Adding Defeasibile Reasoning to Datalog and RDFox

Project Proposal

Joshua J. Abraham*
University of Cape Town
Department of Computer Science
joshuasl8a@gmail.com

Thomas Pownall†
University of Cape Town
Department of Computer Science
pwntho001@myuct.ac.za

Supervisor: Thomas Meyer‡
University of Cape Town
Department of Computer Science
tmeyer@cs.uct.ac.za

2nd Reader: Hussein Suleman§
University of Cape Town
Department of Computer Science
hussein@cs.uct.ac.za

ABSTRACT

The ability for logical reasoners to handle exceptions is a well-known problem within the domain of Artificial Intelligence. One solution to this problem is to grant reasoners the capability for computing defeasible reasoning, which is a form of non-monotonic reasoning that is used to deal with exceptions. This paper proposes a project that will see the integration of defeasible reasoning into classical datalog - a declarative logic programming language, used for database querying and logical reasoning - as well as RDFox - a Resource Description Framework (RDF) triple store, developed at the University of Oxford [1], which supports datalog reasoning. In this paper, we give a brief introduction to the main topics that constitute the basis of this project, as well as a background on relevant theory and tools that will need to be understood in order to follow the discussion of this project. We also provide an exact project description (which also contains the motivation for this project), followed by a concise problem statement, to sum up this project. We then discuss the procedures and methods used in, the legal and ethical issues posed by, the work related to, the outcomes expected from and the procedural plan of this project. Finally, we conclude with a few final words to sum up the ideas posed in this project proposal.

CCS CONCEPTS

• **Artificial Intelligence** → *Knowledge representation*; *Reasoning*;

KEYWORDS

Defeasible Reasoning, Datalog, RDFox

*Joshua is the student who will be heading up the implementational side of this project.
†Thomas P. is the student who will be heading up the theoretical side of this project.
‡Thomas M. will play the role of research supervisor and student mentor to the students involved in this research.
§Hussein Suleman will play the role of second reader to ensure quality of research.

1 INTRODUCTION

Logic and reasoning have been conceptualised since before the days of Plato, through to Aristotle, medieval thinkers and is still a topic of discussion among present-day philosophers, mathematicians, scientists, etc. There are many varying forms of logic and reasoning. For human beings, it is relatively common and quite natural to adjust our views and beliefs based on new information. Even when the new information we face is seemingly contradictory to what we already know, the logical mind has a natural ability and tendency to rationalise this information and either add it to our base of knowledge or reject it entirely. However, when trying to simulate the efficiency and effectiveness of this same reasoning process into a program or machine, this task becomes considerably more challenging. Imagine, if you will, being told that all birds fly and that some animal, called Tweety, is a bird. From this knowledge base, it is reasonable to deduce that Tweety can fly. However, if it is then learned that Tweety is, in fact, a penguin or even an ostrich, this would raise clear contradictions (as penguins and ostriches are both flightless birds). Similar, seemingly contradictory, occurrences can be found in many places, for e.g. Chisholm's paradox of contrary-to-duty imperatives [2][3], in many instances within the domains of law [4] and various other scenarios. The type of reasoning called upon to handle situations of this ilk is known as defeasible, or non-monotonic, reasoning - this is the type of reasoning which we wish to explore in this project.

Defeasibile reasoning is a form of default reasoning employed when faced with seemingly contradictory information, opposed to what is already known or has already been reasonably deduced. This type of reasoning is often used in law, where contracts can be annulled in light of new evidence; in medicine, where medical diagnoses can be reevaluated due to the revelation of new symptoms; in science, where scientific theories can be falsified by ascertaining new experimental results; etc. Thus, it is clear to see the implicational benefits that would emerge from granting programs and machines the power to employ this type of reasoning - this constitutes one of the motivations behind this paper's work.

2 BACKGROUND

In this section, we explain all of the technical terms and tools that need to be understood before discussing this project and its problem statements. Each of the two elements of the project have their own necessary information. For the theory element, the reader needs to understand the KLM properties, as these are vital for this element. We want to ensure, once propositional logic has been extended for Defeasible Reasoning, that the KLM properties are satisfied. For the implementation element of the project, the reader needs to understand both the datalog declarative programming language and RDFox, as they are both vital in constructing the defeasible datalog platform. We discuss these in the next 3 subsections.

2.1 KLM Properties

In terms of notation, we use common logical symbols. Additionally, to represent a defeasible statement, we make use of the $\rightsquigarrow$ symbol - where A $\rightsquigarrow$ B implies that if A then *typically* B. Below we list the KLM properties, which will be used throughout our project:

- *Reflexivity* - $C \rightsquigarrow C$
- *Cumulative Transitivity* - $C \rightsquigarrow D$ and $C \wedge D \rightsquigarrow F \implies C \rightsquigarrow F$
- *Cautious Monotony* - $C \rightsquigarrow D$ and $C \rightsquigarrow F \implies C \wedge D \rightsquigarrow F$
- *Left Logical Equivalence* - $C \rightsquigarrow F \rightsquigarrow C \iff D \implies D \rightsquigarrow F$
- *Right weakening* - $C \rightsquigarrow D$ and $D \rightsquigarrow F \implies C \rightsquigarrow F$
- *Left Disjunction* - $C \rightsquigarrow F$ and $D \rightsquigarrow F \implies C \vee D \rightsquigarrow F$
- *Rational Monotony* - $C \rightsquigarrow F$ and $C \not\rightsquigarrow \neg D \implies C \wedge D \rightsquigarrow F$

2.2 Datalog

The declarative logic programming language, datalog, is most often used as a query language for deductive databases. It is based upon predicate logic - a proven formalisation of reasoning, which is a level above propositional logic, yet below defeasible logic. Being based on predicate logic reasoning makes classical datalog sound, complete and, more importantly for the purposes of this paper, a well-established model for propositional logic reasoning. Datalog, however, is not Turing complete and is thus mainly used in the areas of logic programming; knowledge representation and database querying, although it has recently been found useful in a number of other domains such as data integration and data extraction. Furthermore, datalog is a syntactic subset of its precursor, prolog.

2.3 RDFox

Since there is an unthinkably large amount of data flowing through the World Wide Web at any given time, it is a proportionally large benefit to have a structured form of data describing this data (metadata) to improve the discovery of and access to the data it's describing. However, this metadata requires a defined syntax and structure, in order to be machine-readable. The Resource Description Framework (RDF) was developed in collaboration by members of the World Wide Web Consortium [5], as a solution to this issue. In his article, Miller [6] gives a thorough description on what exactly RDF is and how it works. For the purposes of this paper all that needs to be known, concerning RDF, is that a RDF store (triple store) is simply a database built with the intent of storing and retrieving triples through semantic queries.

RDFox is an RDF store developed at the University of Oxford [1]. RDFox is centralised (i.e. RDF data is stored on main memory) and allows for large growth within the RDF knowledge representation store. Triples can be added to a RDFox database by means of importing them or scheduling them for incremental addition or deletion. Datalog rules are added in the same manner. This method of adding triples and rules to a database allows RDFox to compute parallel datalog reasoning, by means of incremental materialisation. This simply means that when the triple, <Tweety - is - bird>, is explicitly stored in the database and the rule, bird $\implies$ flight is stored in the same database, then the triple, <Tweety - can - fly>, is implicitly generated (materialised) as well. This check for inference is done incrementally, for each new triple and/or rule that enters the database. RDFox also supports SPARQL query answering [7].

3 PROJECT DESCRIPTION AND MOTIVATION

Even though there has been a relatively exceeding amount of work done in the domain of non-monotonic declarative logical programming, we have found there to still exist some gaps in the literature. Among these gaps is the lack of a formal integration of defeasibility into the widely used declarative logic programming language, datalog. We have found that datalog has become quite popular and, even though it isn't a multi-purpose programming language, it finds uses in a wide variety of fields. These include, but are not limited to, data integration; information extraction; networking; program analysis; security; cloud computing; etc [8]. Furthermore, due to its functionally specific nature, it is easier to make full use of efficient algorithms that are developed for query resolution, when extending its computing capabilities - which is one of the motivations behind choosing datalog for the purposes of this paper.

Regarding RDFox, there exists the similar issue of no formal capabilities for handling instances which require defeasible reasoning. We believe that, due to its extensive features, RDFox is an immensely valuable tool when working with data storage and database querying and is an essential part of many works. This is especially evident when noting that RDFox is currently being used by multinational oil and gas companies, electric utility companies, health care consortiums, etc [1]. In addition to this, it is our belief that RDFox would be greater purposed if it had the capability for defeasible reasoning. With these points in mind, we now describe, with exactness, our project, as well as the purpose of it.

3.1 Description of the Project

This project involves both a theoretical component, as well as an implementational one. From the theoretical side, we focus on extending datalog with the defeasible reasoning that will allow for the application of defeasible reasoners in real world models, as opposed to purely theoretical formalisations. Furthermore, we ensure that

the properties of propositional logic and defeasible reasoning still hold once the *defeasible datalog* has been developed. From the implementational side, we will endeavour to extend the expressivity of RDFox by means of creating a wrapper which, when used, will afford RDFox the ability to correctly reason every and all defeasible instance that it encounters.

Let it be noted that, if one truly desires, one could view the implementational component of this project as a proof of concept for the theoretical component of this project - a secondary verification, as it were. However, for our purposes, we view each component of this project as a separate entity, entirely disjoined from each other, so as to allow for the individual assessment of the two researchers involved in said project. Furthermore, this satisfies, not the case of one component verifying another, but that of both components simultaneously verifying each other - that is, if they both work out as they are expected to.

3.2 Motivation for the Project

We believe that our project will have an impact in the field of Artificial Intelligence, regarding knowledge representation and reasoning, by further investigating and allowing large data stores to reason with exceptions and make logical decisions. We believe that, in so doing, not only will we help improve the understanding of defeasible descriptive logics and datalog, but that our work will have many other implementational benefits as well. Further motivation is provided by the gap that we have found in the literature and the fact that the documentation of such a project has not yet been completed.

Furthermore, although efforts to implement defeasibility in description logics have already been made (this is discussed in this paper, in more detail, in the section regarding its related works), this has never been done for the entirety of datalog, nor has it been done for the datalog reasoner, RDFox. This will be beneficial, since incorporating defeasibility with the high scalability and fast retrieval rates of RDFox is expected to significantly assist the research in exception handling within the domain and related domains of Artificial Intelligence. We also believe that the improved functionality of RDFox will benefit those already making use of the RDF triple store.

4 PROBLEM STATEMENT

In this section we explicitly address what the problem statement of this project is. We also provide include the two main research questions posed by this paper, which sum up the work to be done in this project.

4.1 Statement and Subsequent Questions Being Asked

There are two problems that this project elects to face. The first is the fact that there currently exists no means of concrete proof that the classical datalog declarative logic programming language can be fully extended to allow for correct defeasible reasoning by means of computation. The second is that the current working version (as of the creation of this proposal) of the RDF store and datalog reasoner, RDFox, has no capabilities to deal with defeasible circumstances.

Given this, the research questions posed in this project can be stated as follows:

(1) Can the entirety of the Datalog declarative programming language be extended to allow for correct reasoning with defeasibile instances and can it be shown that the means of this extension is both sound and complete?

(2) Can a defeasibile Datalog reasoner be created by modifying the current working version of RDFox and can this reasoner be shown to be capable of reasoning with defeasibile instances.

5 PROCEDURES AND METHODS: EXPERIMENTS, DESIGN, ETC

The project has two distinct elements, that is the theory and the implementation. There is little dependency between these two elements and this reflects in the procedures and methods. Each subsection here within will further be divided into these two elements and discussed accordingly.

5.1 Approach

Theory:
We approach the theoretical element of this project by investigating the algorithm for extending propositional logic with defeasible reasoning. This includes clearly defining the algorithm for our uses and then proving it is correct for the KLM properties.

Implementation:
We approach the implementation element of this project by developing a platform for configuring propositional logic knowledge bases with defeasible statements. The platform will then apply the algorithm to incorporate the defeasible statements into it. We can then use the platform to test the knowledge base to find entailments from the information.

5.2 Evaluation procedure

Theory:
The theoretical element requires no testing as its correctness follows from mathematical and logical reasoning. To ensure the correctness of the mathematical and logic reasoning requires contribution from experts in the field.

Implementation:
The implementation element requires testing in two aspects. Firstly, unit tests must be created and run to ensure the platform works as expected at every stage of the project. This follows similarly from agile methodologies and it ensures the platform is in working order. Secondly, the platform's user interface must be tested to ensure it is user-friendly and to ensure the front-end development works as expected.

5.3 Evaluation criteria

We determine that the project's success can be evaluated by the following criteria. Due to the nature of the project and its two elements, the criteria is similarly divided.

Theory:
The algorithm to be used is correct in terms of complying with the KLM properties.

Implementation:
The unit tests for the extension of propositional logic with defeasible reasoning all pass showing the expected output.
The platform's user interface is working and further possess criteria as follows,

- Ability to create new knowledge bases
- Ability to query knowledge base with reasoning to discover logical entailment
- The interface is easy to use
- The interface looks pleasing to users

6 ETHICAL, PROFESSIONAL AND LEGAL ISSUES ADDRESSED.

6.1 Ethical Issues

Our project poses minimal ethical issues. The only apparent possible issue is the exploitation of software prototype testers during the software prototype testing phase of the project. To address this ethical issue we will obtain ethical clearance from the UCT Human Research Ethics Committee.

6.2 Professional Issues

There are no apparent professional issues with the project. The project will be completed to a high quality to provide a platform for further studies in the research field.

6.3 Legal Issues

The software used in our project is open source. To avoid legal issues the software we create will be open source. The software produced will also be the intellectual property of Joshua Abraham and UCT.

7 RELATED WORK

We discuss here the work that has previously been done, more specifically, those that are related to the work done and proposed in this paper.

Since its conception and formalisation, defeasibility has been widely investigated. The notion of the rational closure of a positive knowledge base, K, containing typical inclusions, e.g. penguin $\rightsquigarrow$ fly, was first introduced by Lehmann [16]. This was further developed by Lehman and Magidor [17], who also presented an algorithm to compute this. More notably, however, is the research done in extending description logics with the notion of defeasibility. Britz et al. [18] propose an operator to represent defeasible subsumption operations (which is represented as $\rightsquigarrow$ in this paper) in description logics, as well as describe the workings of it. Prior to this, Casini and Stracia [14] had already extended the algorithm in (Lehman and Magidor 1992) to description logics and showed that it reduces to a sequence of classical entailment operations. Casini et al. [15], however, provide a reformulated version of this algorithm by incorporating the defeasible subsumption operator in Britz et al. [18]. Along with this, many others have endeavoured to implement defeasibility into description logics and ontologies. These include, but are not limited to: Moodley, Meyer and Varzinczak [9]; Bryant and Krause [10]; Niemela [11]; Garcia and Simari [13]; etc. In these, concepts of efficient defeasible reasoning; various defeasible reasoning implementations; defeasible reasoning in description logics ontologies; and defeasible logic programming; etc. are tackled. Furthermore, defeasible reasoning has even been extended to datalog specifically, by Martinez and Deagustini [12] - let it be noted, however, that this has only been done for a subset of datalog and not datalog in it's entirety.

By virtue, alone, of the amount of work being put into this specific area of artificial intelligence and reasoning, it is reasonable to deduce that work done in this field could institute significant effects and/or benefits. Furthermore, we believe that this is indeed the case. Our work is not only to implement defeasibility into a subsection of datalog, but into datalog in its entirety and to show that this implementation stands correct. Furthermore we extend RDFox, the datalog RDF triple store, to allow it the ability to reason with this defeasible datalog - non of which has been done before. We believe that, in so doing, this will not only improve understanding of defeasible descriptive logics and datalog, but will have many other implementational benefits as well.

8 OUTCOMES: EXPECTED IMPACT: EXPECTED RESULTS, EFFECTS; KEY SUCCESS FACTORS.

In this section, we discuss what the expected outcomes of this project are, as well as how we will measure and determine if the project was a success or not. The outcomes discussed here are closely related to the statements made and questions posed in section 4 of this paper.

At the end of this project, we hope to have achieved two main goals, at the minimum. Firstly, regarding the theoretical aspect of this project, we hope to integrate defeasible reasoning into the entirety of the classical datalog declarative logic programming language. Secondly, regarding the implementational aspect of this project, we hope to modify the RDFox software with the intent of allowing it the ability to compute any and all defeasible instances which the datalog reasoner may encounter. Furthermore, we will provide a range of test cases to show that our extended version of the RDFox reasoner works as it is expected to. These test cases will cover as many varying scenarios as we are able to concoct. There

is, however, no way of testing whether our test cases cover every scenario possible - as is the case with most test case sets.

The success of this project is determined by, not only the accomplishment of the two key success factors mentioned above, but also whether the constituents of those factors are provably correct. That is to say that our theoretical integration is both sound and complete and our implementational integration is both bugless and correctly handles all the test cases that we will provide.

9 PROJECT PLAN

In this section we discuss the main deliverables of this project, how we manage potential risks, the project timeline, the resources that are required to complete this project, the project milestones and finally how the work of this project is allocated between the two student researchers involved.

9.1 Deliverables

The main deliverables set out by the department for the honours project, as well as the due dates for these deliverables, are listed in appendix A.

9.2 Risk Management

Our risk management consists of identifying the risks involved in doing this project; a probability rating for the likelihood of each risk; the impact each risk would have on this project and finally mitigation, monitoring and management plans for each risk. Mitigation is aimed at preventing the risk from occurring. Monitoring is aimed at tracking the increasing likelihood of each risk. Management is aimed at the steps to take when a risk occurs. These result are all shown, by means of a risk matrix, in appendix B.

9.3 Timeline

There are two elements of this project – theoretical and implantation. The theoretical element is showing that the algorithm chosen for extending propositional logic with defeasible reasoning maintains certain properties. The implementation element is developing a platform for extending propositional logic with the algorithm to achieve defeasible reasoning. The only overlap between the two elements is with the chosen algorithm. Appendix C shows a Gantt chart representing the project timeline with the above in mind.

9.4 Required resources

We will be using textbooks and research papers, 2 standard laptops (of our own owning), as well as open source software. Many of the required research papers are in the reference section of this proposal. The open source software to be used are listed below:

- RDFox (in which, the datalog language is embedded)
- Various programming languages and their related compilers
- Microsoft Visual Studio and Eclipse IDE

9.5 Milestones

The milestones we have targeted for our project include all of the deliverables set out by the department A. Further milestones, which we have set for ourselves and that can be seen on the timeline in appendix C, follow bellow, in no particular order.

Regarding the theoretical component of the project:

- Investigate and research around what has been shown about the properties we are interested in for this project.
- Investigate the algorithm used to extend propositional logic.
- Determine a procedure for showing the algorithm holds for the interested properties.
- Prove that the algorithm holds for the interested properties.

Regarding the theoretical component of the project:

- Gain a fundamental understanding for the workings of the datalog declarative logic programming language - this includes, mainly, the syntax of datalog.
- Gain a fundamental understanding for the workings of RDFox.
- Gain an understanding for creating wrappers for the RDFox datalog reasoner - this includes experimentation with and testing of various currently existing RDFox wrappers.
- Create the wrapper which extends RDFox with defeasible datalog reasoning capabilities.
- Create test cases, purposed to test the created RDFox wrapper.
- Use the created test cases to test the RDFox wrapper for correctness in handling various defeasible instance based situations.
- Identify and eradicate any persisting bugs in the RDFox wrapper.

9.6 Work Allocation

There are two core elements to this project, that is the theory and the implementation. Thomas Pownall will perform the theory section. This section involves investigating the proposed algorithm for extending propositional logic and ensuring it complies with the previously identified properties we are interested in. Joshua Abraham will perform the implementation. That is, he will develop a platform for creating propositional knowledge bases that allow for defeasible statements. This implementation will make use of RDFox for scalability and the identified algorithm for extending propositional logic with defeasible reasoning.

10 CONCLUSION

The ability to model real world information in elaborate and simple knowledge bases is extremely powerful. It allows us to model a variety of situations and to draw logical conclusions from them using a formalised process of reasoning. One of the most useful features of logical reasoning is entailment and the ability to draw more implicit information from our knowledge base. Furthermore, it

is extremely important that our reasoning does not fail when given new facts. This is often the case in real world modelling where new facts can contradict previously entailed information. The ability to handle such contradictions makes defeasible reasoning extremely valuable for modelling real world information. It is important to note, however, that our defeasible reasoning must still comply with the properties that we desire from propositional logic, otherwise it is not useful. Further, a platform for creating and performing defeasible reasoning should be created to make it applicable to the real world.

Our proposal outlines a plan for accomplishing the above as well as the approach that will be taken and the criteria that will be considered for a successful project. The implemented platform will provide a useful resource for modelling real world knowledge and the theory will provide a platform for further research into defeasible reasoning.

A DELIVERABLES

Date	Deliverable
22/05/2018	Project Proposal
28/05/2018	Project Proposal Presentation
11/06/2018	Revised Proposal Submission
15/06/2018	Project Web Presence
23/07/2018	Initial Software Feasibility Demonstration
27/08/2018	Final Complete Draft
06/09/2018	Project Paper Final Submission
07/09/2018	Project Code Final Submission
17/09/2018	Final Project Demonstration
19/09/2018	Project Poster Due
26/09/2018	Project Web Page
03/10/2018	Reflection Paper

Table 1: Table of Deliverables

B RISK TABLE

Risk	Probability	Impact	Mitigation	Monitoring	Management
Scope creep	6/10	Moderate	Speak with supervisor and get feedback from 2nd reader. Be sensible and cautious about adding additional functionality.	Use the project timeline to ensure there is enough time for each function.	Remove unnecessary functionality and focus on core functions.
Development takes longer than expected due to poor time management and lack of experience.	4/10	Moderate	Speak with supervisor to gain knowledge and experience. Allow slack time in timeline for potential problems that may arise during development.	Use project timeline to identify if a task is taking too long.	Identify core functions that must be produced. Use slack time to get back on schedule.
A team member does not complete their part of the project.	3/10	Moderate	Ensure each team member's part can be a standalone project.	Communicate with team members to ensure they are going to finish.	Since project part can be stand alone project the other team member will be able to complete their part.
Losing work due to theft or computer failure.	2/10	Significant	Create frequent backups of work. Use cloud services so work is always available.	Be vigilant and ensure to remember computer when leaving work station.	Restore from one of the previous back ups or use university workstation and cloud services.
Some of the required resources become unavailable.	2/10	Severe	Use open source software and have copies of resources saved. Ensure resources can be removed without severe disruption.	Check emails to see if resources have announced changes to availability.	Find replacement resources and adapt to the new resources.

Table 2: Matrix Table of Risks and Risk Management Precautions

C GANTT CHART

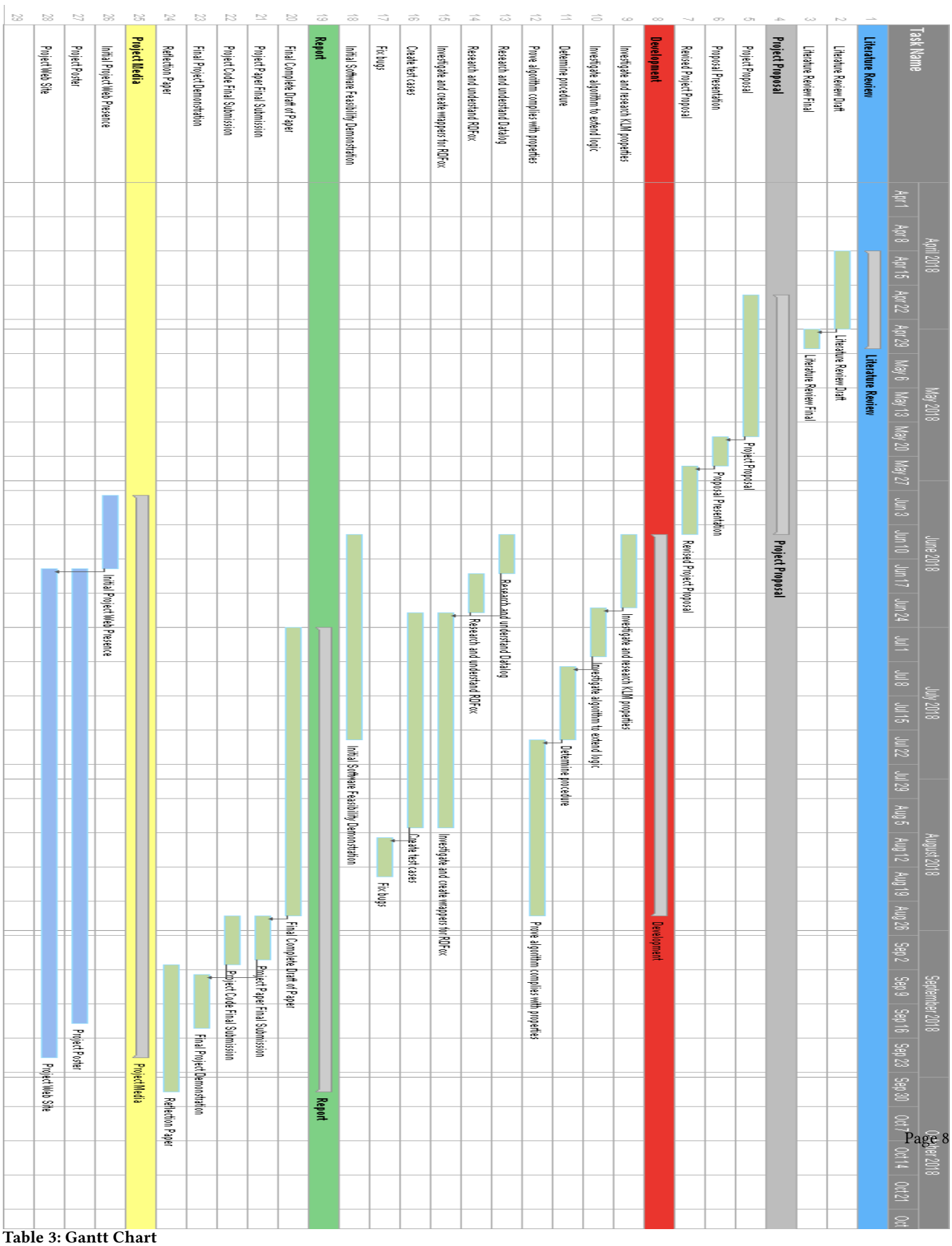

Table 3: Gantt Chart

REFERENCES

[1] Nenov Y., Piro R., Motik B., Horrocks I., Wu Z., Banerjee J. (2015) RDFox: A Highly-Scalable RDF Store. In: Arenas M. et al. (eds) The Semantic Web - ISWC 2015. Lecture Notes in Computer Science, vol 9367. Springer, Cham.

[2] Arregui, A. (2017). *Chisholm's Paradox in Should-Conditionals. SALT 27*: the University of Maryland, College Park. doi: http://dx.doi.org/10.3765/salt.v18i0.2477

[3] Prakken, H. and Sergot, M. (1995) *Contrary-to-duty obligations.*

[4] Prakken, H. and Sartor, G. (1996). A dialectical model of assessing conflicting arguments in legal reasoning. *Artificial Intelligence and Law* 4 (3-4), pp. 331-368. doi: 10.1007/BF00118496

[5] RDF Working Group. (2014). *RDF*. Retrieved May 2018 from https://www.w3.org/RDF/

[6] Miller, E. (2005). An Introduction to the Resource Description Framework. *Bulletin of the Association for Information Science and Technology* 25(1) pp. 15-19

[7] (March 2013) *W3C: SPARQL Query Language for RDF*. Retrieved May 2018 from ://www.w3.org/TR/rdf-sparql-query/#acknowledgements

[8] Shan Shan Huang, Todd Jeffrey Green, and Boon Thau Loo. 2011. Datalog and emerging applications: an interactive tutorial. In Proceedings of the 2011 ACM SIGMOD International Conference on Management of data (SIGMOD '11). ACM, New York, NY, USA, 1213-1216. DOI: https://doi.org/10.1145/1989323.1989456

[9] Moodley, K., Meyer, T. and Varzinczak, I. J. (2012). A defeasible reasoning approach for description logic ontologies. In Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference (SAICSIT '12). ACM, New York, NY, USA, 69-78. DOI=http://dx.doi.org/10.1145/2389836.2389845

[10] Bryant, D. and Krause, P. (2004). A review of current defeasible reasoning implementations. The Knowledge Engineering Review, Vol. 00:0,pp. 1–24, Cambridge University Press. doi: 10.1017/S000000000000000. Printed in the United Kingdom.

[11] Niemela, I. (1995). Towards efficient default reasoning. In Proceedings of the 14th international joint conference on Artificial intelligence - Volume 1 (IJCAI'95), Chris S. Mellish (Ed.), Vol. 1. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 312-318.

[12] Martinez, M. V., Deagustini, C. A. D., Falappa, M. A. and Simari, G. R. (2014). Inconsistency-Tolerant Reasoning in Datalog. *In Advances in Artificial Intelligence* – IBERAMIA 2014, Ana L.C. Bazzan and Karim Pichara (Eds.). Springer International Publishing, Cham, 15–27.

[13] García, A.J., and Simari, G.R. (2014). Defeasible logic programming: DeLP-servers, contextual queries, and explanations for answers. *Argument & Computation*, 5, 63-88.

[14] Casini, G. and Stracia, U. (2010). Rational Closure for Defeasible Description Logics. In: Janhunen T., Niemelä I. (eds) Logics in Artificial Intelligence. JELIA 2010. *Lecture Notes in Computer Science*, vol 6341. Springer, Berlin, Heidelberg

[15] Casini G., Meyer T., Moodley K., Sattler U., Varzinczak I. (2015). Introducing Defeasibility into OWL Ontologies. In The Semantic Web - ISWC 2015. Arenas, M., et al. (eds). *Lecture Notes in Computer Science*, vol 9367. Springer, Cham, 409-426.

[16] Lehmann, D., 1989, "What does a conditional knowledge base entail?," in Proceedings First International Conference on Principles of Knowledge Representation and Reasoning, R. Brachman and H.J. Levesque, eds., Toronto, Ontario.

[17] Lehmann, D. and Magidor, M. (1992). What does a conditional knowledge base entail? *Artificial Intelligence* 55, 1 (1992), 1–60.

[18] Katarina Britz, Thomas Meyer, and Ivan Varzinczak. 2011. Semantic foundation for preferential description logics. In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol. 7106 LNAI. 491–500.