.

Consider a set of propositions $\mathcal{P} = \{p, q, r\}$, Table 3 shows a possible ranked interpretation form $\mathcal{P}$.

∞	$\overline{\mathsf{pqr}}$ $\overline{\mathsf{p}}\mathsf{q}\overline{\mathsf{r}}$
2	$\overline{\mathsf{p}}\mathsf{qr}$ $\mathsf{p}\overline{\mathsf{qr}}$
1	pqr $\mathsf{pq}\overline{\mathsf{r}}$ $\mathsf{p}\overline{\mathsf{qr}}$
0	$\overline{\mathsf{pqr}}$

Table 3: Ranked interpretations for $\mathcal{P} = \{p, q, r\}$

Table 3 shows that $\mathcal{R}(\overline{\mathsf{pqr}}) = 0$ which shows that $\overline{\mathsf{pqr}}$ is most typical while $\overline{\mathsf{pqr}}$ and $\overline{\mathsf{p}}\mathsf{q}\overline{\mathsf{r}}$ are impossible as their rank is ∞.

3.2 Rational Closure

Rational closure in defeasible reasoning is crucial for deriving conclusions from explicit and implicit statements within a knowledge base. It allows us to conduct rational inference on defeasible knowledge bases [7], ensuring that they are logically sound and consistent. One key aspect of rational closure is to assign a ranking to each in the knowledge base, thereby extending preferential reasoning [22].

Formally, rational closure involves determining whether the defeasible implication is entailed by a knowledge base using a method that adheres to the rationality properties proposed by Lehmann and Magidor. These properties ensure that conclusions drawn from defeasible implications remain consistent and coherent, even when exceptions exist. Rational closure is the most conservative form

of defeasible entailment. This means that anything entailed by rational closure will also be entailed by the other common forms of defeasible entailment [20]. The rational closure process can be defined semantically or algorithmically, as outlined in [5].

3.2.1 ***Minimal Ranked Entailment.*** Minimal ranked entailment involves establishing a partial ordering $\preceq_{\mathcal{K}}$ of all ranked models of knowledge base $\mathcal{K}$. The ordering is defined such that for any two ranked interpretations $\mathcal{R}_1$ and $\mathcal{R}_2$, $\mathcal{R}_1 \preceq_{\mathcal{K}} \mathcal{R}_2$ if for every valuation $v \in \mathcal{U}$, $\mathcal{R}_1(v) \leq \mathcal{R}_2(v)$ [5]. The most typical models are ranked lower, which leads to identifying a minimal element [10]. This minimal element is denoted as $\mathcal{R}^{\mathcal{K}}_{RC}$. Minimal ranked entailment, denoted by $\approx$, is then formulated based on a defeasible knowledge base $\mathcal{K}$ and its corresponding ranked interpretation $\mathcal{R}^{\mathcal{K}}_{RC}$. For any defeasible implication $\alpha \vdash \beta$, $\mathcal{K} \approx \alpha \vdash \beta$ holds if and only if $\mathcal{R}^{\mathcal{K}}_{RC}$ entails $\alpha \vdash \beta$. Based on ranked interpretations, this approach to entailment is shown to have similarities to preferential entailment.

3.2.2 ***Materialisation.*** To ascertain the exceptional nature of a statement, it is imperative to materialise defeasible statements [7]. This involves defining the counterpart of each defeasible implication $\alpha \vdash \beta$ as the propositional formula $\alpha \rightarrow \beta$. The collection of all such counterparts for $\mathcal{K}$ is represented as $\mathcal{K}$. This is formally denoted as $\mathcal{K} = \{\alpha \rightarrow \beta : \alpha \vdash \beta \in \mathcal{K}\}$. A propositional statement α is deemed exceptional in relation to a knowledge base $\mathcal{K}$ if and only if $\mathcal{K} \models \neg\alpha$, meaning α is false in all the most typical interpretations across every ranked model in $\mathcal{K}$.

3.2.3 ***Base Rank Algorithm.*** The base rank algorithm separates formulas in the knowledge base into ranks, assigning ranks based on the exceptional subsets derived from the defeasible statements and classical propositional logic statements. The base rank is the initial step to algorithmically determining whether a defeasible implication is encompassed within the rational closure of a knowledge base. As detailed in [5], the base rank outlined in Algorithm 1 is the first of the two sub-algorithms used for rational closure.

Algorithm 1: `BaseRank` [8]

Input: A knowledge base $\mathcal{K}$
Output: An ordered tuple $(\mathcal{R}_0, \ldots, \mathcal{R}_{n-1}, \mathcal{R}_\infty, n)$

$i := 0$;
$\mathcal{R}_\infty := \mathcal{K} \setminus \{\alpha \vdash \beta \in \mathcal{K}\}$;
$\mathcal{E}_0 := \{\alpha \vdash \beta \in \mathcal{K}\}$;
repeat
  $\mathcal{E}_{i+1} := \{\alpha \vdash \beta \in \mathcal{E}_i \mid \mathcal{R}_\infty \cup \mathcal{E}_i \models \neg\alpha\}$;
  $\mathcal{R}_i := \mathcal{E}_i \setminus \mathcal{E}_{i+1}$;
  $i := i + 1$;
until $\mathcal{E}_{i-1} = \mathcal{E}_i$;
$\mathcal{R}_\infty := \mathcal{R}_\infty \cup \mathcal{E}_{i-1}$;
$n := i - 1$;
return $(\mathcal{R}_0, \ldots, \mathcal{R}_{n-1}, \mathcal{R}_\infty, n)$;

The process starts by separating the defeasible statements from classical statements. The classical statements are added to the infinite rank $\mathcal{R}_\infty$, while the classical statements are added to the sequence element $\mathcal{E}_0$. Subsequent elements, $\mathcal{E}_i$, are derived by filtering out statements of the form $\alpha \vdash \beta$ from the previous element $\mathcal{E}_{i-1}$, where α can be demonstrated as false based on the existing statements. This process continues until no new statements can be derived, with $\mathcal{E}_n$ usually denoted as $\mathcal{E}_\infty$ representing the final unique element.

A ranking is constructed based on the difference between consecutive sequence elements. Classical statements and statements from the final element $\mathcal{E}_\infty$ occupy the infinite rank. A statement $\alpha \vdash \beta$ is assigned a base rank i if it resides in rank i, signifying that in any ranked interpretation derived from the corresponding materialisation $\mathcal{E}_i$, $\alpha \rightarrow \beta$ will be valid in at least one interpretation belonging to the most typical rank.

The second sub-algorithm shown in Algorithm 2 checks defeasible entailment using rational closure. Given a defeasible implication $\alpha \vdash \beta$, check if the statements in all ranks entail $\neg\alpha$. Then, systematically eliminate sets of classical implications from higher ranks until identifying a rank where $\neg\alpha$ is not entailed. Finally, assess whether the remaining statements entail $\alpha \rightarrow \beta$. If only the bottom rank remains and the statements entail $\neg\alpha$, conclude that $\mathcal{K}$ defeasibly entails $\alpha \vdash \beta$.

Algorithm 2: `RationalClosure` [8]

Input: A knowledge base $\mathcal{K}$ and a DI $\alpha \vdash \beta$
Output: true if $\mathcal{K} \approx_{RC} \alpha \vdash \beta$, otherwise **false**

$(\mathcal{R}_0, \ldots, \mathcal{R}_{n-1}, \mathcal{R}_\infty, n) =:$ `BaseRank`$(\mathcal{K})$;
$i := 0$;
$\mathcal{R} := \bigcup_{j=0}^{j<n} \mathcal{R}_j$;
while $\mathcal{R}_\infty \cup \mathcal{R} \models \neg\alpha$ ***and*** $\mathcal{R}_\infty \neq \emptyset$ **do**
  $\mathcal{R} := \mathcal{R} \setminus \mathcal{R}_i$;
  $i := i + 1$;
end
return $\mathcal{R}_\infty \cup \mathcal{R} \models \alpha \rightarrow \beta$

To illustrate the Rational Closure algorithm, let's look at the following example.

Example 3. Suppose one has the knowledge base $\mathcal{K}$ which has the following statements $p \rightarrow b, s \rightarrow b, b \vdash f, b \vdash w, b \vdash e$ and $p \vdash \neg f$ which mean *"penguins are birds"*, *"Skipper is a penguin"*, *"birds typically fly"*, *"birds typically have wings"*, *"birds typically have eyes"*, and *"penguins typically don't fly"* respectively.

Consider the statement, *"Skipper typically does not fly"* (i.e., $s \vdash \neg f$). Using the `RationalClosure` algorithm, we first compute initial rankings using the `BaseRank` algorithm.

We add classical formulas in infinite rank $\mathcal{R}_\infty = \{p \rightarrow b, s \rightarrow p\}$.

$\mathcal{E}_0 = \{b \vdash f, b \vdash w, b \vdash e, p \vdash \neg f\}$
$\mathcal{E}_1 = \{p \vdash \neg f\}$ Since $(\mathcal{R}_\infty \cup \mathcal{E}_0 \models \neg p)$
$\mathcal{E}_2 = \{\}$
$\mathcal{E}_3 = \{\}$ We stop since $\mathcal{E}_3 = \mathcal{E}_2$ ($\mathcal{E}_2$ is the last element)

The last element is unique and is usually denoted as $\mathcal{E}_\infty$. Therefore $\mathcal{E}_\infty = \mathcal{E}_2 = \{\}$. The initial ranking shown in Table 4 is computed as follows $\mathcal{R}_0 = \mathcal{E}_0 \setminus \mathcal{E}_1$, $\mathcal{R}_1 = \mathcal{E}_1 \setminus \mathcal{E}_\infty$, and $\mathcal{R}_\infty = \mathcal{R}_\infty \cup \mathcal{E}_\infty$.

0	$b \vdash f, b \vdash w, b \vdash e$
1	$p \vdash \neg f$
2	$p \rightarrow b, s \rightarrow p$

Table 4: Initial ranking of statements for Example 3

Next, we check if $\mathcal{R}_0 \cup \mathcal{R}_1 \cup \mathcal{R}_\infty \models \neg s$. Since this is true, we remove $\mathcal{R}_0$. We then verify if $\mathcal{R}_1 \cup \mathcal{R}_\infty \models \neg s$. As this is false, we stop removing ranks and check if $\mathcal{R}_1 \cup \mathcal{R}_\infty \models s \rightarrow \neg f$. Since this entailment holds, the `RationalClosure` algorithm returns **true**.

Example 4. Suppose one has the knowledge base in Example 3. Consider the statement, *"Skipper typically has wings"* (i.e., $s \vdash w$). `BaseRank` algorithm constructs the initial ranking in Table 4.

Next, we check if $\mathcal{R}_0 \cup \mathcal{R}_1 \cup \mathcal{R}_\infty \models \neg s$. Since this is true, we remove $\mathcal{R}_0$. We then verify if $\mathcal{R}_1 \cup \mathcal{R}_\infty \models \neg s$. As this is false, we stop removing ranks and check if $\mathcal{R}_1 \cup \mathcal{R}_\infty \models s \rightarrow w$. This entailment does not hold as $\mathcal{R}_0$ has been removed. Thus, `RationalClosure` algorithm returns **false**.

3.3 Lexicographic Closure

Lexicographic closure is a form of defeasible entailment first introduced by Lehmann [17]. Lexicographical closure is a nonmonotonic reasoning procedure that builds on the concepts of rational closure [6] and refines it [17]. It is used to derive conclusions from explicit knowledge and default information. Lexicographic closure can be defined in terms of ranked interpretation and ranking formulas. Both of these definitions use a *"seriousness"* ordering, in which we prefer ordering a "lighter" set of defaults over a more serious one [17]. Two criteria must be considered when determining the seriousness of two sets: cardinality and specificity.

3.3.1 ***Ranked Interpretations.*** Specificity is considered a primary criterion for generating ranked interpretations for lexicographic closure. The specificity of statements is determined by their rank (a higher rank is more specific while a lower rank is less specific). Thus, we prefer violating a lower rank than a higher rank. After using specificity, we can refine the ordering by checking the number of statements violated by each interpretation in rank. Given the knowledge base $\mathcal{K}$ with order k, we construct a $k+1$-tuple of natural numbers $n_D = \langle n_0, n_1, \ldots, n_k \rangle$ for each subset $D \subseteq \mathcal{K}$, where n_0 is the number of formulas that have rank ∞ and n_i is the number of formulas that have rank $k-1$. This ordering is a "strict modular partial ordering denoted by $\prec_S$ (seriousness ordering)" [17]. Using these tuples, we can define ordering on subsets $D_1, D_2 \subseteq \mathcal{K}$ by $D_1 \prec_S D_2$ if $n_{D_1} < n_{D_2}$, where $<$ is the lexicographic ordering of tuples (e.g. $\langle 0,1,1,0 \rangle < \langle 1,1,0,1 \rangle$). We can therefore use this ordering to impose an order $\preceq_{LC}$ on interpretations, $\mathcal{K}$. For interpretations $u, v \in \mathcal{U}$, the ordering $\prec_{LC}$ can be defined as: $u \prec_{LC} v$ iff $V(u) \prec_{LC} V(v)$ where, $V(w) \subseteq \mathcal{K}$ is the number of defeasible implication violated by w [17]. This definition can be used to generate ranked interpretations (denoted $\mathcal{R}^{\mathcal{K}}_{LC}$) that correspond to the lexicographic closure.

3.3.2 ***Ranked Formulas.*** Lexicographic closure can also be defined using an algorithm in Algorithm 3. One starts by ranking formulas in the knowledge base according to the `BaseRank` algorithm, which considers specificity. The ranking is then refined by weakening the formulas. The idea is that, instead of removing the whole level when there is an inconsistency, one should remove the formula causing the inconsistency by weakening the most typical rank. When checking if $\mathcal{K} \mathrel{|\!\sim} \alpha \vdash \beta$, the algorithm starts by checking whether $\mathcal{R}_0 \cup \mathcal{R}_1 \cup \cdots \cup \mathcal{R}_{n-1} \cup \mathcal{R}_\infty \models \neg\alpha$. If this condition is met, the algorithm weakens the lowest finite instead of immediately discarding all statements at this rank.

Algorithm 3: `LexicographicClosure` [8]

Input: A knowledge base $\mathcal{K}$ and a DI $\alpha \vdash \beta$
Output: **true** if $\mathcal{K} \mathrel{|\!\sim}_{LC} \alpha \vdash \beta$, otherwise **false**

$(\mathcal{R}_0, \ldots, \mathcal{R}_{n-1}, \mathcal{R}_\infty, n) =:$ `BaseRank`$(\mathcal{K})$;
$i := 0$;
$\mathcal{R} := \bigcup_{j=0}^{j<n} \mathcal{R}_j$;
while $\mathcal{R}_\infty \cup \mathcal{R} \models \neg\alpha$ ***and*** $\mathcal{R}_\infty \neq \emptyset$ **do**
  $\mathcal{R} := \mathcal{R} \setminus \mathcal{R}_i$;
  $m := |\mathcal{R}_i| - 1$;
  $\mathcal{R}_{i,m} := \bigvee_{X \in \text{Subsets}(\mathcal{R}_{i,m})} \bigwedge_{x \in X} x$;
  while $\mathcal{R}_\infty \cup \mathcal{R} \cup \{\mathcal{R}_{i,m}\}$ ***and*** $m > 0$ **do**
    $m := m + 1$;
    $\mathcal{R}_{i,m} := \bigvee_{X \in \text{Subsets}(\mathcal{R}_{i,m})} \bigwedge_{x \in X} x$;
  end
  $\mathcal{R} := \mathcal{R} \cup \{\mathcal{R}_{i,m}\}$;
  $i := i + 1$;
end
return $\mathcal{R}_\infty \cup \mathcal{R} \models \alpha \rightarrow \beta$

We take subsets of the rank to weaken a rank, with each subset being of size $x-1$ (where x is the number of formulas in the lowest finite rank); for each subset, we apply conjunction ($\wedge$) between the formulas and then use disjunction ($\vee$) to combine the resulting formulas from the subsets. This weakening process continues iteratively, checking subsets of decreasing size until either the entailment no longer holds or the size of the subsets reaches 0 (at which point the resulting formula is a tautology, $\top$), and the rank is discarded entirely. The process is then repeated for the next lowest rank of formulas.

Example 5. Suppose we have a rank $\mathcal{R}_0$ with the statements $\{a \vdash x, a \vdash y, a \vdash z\}$. The rank size is $|\mathcal{R}_0| = 3$. For subsets of size 2, we obtain the refined rank $\mathcal{R}_{0,2} = (a \vdash x \wedge a \vdash y) \vee (a \vdash x \wedge a \vdash z) \vee (a \vdash y \wedge a \vdash z)$. For subsets of size 1, we have $R_{0,1} = a \vdash x \vee a \vdash y \vee a \vdash z$. For subsets of size 0, we have $\mathcal{R}_{0,0} = \top$, meaning that the rank $\mathcal{R}_0$ is discarded entirely because it can no longer be weakened.

To illustrate the difference between the Lexicographic Closure and Rational Closure algorithm, let's look at the following example.

Example 6. Suppose one has the knowledge base in Example 3, i.e., $\mathcal{K} = \{p \rightarrow b, s \rightarrow b, b \vdash f, b \vdash w, b \vdash e, p \vdash \neg f\}$. Consider the statement, *"Skipper typically has wings"* (i.e., $s \vdash w$). `BaseRank` algorithm constructs the initial ranking in Table4.

With the same knowledge base $\mathcal{K}$ and query formula $s \vdash w$ in Example 4, the `RationalClosure` algorithm returns **false**.

For Lexicographic Closure, we $\mathcal{R}_0 \cup \mathcal{R}_1 \cup \mathcal{R}_\infty \models \neg s$. Since this is true, we refine $\mathcal{R}_0$ by weakening its formulas shown in Table 5. Then we check again whether $\mathcal{R}_0 \cup \mathcal{R}_1 \cup \mathcal{R}_\infty \models \neg s$. As this is not

true, we stop refining ranks and check if $\mathcal{R}_0 \cup \mathcal{R}_1 \cup \mathcal{R}_\infty \models s \rightarrow w$. Since this entailment holds, the Lexicographic Closure algorithm returns **true**.

0	$(b \vdash e \wedge b \vdash f) \vee (b \vdash e \wedge b \vdash w) \vee (b \vdash f \wedge b \vdash w)$
1	$p \vdash \neg f$
2	$p \rightarrow b, s \rightarrow p$

Table 5: Refined ranking of statements for Example 6

4 SYSTEM DESIGN AND IMPLEMENTATION

4.1 Requirements

A systematic approach was employed to gather the necessary requirements to build the defeasible reasoning software. This involved gathering information to define the software's functional and non-functional needs.

4.1.1 ***Sources of Information.*** To accurately capture the requirements, we gathered information from various sources. We held regular meetings with our supervisor to discuss the software's objectives and key functionality. These discussions were crucial in defining the project's scope, identifying critical features and understanding the priorities. We studied tools similar to the one being developed, mainly focusing on their strengths and limitations. For instance, analysing the tools developed by Hamayobe [12] and Wang [25] provided insights into implementing and improving how the reasoning process is presented to users. We examined the algorithms by Everett et al. [8], which helped us determine how to explain the defeasible reasoning process.

4.1.2 ***Functional Requirements.*** The primary functional requirement was that the application perform and display defeasible reasoning processes, including Base Rank, Rational Closure, and Lexicographic Closure operations. The application was also required to have a user-friendly interface to allow users to interact easily with and understand the reasoning process. The application should also allow users to input and modify the knowledge base and queries. The application should display the reasoning process to show how conclusions were derived using different algorithms.

4.1.3 ***Non-Functional Requirements.*** The application was required to handle errors to ensure the system remains stable and reliable even under exceptional circumstances. The user interface must be intuitive and user-friendly to allow users to interact with the application and understand the reasoning process.

4.2 Architecture

The web application was developed using Java for the back end and TypeScript for the front end. It follows a layered architecture, organising related functionalities into layers where each layer provides services to the layer above. As shown in Figure 1, this architecture pattern was chosen to address the separation of concerns and ensure modularity and ease of use.

For the front end, the application uses a JavaScript framework called *React*, which is part of the *presentation layer*. This layer consists of *React* components that make up the user interface. The user interface allows users to input data and request results, which are then displayed. The presentation layer also utilises third-party JavaScript libraries to render mathematical equations in TEX format. When users query for results, the presentation layer sends requests to the application layer and updates the user interface based on the responses.

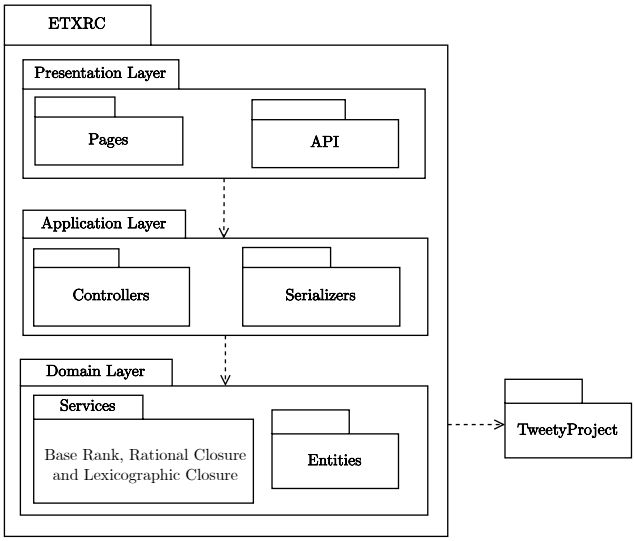

Figure 1: Layered Architecture of the web application.

The back end of the web application uses a lightweight Java framework called *Javalin* to create a REST API. It consists of the *application* and *domain layers*. The application layer includes serializers that convert Java objects to JSON objects and vice versa. It also contains controllers responsible for handling requests from the presentation layer. The controllers use the serialisers to manage request and response data and employ services from the *domain layer* to provide appropriate responses.

The domain layer defines entities that represent core domain rules and services the application layer uses. These services include reasoning algorithms such as the Base Rank Algorithm, Rational Closure Algorithm, and Lexicographic Closure Algorithm. This layer also uses the third-party Java library *TweetyProject* [23, 24] to determine entailment.

4.3 System Implementation

The defeasible reasoning software was implemented as a web application, leveraging a layered architecture to maintain modularity and facilitate the separation of concerns. The software stack included Java for the back end, TypeScript and React for the front end, and various third-party libraries to handle specific computational tasks.

4.3.1 ***Back end Implementation.*** The application's back end was developed using Java, a robust and widely used programming language. The Javalin framework was chosen for its lightweight nature and simplicity in setting up RESTful APIs, essential for communication between the front and back ends. The back end was divided into several layers.

The *application layer* acted as the intermediary between the presentation layer and the core business logic. It housed controllers

responsible for handling HTTP requests from the front end. These controllers processed incoming data, invoked the necessary services, and returned the appropriate responses. The *domain Layer* encapsulated the core logic of the defeasible reasoning software. It implemented the reasoning algorithms - Base Rank, Rational Closure, and Lexicographic Closure in the *TweetyProject* library. These algorithms are the services that the controllers use. This layer also defined the entities representing the knowledge base, rankings, queries, and other core concepts. The entities and services in the domain layer use the proposition logic library provided by the *Tweety* project. The *TweetyProject* library, a powerful tool for knowledge representation and reasoning, was integral to implementing the reasoning algorithms [23, 24]. It provided the necessary infrastructure for logical operations, allowing the system to perform complex reasoning tasks efficiently. Custom extensions were developed on top of the *TweetyProject* framework to accommodate the requirements of defeasible reasoning. These extensions allowed the software to handle defeasible implications and other logic forms not natively supported by the library.

4.3.2 ***Frontend Implementation.*** The front end was developed using TypeScript, a strongly typed superset of JavaScript. React, a popular JavaScript library for building user interfaces was employed to create a dynamic and responsive UI. The choice of React was motivated by its component-based architecture, which allowed for the modular development of the interface. Each component was responsible for a specific UI part, such as data input forms, query submission buttons, and result displays (for the summary, base rank, rational closure, and lexicographic closure). Figure 2 shows the frontend interface of the web tool.

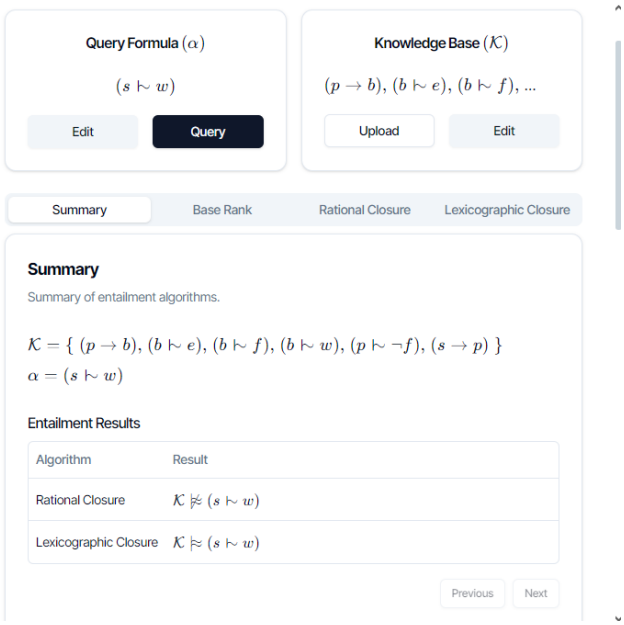

Figure 2: Query formula, knowledge base and result display

Users could input their knowledge base either manually or by uploading a file. The front end validates input formats and ensures the knowledge base is correctly structured before submission. Users can enter queries related to the knowledge base, which are then sent to the back end for processing. The results of the reasoning process, including the steps involved in deriving conclusions, were displayed in a user-friendly format by rendering mathematical equations in TEX format.

4.4 Data Flow and Interaction

The REST API facilitated the front and back-end interaction, allowing efficient data flow between the two layers. The typical workflow involved the following steps:

(1) Users entered a knowledge base and a query through the front-end interface shown in Figure 3. The input data was first validated locally to meet the required format and structure.

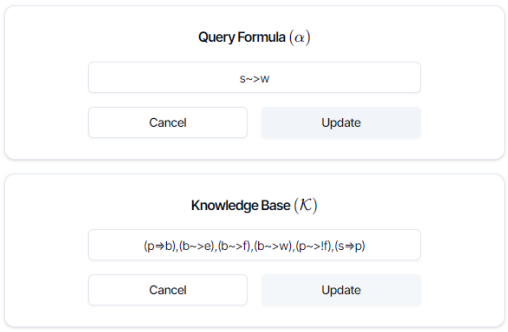

Figure 3: User input forms for query and knowledge base.

(2) The input data (knowledge base and query formula) is sent to the backend to be checked for validity. Once validated, it is returned to the front end, and the interface is updated with rendered mathematical formulas as shown in Figure 4.

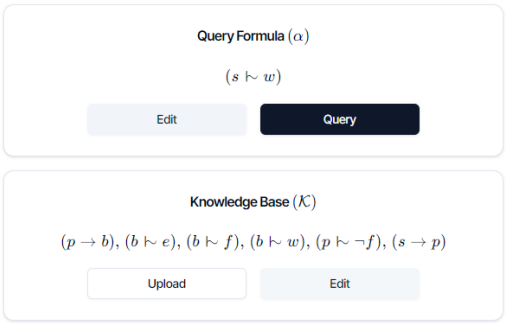

Figure 4: Rendered math formulas for inputs.

(3) After pressing the "query" button, the validated knowledge base is sent to the backend to calculate the base rank. Once the base rank is computed and sent to the front end. Then, the front end initiates a request with the base rank and query formula as input data. The request involves computing rational entailment and lexicographic entailment. Subsequently, the backend runs these reasoning algorithms based on the input data and sends the results to the frontend.

(4) The front end received the results, saved them to local storage in the front end and updated the UI accordingly. The reasoning process and conclusions were displayed in a clear and structured manner. Figure 5 summarises the results, including

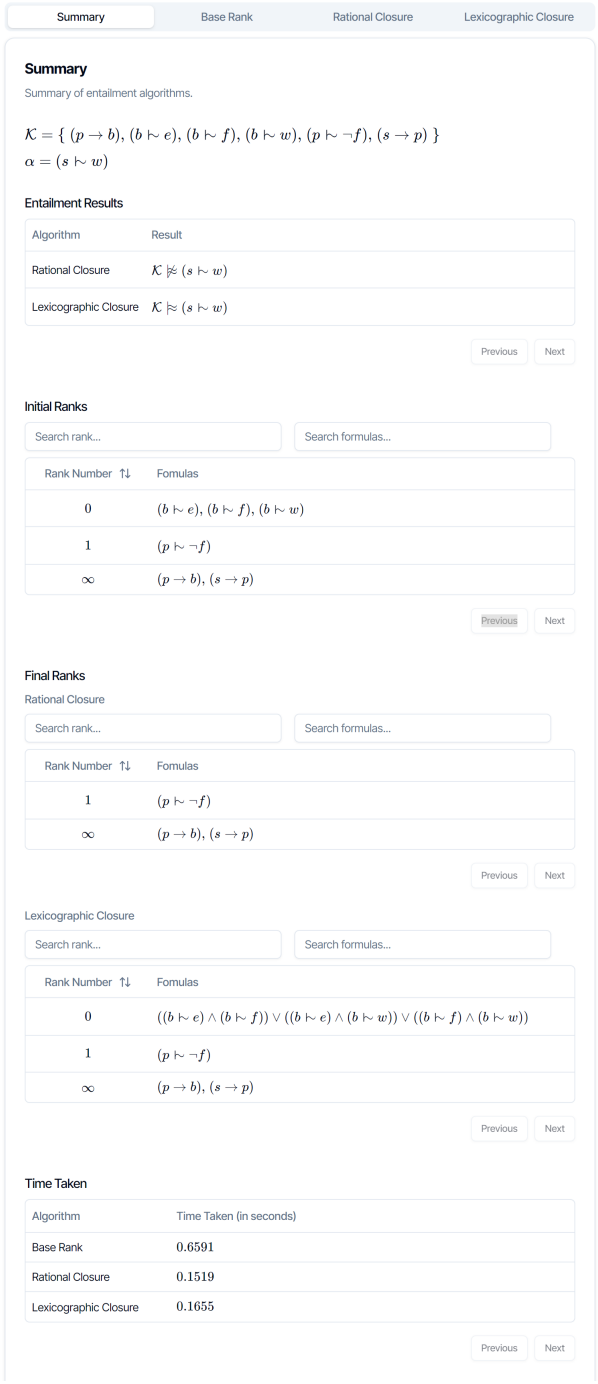

Figure 5: Summary of results.

entailment results, initial and final ranks for rational and lexicographic closures, and the time taken for each algorithm. More screenshots (for base rank, rational closure, and lexicographic closure algorithms) can be found in Appendix B.

4.5 Testing, Evaluation, and Execution

A rigorous testing strategy was implemented to ensure the defeasible reasoning software met all functional and non-functional requirements. This included unit testing and integration testing. Each system component was subjected to unit testing, from the individual reasoning algorithms in the domain layer to the API endpoints in the application layer. The JUnit framework was used to test the backend, ensuring that each algorithm produced the correct outputs for various inputs, including edge cases. Integration tests were conducted to ensure that the different layers of the application worked together seamlessly. These tests simulated real-world scenarios, where users would input data through the front end, trigger backend processing, and receive results. The tests verified that data was correctly passed between the front and back end and that the system functioned as intended.

Extensive testing was also conducted to ensure the system handled invalid inputs and other exceptional conditions. The error handling mechanisms implemented in both the front and back end were validated to ensure the system could recover gracefully from errors and provide informative feedback.

We conducted all tests by providing different knowledge bases to the web tool. The test results and the feedback from our supervisor were enough to verify the correctness of the algorithms implemented by our application.

5 DISCUSSIONS

This paper aimed to develop web application software that incorporates rational closure and lexicographic closure and showcases the defeasible reasoning process. This objective has been accomplished. However, to achieve this, the algorithms required modification in their inputs. The rational closure in Algorithm 2 and lexicographic closure in Algorithm 3 initially calculated the base rank, which needed to be changed to avoid redundant calculations. This adjustment was necessary because the `BaseRank` algorithm can take longer to compute for a knowledge base with numerous exceptional statements. To compare the computational time of these two algorithms, the inputs of their implementations were modified to include an ordered tuple $(\mathcal{R}_0, \ldots, \mathcal{R}_{n-1}, \mathcal{R}_\infty, n)$ and a defeasible implication $\alpha \vdash \beta$. This means the initial ranks are constructed first and then used as input to these algorithms. These modified algorithms are in Appendix A.

The results of the defeasible reasoning are categorized into four sections. The first section is the "Summary," which displays the entailment results ($\mathcal{K} \models \alpha \vdash \beta$ or $\mathcal{K} \not\models \alpha \vdash \beta$), the initial ranking created by the `BaseRank` algorithm, the final ranking, and the time taken for each algorithm. The second section is the "Base Rank," illustrating the steps involved in constructing the initial ranking. The third section is the "Rational Closure," detailing the steps taken to check for entailment using the `RationalClosure` algorithm. The fourth section is the "Lexicographic Closure," outlining the steps taken to check for entailment using the `LexicographicClosure` algorithm. Users can navigate via tabs to access the results of each

section. This tool can be utilized as a debugging tool to identify exceptional statements in the knowledge base and the distinctions between lexicographic closure and rational closure. As lexicographic closure is an expansion of rational closure, the tool can also pinpoint issues with rational closure when querying specific statements.

6 CONCLUSIONS

This paper introduces knowledge representation and reasoning and explains how propositional logic is used to represent knowledge. It further discusses the differences between classical entailment and defeasible entailment using examples. By delving into the KLM framework, we have shown how defeasible reasoning acknowledges exceptions and uncertainties, providing a more nuanced approach than classical reasoning systems. The KLM framework's reliance on nonmonotonic reasoning allows conclusions to be tentative, accommodating the dynamic nature of real-world information.

We have reviewed various methods of extending the KLM approach, including rational closure and lexicographic closure. Each method handles exceptions differently. This paper implements algorithms by Everett et al. [8] to compute defeasible entailment. Rational Closure offers foundational techniques, while Lexicographic Closure refines these approaches by introducing a seriousness ordering, thus providing more nuanced control over defeasible entailments.

As discussed, developing a web application tool represents a significant step towards explaining a defeasible reasoning process. It can also be used as a debugger to identify exceptional information in knowledge bases and differences or issues in the entailment algorithms.

Future work includes extending the software by focusing on other forms of defeasible entailment, such as Relevant Closure.

REFERENCES

[1] Grigoris Antoniou and Mary-Anne Williams. 1997. *Nonmonotonic Reasoning*. MIT Press. https://doi.org/10.7551/mitpress/5040.001.0001

[2] Mordechai Ben-Ari. 2012. *Mathematical Logic for Computer Science*. Springer Science & Business Media.

[3] Ronald J Brachman and Hector J Levesque. 2004. *Knowledge Representation and Reasoning*. Elsevier.

[4] Giovanni Casini, Thomas Meyer, and Ivan Varzinczak. 2018. Defeasible entailment: From rational closure to lexicographic closure and beyond. In *Proceeding of the 17th International Workshop on Non-Monotonic Reasoning (NMR 2018)*. 109–118.

[5] Giovanni Casini, Thomas Meyer, and Ivan Varzinczak. 2019. Taking defeasible entailment beyond rational closure. In *European Conference on Logics in Artificial Intelligence*. 182–197.

[6] Giovanni Casini and Umberto Straccia. 2012. Lexichographic Closure for Defeasible Description Logics. *CEUR Workshop Proceedings* 969, xx–xx.

[7] Victoria Chama. 2019. *Explanation For Defeasible Entailment*. Master's Thesis. Faculty of Science, University of Cape Town.

[8] Lloyd Everett, Emily Morris, and Thomas Meyer. 2022. Explanation for KLM-Style Defeasible Reasoning. In *Artificial Intelligence Research*, Edgar Jembere, Aurona J. Gerber, Serestina Viriri, and Anban Pillay (Eds.). Springer International Publishing, Cham, 192–207.

[9] Kenneth G. Ferguson. 2003. Monotonicity in Practical Reasoning. *Argumentation* 17, 3 (September 2003), 335–346. https://doi.org/10.1023/A:1025164703468

[10] Laura Giordano, Valentina Gliozzi, Nicola Olivetti, and Gian Luca Pozzato. 2015. Semantic characterization of rational closure: From propositional logic to description logics. *Artificial Intelligence* 226 (2015), 1–33.

[11] Cordelia Hall and John O'Donnell. 2000. *Propositional Logic*. Springer London, London. 35–87 pages. https://doi.org/10.1007/978-1-4471-3657-6_2

[12] Chipo Hamayobe. 2023. *Defeasible Entailment and Explanations*. BSc (Honours) Project. University of Cape Town, Computer Science Department. https://projects.cs.uct.ac.za/honsproj/cgi-bin/view/2023/hamayobe_morule.zip/resources/Final_Project_Chipo.pdf

[13] Michael Harrison and Thomas Meyer. 2020. DDLV: A system for rational preferential reasoning for datalog. *South African Computer Journal* 32, 2 (2020). https://doi.org/10.18489/sacj.v32i2.850

[14] Adam Kaliski. 2020. *An Overview of KLM-Style Defeasible Entailment*. Master's Thesis. Faculty of Science, University of Cape Town.

[15] Kevin C. Klement. 2004. Propositional Logic. In *Internet Encyclopedia of Philosophy*. Internet Encyclopedia of Philosophy.

[16] Sarit Kraus, Daniel Lehmann, and Menachem Magidor. 1990. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence* 44, 1-2 (1990), 167–209.

[17] Daniel Lehmann. 1995. Another perspective on default reasoning. *Annals of Mathematics and Artificial Intelligence* 15, 1 (01 03 1995), 61–82. https://doi.org/10.1007/BF01535841

[18] Daniel Lehmann and Menachem Magidor. 1992. What does a conditional knowledge base entail? *Artificial Intelligence* 55, 1 (1992), 1–60. https://doi.org/10.1016/0004-3702(92)90041-U

[19] Maqhobosheane Mohlerepe. 2024. *Investigating Optimisation Techniques for Rational Closure in Defeasible Reasoning*. BSc (Honours) Project. University of Cape Town, Computer Science Department.

[20] Matthew Morris, Tala Rossa, and Thomas Meyer. 2020. Algorithmic definitions for KLM-style defeasible disjunctive Datalog. *SACJ* 32, 2 (December 2020). Research Article.

[21] Mamodike Sadiki. 2024. *A Study of Parameters and Optimization Strategies in Defeasible Knowledge Base Generation*. BSc (Honours) Project. University of Cape Town, Computer Science Department.

[22] Luke Slater and Thomas Meyer. 2023. Extending Defeasible Reasoning Beyond Rational Closure. In *Artificial Intelligence Research*, Anban Pillay, Edgar Jembere, and Aurona J. Gerber (Eds.). Springer Nature Switzerland, Cham, 151–171.

[23] Matthias Thimm. 2014. Tweety - A Comprehensive Collection of Java Libraries for Logical Aspects of Artificial Intelligence and Knowledge Representation. In *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR'14)* (Vienna, Austria).

[24] Matthias Thimm. 2017. The Tweety Library Collection for Logical Aspects of Artificial Intelligence and Knowledge Representation. *Künstliche Intelligenz* 31, 1 (March 2017), 93–97.

[25] Steve Wang. 2022. *Defeasible Justification for the KLM Framework*. Master's Thesis. Faculty of Science, University of Cape Town.

A ALGORITHMS

Algorithm 4: `ModifiedRationalClosure`

Input: An ordered tuple $(\mathcal{R}_0, \ldots, \mathcal{R}_{n-1}, \mathcal{R}_\infty, n)$ and a DI $\alpha \vdash \beta$
Output: **true** if $\mathcal{K} \approx_{RC} \alpha \vdash \beta$, otherwise **false**

$i := 0$;
$\mathcal{R} := \bigcup_{j=0}^{j<n} \mathcal{R}_j$;
while $\mathcal{R}_\infty \cup \mathcal{R} \models \neg\alpha$ ***and*** $\mathcal{R}_\infty \neq \emptyset$ **do**
    $\mathcal{R} := \mathcal{R} \setminus \mathcal{R}_i$;
    $i := i + 1$;
end
return $\mathcal{R}_\infty \cup \mathcal{R} \models \alpha \rightarrow \beta$

Algorithm 5: `ModifiedLexicographicClosure`

Input: An ordered tuple $(\mathcal{R}_0, \ldots, \mathcal{R}_{n-1}, \mathcal{R}_\infty, n)$ and a DI $\alpha \vdash \beta$
Output: **true** if $\mathcal{K} \approx_{LC} \alpha \vdash \beta$, otherwise **false**

$i := 0$;
$\mathcal{R} := \bigcup_{j=0}^{j<n} \mathcal{R}_j$;
while $\mathcal{R}_\infty \cup \mathcal{R} \models \neg\alpha$ ***and*** $\mathcal{R}_\infty \neq \emptyset$ **do**
    $\mathcal{R} := \mathcal{R} \setminus \mathcal{R}_i$;
    $m := |\mathcal{R}_i| - 1$;
    $\mathcal{R}_{i,m} := \bigvee_{X \in \text{Subsets}(\mathcal{R}_{i,m})} \bigwedge_{x \in X} x$;
    while $\mathcal{R}_\infty \cup \mathcal{R} \cup \{\mathcal{R}_{i,m}\}$ ***and*** $m > 0$ **do**
        $m := m + 1$;
        $\mathcal{R}_{i,m} := \bigvee_{X \in \text{Subsets}(\mathcal{R}_{i,m})} \bigwedge_{x \in X} x$;
    end
    $\mathcal{R} := \mathcal{R} \cup \{\mathcal{R}_{i,m}\}$;
    $i := i + 1$;
end
return $\mathcal{R}_\infty \cup \mathcal{R} \models \alpha \rightarrow \beta$

B SUPPLEMENTARY INFORMATION

B.1 Screenshots

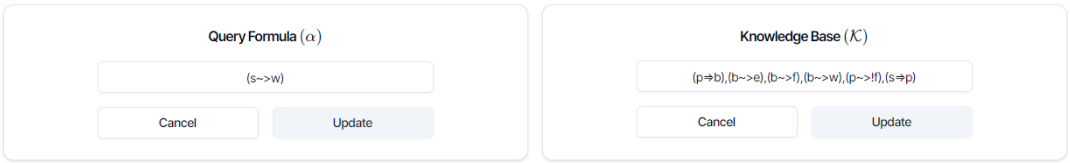

Figure 6: Forms for query formula and knowledge base

Figure 7: Form to upload a knowledge base file

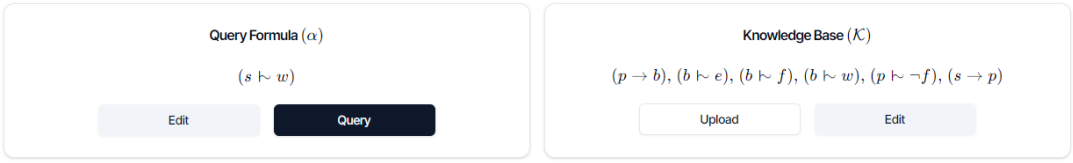

Figure 8: Math formulas for query and knowledge base

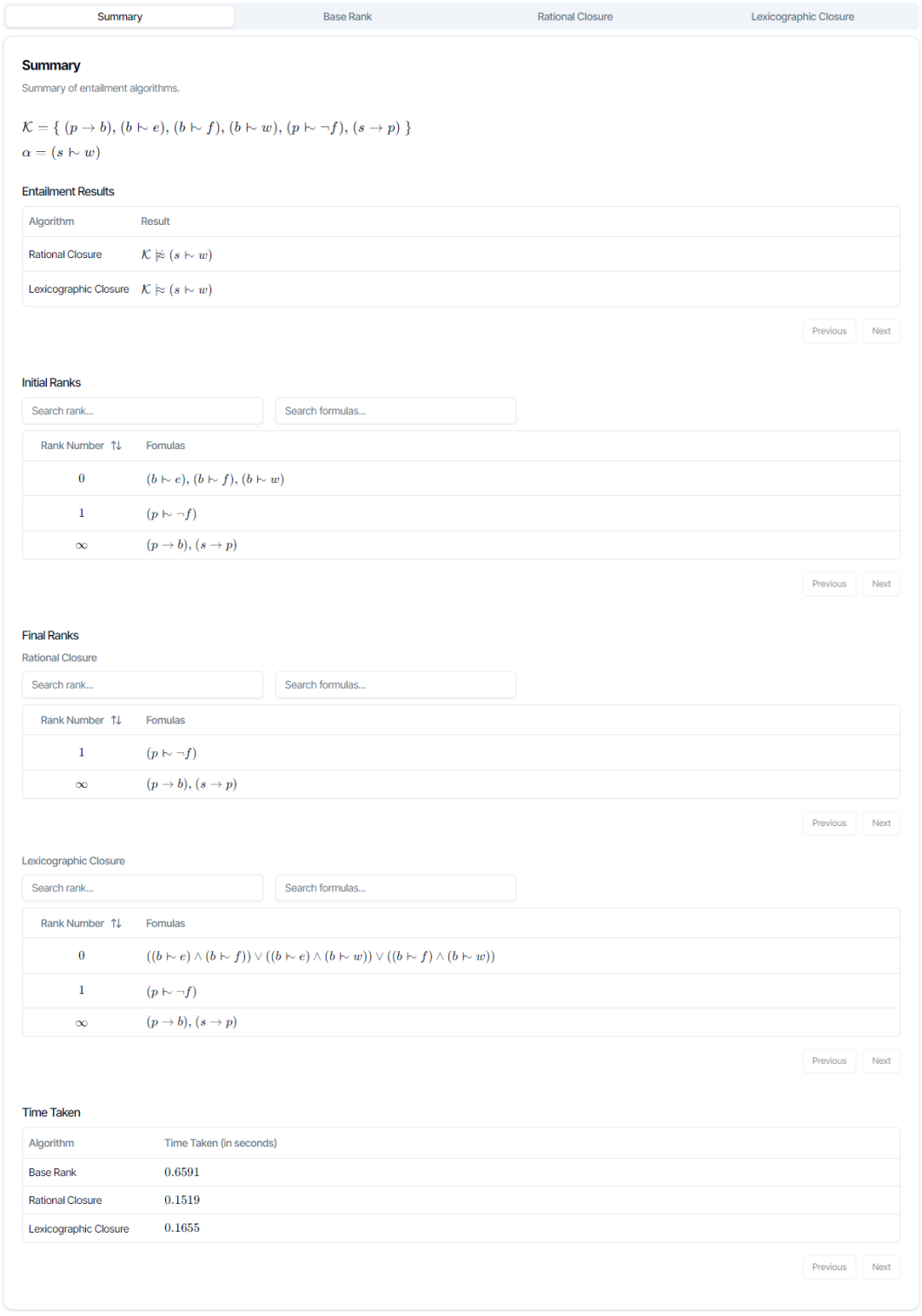

Figure 9: Summary of results

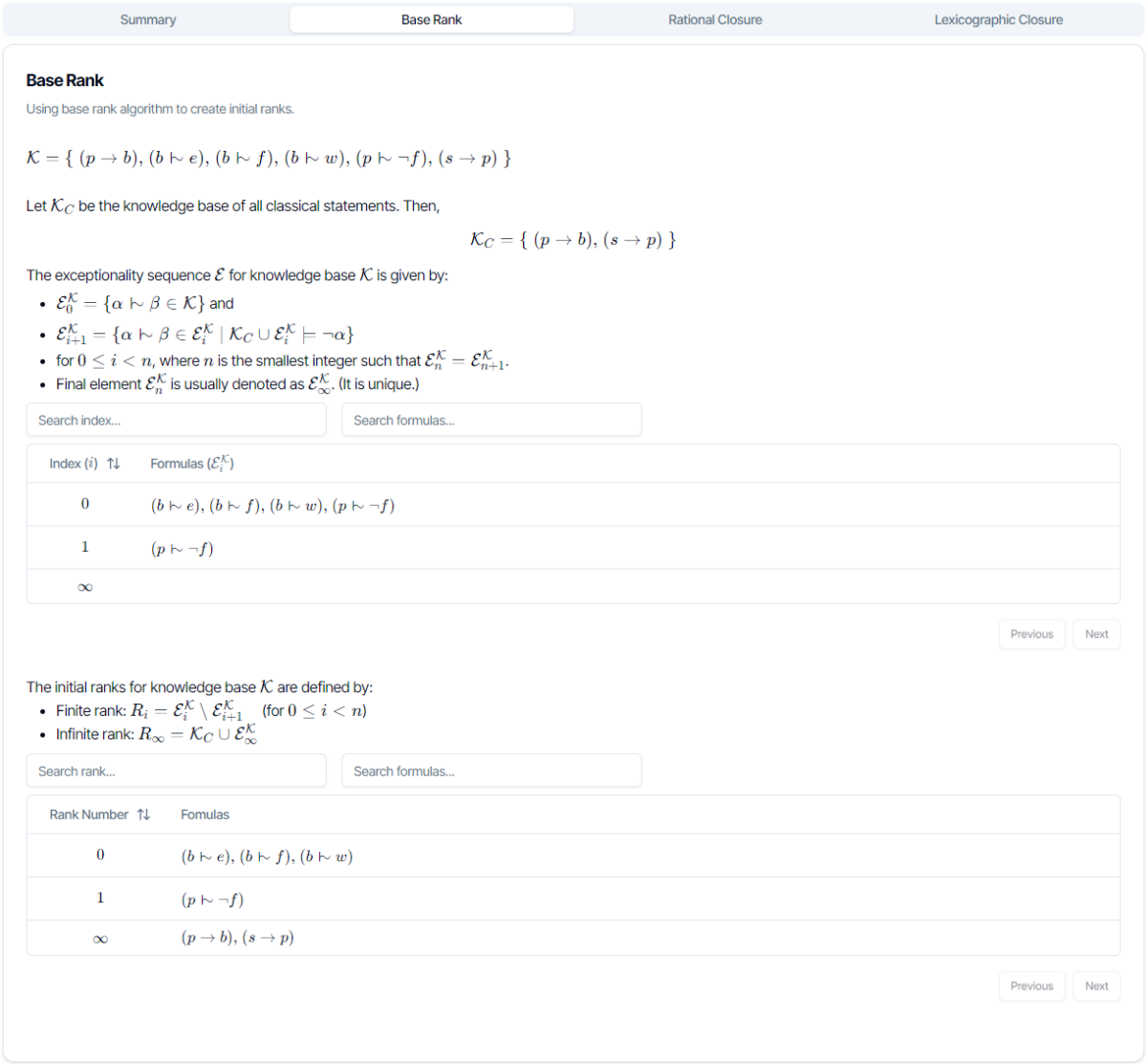

Figure 10: Base rank results

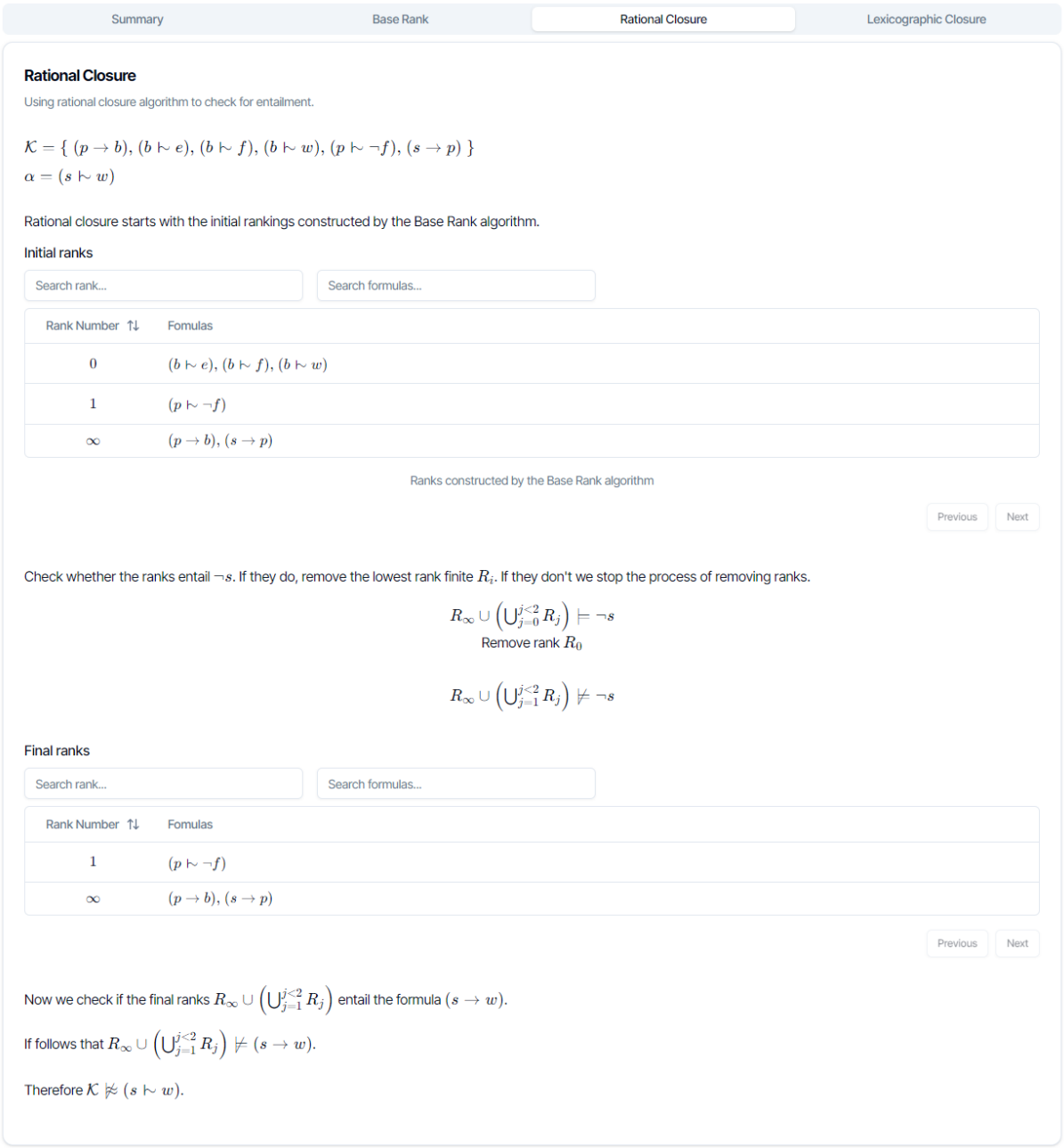

Figure 11: Rational closure results

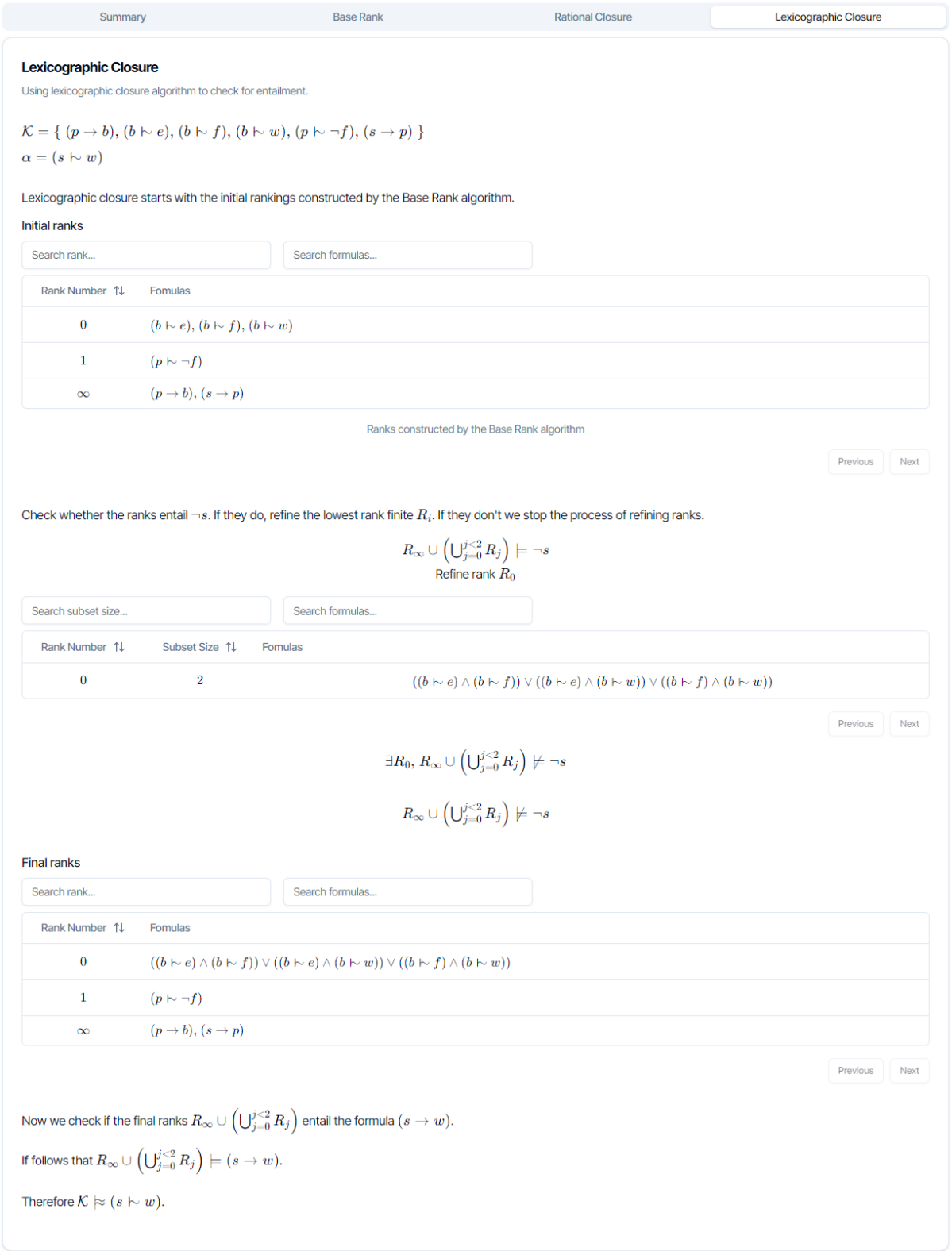

Figure 12: Lexicographic closure results