Literature Review for Defeasible Conditional using Answer Set Programming

Jack Mabotja
Department of Computer Science
The University of Cape Town
Cape Town, South Africa
mbtjac003@myuct.ac.za

ABSTRACT

Defeasible reasoning can help agents represent and reason about exceptional and incomplete information. Many approaches to defeasible reasoning have been formalized. We review works about Answer Set Programming (ASP) encodings of these approaches and study their performance. They prove to perform better than their non-ASP counterparts.

CCS CONCEPTS

• Artificial intelligence → Knowledge representation; Reasoning; • Logic programming → Answer Set Programming.

KEYWORDS

Knowledge representation and reasoning, Defeasible Reasoning, Rational closure, Answer Set Programming

1 INTRODUCTION

Knowledge Representation and Reasoning (KRR) is a subfield of *Artificial Intelligence (AI)* concerned with how knowledge can be comprehensively represented, stored, and manipulated in a computer system in a way that enables intelligent reasoning and problem-solving [6]. Given a vast amount of information about a particular subject —facts, rules, concepts, relationships, and more; KRR is about finding effective ways to organize and structure this information so that a computer can understand it and use it to solve problems, make decisions, and answer questions.

The main goal of this endeavour is to equip computers with the same reasoning capabilities as human beings. Humans organize and store knowledge in their minds using complex mental models, concepts, and relationships. This gives us the ability to tackle complex problems in various domains, such as science, engineering, and everyday life. KRR seeks to emulate certain aspects of these cognitive abilities in computer systems to develop intelligent systems that can perform tasks that require human-like intelligence and understanding.

Two example approaches to Knowledge Representation and Reasoning (KRR) are the Logical Representation Scheme and the Network Representation Scheme [8]. The Logical Representation Scheme utilizes formal logic to represent and reason about knowledge. In this approach, knowledge is encoded using logical symbols and inference rules are applied to derive new conclusions. On the other hand, the Network Representation Scheme represents knowledge as a network of interconnected nodes. In this scheme, each node corresponds to a concept or entity, and relationships between concepts are depicted by edges connecting the nodes. This network structure facilitates the representation of complex relationships and hierarchical structures within the knowledge domain.

This review focuses on the logical representation approach to KRR. At its core, the logical representation approach provides a systematic and rigorous framework for expressing knowledge in a precise and structured manner. Knowledge is encoded as a collection of statements or propositions, each of which is expressed using logical symbols and operators [10]. These statements capture relationships, constraints, and facts about the domain being modeled.

One of the key strengths of the logical representation approach lies in its ability to support deductive reasoning. By applying inference rules and logical principles, we can derive new conclusions from existing knowledge [6]. This allows us to explore the implications of given knowledge domains and draw logical conclusions based on the information available.

The two classes of logical formalisms that we look at are *classical propositional logic* and *defeasible logic*. Propositional logic is a reasoning formalism that is defined by a set of connectives and a set of propositional statements–which is referred to as a *knowledge base*–that can be used to construct more complex statements [10].

A key characteristic of propositional logic is that it is monotonic [10]. This means that previous inferences cannot be retracted when new information is added to a knowledge base. This inability to revise inferences upon attaining new information presents a challenge when working with incomplete or exceptional information. *Defeasible logic* aims to address this challenge by adding the ability to reason with exceptional information.

The most popular defeasible logic framework is the KLM Framework [12]. It extends propositional logic by adding the connective $\mid\!\sim$ to the set of propositional connectives. This connective gives previously propositional logic knowledge bases the ability to handle exceptional information. We will give a proper description of these formalisms in the next section.

This review is for a research project whose objective is to implement rational closure, as defined in the KLM framework [12] for defeasible reasoning. The goal is to use Answer Set Programming to encode the rational closure algorithm. This will be followed by a preliminary empirical analysis of performance. After this, attempts will be made to optimize the implementation.

The rest of the review will be structured as follows: the next section provides background information on the topics in question. This includes Propositional Logic, Defeasible Logic, and Answer Set Programming. Sections 3, 4, and 5 discuss the literature on works related to our own.

Section 3 discusses works that set out to implement defeasible reasoning formalisms and frameworks using ASP semantics. Section 4 discusses works that implemented defeasible reasoning frameworks using semantics other than the answer set semantics. Section 5 discusses works that attempted to optimize the rational closure algorithm for practical implementation.

Section 6 consolidates the reviewed studies. Section 7 will closes the review by providing conclusions that can be drawn from the works reviewed and discussed, and also final remarks about the project moving forward.

2 BACKGROUND

2.1 Propositional Logic

Propositional logic is a basic formal system used in KRR to represent and reason about statements. It deals with statements that can be either true or false, and how to combine them to form more complex statements [10].

2.1.1 Syntax

The basic building blocks of propositional logic are statements called propositional *atoms* [10,11]. They represent simple statements like "It's raining" or "The sun is shining", that cannot be broken down further. Propositional atoms are usually denoted using lowercase Latin letters such as p, q, and r. Using this denotation scheme, the previous statements can be denoted by p and q respectively.

The set of all propositional atoms is denoted as $\mathcal{P}$. This set is accompanied by the set of connectives: $\{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$. These two sets can be combined to create more complex statements, which are often referred to as *formulas*. The set of all formulas is typically denoted as $\mathcal{L}$. The formulas are denoted using lowercase Greek letters such α and β.

In extending the previous example, $\mathcal{P} := \{p, q\}$. The two atoms can be combined using the connectives $\rightarrow$ and $\neg$ to yield the formula $p \rightarrow \neg q$, which can be read as "The sun does not shine when it rains." This formula can be denoted as $\alpha \in \mathcal{L}$.

2.1.2 Semantics

The semantics defines the meaning of logical expressions, such as propositions, and formulas. Semantics seeks to define what it means for a logical expression to be true or false, based on the interpretation of the symbols and rules of the logical system [10]. Table 1 summarizes the semantics of the propositional connectives.

Table 1: The semantics of propositional connectives

Name	Symbol	Example	Meaning
Negation	$\neg$	$\neg p$	not p
Disjunction	$\vee$	$p \vee q$	p or q
Conjunction	$\wedge$	$p \wedge q$	p and q
Implication	$\rightarrow$	$p \rightarrow q$	if p then q
Bi-implication	$\leftrightarrow$	$p \leftrightarrow q$	p if and only if q

An *atom* can have a truth value of either true (T) or false (F). When an atom has a truth value of T, it is said to be *satisfied*. An atom gets assigned a truth value by a function called a *valuation*. A valuation is a function that maps every $p \in \mathcal{P}$ to a truth value $u \in \{T, F\}$. If a valuation u maps an atom p to T (i.e. $u(p) = T$), then the valuation is said to satisfy the atom, denoted as $u \Vdash p$.

This notion of satisfiability can be extended to any $\alpha, \beta \in \mathcal{L}$ where the satisfiability of a formula γ by a valuation v will depend on the individual atoms constituting γ and the connectives used to combine them. A valuation u that satisfies a formula α, is referred to as a model of α.

Let $\mathcal{U}$ be the set of all valuations for the language $\mathcal{L}$. The set of all models of a formula α is denoted as $[\![\alpha]\!]$, and defined as $[\![\alpha]\!] := \{u \in \mathcal{U} \mid u \Vdash \alpha\}$. In simple terms, $[\![\alpha]\!]$ is the set of all valuations $u \in \mathcal{U}$ that satisfy α.

Recall that a finite set $\mathcal{K} \subseteq \mathcal{L}$ of propositional formulas is referred to as a *knowledge base*. A valuation u satisfies $\mathcal{K}$, denoted $u \Vdash \mathcal{K}$, if and only if it satisfies every statement $\alpha \in \mathcal{K}$. A formula $\alpha \in \mathcal{L}$ is said to be entailed by $\mathcal{K}$ (denoted as $\mathcal{K} \models \alpha$) if and only if for every $u \in \mathcal{U}$ such that $u \Vdash \mathcal{K}$, it is the case that $u \Vdash \alpha$. This means that a knowledge base entails a formula if every valuation that satisfies it, also satisfies the formula. Another way of stating this is $\mathcal{K} \models \alpha$ *iff* $[\![\mathcal{K}]\!] \subseteq [\![\alpha]\!]$. This is the definition of classical entailment.

While classical entailment is a sufficiently powerful and expressive formalism for most cases, it does have a property that limits its applications in certain cases. This property is *monotonicity*. Monotonicity states that the addition of new information to our knowledge base should never lead to the detraction of inferences made prior to the addition of this new information [10].

To fully understand why monotonicity can sometimes be an undesirable property for a logic formalism, consider the following example. Suppose we have a knowledge base $\mathcal{K}$ defined as follows:

$$\mathcal{K} := \{b \rightarrow f, p \rightarrow b, p \rightarrow \neg f, t \rightarrow b\}$$

where
b is bird, f is flies, p is penguin, and t is Tweety, a name given to some bird. A deduction that can be made from the above knowledge base is that Tweety can fly, denoted as $t \rightarrow f$. Given the above information, this is a sound deduction.

If we go on to add the statement $t \rightarrow p$, specifying the specific type of bird that Tweety is, an inference that can be made is $t \rightarrow \neg f$. This is also sound since Tweety is a penguin, a specific bird type, and one that cannot fly. Because of monotonicity, the previous inference, which contradicts the new inference cannot be retracted. The presence of both inferences leads to a further inference $\neg t$, which says that Tweety does not exist since Tweety has contradictory properties.

This behaviour is undesirable, especially since most knowledge bases might have exceptional and contradictory information. This necessitates the development and study of *non-monotonic* logic formalisms. Non-monotonic logics are a broad class of formal systems that allow for conclusions to be withdrawn in the presence of new information, contrary to classical logic [7]. This review considers defeasible logic.

2.2 Defeasible Logic

Defeasible logics are a specific type of non-monotonic logic that deal with reasoning about rules that are generally true but can be overridden or defeated by specific conditions or exceptions [7]. Defeasible reasoning is used to model everyday reasoning, where conclusions are drawn based on general principles but can be revised in light of new information or exceptions.

There are numerous approaches to defeasible reasoning, but the most popular and the one considered in this work, is the KLM framework [12]. Kraus, Lehmann, and Magidor extended propositional logic to give it non-monotonic capabilities. Their framework has "been successful due to its elegance and robustness" [16].

They extended propositional logic by adding an operator to denote the concept of defeasible implication ($\mid\!\sim$). This operator can be used to construct statements called *conditional assertions*. Conditional assertions can be defeated by other statements in a given knowledge base. A conditional assertion has the form: $\alpha \mid\!\sim \beta$, and is read "β defeasibly follows from α".

They also introduced the notion of *rational closure*. Rational closure takes a set of conditional assertions and a defeasible logic and provides a precise definition of the kind of inferences that can be made from the conditional knowledge base [16].

2.3 Rational Closure

Rational closure is used in the KLM framework to perform defeasible entailment computations over defeasible knowledge bases, and it is denoted using $\mid\!\approx$. When computing the rational closure of a defeasible knowledge base, all statements in the knowledge base are assigned a ranking [7]. This ranking uses the idea of *exceptionality*, which determines the specificity of a given statement in the knowledge base. Once the statements have been ranked, they are then treated as classical statements.

To determine the exceptionality of statements, a *materialisation* of defeasible statements is performed. Materialisation converts a defeasible statement to its classical counterpart [7,10]. For example, the materialisation of $\alpha \mid\!\sim \beta$ is $\alpha \rightarrow \beta$. The materialisation counterpart of a defeasible knowledge base $\mathcal{K}$ is computed by replacing every defeasible implication $\alpha \mid\!\sim \beta \in \mathcal{K}$ with $\alpha \rightarrow \beta$, and is denoted as $\vec{\mathcal{K}}$ [10].

In summary, to compute the rational closure of a defeasible knowledge base $\mathcal{K}$, compute the materialization counterpart $\vec{\mathcal{K}}$ of the knowledge base. Determine the exceptionality of all statements in $\vec{\mathcal{K}}$, and then rank them according to their exceptionality [7]. This gives the rational closure. The rational closure can then be used to compute defeasible entailment checks.

2.4 Answer Set Programming

Answer Set Programming (ASP) is a declarative programming paradigm oriented towards difficult (primarily NP-hard) search problems [4,13]. It is based on the stable model semantics of logic programming [4,13].

In ASP, a problem is specified as a logic program, which consists of rules and facts. Rules define how to derive new facts from existing ones, and facts are statements that are considered to be true. The program is then used to compute *answer sets* (also called *stable models*), which are sets of facts that are consistent with the rules and represent possible solutions to the problem [13]. ASP is particularly well-suited for solving combinatorial search problems [13], such as scheduling, planning, and configuration problems.

The process of solving a problem using ASP is as follows: (1) The problem to be solved is encoded using answer set semantics, the result of this is an ASP program; (2) A *grounder* is used to generate a finite propositional representation of the ASP program; (3) A solver is used to compute the stable models of the propositional program [4]. The stable models can be interpreted as solutions to the original problem [3].

3 DEFEASIBLE REASONING IMPLEMENTATIONS USING ANSWER SET SEMANTICS

ASP provides an expressive language for representing complex knowledge and reasoning tasks. It is well-suited for modelling defeasible reasoning problems that involve uncertainty and exceptions.

It also has efficient solvers, such as Clingo, which can efficiently find answer sets for complex ASP programs. It is for the aforementioned reasons that ASP is used by researchers to encode novel or pre-existing defeasible reasoning formalisms. Some of these formalisms and encodings are provided theoretically while some are also implemented programmatically.

3.1 THEORETICAL ASP ENCODINGS

Lim et. al. [14] developed a defeasible logic Domain Specific Language (DSL) called L4. L4 is intended to represent legal texts and norms and ways of reasoning about them. In defining L4, they used a different notion of defeasibility than the one we focus on in this work. They define the notion of *rule modifiers*, which “limit the applicability of rules and make them defeasible”.

These rule modifiers are based on the terminology often used in law texts. They can be used to override one rule with another when these rules contradict each other. They also provided classical and ASP semantics for these modifiers. They went on to provide two classical logic encodings and a more elaborate ASP encoding.

It should be noted that the ASP encoding they provided has not been implemented programmatically. Only one of the classical encodings has been implemented. This means that it is not known yet if the ASP encoding will provide a significantly better performance than the classical encodings. They also do not provide empirical experimentations of the implemented classical encodings. Although the L4 system seems to work on paper, it is not known yet how it will perform in a computer system. The language is still under development, together with its implementation using ASP encoding.

In another study, Wan et. al. presented a framework in [19] to unify the multiple approaches for the defeasibility of disjunctive logic programs. As opposed to L4, it is not specific to a given domain. The framework, Answer Set Programming with Defaults and Argumentation Theories (ASPDA) is an extension of their earlier work on LPDA [18], a framework which unified defeasible reasoning approaches that rely on well-founded semantics.

LPDA uses the concept of *argumentation theories*, which are sets of axioms (or arguments) that determine when certain rules should be considered defeated by other rules, as an abstraction for defeasibility. LPDA could not capture most defeasible approaches as they are based on stable model semantics. ASPDA uses answer set semantics which are more general than well-founded semantics [19]. As a result, LD-PDA and ASPDA both capture most approaches to defeasible reasoning.

They show that an ASPDA program can be reduced to an ASP program, which means that ASP solvers can be used to solve ASPDA problems. Like the L4 DSL, no programmatic implementation was provided.

The above studies suffer from similar limitations. They both lack programmatic implementations. They can therefore only be reasoned about from a theoretical perspective. Another thing worth considering is that the frameworks are different approaches to defeasible reasoning and one of them is domain-specific, while the other is more general. This makes direct comparisons difficult. The next section focuses on ASP encodings of defeasible reasoning frameworks that were implemented programmatically.

3.2 PRACTICAL ASP ENCODINGS

Alsinet et. al. proposed an implementation of a defeasible reasoning framework called Recursive Possibilistic Defeasible Logic Programming (RP-DeLP) using ASP encodings [2].

RP-Delp is an extension of Possibilistic Defeasible Logic Programming (P-DeLP), where rules have levels of strength, which are formalized as degrees of possibilistic necessity [1]. RP-DeLP adds recursive warrant semantics to P-DeLP, which are based on the idea of collective conflict among arguments.

They developed an ASP interpreter for RP-DeLP in [2], which takes a knowledge base (i.e. program) as input and returns a pair of sets of warranted and blocked conclusions with maximum strength.

They tested its performance empirically against an SAT implementation (using MiniSAT, an efficient SAT solver) they developed alongside the ASP implementation (using Clingo and the Potassco suite). The results showed that the ASP implementation could solve RP-DeLP programs faster than the SAT implementation.

Finally, Ben-Naim et. al. proposed an ASP encoding of prioritized removed set revision (PRSR) [5]. PRSR extends Removed Sets Revision (RSR), which removes some formulas from a knowledge base when new information is added to restore consistency. The encoding was applied to a geographical problem where water height at different locations in flooded valleys was assessed.

The valley was segmented into 120 different compartments. They encoded the RSR framework using ASP and MiniSAT. Their empirical results show that the ASP encoding and the SAT encoding behaved similarly for a few compartments but the ASP encoding outperformed the SAT encoding as more compartments were added. The addition of priorities to the RSR encoding reduced the running time for the computing of answer sets.

The empirical results of both encodings show that the ASP encodings performed better than their non-ASP counterparts. Based on this, it can be expected that ASP encodings of other defeasible reasoning frameworks can be expected to perform well. However, both the above comparisons were against SAT implementations and therefore these results cannot be generalized to other non-SAT implementations. To explore this implication further, the next section discusses a defeasible reasoning implementation that does not use ASP semantics.

4 NON-ASP DEFEASIBLE REASONING IMPLEMENTATIONS

In [9], Harrison and Meyer developed a system to extend Datalog to enable it to handle exceptional knowledge. They defined two extensions to Datalog. The first one is a practical defeasible Datalog language to implement defeasible reasoning in Datalog. The second one is a theoretical defeasible reasoning Datalog language that they used to prove that their approach meets the KLM requirements for a rational consequence relation. To this end, they redefined the rational closure algorithm for their approach.

They provided a Java implementation of their approach, called DDLV. DDLV can create, edit and query a defeasible Datalog program. It uses DLV, "a Disjunctive Logic Programming (DLP) system that uses an extended Disjunctive Datalog as its kernel language", to perform classical Datalog entailment checks. They then performed an empirical evaluation of DDLV using the Defeasible-Infereence Platform for Description Logics (DIP) as a benchmark. DIP is a defeasible reasoner developed in [15] by Meyer et. al. DIP is a different language than DDLV, but was chosen for the comparison because it has a similar preferential reasoning approach to DDLV and there were no other referential reasoning systems for defeasible Datalog to compare DDLV to.

The empirical results show that DDLV's ranking compilation performance proved to be considerably better than DIP's performance. However, DIP, on average, out-performs DDLV in defeasible entailment checks even though DDLV performs better in the worst case.

Similar to the ASP-encodings of defeasible reasoning, DDLV also performed better than its counterpart. Although the ASP encodings performed better than the SAT encodings, they may not perform significantly better than other non-SAT encodings, like DDLV.

5 RATIONAL CLOSURE OPTIMIZATIONS

When exploring the available literature on KLM-style defeasible reasoning, we found very little work on possible optimizations of the rational closure algorithm. In [17], Slater and Meyer addressed the computational and storage limitations of implementations of minimal ranked entailment, which would otherwise be impractical.

They propose reduced minimal ranked entailment to address the aforementioned issue. Their approach uses reduced ordered binary decision diagrams to eliminate redundant information and thus enhance computational efficiency and reduce memory requirements. Although it is expected to perform better than the original minimal ranked entailment approach, Slater and Meyer did not provide an empirical validation of their approach.

6 DISCUSSION

The implementation of defeasible reasoning using Answer Set Programming (ASP) offers a promising avenue for addressing complex reasoning tasks in various domains. The preceding sections provided a review of prior works, most of which employed ASP-based formalisms for defeasible reasoning, highlighting both theoretical frameworks and practical implementations.

These works are not strictly related to our project but they do offer useful insights. The most prominent one is that ASP encodings of defeasible reasoning formalisms seem to computationally perform better than SAT encodings. It is however not known if these results can be extrapolated to other non-SAT encodings of defeasible reasoning. It is also worth noting that the vast number of different defeasible formalisms make it difficult to compare their ASP encodings. While an ASP encoding might perform well for one defeasible reasoning approach, it might not perform as well for some other approach.

7 CONCLUSIONS

ASP is a declarative programming paradigm, in which problems are defined in terms of their solutions [13]. Solutions to declarative programs are found through a process of automated reasoning. This review explored problems where defeasible reasoning frameworks were encoded using answer set semantics. Of these encodings, those that were implemented were proven to perform better than their counterparts. Although these results may not be universal, they show that an encoding of KLM-style rational closure using answer set semantics is worth pursuing.

An optimization of the rational closure algorithm was also explored. It promises to improve the computational performance and space requirements of the implementation of the rational closure algorithm.

REFERENCES

[1] Teresa Alsinet, Ramón Béjar, and Lluis Godo. 2010. A characterization of collective conflict for defeasible argumentation. 2010. 27–38. https://doi.org/10.3233/978-1-60750-618-8-27

[2] Teresa Alsinet, Ramón Béjar, Luis Godo, and Francesc Guitart. 2012. Using Answer Set Programming for an Scalable Implementation of Defeasible Argumentation. In *2012 IEEE 24th International Conference on Tools with Artificial Intelligence*, 2012. 1016–1021. https://doi.org/10.1109/ICTAI.2012.171

[3] Christian Anger, Kathrin Konczak, Thomas Linke, and Torsten Schaub. 2005. A Glimpse of Answer Set Programming. *KI* 19, (2005), 12–.

[4] Theofanis I. Aravanis and Pavlos Peppas. 2017. Belief Revision in Answer Set Programming. In *Proceedings of the 21st Pan-Hellenic Conference on Informatics (PCI '17)*, 2017. Association for Computing Machinery, Larissa, Greece. https://doi.org/10.1145/3139367.3139387

[5] Jonathan Ben-Naim, Salem Benferhat, Odile Papini, and Eric Würbel. 2004. An Answer Set Programming Encoding of Prioritized Removed Sets Revision: Application to GIS. In *9th European Conference Logics in Artificial Intelligence (JELIA 2004) (Lecture Notes in Computer Science book series (LNCS)*, September 2004. 604–616. https://doi.org/10.1007/978-3-540-30227-8_50

[6] Ronal Brachman and Hector Levesque. 2004. *Knowledge Representation and Reasoning*. Morgan Kaufmann Publishers, San Francisco, CA.

[7] Victoria Chama. 2020. Explanation for defeasible entailment. Retrieved from http://hdl.handle.net/11427/32206

[8] Sampada Gulavani. 2019. Knowledge Representation Approaches in Artificial Intelligence. *International Journal of Science and Research* 8, 10 (2019), .

[9] Michael Harrison and Thomas Meyer. 2020. DDLV: A system for rational preferential reasoning for Datalog. *South African Computer Journal* 32, (2020), 184–217. https://doi.org/10.18489/sacj.v32i2.850

[10] Adam Kaliski. 2020. An overview of KLM-style defeasible entailment. Retrieved from http://hdl.handle.net/11427/32743

[11] Hans Kleine Büning and Theodor Lettmann. 1999. *Propositional logic - deduction and algorithms*. Cambridge University Press.

[12] Sarit Kraus, Daniel Lehmann, and Menachem Magidor. 1990. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial intelligence* 44, 1–2 (1990), 167–207.

[13] Vladimir Lifschitz. 2019. *Answer Set Programming* (1st ed.). Springer Publishing Company, Incorporated.

[14] How Khang Lim, Avishkar Mahajan, Martin Strecker, and Meng Weng Wong. 2022. Automating Defeasible Reasoning in Law. *ArXiv* (2022). Retrieved from https://api.semanticscholar.org/CorpusID:248811071

[15] Kody Moodley, Thomas Meyer, and Uli Sattler. 2014. DIP: A Defeasible-Inference Platform for OWL Ontologies. In *DL 2014 (CEUR Workshop Proceedings)*, 2014. CEUR Workshop Proceedings, Germany, 671–683.

[16] Kody Moodley, Thomas Meyer, and Ivan José Varzinczak. 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)*, 2012. Association for Computing Machinery, Pretoria, South Africa, 69–78. https://doi.org/10.1145/2389836.2389845

[17] Luke Slater and Thomas Meyer. 2023. Extending Defeasible Reasoning Beyond Rational Closure. In *Artificial Intelligence Research*, 2023. Springer Nature Switzerland, Cham, 151–171. https://doi.org/10.1007/978-3-031-49002-6_11

[18] Hui Wan, Benjamin Grosof, Michael Kifer, Paul Fodor, and Senlin Liang. 2009. Logic Programming with Defaults and Argumentation Theories. 2009. 432–448. https://doi.org/10.1007/978-3-642-02846-5_35

[19] Hui Wan, Michael Kifer, and Benjamin Grosof. 2010. Defeasibility in Answer Set Programs via Argumentation Theories. In *Web Reasoning and Rule Systems*, 2010. Springer Berlin Heidelberg, Berlin, Heidelberg, 149–163. https://doi.org/10.1007/978-3-642-15918-3_12