Defeasible DLV

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

Tala Ross
rsstal002@myuct.ac.za
University of Cape Town
Cape Town, South Africa

ABSTRACT

Datalog is a declarative logic programming language that uses classical logical reasoning as its basic form of reasoning. DLV is a disjunctive logic programming language, which forms an extension of Datalog. Defeasible reasoning is a form of non-classical reasoning that is able to deal with exceptions to general assertions in a formal manner. Since DLV uses classical reasoning, it is currently not able to handle defeasible implications and exceptions. We aim to extend the expressivity of DLV by incorporating KLM-style defeasible reasoning into classical DLV.

CCS CONCEPTS

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

KEYWORDS

artificial intelligence, knowledge representation and reasoning, defeasible reasoning, rational closure, relevant closure, lexicographic closure, Datalog, DLV

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 draw before. This form of reasoning can be too weak to model certain systems. To illustrate this, consider this example where the following statements are made:

(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, and knowing that '*tutors pay taxes*' and '*tutors do not pay taxes*', it must then conclude that no tutors exist. 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].

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. The language $\mathcal{L}$ of propositional logic is the set of all formulas, denoted by α or β, which can be formed by combining propositional atoms from a finite set $\mathcal{P}$ of atoms. This is done recursively using Boolean operators, as follows: $\alpha ::= \top \mid \bot \mid p \mid \neg\alpha \mid \alpha \wedge \alpha \mid \alpha \vee \alpha \mid \alpha \rightarrow \alpha \mid \alpha \leftrightarrow \alpha$.

An *interpretation* is a function $I : \mathcal{P} \rightarrow \{T, F\}$ which assigns a single truth value to each atom. A formula $\alpha \in \mathcal{L}$ is satisfied by an interpretation I, denoted $I \Vdash \alpha$, if it can be evaluated to true by I in the usual recursive truth-functional way, corresponding to the "meaning" of the Boolean operators. For a knowledge base $\mathcal{K}$ of formulas, we define $[\![\mathcal{K}]\!] = \{I : I \Vdash \alpha, \alpha \in \mathcal{K}\}$. Reasoning in propositional logic involves answering questions such as '*Is α a logical consequence of knowledge base K?*'. That is, we check if $[\![\alpha]\!] \subseteq [\![\mathcal{K}]\!]$.

2.2 The KLM Approach

There are many different formalizations of non-monotonic reasoning. The *KLM approach* [11] is an axiomatic approach based on the concept of plausible inference, and is currently one of the most preferred methods [4, 15]. In this approach, plausible inference is represented by a defeasible implication operator of the form $\alpha \mid\!\sim \beta$, which means that α is typically a good enough reason to believe β.

2.3 The KLM Properties

Unlike classical entailment, defeasible entailment is not unique. There exist multiple formalizations of defeasible entailment, such as *rational closure* [13], *relevant closure* [4] and *lexicographic closure* [12]. The KLM framework provides a list of rationality properties, which Lehmann and Magidor [13] argued must be satisfied by defeasible entailment methods. This provides a way of differentiating between acceptable and non-acceptable methods of entailment. The *KLM properties* are listed below:

$$\text{(Ref) } \mathcal{K} \approx \alpha \mid\!\sim \alpha \qquad \text{(LLE) } \frac{\alpha \equiv \beta,\ \mathcal{K} \approx \alpha \mid\!\sim \gamma}{\mathcal{K} \approx \beta \mid\!\sim \gamma}$$

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

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

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

2.4 Rational Closure

Rational closure is in many ways the most simple and intuitive way of defining defeasible entailment. Here, we present two equivalent definitions for entailment checking within propositional logic. The first is a semantic definition, based on underlying structures we call *Ranked Interpretations*. The second is an algorithmic definition, which we refer to as the *Base Rank Algorithm*.

2.4.1 Semantic Definition. *Ranked interpretations* [6] are functions $R : U \longrightarrow \mathbb{N} \cup \{\infty\}$, which are used to define the semantics of KLM-style defeasible implications. They organize interpretations into ranks, in order of decreasing typicality. No level is empty, starting at level 0 with the most typical interpretations, and ending at level n for some $n \in \mathbb{N}$ with the least typical interpretations. A final level of infinite rank is added for all the impossible interpretations. A ranked interpretation R satisfies α, denoted $R \Vdash \alpha$, if α is true in all non-impossible levels of R. We can then say that $R \Vdash \alpha \mid\sim \beta$ if in the most 'normal' level where α holds, β also holds.

As an example, figure 1 gives a ranked interpretation for $\mathcal{P} = \{\mathrm{b}, \mathrm{f}, \mathrm{p}\}$ satisfying $\mathcal{K} = \{\mathrm{p} \rightarrow \mathrm{b}, \mathrm{b} \mid\sim \mathrm{f}, \mathrm{p} \mid\sim \neg\mathrm{f}\}$ [6]. For easier reading, we omit the valuations with rank ∞ in our graphical representations of ranked interpretations. Note that the presence of an atom (e.g. p) represents it being true in the interpretation, whereas the presence of a barred atom (e.g. $\overline{\mathrm{p}}$) represents it being false in the interpretation.

2	pbf
1	$\overline{\mathrm{p}}\mathrm{b}\overline{\mathrm{f}}$ $\mathrm{p}\mathrm{b}\overline{\mathrm{f}}$
0	$\overline{\mathrm{p}}\overline{\mathrm{b}}\overline{\mathrm{f}}$ $\overline{\mathrm{p}}\overline{\mathrm{b}}\mathrm{f}$ $\overline{\mathrm{p}}\mathrm{b}\mathrm{f}$

Figure 1: A ranked interpretation for $\mathcal{P} = \{\mathrm{b}, \mathrm{f}, \mathrm{p}\}$.

We can define a partial order $\preceq_{\mathcal{K}}$ on all ranked interpretations of a knowledge base $\mathcal{K}$ and find a unique minimal ranked interpretation $R_{\mathcal{K}}$ with respect to $\preceq_{\mathcal{K}}$. [10]. We say that $\mathcal{K}$ defeasibly entails $\alpha \mid\sim \beta$, denoted $\mathcal{K} \mid\approx \alpha \mid\sim \beta$, if $R_{\mathcal{K}} \Vdash \alpha \mid\sim \beta$. This form of defeasible entailment satisfies the KLM-properties [13].

2.4.2 Algorithmic Definition. The *Base Rank Algorithm* provides a means for checking defeasible entailment that is equivalent to the above semantic definition [9]. In this algorithm, we first construct a ranking $\mathcal{K}_R$ of all the statements in $\mathcal{K}$. This is done as follows:

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

We end up with a sequence such that $\Sigma_n \subset \Sigma_{n-1} \subset \ldots \subset \Sigma_0$ We then assign $\Sigma_{n-1} - \Sigma_n$ to the bottom rank, $\Sigma_{n-2} - \Sigma_{n-1}$ to second from the bottom, and so on. We end up with a ranking of $\mathcal{K}$ where the classical statements appear on the bottom; the 'infinite' level. The higher up in the ranking you go, the more 'general' the statements become.

Now, to check entailment, perform the following steps:

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

2.5 Relevant Closure

While rational closure is mostly intuitive and makes sense, it can seem unnecessary to throw away an entire level of statements when we have a conflict. While it is true that a statement within the level is likely causing the conflict, there are other statements in the level that have no effect on the conflict occurring. To solve this, the algorithm of relevant closure introduces the notion of *relevance*. Intuitively, we say that $\gamma_1 \mid\sim \gamma_2$ is *relevant* to a query $\alpha \mid\sim \beta$ if we can use $\gamma_1 \mid\sim \gamma_2$ to prove $\neg\alpha$ (a conflict).

So when we're checking if $\mathcal{K} \mid\approx \alpha \mid\sim \beta$, we define $R \subseteq D$ to be all defeasible statements relevant to the query $\alpha \mid\sim \beta$. Then, we simply perform the rational closure algorithm as before. However, when throwing away the highest level of the ranking, we only throw away those statements that appear in R.

While this method seems to make sense, Casini et al[4] showed that for description logics, relevant closure does not satisfy the KLM properties of *Or*, *CM* and *RM*. This means it is unlikely that the adapted version for DLV will satisfy those properties either.

2.6 Lexicographic Closure

Intuitively, lexicographic closure can be seen as a finer grained version of rational closure. We check defeasible entailment $\mathcal{K} \mid\approx \alpha \mid\sim \beta$ in a manner similar to that in the rational closure algorithm. However, if $\vec{\mathcal{K}} \models \neg\alpha$ then we do not throw away the whole top level of statements. Instead, we consider all possible combinations of statements from the top layer and throw away the smallest number of statements such that $\vec{\mathcal{K}} \not\models \neg\alpha$.

If such a combination exists, we return $\vec{\mathcal{K}} \models \alpha \rightarrow \beta$. Otherwise, we throw away the whole top level and repeat the process. Similarly to in the rational closure algorithm, if only the bottom level remains and $\vec{\mathcal{K}} \models \neg\alpha$, then we return 'true'. Unlike relevant closure, lexicographic closure for propositional logic has been shown to satisfy all of the KLM properties [12].

3 PROJECT DESCRIPTION

Datalog is a declarative logic programming language, whose language is made up of function-free Horn clauses $h \leftarrow l_1 \wedge \cdots \wedge l_m$, where h and all l_i are literals. DLV [14] is an extension of Datalog which allows disjunction ($\vee$) in the heads of rules in the form $a_1 \vee \cdots \vee a_n \leftarrow b_1 \wedge \ldots \wedge b_m$, where all a_i and b_i are literals. Datalog and DLV both use classical logical reasoning as their basic form of reasoning. Hence, they are currently not able to handle defeasible implications and exceptions.

The ability to reason with uncertainty and exceptions is a desired property of many systems and a major topic in Artificial Intelligence [5]. It allows for a more expressive, common-sense and human-like style of reasoning. This project aims to achieve this and extend the expressivity of DLV by incorporating KLM-style defeasible reasoning into classical DLV.

Defeasible entailment is not unique, with multiple definitions for computing entailment existing. So, the KLM [11] properties are used to specify how any correct form of defeasible entailment should behave for propositional logic. In order to extend KLM-style defeasible reasoning, the KLM properties first need to be extended for DLV to provide a way of evaluating the correctness of the different defeasible entailment algorithms for DLV.

The most basic form of defeasible entailment which satisfies these properties (for propositional logic), Rational Closure [13], should be extended to DLV to satisfy the modified KLM properties. However, Rational Closure is usually too weak from an inferential point of view [4]. To consider a more expressive form of defeasible entailment, Relevant Closure [4] should be extended. However, Relevant Closure does not satisfy all the KLM properties (for propositional logic), and hence, should not satisfy them for DLV either. Finally, Lexicographic Closure [12], a more expressive form of defeasible entailment which does satisfy all the KLM properties (for propositional logic) should be extended.

3.1 Why is it Important?

Datalog is a popular query language for deductive databases. DLV is considered to be a state-of-the-art implementation of disjunctive logic programming (DLP) [8]. In the context of deductive databases, DLP is recognized as a valuable tool for reasoning due to its ability to support natural modeling of incomplete knowledge [1]. As an extension of Datalog, DLV can solve all problems which are solved using Datalog, and additionally, some problems which cannot be solved using disjunction-free Datalog. The inclusion of disjunction can also allow for representing problems in a simpler and more natural fashion in DLV than could be done using Datalog [8]. However, there are many problems where classical reasoning is inappropriate and defeasible reasoning is required. Thus, the extension of DLV to allow for defeasible reasoning should further increase its applications by allowing for a more common-sense form of reasoning.

3.2 Possible Issues and Difficulties

The following have been identified as possible issues and difficulties for the project:

- There are no clear steps for extending the KLM properties for DLV.
- There are no guidelines for determining new algorithms for defeasible entailment in DLV and proving their correctness using KLM properties.
- It is difficult to estimate the time for finding the algorithms or proving their correctness.
- Additional properties, beyond the KLM properties, may be required to properly constrain an extension of defeasible reasoning in DLV. Hence, it will be difficult to fully assess the acceptability of the new algorithms for defeasible entailment in DLV.

4 PROBLEM STATEMENT

4.1 Aims

The primary aims of this project are to:

- Find and justify an extension of the KLM properties for DLV.
- Find satisfactory algorithms for rational closure, relevant closure and lexicographic closure in DLV.
- Prove the correctness and computational complexity of the rational closure, relevant closure and lexicographic closure algorithms for DLV.

Since it is difficult to estimate the time required to find these algorithms and prove their correctness, it is possible that these aims are met sooner than expected. If this is the case, additional aims for the project are to implement rational closure, relevant closure or lexicographic closure in the DLV system.

4.2 Research Questions

The problem to be solved is summarized in the following central research questions, which are separated into the questions to be answered by each group member.

4.2.1 *Matthew Morris.*

- Can an extension of the KLM properties for DLV be found?
- Can a provably correct algorithm be found for extending relevant closure to DLV?

4.2.2 *Tala Ross.*

- Can a provably correct algorithm be found for extending rational closure to DLV?
- Can a provably correct algorithm be found for extending lexicographic closure to DLV?

4.2.3 *Additional Research Questions.* Again, since it is possible that these research questions are answered sooner than expected, we pose the following additional research questions:

- How can the rational closure algorithm be implemented in the DLV system?
- How can the relevant closure algorithm be implemented in the DLV system?
- How can the lexicographic closure algorithm be implemented in the DLV system?

5 PROCEDURES AND METHODS

The problem is divided into sections for extending the KLM properties to DLV and extending the defeasible entailment algorithms to

DLV. In addition, sections for implementing these algorithms may be included.

5.1 Extending KLM Properties

In order to extend the KLM properties, we will first study how the KLM properties for propositional logic were extended for description logics. We will then consider how the differences between these logics and the logic of DLV impacts the definition of the properties for DLV.

5.1.1 Testing and Analysis. It must be ensured that the definition of the extension of KLM properties is reasonable. This will be achieved by having an expert in the field review the definitions.

5.1.2 What can this be used for? The extended KLM properties will be used in the proofs of correctness of the defeasible entailment algorithms for DLV. Additionally, they could be used as rules for deciding whether or not other methods of defeasible entailment for DLV are acceptable.

5.2 Extending Defeasible Entailment Algorithms

In order to extend the rational closure algorithm, we will first study the algorithm for rational closure in propositional logic and description logics. We will consider how the differences between these logics and the logic of DLV impacts the approach to the rational closure algorithm for DLV. To prove the correctness of the algorithm, we must show that the output of the algorithm satisfies all of our extended KLM properties for DLV.

We will approach extending the relevant closure and lexicographic closure algorithms in a similar manner, with additional considerations of how approaches for the extension for rational closure for DLV can be applied in these cases. We will again prove correctness of the algorithms by showing that the output of the algorithm satisfies all of our extended KLM properties, and then prove computational complexity results for all these algorithms.

5.2.1 Testing and Analysis. The correctness of the algorithms will be given in their proofs. These proofs cannot be tested and should instead be verified by an expert in the field. The algorithms will be compared and analyzed using the extended KLM properties as well as their computational complexities. Furthermore, if they were developed, the prototype implementations could be used for comparing and analyzing the different algorithms' real-world performances.

5.2.2 What can this be used for? The produced defeasible entailment algorithms could be implemented to allow for defeasible reasoning in the DLV system. These implementations could be used to reason about systems with uncertain and incomplete information, and hence, would have real-world applications.

5.3 Implementation

Provided the other aspects of the project have been completed, we will attempt bare-bones implementations of the various algorithms within DLV. The initial challenge will be working with the interior operations of DLV to try and add functionality for defeasible entailment checking. We will use the built-in DLV libraries to do classical entailment checks as part of the algorithms. Once we have implemented rational closure, relevant closure and lexicographic closure should be quick to implement and will only require some adaptation of the existing code.

5.3.1 Testing and Analysis. The correctness of these implementations will be shown by our previous work in proving the correctness of the algorithms. We will be able to verify that the systems behave as expected by testing them on several real-world examples that we will provide to them.

Furthermore, we will be able to test how long the algorithms take to run on reasonably sized data-sets. Since the implementation is not a core focus of the project, it is likely that they will be highly unoptimized and run relatively slowly.

Finally, we will be able to compare the differences in defeasible entailment between the three algorithms we are considering. While evaluating these differences is beyond the scope of our project, possible future work may involve comparing the three approaches within the context of DLV. This comparison could allow for the identification of a defeasible entailment definition that makes the most sense within DLV.

5.3.2 What can this be used for? Since our implementation will likely be unoptimized, it is unlikely to be useful within a practical environment. However, it could be useful as a prototype for DLV users. They would be able to see if extending the language with defeasible reasoning would allow them to solve problems they previously could not, or better code answers to solved problems.

6 ETHICAL, PROFESSIONAL AND LEGAL ISSUES

Due to the theoretical nature of the project, it poses no inherent ethical issues. Although, if the algorithms were to be implemented and tested in real world cases, possible ethical issues could arise during user testing if the test users did not give consent or were exploited. Furthermore, there are no foreseeable professional issues with the project. In fact, the project provides opportunity for further study in the area of defeasible reasoning for DLV and other description logics.

The current DLV system is open source with a custom license, making it free for academic and non-commercial educational use. If any of the algorithms are implemented for DLV, the extended version of the DLV system will remain open source under this license. Thus, any issues regarding intellectual property rights will be avoided. This would also allow for the use of the extended system in further research. There are no other foreseeable legal issues.

7 RELATED WORK

Kraus, Lehmann and Magidor (KLM) [11] introduced preferential reasoning, KLM-style defeasible implications and the KLM properties. Lehmann and Magidor [12] 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.

Lehmann [12] introduced lexicographic closure for propositional logic. This was shown to be another method for computing defeasible entailment which is strictly less conservative than rational closure while still satisfying all the KLM properties. Casini et al[6] presented a systematic approach for enriching propositional logic with a defeasible implication connective, and described algorithms for rational closure and lexicographic closure within this framework. Casini et al[7] presented an algorithm for lexicographic closure for description logics. Casini et al[4] introduced relevant closure for description logics; yet 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 ANTICIPATED OUTCOMES

8.1 Expected Impact

A successful project will impact the field of Knowledge Representation and Reasoning. An extension of existing knowledge will be provided by extending the KLM properties and KLM-style defeasible reasoning for DLV. Ontology engineering will be impacted by the provision of algorithms for defeasible reasoning which can be implemented in DLV. This will allow for modeling in ways that were previously not possible due to the limitations of classical reasoning.

A successful rational closure algorithm will provide an acceptable way to compute defeasible entailment which is of minimal computational complexity. A successful relevant closure algorithm will provide a more common-sense way to compute defeasible entailment, albeit one which does not satisfy the KLM properties. A successful lexicographic closure algorithm will provide a more common-sense but more complex way to compute defeasible entailment. Furthermore, this should satisfy the KLM properties.

8.2 Key Success Factors

The success of the project will be judged through two factors. The first will be whether the adapted algorithms satisfy the KLM properties for DLV. The second will be the computational complexity of the adapted algorithms. In addition, if implementations of the algorithms within DLV are provided, success will be judged by how optimized the implementations are.

8.3 System

As stated before, there is potential for software deliverables. The key features of the deliverables will be being able to compute defeasible entailment within DLV. Major design challenges for the software will include; initially discovering how to edit the DLV compiler, making the correct calls to the core DLV libraries, and providing relatively optimized implementations.

9 PROJECT PLAN

9.1 Risks

A list of risks for the project and their allocated identification numbers (or risk numbers) is given in Appendix A.1. The table also ranks the probability of each risk occurring and the impact that it would have on the project if it did happen to occur. Approaches for mitigating, monitoring, and managing each risk are given in Appendix A.2. Based on the analysis of possible risks in Appendix A, the project has relatively low risk and all foreseeable risks can be mitigated.

9.2 Timeline and Milestones

The "development" phase of the project will run for 7 weeks; from 8 July 2019 to 23 August 2019. We will present our final project and findings between 2 - 13 September 2019. The timeline, tasks and milestones for the project can be seen in Appendix B; in our Gantt chart (see Appendix B.1) and Tasks and Milestones table (see Appendix B.2).

9.3 Resources required

No special resources are required for the theoretical components of the project. However, if the implementation of an algorithm for the DLV system is attempted, then various tools and software will be required.

We will need access to the core code of DLV. It is written in C++, and hence, C++ should be used to implement the defeasible reasoning algorithms. The Eclipse IDE will be used since it is a familiar development environment for both group members and is widely considered a good IDE for C++ development.

9.4 Deliverables

We will present formalizations of the KLM properties, the rational closure algorithm, the relevant closure algorithm and the lexicographic closure algorithm (all for DLV). We will also provide formal proofs that the algorithms in question satisfy the adapted KLM properties. Finally, if the project timeline allows, we will deliver bare-bones implementations of the algorithms within DLV. Other deliverables are the literature review, project proposal, proposal presentation, feasibility demonstration, final report, final demonstration and project poster.

9.5 Work Allocation

The project work is divided into 8 logical sections as follows:

- The extension and justification of the KLM properties for DLV (1).
- The definition and proof of correctness of the algorithms for rational closure (2), relevant closure (3) and lexicographic closure (4) in DLV.

with the possibility of the additional sections:

- The implementation of the algorithms for rational closure (5), relevant closure (6) and lexicographic closure (7) in the DLV system.

Initially, section (1) will be completed by Matthew Morris and section (2) by Tala Ross. After these sections are completed, Matthew Morris will complete section (3), and possibly sections (5) and (6). Tala Ross will complete section (4), and possibly section (7).

REFERENCES

[1] Chitta Baral and Michael Gelfond. 1994. Logic programming and knowledge representation. *The Journal of Logic Programming* 19-20, SUPPL. 1 (may 1994), 73–148. https://linkinghub.elsevier.com/retrieve/pii/0743106694900256

[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] Giovanni Casini and Umberto Straccia. 2012. Lexicographic closure for defeasible description logics. *CEUR Workshop Proceedings* 969 (2012), 28–39.

[8] Thomas Eiter, Wolfgang Faber, Christoph Koch, Nicola Leone, and Gerald Pfeifer. 2000. *DLV - A System for Declarative Problem Solving.* Technical Report. Institut fur Informationssysteme. 6 pages. http://arxiv.org/abs/cs/0003036

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

[10] L. Giordano, V. Gliozzi, N. Olivetti, and G.L. Pozzato. 2015. Semantic characterization of rational closure: From propositional logic to description logics. *Artificial Intelligence* 226 (sep 2015), 1–33. https://doi.org/10.1016/J.ARTINT.2015.05.001

[11] 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

[12] 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

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

[14] Nicola Leone, Gerald Pfeifer, Wolfgang Faber, Thomas Eiter, Georg Gottlob, Simona Perri, and Francesco Scarcello. 2006. The DLV system for knowledge representation and reasoning. *ACM Transactions on Computational Logic* 7, 3 (jul 2006), 499–562. https://doi.org/10.1145/1149114.1149117

[15] 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 RISKS AND RISK MANAGEMENT

A.1 Risk Identification

ID	Risk	Probability	Impact
1	Supervisor is unavailable	Very unlikely	Moderate
2	Partner is unable to complete their part of the project	Possible	Moderate
3	Disagreement and conflict in the group	Very Unlikely	Significant
4	Poor communication in the group	Unlikely	Moderate
5	Poor time management	Possible	Severe
6	Adapted algorithm does not satisfy KLM properties.	Unlikely	Significant
7	KLM properties do not adapt well to DLV.	Very unlikely	Severe
8	Cannot suitably extend algorithms to DLV.	Unlikely	Severe
9	KLM properties are not sufficient to constrain DLV entailment.	Possible	Moderate

Figure 2: Ranking of risks according to probability and impact

A.2 Risk Management

ID	Mitigation	Monitoring	Management
1	Check in advance for supervisor availability	Confirm meetings through email and calender invites	Continue to work independently
2	Collaborate when partner gets stuck on something	Check in on partner's progress	Make sure the work is evenly split with a logical divide and guarantee that sections can be completed without the other
3	Agree on parameters for collaboration up front	Make sure both parties are always satisfied with the situation	Involve supervisor in the disagreements
4	Keep partner informed on progress with regular updates	Check in as to whether partner understands what work we are currently doing	Resolve miscommunications clearly through written text such as email
5	Create initial project timeline and stick to it	Check whether we are meeting the milestones we have set by their deadlines	Restructure project timeline with new estimates
6	Show algorithm to supervisor for approval before attempting to prove that the KLM properties hold	Initial proof sketches before fully diving into the proofs	Adapt algorithm so that it will satisfy the properties
7	Read similar work where the KLM properties were adapted and see how it was done	Check whether properties are making sense in the context of DLV, and not just syntactically	
8	Present adapted algorithm to supervisor for comments and editing	Check whether all needed library calls can be accessed within DLV	Adapt the algorithm so that it can be implemented within DLV
9	No mitigation possible, the KLM properties are widely agreed upon to be useful and DLV itself cannot be changed	Investigate the nature of DLV to identify other properties we might want it to have that the KLM properties do not constrain	Add additional properties to constrain defeasible entailment

Figure 3: Approaches for mitigation, monitoring and management of risks

B TIMELINE AND MILESTONES

B.1 Gantt Chart

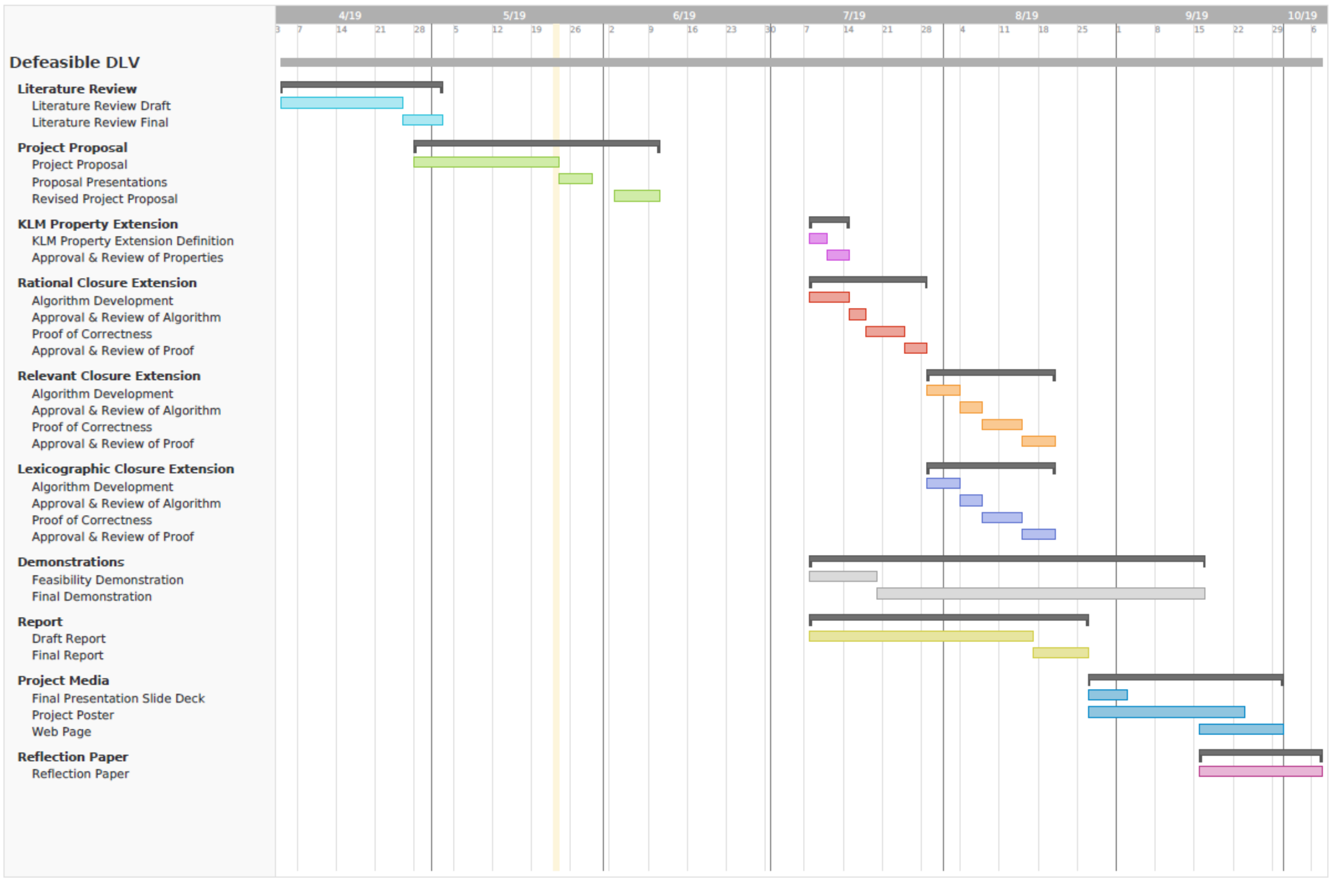

Figure 4: Gantt chart

B.2 Tasks and Milestones

Milestone/Task	Start Date	Finish Date
Literature Review	**04-04-2019**	**02-05-2019**
Literature Review Draft	04-04-2019	25-04-2019
Literature Review Final	26-04-2019	02-05-2019
Project Proposal	**28-04-2019**	**10-06-2019**
Project Proposal	28-04-2019	23-05-2019
Proposal Presentation	24-05-2019	29-05-2019
Revised Project Proposal	03-06-2019	10-06-2019
KLM Property Extension	**08-07-2019**	**14-07-2019**
KLM Property Extension Definition	08-07-2019	10-07-2019
Approval & Review of Properties	11-07-2019	14-07-2019
Rational Closure Algorithm Extension	**08-07-2019**	**28-07-2019**
Algorithm Development	08-07-2019	14-07-2019
Approval & Review of Algorithm	15-07-2019	17-07-2019
Proof of Correctness	18-07-2019	24-07-2019
Approval & Review of Proof	25-07-2019	28-07-2019
Relevant Closure Algorithm Extension	**29-07-2019**	**20-08-2019**
Algorithm Development	29-07-2019	03-08-2019
Approval & Review of Algorithm	04-08-2019	07-08-2019
Proof of Correctness	08-08-2019	14-08-2019
Approval & Review of Proof	15-07-2019	20-08-2019
Lexicographic Closure Algorithm Extension	**29-07-2019**	**20-08-2019**
Algorithm Development	29-07-2019	03-08-2019
Approval & Review of Algorithm	04-08-2019	07-08-2019
Proof of Correctness	08-08-2019	14-08-2019
Approval & Review of Proof	15-07-2019	20-08-2019
Demonstrations	**08-07-2019**	**16-09-2019**
Feasibility Demonstration	08-07-2019	19-07-2019
Final Demonstration	20-07-2019	16-09-2019
Report	**08-07-2019**	**02-09-2019**
Draft Report	08-07-2019	16-08-2019
Final Report	17-08-2019	26-08-2019
Project Media	**27-08-2019**	**30-09-2019**
Final Presentation Slide Deck	27-08-2019	02-09-2019
Project Poster	27-08-2019	23-09-2019
Web Page	16-09-2019	30-09-2019
Reflection Paper	**16-09-2019**	**07-10-2019**
Reflection Paper	16-09-2019	07-10-2019

Figure 5: Tasks and milestones