Scalable Defeasible Reasoning V2 with focus on Rational and Lexicographic Closure

Research Proposal

Dhiresh Thakor Vallabh
University of Cape Town
Cape Town, South Africa
THKDHI001@myuct.ac.za

Evashna Pillay
University of Cape Town
Cape Town, South Africa
PLLEVA005@myuct.ac.za

ABSTRACT

Knowledge representation and reasoning is a methodology in artificial intelligence (AI). This methodology involves storing some information about the world into a machine (a knowledge base) and it creates conclusions based on the knowledge given. Defeasible reasoning is a non-classical form of reasoning that expands upon the classical form by being able to reason about new and/or conflicting information. This is done by accepting information that is typically true. Entailment checking then identifies if the new information is deduced by the knowledge base. There is currently few practical implementations of defeasible entailment for propositional logic. This project will investigate the improvements that can be made to the scalability of defeasible entailment algorithms and implement these new algorithms in such a way that it works for a larger framework. We also aim to implement a system that generates knowledge bases for testing purposes and investigate how it affects the performance of defeasible entailment.

CCS CONCEPTS

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

KEYWORDS

artificial intelligence, knowledge representation and reasoning, defeasible reasoning, Boolean satisfiability solving

1 INTRODUCTION

Knowledge representation (KR) is a fundamental part of artificial intelligence. KR utilizes symbols to represent knowledge in which inferences are made and new elements of knowledge are formed about the world. To represent information, we will be differentiating between propositional logic and defeasible reasoning. Propositional logic is monotonic as previously drawn conclusions cannot be contradicted by new information [15]. To illustrate this logic, we will use the common example, "Birds can fly." We know birds have wings. Penguins are birds as they have wings, therefore penguins can fly. However, we also know that penguins cannot fly. It is evident that propositional logic does not have the ability to model human reasoning and limits any possibility of exceptions in the knowledge base. Thus, our research will focus on non-monotonic reasoning, that allows for a more common-sense" approach to reasoning in which elements of knowledge are typically true and not always definite [2]. defeasible reasoning addresses atypical reasoning scenarios in which humans fundamentally think differently, this may be due to their beliefs, context, or other circumstances. The reasoner can temporarily dismiss commonly held beliefs when presented with new contradictory information.

2 BACKGROUND

2.1 Propositional Logic

Propositional logic [15] is a framework that represents information about the world as logical statements known as *formulas*. Each formula is made up of *propositional atoms* which can be assigned *truth values* (true or false). Formulas can connect propositional atoms together using *connectives* ($\neg, \wedge, \vee, \rightarrow, \leftrightarrow$). With $\alpha, \beta \in \mathcal{L}$ where $\mathcal{L}$ is the set of all formulas we can create $\neg\alpha$, $\alpha \wedge \beta$, $\alpha \vee \beta$, $\alpha \rightarrow \beta$, $\alpha \leftrightarrow \beta$. An *interpretation* is a total function $I : P \mapsto \{T, F\}$ that takes a formula, $\alpha \in \mathcal{L}$, and assigns each propositional atom in α a truth value. The set of all of these interpretations is represented as $\mathcal{W}$. The truth value of an interpretation, created by α, is denoted as $I(\alpha)$. If $I(\alpha)$ is true for the formula α, then I satisfies α. This is denoted as $I \Vdash \alpha$. A *knowledge base* is a finite set of propositional formulas. A knowledge base, $\mathcal{K}$, is satisfied by an interpretation, I, if and only if for every $\alpha \in \mathcal{K}$, I satisfies α. If at least one interpretation satisfies $\mathcal{K}$, then that I is a *model* of $\mathcal{K}$.

2.2 Entailment

A knowledge base, $\mathcal{K}$, *entails* a formula, α, (written as $\mathcal{K} \models \alpha$) when every model of $\mathcal{K}$ is satisfied by α. Propositional logic, and therefore propositional entailment, is monotonic. This means that all logical statements and conclusions in the knowledge base are infallible. This means that any new information that creates a contradiction can never be made as an exception [18]. However, the knowledge base must still reconcile with this new information and can form incorrect conclusions. This makes knowledge base generation redundant which is why we require a nonmonotonic approach [11] [13]. The KLM style framework is our preferred choice.

When checking if $\mathcal{K} \models \alpha$ can be done by using a satisfiability (SAT) solver [18]. The SAT solver reduces entailment into satisfiability by checking if the models of $\mathcal{K}$ satisfy $\mathcal{K} \cup \neg\alpha$. This will be further discussed in the relevant works section.

2.3 KLM Approach to Defeasible Reasoning

The KLM Approach models statements written in the form of $\alpha \mid\sim \beta$, which is interpreted as "α typically implies β". This means that we are prepared to conclude β from α unless we receive contradictory information. There are multiple methods used to deduce information from a defeasible knowledge base, this notion is referred to as defeasible entailment. Defeasible entailment determines whether a defeasible implication is entailed by a knowledge base, the approach used must adhere to the rationality properties defined by Lehmann and Magnidor [10]. Two methods which fall within the KLM approach are rational closure and lexicographic closure, which our research will address.

2.4 Rational Closure (RC)

Rational closure is an LM-rational method of defeasible entailment proposed by Lehmann and Magidor [7]. Before providing an overview on rational closure, the following concepts will be defined; minimal ranked entailment, materialisation and Base Rank Algorithm.

2.4.1 Minimal Ranked Entailment. Ranked entailment is similar in principle to ranked interpretations, the lower the ranked model the more likely there exists a minimum ranked model in the ordering. This principle involves a partial order for all ranked models of knowledge base K and ranks the interpretations by how likely it is. The rational closure of a knowledge base K is based on determining the minimum ranked model [7], this must satisfy some defeasible implication for the knowledge base K to entail it [3].

2.4.2 Materialisation. The material counterpart of a defeasible implication $\alpha \mid\sim \beta$ is the propositional formula $\alpha \rightarrow \beta$ [4]. The material counterpart of a defeasible knowledge base $\mathcal{K}$ is denoted as $\overrightarrow{\mathcal{K}}$ and represents the set of material counterparts, $\alpha \rightarrow \beta$, for every defeasible implication $\alpha \mid\sim \beta \in \mathcal{K}$.

2.4.3 Base Rank Algorithm. The initial step performed when implementing the rational closure of a knowledge base $\mathcal{K}$ is the base rank algorithm [16]. This determines the minimum ranked model.

Pseudo-code of the algorithm is shown below:

(1) Separate the classical statements $\mathcal{K}_C$ from the defeasible statements $\mathcal{K}_\mathcal{D}$.

(2) Materialise $\mathcal{K}_\mathcal{D}$ into $\overrightarrow{\mathcal{K}}_\mathcal{D}$. $\overrightarrow{\mathcal{K}}_\mathcal{D}$ in $\mathcal{E}_0^K$ and $\mathcal{K}_C$ as is.

(3) For each $\alpha \rightarrow \beta \in \mathcal{E}_0^K$, determine if $\mathcal{E}_0^K \cup \mathcal{K}_C \models \neg\alpha$.

(4) If true, α is exceptional and all formulas within $\mathcal{E}_0^K$ with α are moved to $\mathcal{E}_1^K$.

(5) Rank $\mathcal{R}_0^K$ is assigned to the formulas $\mathcal{E}_0^K \backslash \mathcal{E}_1^K$.

(6) Repeat the process for the next subset, i.e., $\mathcal{E}_1^K$ until no more formulas need to be assigned.

(7) Add the final rank $\mathcal{R}_\infty^\mathcal{K}$ to store $\mathcal{K}_C$.

2.4.4 Rational Closure Algorithm. $\mathcal{R}^\mathcal{K}$ from the base rank algorithm will be used, given a knowledge base $\mathcal{K}$ entails some defeasible implication $\alpha \mid\sim \beta$.

Pseudo-code of the algorithm is shown below:

(1) Check if $\overrightarrow{\mathcal{K}}$ entails $\neg\alpha$.

(2) If this is not the case, α is compatible with $\mathcal{K}$. Then check if $\overrightarrow{\mathcal{K}}$ entails $\alpha \rightarrow \beta$.

(3) If this is the case, α is incompatible with $\mathcal{K}$. The most preferred rank is withdrawn from $\mathcal{R}^\mathcal{K}$. The resulting knowledge base will be denoted as $\mathcal{K}'$.

(4) If $\mathcal{R}^{\mathcal{K}'}$ is an empty set, then $\mathcal{K} \not\models \alpha \mid\sim \beta$, else if $\mathcal{R}^{\mathcal{K}'}$ contains at least one rank, we return to (1) with $\mathcal{K}'$.

2.5 Lexicographic Closure (LC)

Lexicographic closure [12] is different from rational closure in that lexicographic closure follows presumptive reading whilst rational closure follows prototypical reading. Presumptive reading means that atypical formulas are able to inherit the properties of more typical formulas. Lexicographic closure shows this change by changing how rankings are formed. Lexicographic closure still uses the Base Rank Algorithm however formulas within the ranks are also ranked where the more typical formulas are ranked higher. By doing this, we can check and remove one statement within a rank instead of the whole rank. If all of the statements are removed then the whole rank is removed.

This method will be shown by giving the classic penguin example of entailment that compare rational closure to lexicographic closure. Given the knowledge base:

$$\mathcal{K} = \{b \mid\sim f, p \rightarrow b, b \mid\sim w, p \mid\sim \neg f\}$$

We can use the Base Rank Algorithm to create the following ranks:

∞	$p \rightarrow b$
1	$p \rightarrow \neg f$
0	$b \rightarrow f, b \rightarrow w$

Table 1: Ranks of $\mathcal{K}$

We then wish to query if $p \mid\sim w$ is entailed by the knowledge base:

2.5.1 Rational Closure Method.

(1) Query if $\overrightarrow{\mathcal{K}} \models \neg p$. $\neg p$ is entailed so we remove rank 0, which gives:

∞	$p \rightarrow b$
1	$p \rightarrow \neg f$
0	~~$b \rightarrow f, b \rightarrow w$~~

Table 2: Ranks of $\mathcal{K}$ After Rank 0 Removal

(2) We then query if $\neg p$ is entailed by the remaining statements in the ranks. $\neg p$ is not entailed and so we check if $\overrightarrow{\mathcal{K}}$ entails the statement $p \rightarrow w$. In this case it does not, therefore $\mathcal{K} \not\models p \mid\sim w$.

2.5.2 Lexicographic Closure Method.

(1) Create ranks within the ranks that are based on all possible rankings. In this case only rank 0 needs to change.

∞	$p \rightarrow b$
1	$p \rightarrow \neg f$
0	$b \rightarrow f$
	$b \rightarrow w$

Table 3: Lexicographic Ranking 1

∞	$p \rightarrow b$
1	$p \rightarrow \neg f$
0	$b \rightarrow w$
	$b \rightarrow f$

Table 4: Lexicographic Ranking 2

(2) Query if $\overrightarrow{\mathcal{K}} \models \neg p$. $\neg p$ is entailed so we remove the top statement in each ranking.

∞	$p \rightarrow b$
1	$p \rightarrow \neg f$
0	~~$b \rightarrow f$~~
	$b \rightarrow w$

Table 5: Lexicographic Ranking 1 After Removal

∞	$p \rightarrow b$
1	$p \rightarrow \neg f$
0	~~$b \rightarrow w$~~
	$b \rightarrow f$

Table 6: Lexicographic Ranking 2 After Removal

(3) We then query if the remaining statements entail $\neg p$. In this case, ranking 1 does not and so we check if the statements entail $p \rightarrow w$. The statements do entail $p \rightarrow w$ and so we conclude that $\mathcal{K} \mid\approx p \mid\sim w$.

This has shown us that lexicographic closure uses presumptive reasoning and presumed that penguins do have wings. This is the opposite of rational closure which used prototypical reasoning to come to the opposite conclusion.

3 PROJECT DESCRIPTION

3.1 Overview of Problem

Currently, research for checking satisfiability within defeasible reasoning in the propositional context has mainly been theoretical. In terms of practical implementations, there is only the previous year's SCADR project [1] [8] [18]. This means that there is still a large gap in practical research especially when compared to reasoners for other types of logic such as classical reasoning and description logics.

While these other reasoners do exist, they are very limited in their own ways. Classical reasoners has monotonicity which fails under contradicting information and description logics [2] would be too impractical to translate into propositional logic.

This also means that more work can still be done to improve the scalability and efficiency on rational [13] and lexicographic [12] algorithms from last year's project. Another key issue is that we can hopefully build upon the knowledge base generation work from last year's work and continue to see how parameters affect the efficiency of the project.

3.2 The Importance of the Problem

With research primarily focused on the theoretical aspect of non-monotonic reasoning, there is little to no implementation of automated defeasible reasoning. This research project which builds on the previous year's SCADR project [9] will recognize aspects of the theoretical nature of work done on defeasible reasoning and analyse its feasibility when implementing the automated model. Furthermore, an implementation of automated defeasible reasoning would signify a large contribution to the logic-based AI community (i.e., robotics and medical decision-making systems etc).

4 PROBLEM STATEMENT

4.1 Research Questions [9]

4.1.1 Evashna Pillay.

(1) How does the execution time of the optimised rational closure algorithm compare to the previous year's SCADR project implementation?

(2) What is the relationship between the performance of the rational closure algorithm and fixed-sized knowledge bases with different parameters (i.e., number of ranks and varying rank distributions)?

(3) What is the experience of testing the rational closure algorithm for completeness with the knowledge base generator?

4.1.2 Dhiresh Thakor Vallabh.

(1) How does the execution time of the optimised lexicographic closure algorithm compare to the previous year's SCADR project implementation?

(2) What is the relationship between the performance of the lexicographic closure algorithm and fixed-sized knowledge bases with different parameters (i.e., number of ranks and varying rank distributions)?

(3) What is the experience of testing the lexicographic closure algorithm for completeness with the knowledge base generator?

4.2 Research Aims

(1) To develop an optimised reasoner console-application based on some defeasible knowledge base that implements the rational closure algorithm.

(2) To refine the optimisation approaches used to increase the scalability of the rational closure implementation compared to the previous year's SCADR project.

(3) To develop an optimised reasoner console-application based on some defeasible knowledge base that implements the lexicographic closure algorithm.

(4) To refine the optimisation approaches used to increase the scalability of the lexicographic closure implementation compared to the previous year's SCADR project.

(5) To develop an application for the generation of knowledge bases that will be utilized during the testing of the optimised reasoner applications for rational and lexicographic implementations.

5 PROCEDURES AND METHODS

The project work is divided into three parts. Namely, the defeasible reasoners that use rational closure and lexicographic closure and the knowledge base generator. The defeasible reasoners will be built in Java and the knowledge base generator will be built using Scala.

We will integrate the KLM style for propositional logic with defeasible reasoning to create our algorithms for both reasoners. The knowledge base generator will generate a knowledge base which will be used for testing purposes. This knowledge base and a defeasible logic statement will be given as input to either of the defeasible reasoners. The reasoners will then use their algorithms and a satisfiability (SAT) solver to check for entailment and return the output. We will then measure metrics that can be used to compare the performance of these tools.

5.1 Knowledge Base Generator

Before starting development on the knowledge base generator, we will develop an understanding if how classical knowledge bases are structured. We will then see how KLM properties and defeasible reasoning is integrated into the classical structure to create a defeasible knowledge base. We can then look into the previous year's knowledge base generator [1] [8] [18] to build our understanding of what worked and where improvements can be made.

The previous year's work can also be used to identify what parameters they have used and where we can add changes. This will require an understanding of how rational closure and lexicographic closure works. These parameters can affect two aspects of the knowledge base: the ranking of the defeasible statements in the knowledge base and the defeasible entailment checking. These two aspects will give an indication of the efficacy of different parameters. Once we use all of these parameters collectively, we will then use metrics to measure how successful these parameters are.

5.2 Rational and Lexicographic Reasoners

To create these reasoners, we will need to have an understanding of how each algorithm works and how it can be reduced to checking for satisfiability (for the SAT solver). We will also study the previous year's work [1] [8] [18] to see how they implemented these algorithms. We will ensure correctness in our algorithms by generating test knowledge bases, inputting defeasible statements and checking the output. Once the initial versions are developed, we can add the optimisation techniques from last year's project and add our own techniques to improve the performance.

Metric testing and comparisons will be done on the initial version, the previous year's version, and the newer versions. These metrics will measure metrics like computational resource utilisation and execution time. We will also do this testing using different knowledge bases that vary in structure. These structures will vary how many ranks are created and how statements are added to ranks. This testing will be automated by creating a test-bench.

These reasoners will be executable through the command line. If all aspects of the project are completed, we could potentially wrap these tools into an API to build a singular, large framework.

5.3 Possible Challenges

- Finding an intuitive SAT solver that is compatible with propositional logic formats.
- Identifying a more optimal, generic solution to knowledge base generation for different logics.
- Identifying useful optimisation techniques that also improve upon the previous year's work.
- Identifying knowledge base generation parameters that will create knowledge bases of varying structures (how ranking is performed).

6 ETHICAL, PROFESSIONAL, AND LEGAL ISSUES

The algorithms that we are using were from publicly released academic literature therefore we are not infringing on any intellectual property rights. We will only be required to appropriately acknowledge the creators of these algorithms. The Java and Scala languages, that we will be using, uses OpenJDK which has a General Public License with the Classpath exception. This allows us to use the OpenJDK facilities and executables to our own discretion. Therefore, there will be no legal issues in this regard. We also do not require any user testing and we do not require private data. This means that there will be no ethical issues or possible breaching of the POPI act. In conclusion, there are no obvious signs of future legal, professional, or ethical issues arising in our project.

7 RELATED WORK [9]

RC and LC reduce to a number of checks pertaining to classical entailment. Therefore, Boolean satisfiability is applicable to our research. Boolean satisfiability (SAT) determines whether there exists an assignment satisfying a given Boolean formula [[5], [19]]. In analyzing the SAT solvers for classical reasoning we can determine which SAT solvers are appropriate for our defeasible reasoning model.

7.1 Classical SAT Solvers

7.1.1 Semantic Tableaux. The tableau is constructed by decomposing a formula into sets of atomic literals, resulting in a tree-like tableau [15]. Each branch ends with a complementary pair of formulas, that is referred to as a closed branch, or contains a set of

non-contradictory literals, that is referred to as an open branch. Each open branch represents a model for the given formula. When no further decomposition can take place, the construction is complete. The initial formula is found to be unsatisfiable when there is a clash. A clash occurs when literals for an atom are found within the same subset, this results in a contradiction. To determine a formula's satisfiability, a completed tableau is required.

7.1.2 DPLL. DPLL is a backtracking algorithm [6]. The algorithm allows for the input of a propositional formula in CNF format, in which true is returned if the formula is satisfiable or false is returned if the formula is unsatisfiable. A branching procedure is executed where some atom in α is set to a random truth value. Branching continues until an assignment satisfies α and true is returned, or α is false and the algorithm backtracks. Backtracking involves retrieving the recent branching assignment and re-branching with a different assignment. If there are no new assignments to branch to, then backtrack. If there exists no new branches to be taken and we cannot backtrack, false is returned and α is unsatisfiable.

7.1.3 Conflict-Driven Clause Learning (CDCL). The creation of the CDCL [14] was motivated by the DPLL SAT solver. The key difference between the CDCL SAT solver and the DPLL SAT solver is that backtracking in the CDCL SAT solver is non-chronological [6]. Conflicts caused by variable assignments are cached. This leads to optimized efficiency and execution.

7.2 Defeasible SAT Solvers

SAT solvers only cover a limited scope of reasoning, specifically classical. There is no SAT solver which deduces if a defeasible statement is satisfiable or unsatisfiable in relation to the rational closure or lexicographic closure of a defeasible knowledge base [12].

8 ANTICIPATED OUTCOMES

8.1 Software Developed [9]

- A console-application which improves the execution time of the rational closure implementation.
- An console-application which improves the execution time of the lexicographic closure implementation.
- An application to generate defeasible knowledge bases taking in to account several parameters (i.e., size, number of ranks and varying rank distribution etc).

8.2 Expected Impact

Argumentation theory (AT) is a branch of logic-based artificial intelligence that serves as the foundation for computational models of defeasible reasoning. AT has emerged as a promising research area for adopting defeasible reasoning, based on promising results [17]. However, applications are typically ad-hoc frameworks that do not include all of the layers and procedures necessary in an argumentation process, constraining their usability across domains. This research project will provide a complete defeasible-based framework, from assignment formation through conflict resolution to the final conclusion (i.e., true or false) [9].

8.3 Evaluation of the System

To determine whether our reasoners are successful we will create specific test cases, query sets and include system time calls to measure the execution time of each reasoner.

8.3.1 Test Cases. The knowledge base generator will only include defeasible statements in the form of $\alpha \mid\sim \beta$. The knowledge bases used will differ in number of ranks (i.e., 10, 50 and 100) and varying statement distribution (i.e., 50, 100, 200) [8].

8.3.2 Query Sets [1]. We will create three query sets:

- A set of queries in which the antecedents are equally unique and equally repeated.
- A set of queries which result in all the ranks except the final rank being retracted and the antecedents becomes consistent with the knowledge base.
- A set of queries whose antecedents all become consistent with the knowledge base after the first rank has been retracted.

8.4 Key Success Factors

- The defeasible reasoner developed outputs the correct result which can be validated using the knowledge base generator and test cases.
- The approaches used to improve the performance and scalability of the rational closure algorithm is faster in execution time/performance compared to the previous year's SCADR project [8].
- The approaches used to improve the performance and scalability of the lexicographic closure algorithm is faster in execution time/performance compared to the previous year's SCADR project [18].
- The application for the generation of knowledge bases is valid and useful for testing the optimised reasoners for rational and lexicographic implementations [1].

9 PROJECT PLAN

9.1 Risks

See subsection A.1 of the Appendix.

9.2 Timeline

See subsection B.2 of the Appendix.

9.3 Resources Required

We require access to the previous year's SCADR project to establish the scope of their research [9]. We will use their results for efficiency and performance of the rational and lexicographic closure as a benchmark to improve on. We will be using two separate laptops to write up our research, as well as develop the defeasible reasoners and the knowledge base generator. We will utilize publicly available SAT solvers and lastly, the IDE used will support both Java and Scala (OpenJDK) development.

9.4 Deliverables

The main project deliverables include the literature review, project proposal presentation, project proposal, software feasibility demonstration, final project paper, final project demonstration, project poster, and project webpage. The project itself will produce multiple versions of the rational closure and lexicographic closure reasoners. The initial version will be an amalgamation of the previous year's SCADR project [1] [8] [18] and our own naive approach. Newer versions will integrate optimisation techniques that improve scalability. The project will also produce a knowledge base generator which will be used for testing. We will also produce metrics that relate to every version of our reasoners and compare them to each other and the previous year's work. This is to compare the efficiency and scalability of the different versions and show the degree of success of our project compared to the previous year.

9.5 Milestones

See B.1 of the Appendix.

9.6 Work Allocation

Student	Work Allocation
Evashna Pillay	Development of the defeasible reasoner that uses the rational closure algorithm. Extensive research will be done for optimisation techniques to improve upon the previous SCADR project.
Dhiresh Thakor Vallabh	Development of the defeasible reasoner that uses the lexicographic closure algorithm. Extensive research will be done for optimisation techniques to improve upon the previous SCADR project.

Both members will develop the knowledge base generator which will provide test cases to test the scalability and efficiency of both algorithms. Potential work can be done to wrap the tools into an API which will be done by both members.

REFERENCES

[1] Aiden Bailey. 2021. Scalable Defeasible Reasoning. (2021), 1–16. https://projects.cs.uct.ac.za/honsproj/cgi-bin/view/2021/bailey_hamilton_park.zip/files/BLYAID001_SCADR.pdf

[2] Giovanni Casini, Thomas Meyer, Kodylan Moodley, and Ivan Varzinczak. 2013. Towards practical defeasible reasoning for description logics. (2013), 1–13. http://ceur-ws.org/Vol-1014/paper_17.pdf

[3] Giovanni Casini, Thomas Meyer, and Ivan Varzinczak. 2018. Defeasible Entailment: from Rational Closure to Lexicographic Closure and Beyond. *In 17th International Workshop on Non-Monotonic Reasoning (NMR)* (2018), 109–118. http://pubs.cs.uct.ac.za/id/eprint/1304

[4] Giovanni Casini, Thomas Meyer, and Ivan Varzinczak. 2019. Taking defeasible entailment beyond rational closure. *European Conference on Logics in Artificial Intelligence*, 182–197. https://doi.org/10.1007/978-3-030-19570-0

[5] Stephen A. Cook. 1971. The complexity of theorem-proving procedures. *In Proceedings of the third annual ACM symposium on Theory of computing* (1971), 151–158. https://doi.org/10.1145/800157.805047

[6] Martin Davis, George Logemann, and Donald Loveland. 1962. A Machine Program for Theorem-Proving. *Commun. ACM* 5 (1962), 394–397. Issue 7. https://doi.org/10.1145/368273.368557

[7] 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 (2015), 1–33. https://doi.org/10.1016/j.artint.2015.05.001

[8] Joel Hamilton. 2021. An Investigation into the Scalability of Rational Closure. (2021), 6–7. https://projects.cs.uct.ac.za/honsproj/cgi-bin/view/2021/bailey_hamilton_park.zip/files/SCADR_HMLJOE001.pdf

[9] Joel Hamilton, Daniel Park, and Aiden Bailey. 2021. Scalable Defeasible Reasoning. (2021), 1–10. https://projects.cs.uct.ac.za/honsproj/cgi-bin/view/2021/bailey_hamilton_park.zip/files/SCADR_Project_Proposal.pdf

[10] Adam Kaliski. 2020. An Overview of KLM-Style Defeasible Entailment. (2020), 52–83. http://hdl.handle.net/11427/32743

[11] Sarit Kraus, Daniel Lehmann, and Menachem Magidor. 1990. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial intelligence* 44 (1990), 167–207. https://doi.org/10.1016/0004-3702(90)90101-5

[12] Daniel Lehmann. 1995. Another perspective on default reasoning. *Annals of Mathematics and Artificial Intelligence* 15, 1 (1995), 61–82. https://doi.org/10.48550/arXiv.cs/0203002

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

[14] Joaão P. Marques-silva and Karem A. Sakallah. 1999. Grasp: A search algorithm for propositional satisfiability. *IEEE Trans. Comput.* 48 (1999), 506–521. https://doi.org/10.1109/12.769433

[15] Ben-Ari Mordechai. 2012. *Propositional Logic: Formulas, Models, Tableaux* (3 ed.). Springer London, 1, 7–47. https://doi.org/10.1007/978-1-4471-4129-7

[16] Matthew Morris, Tala Ross, and Thomas Meyer. 2020. Algorithmic definitions for KLM-style defeasible disjunctive Datalog. *South African Computer Journal* 32, 2 (2020), 141–160. https://doi.org/10.18489/sacj.v32i2.846

[17] Lucas Rizzo. 2017. On Demonstrating the Impact of Defeasible Reasoning in Practice via a Multi-layer Argument-based Framework. *AAMAS '17: Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems* (2017), 1857–1858. https://dl.acm.org/doi/10.5555/3091125.3091471

[18] Joon Soo Park. 2021. Scalable Defeasible Reasoning. (2021), 1–12. https://projects.cs.uct.ac.za/honsproj/cgi-bin/view/2021/bailey_hamilton_park.zip/files/PRKJOO001_SCADR_Final_Paper.pdf

[19] Yakir Vizel, Georg Weissenbacher, and Sharad Malik. 2015. Boolean Satisfiability Solvers and Their Applications in Model Checking. *Proc. IEEE* 103 (2015), 2021–2035. https://doi.org/10.1109/JPROC.2015.2455034d

Appendix

A RISKS

A.1 Risk Identification, Mitigation, Monitoring, and Management

ID	Risk	$\mathcal{P}$ (1-5)	$\mathcal{I}$ (1-5)	Mitigation	Monitoring	Management
1	Supervisor unavailable as well as team members unable to contact or meet supervisor in a timely manner.	3	4	Determine when it is suitable to meet with the supervisor every two weeks, this will ensure an agreed upon scheduled interaction.	Ask about the supervisor's schedule before ending meetings to know when they are available next.	Correspond regularly with supervisor to determine in advance when they will be available or not.
2	Quality of work impacted by poor time management due to differing schedules of team members.	2	4	Create mini deadlines for deliverables before the hard deadline, this will ensure work is being done regularly.	Check upcoming deadlines regularly to remember deadline dates.	Team members should meet and provide the work they have completed, this way every member is held accountable.
3	Inadequate understanding of the research topic as a result of its complex theoretical nature.	2	5	Gain a thorough understanding of the content early, and identify and address areas of uncertainty.	Members question each other on their knowledge of the topic before moving on to a new topic.	Meet with team to discuss confusion and if necessary contact supervisor for further guidance.
4	Experiencing conflict within the team.	1	3	Maintain openness and regular communication to address concerns as they occur.	Members ask each other how they feel about certain topics and/or issues.	Team must convene to voice concerns and provide solutions, if necessary seek advice from supervisor.
5	Viewing of past research and deliverables made difficult due to load-shedding (i.e., unreliable internet connection) or access restrictions.	4	3	Save files to local drive, this offline copy can be accessed at any time given one's laptop is charged.	Keep track of the load shedding schedule as much as possible and check if papers require access.	Ensure supervisor has granted access to past papers (and use UCT credentials).
6	Team member falling ill and unable to complete deliverables.	2	4	Ensure team members follow the mini deadline schedule, so that when a team member falls ill, work already completed can be turned over to another team member to complete in full.	Be wary of our own well-being and communicate any health concerns.	Team members must have a legitimate ailment otherwise members will reach out to the supervisor for further action.
7	Team member decides they no longer want to participate in the research project.	1	5	This risk however likely or not cannot be mitigated.	Members communicate their opinions and feelings towards the project.	Team members are responsible for ensuring the work they take on can standalone as a separate project from the group.
8	Unable to improve on the optimization of algorithms from the previous year.	3	3	Research and assess/implement the algorithms used in previous years as well as broadly in this area of study.	Use test metrics to compare the execution times and computational resource utilisation of the project with the previous year's project.	Compare the performance, speed and usefulness of the algorithms implemented in this project with the previous year and optimize accordingly.

$\mathcal{P}$ stands for Probability. $\mathcal{I}$ stands for Impact.

B TIMELINE

B.1 Tasks and Milestones

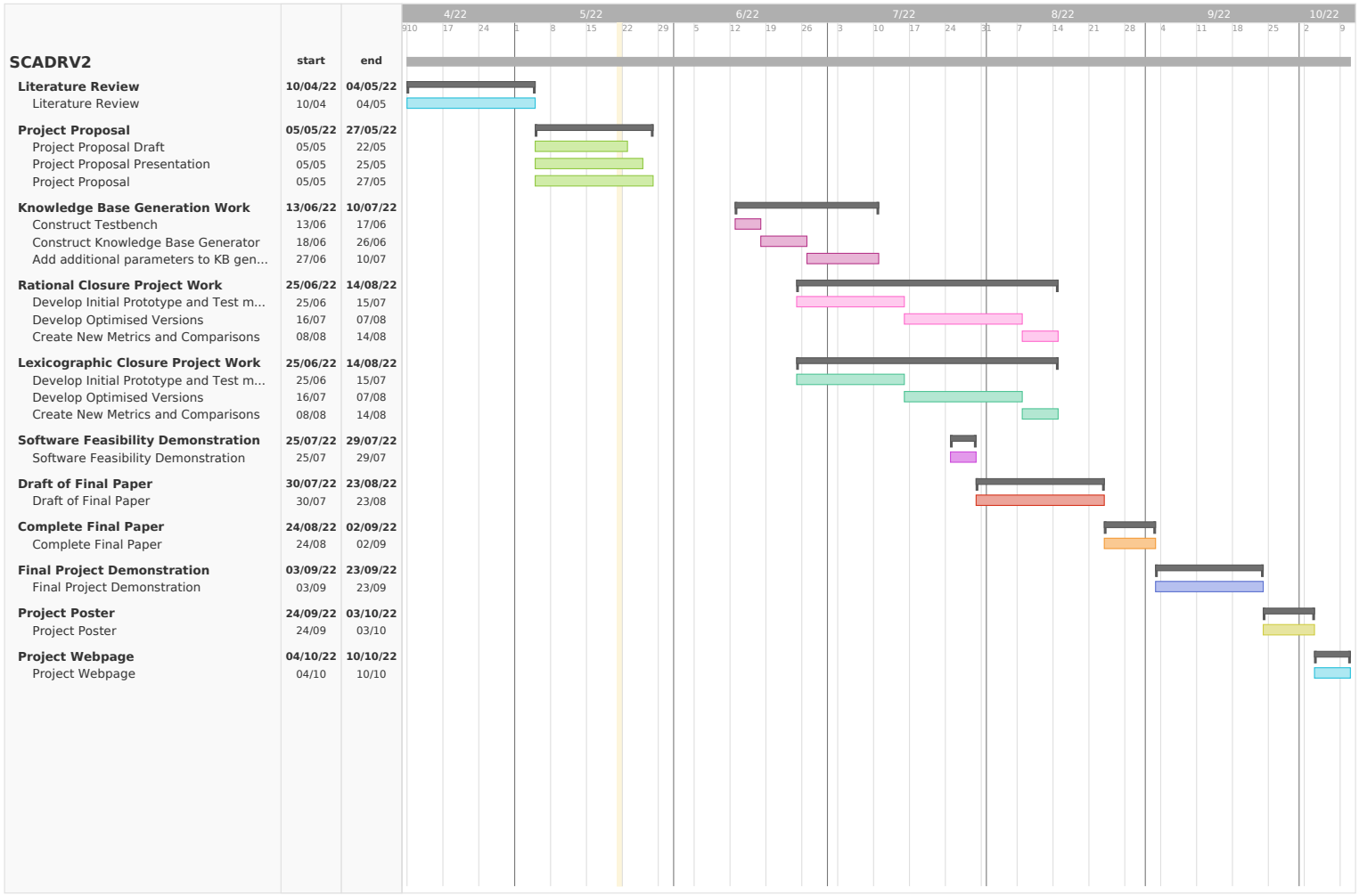
Milestones	Start Date	End Date
PHASE 1: Project Planning	10/04/2022	27/05/2022
Literature Review	10/04/2022	04/04/2022
Project Proposal Draft	05/05/2022	22/05/2022
Project Proposal Presentation	05/05/2022	25/05/2022
Project Proposal	05/05/2022	27/05/2022
PHASE 2: Project Work	13/06/2022	14/08/2022
Knowledge Base Generator	13/06/2022	10/07/2022
Construct Test-bench	13/06/2022	17/06/2022
Construct Knowledge Base Generator	18/06/2022	26/06/2022
Add Parameters to KB Generation	27/06/2022	10/07/2022
Rational Closure Project Work	25/06/2022	14/08/2022
Initial Prototype and Test Metrics	25/06/2022	15/07/2022
Develop Optimised Versions	16/07/2022	07/08/2022
Create New Metrics and Comparisons	08/08/2022	14/08/2022
Lexicographic Closure Project Work	25/06/2022	14/08/2022
Initial Prototype and Test Metrics	25/06/2022	15/07/2022
Develop Optimised Versions	16/07/2022	07/08/2022
Create New Metrics and Comparisons	08/08/2022	14/08/2022
PHASE 3: Project Completion	25/07/2022	10/10/2022
Software Feasibility Demonstration	25/07/2022	29/07/2022
Draft of Final Paper	30/07/2022	23/08/2022
Complete Final Paper	24/08/2022	02/09/2022
Final Project Demonstration	03/09/2022	23/09/2022
Project Poster	24/09/2022	03/10/2022
Project Webpage	04/10/2022	10/10/2022

B.2 GANTT Chart

See last page of Appendix for GANTT Chart.

teamgantt
Created with Free Edition

Printed: 21/05/2022



SCADRV2	start	end
Literature Review	**10/04/22**	**04/05/22**
Literature Review	10/04	04/05
Project Proposal	**05/05/22**	**27/05/22**
Project Proposal Draft	05/05	22/05
Project Proposal Presentation	05/05	25/05
Project Proposal	05/05	27/05
Knowledge Base Generation Work	**13/06/22**	**10/07/22**
Construct Testbench	13/06	17/06
Construct Knowledge Base Generator	18/06	26/06
Add additional parameters to KB gen...	27/06	10/07
Rational Closure Project Work	**25/06/22**	**14/08/22**
Develop Initial Prototype and Test m...	25/06	15/07
Develop Optimised Versions	16/07	07/08
Create New Metrics and Comparisons	08/08	14/08
Lexicographic Closure Project Work	**25/06/22**	**14/08/22**
Develop Initial Prototype and Test m...	25/06	15/07
Develop Optimised Versions	16/07	07/08
Create New Metrics and Comparisons	08/08	14/08
Software Feasibility Demonstration	**25/07/22**	**29/07/22**
Software Feasibility Demonstration	25/07	29/07
Draft of Final Paper	**30/07/22**	**23/08/22**
Draft of Final Paper	30/07	23/08
Complete Final Paper	**24/08/22**	**02/09/22**
Complete Final Paper	24/08	02/09
Final Project Demonstration	**03/09/22**	**23/09/22**
Final Project Demonstration	03/09	23/09
Project Poster	**24/09/22**	**03/10/22**
Project Poster	24/09	03/10
Project Webpage	**04/10/22**	**10/10/22**
Project Webpage	04/10	10/10