Literature Review for Propositional Typicality Reasoning

Andrew Howe-Ely
HWLAND004
University of Cape Town

ABSTRACT

Defeasible reasoning is an extension to reasoning in propositional logic that is able to deal with exceptions to general rules. Using this we can make statements such as "Birds typically fly, but penguins don't" and reason about them. Propositional Typicality Logic (PTL) is a recently proposed extension that allows the notion of typicality to occur anywhere in a propositional statement, rather than in just the antecedent. Reasoning in propositional logic amounts to computing entailment. In defeasible reasoning the Rational Closure is the key concept for entailment. Two forms of entailment have been proposed for PTL, which build upon entailment in defeasible reasoning, namely LM-Entailment and PT-Entailment.

This paper discusses the theory of propositional logic and methods of computing entailment in a defeasible setting. Examples of the different statements that can be made in each setting are given as well as how conclusions are drawn. LM and PT-Entailment are discussed with the practical implementation of a reasoner in mind, using one of them: PT-Entailment. The two forms of entailment have advantages and disadvantages, with more investigation still to be done.

1 INTRODUCTION

The Propositional Typicality Reasoning (PTR) Project will explore the theory of Propositional Typicality Logic (PTL) [4], and the implementation of an algorithm for reasoning in PTL. PTL is a recently proposed logic that builds on the work done on defeasibility and on propositional logic. The challenge of this new language is to develop the theory of how to reason in it.

The notion of typicality and moreover defeasibility allows us to handle exceptionality in a logical system. Typicality adds an additional operator that allows more expressivity to this end. However, some results from defeasible reasoning do not hold, and a new form of entailment is needed in order to reason. Two forms of entailment have been put forward as candidates for reasoning in PTL. The work of the PTR project has been divided between the two forms of entailment for PTL, with the implementation of a reasoner using each.

This review will discuss the topics of propositional logic, defeasible reasoning and PTL, as well as entailment in PTL and the implementation of these algorithms.

Propositional logic is a fundamental concept in mathematics that has been studied for centuries. The concepts it includes are vital to the field of Computer Science, and particularly in the fields of knowledge representation and Artificial Intelligence. Propositional logic has been formalised as a language that joins statements together in a way that is either entirely true or false. From this we can reason about things, and draw conclusions from a set of assumptions. Entailment is the mechanism of reasoning in propositional logic, the next section of this review will explain the basic theory leading up to that.

Defeasible reasoning is an extension to propositional logic where exceptions to rules are allowed to occur. This is done by saying something is typically the case, and therefore having room for exceptionality without contradiction. This way to write statements leads to a different form of entailment and the literature about this topic is discussed in the third section. Allowing for exceptions means we can describe things in a more interesting way, as well as providing a way to reason formally in a manner that humans can informally do very easily.

Another form of defeasible reasoning is introduced in PTL. A new operator is defined in this language, which is an extension of propositional logic. Since this is a newly proposed language there is ongoing and recent research into how to reason in it. The motivation behind PTL is to create a more expressive language that can hopefully capture exceptions and typical behaviour in more ways, as well as reason about these more expressive statements. Different forms of entailment have been proposed in order to compute reasoning in this language. These are essentially two algorithms that will be explored and developed in the PTR project. The practical part of the project will also be discussed where a reasoner needs to be developed.

2 PROPOSITIONAL LOGIC

Propositional logic is a logical system that builds up sentences using atomic propositions that are either True or False. These atoms are joined with logical connectives to build more complex formulas. For the purposes of this review we will be basing the use of propositional logic on the work found in Ben-Ari, Chapter 2 [1]. However, many different books and papers on this topic exist in the literature. Classical propositional logic will provide the basis for the other topics explored in this paper. Propositional logic allows us to make statements about the world and reason about them. This is particularly important in the field of knowledge representation.

Atomic propositions are typically denoted by symbols like p or q. The connectives used are: $\neg$ (negation, or Not), $\wedge$ (conjunction, or And), $\vee$ (disjunction, or Or), $\rightarrow$ (implication) and $\leftrightarrow$ (equivalence). There are other operators but they are not necessary for our purposes. These connectives are also referred to as Boolean operators, where negation is a unary operator with one operand and the rest are binary operators. $\top$ (top, meaning unconditionally true) and $\bot$ (bottom, unconditionally false) are also connectives used to define the language. The operators $\neg$ and $\wedge$ can be used to define the rest of the Boolean operators. For example, the formula $p \rightarrow q$ would be a propositional sentence reading p implies q or if p then q. p is called the antecedent and q is the consequent. We call a finite set of sentences in a language a knowledge base.

An interpretation of a formula A is a function that assigns a truth value to every atom in A. If we have p and q as atoms in formula,

then there are four interpretations where p and q are either True or False, or 1 or 0 depending on the notation used. If there are n atoms in a formula there will be 2^n interpretations.

An example of an interpretation over a set of variables p and q would be if p was False and q was True, represented either as 01, or as $\{\neg p, q\}$. The negation of p represents p being false. Over an interpretation the whole formula can be evaluated as True or False. A truth table is a way of showing the different interpretations and evaluations. The following truth table shown in Figure 1 demonstrates $p \rightarrow q$.

p	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

Figure 1: A truth table for $p \rightarrow q$

We call a formula satisfiable if it evaluates as True for some interpretation of the atoms. We call this satisfying interpretation a Model for A. From this we come to the concept of logical consequence in propositional logic. For a set of formulas U, A is a logical consequence of U if and only if every model of U is a model of A. This is denoted by $U \models A$. This is also referred to as entailment, and is how reasoning is done in propositional logic.

The meaning of the operators with respect to interpretations are as follows: $\neg A$ is true in an interpretation if A is false in it. $A \wedge B$ is true if both A and B are true in the interpretation. $A \vee B$ is true if one or both of A and B are true in the interpretation. $A \rightarrow B$ is true if A is false or if B is true. This operator is also called a conditional. $A \leftrightarrow B$ is true in an interpretation if both $A \rightarrow B$ and $B \rightarrow A$ are true.

The propositional language used, and enriched, in later sections is typically defined in a standard way as follows: $\mathcal{P}$ is a finite set of propositional *atoms*. $p, q, \ldots$ are used as meta-variables for atoms. Propositional sentences are denoted by $\alpha, \beta, \ldots$. If α is in $\mathcal{P}$ it is a sentence. If α is a sentence then so is $\neg\alpha$. If α and β are sentences then so are $\alpha \wedge \beta$, $\alpha \vee \beta$, $\alpha \rightarrow \beta$ and $\alpha \leftrightarrow \beta$. $\mathcal{L}$ denotes the set of all propositional sentences.

An example from the literature, namely Booth et al. [3], recursively defines sentences as: $\alpha ::= p \mid \neg\alpha \mid \alpha \wedge \alpha \mid \top \mid \bot$. The other Boolean connectives ($\vee, \rightarrow, \leftrightarrow, \ldots$) are defined in terms of $\neg$ and $\wedge$.

Reasoning in propositional logic amounts to computing entailment. For this reason, entailment is the most important part of reasoning in propositional logic. The following example will illustrate reasoning using entailment. In the literature this is called the Tweety example. First let our knowledge base $\mathcal{K} = \{p \rightarrow b, b \rightarrow f\}$. b represents being a bird, p a penguin and f being able to fly. The sentences can be interpreted as meaning penguins are birds and birds fly. From this we can reason that $\mathcal{K} \models p \rightarrow f$, i.e. that penguins fly. Say we now add to our knowledge base the sentence $p \rightarrow \neg f$, meaning penguins do not fly. Figure 2 shows a table of different interpretations and whether they are a model for each statement and the knowledge base itself, for this example.

Interpretations			Is a model for		
p	b	f	$p \rightarrow f$	$\neg p$	$\mathcal{K}$
1	1	1	Yes		
1	1	0			
1	0	1	Yes		
0	1	1	Yes	Yes	Yes
1	0	0			
0	1	0	Yes	Yes	
0	0	1	Yes	Yes	Yes
0	0	0	Yes	Yes	Yes

Figure 2: A table of models for $\mathcal{K} = \{p \rightarrow b, b \rightarrow f, p \rightarrow \neg f\}$

Since whenever an interpretation is a model for $\mathcal{K}$ it is a model for each of the two statements shown, they are logically entailed by $\mathcal{K}$. Using classical entailment we reason that $\mathcal{K} \models \neg p$. In other words, a conclusion from this is that there are no penguins, since classical logic does not allow for exceptions. However, we still have the conclusion that penguins fly, since entailment in propositional logic is monotonic and we do not lose conclusions even when we add new information. Clearly this is rather restrictive when thinking about reasoning intelligently, since we want to be able to handle exceptionality. This example will be revisited in the following sections.

Reasoners that use propositional logic compute using this form of entailment. More will be discussed about these in the implementation section.

3 DEFEASIBLE REASONING

Defeasible reasoning extends reasoning in propositional logic by being able to deal with exceptions to rules. This allows for the creation of statements that were not previously able to make in classical propositional logic. Recall the Tweety example from propositional logic, where given an exception that penguins do not fly lead us to conclude there are no penguins. With defeasibility we say that birds typically fly, and then conclude that penguins are simply exceptional birds that do not fly. In order to do this a new connective is introduced: $\mid\sim$. This is a new type of conditional. The statement $a \mid\sim b$ reads as: if a, then typically b. This approach is referred to as the KLM approach and was investigated in Kraus et al. [12].

Much of the literature discusses the property of non-monotonicity that defeasibility possesses. Monotonicity informally means given a set of statements in a knowledge base, if we add more statements, we do not lose any consequences. This is not a characteristic that defeasible reasoning should have since conclusions should be retracted in the case of new conflicting information.

The following example is to compare the Tweety example from propositional logic and use it in defeasible reasoning. Let our knowledge base be $\mathcal{K} = \{b \mid\sim f, p \rightarrow b\}$. Interpreted as: birds typically fly, and penguins are birds. From this we would conclude that penguins typically fly, using ranked entailment. However, if we add the sentence $p \mid\sim \neg f$ to our knowledge base (penguins typically do not fly) we retract the previous conclusion that penguins typically fly, since now penguins are not typical birds, but rather exceptional

ones. This is a display of non-monotonic reasoning, as we retracted a conclusion after the addition of new information and is what is required from defeasible reasoning.

The $\mid\sim$ operator is said to be a preferential conditional since it has the following properties, put forward by the authors Kraus et al.: Reflexivity (Ref), Left Logical Equivalence (LLE), Right Weakening (RW) and Cautious Monotonicity (CM).

$$\text{(Ref)} \quad \alpha \mid\sim \alpha \qquad \text{(LLE)} \quad \frac{\models \alpha \leftrightarrow \beta,\ \alpha \mid\sim \gamma}{\beta \mid\sim \gamma}$$

$$\text{(And)} \quad \frac{\alpha \mid\sim \beta,\ \alpha \mid\sim \gamma}{\alpha \mid\sim \beta \wedge \gamma} \qquad \text{(Or)} \quad \frac{\alpha \mid\sim \gamma,\ \beta \mid\sim \gamma}{\alpha \vee \beta \mid\sim \gamma}$$

$$\text{(RW)} \quad \frac{\alpha \mid\sim \beta,\ \models \beta \rightarrow \gamma}{\alpha \mid\sim \gamma} \qquad \text{(CM)} \quad \frac{\alpha \mid\sim \beta,\ \alpha \mid\sim \gamma}{\alpha \wedge \beta \mid\sim \gamma}$$

Another property that $\mid\sim$ has is Rational Monotonicity (RM). From this we can say it is a rational conditional.

$$\text{(RM)} \quad \frac{\alpha \mid\sim \gamma,\ \alpha \not\mid\sim \neg\beta}{\alpha \wedge \beta \mid\sim \gamma}$$

In Lehmann and Magidor [14] ordered structures called ranked interpretations are used to reason with the defeasible conditional. A ranked interpretation is an ordering of different interpretations and can be written as a partition $(L_0, \ldots, L_n)$. The valuations lower down in the ordering are the ones that are more typical. We say a ranked interpretation is a ranked model of some set of conditionals if it entails every conditional in that set. A new preference relation $\trianglelefteq_{\text{LM}}$ is defined in Lehmann and Magidor's work. This forms a partial order over ranked interpretations. The intuition is that the ranked interpretation with the most typical valuations possible at the bottom of the ranking should be preferred. Using this order there is shown to be a unique minimal ranked interpretation among all the ranked models. Reasoning in this KLM framework is thought of as deriving new conditionals from a set of conditionals C. The Rational Closure of the set C gives us a form of entailment for defeasibility. This is not the only form of entailment for defeasibility but it is the one that the rest of the paper will build on.

To reason using the Rational Closure there are essentially two approaches. The first is an algorithm to determine if a conditional follows from a knowledge base $\mathcal{K}$. The second is the rational closure construction, which builds the unique minimal ranked model. The Rational Closure algorithm is first established in Lehmann and Magidor [14], and then another more detailed approach based on the former is put forward in Booth and Paris [5]. The algorithm determines whether, given a conditional assertion $a \mid\sim b$, it is entailed by a knowledge base $\mathcal{K}$. A ranking of sentences in the knowledge base based on their exceptionality is produced, with the classical statements being put on their own level, L_∞. Sentences are put into different levels depending on their exceptionality, with the less exceptional sentences being put on the lower levels. Defeasible sentences are converted into classical implications. If the antecedent of the statement being checked by the algorithm is False from the knowledge base, the highest level of exceptionality is dropped from the ranking. The algorithm checks if the sentence is entailed by the knowledge base once this condition is met.

From this, the algorithm finds whether the statement is True or False in the knowledge base. For more detail on the algorithm see the papers mentioned.

Figure 3 shows a ranking of the sentences in the knowledge base $\mathcal{K} = \{b \mid\sim f, p \rightarrow b, p \mid\sim \neg f, b \mid\sim w\}$ when using the rational closure algorithm. w represents the property of having wings, with the rest of the sentences the same as before. Using the algorithm; the statement that $p \mid\sim w$ is entailed by this knowledge base.

L_∞ :	$p \rightarrow b$	
L_1 :	$p \mid\sim \neg f$	
L_0 :	$b \mid\sim f$	$b \mid\sim w$

Figure 3: Ranking sentences in $\mathcal{K}$

The Rational Closure construction on the other hand creates a ranked model of interpretations. As mentioned before this model is shown to be the unique minimal model. To construct this: first, interpretations that contradict the classical statements are removed. Then each interpretation is put on the lowest level, then moved up if the interpretation is deemed an exceptional case based on the defeasible sentences in the knowledge base. Once no more interpretations can be moved up a level the construction is complete. Figure 4 shows an example of a ranked interpretation which is the minimal ranked model, using the Tweety example. Computing entailment from this model is then a fairly simple computation.

L_2 :		111	
L_1 :	100		101
L_0 :	000	010	110

Figure 4: A ranked interpretation for $\mathcal{P} = \{b, f, p\}$.

4 PROPOSITIONAL TYPICALITY LOGIC

In Booth et al [4] a new logic for reasoning about typicality is introduced. Propositional Typicality Logic (PTL) enriches propositional logic with a typicality operator. The operator is a unary operator that says that something is the typical case. Unlike the binary defeasible $\mid\sim$, the $\bullet$ operator can be placed anywhere in the formula. This makes PTL more expressive than defeasible reasoning. The Tweety example from before would now read as: typical birds fly. The notation for a statement $\bullet a \rightarrow b$, says the most typical a implies b. The statements $\bullet p \rightarrow q$ and $p \mid\sim q$ are equivalent. But now another example could also have $\bullet p \rightarrow \bullet q$, which makes PTL more expressive. In the original introduction to PTL slightly different notation is used, with $\bar{a}$ (a bar) representing typical a. The language of PTL is that of propositional logic but with the $\bullet$ operator included.

Though PTL is more expressive, we cannot use the same form of entailment as in the defeasible case. Booth et al. [3] show that ranked entailment is not appropriate for PTL. The paper also lists eleven different postulates that a consequence operator could satisfy. A consequence operator is a mapping which relates to a given form

of entailment. For a given entailment, its consequence operator gives the set of statements in a given knowledge base which are entailed by that knowledge base. It is shown that these postulates cannot all be satisfied at once. This impossibility result leads to two new forms of entailment to be proposed. The paper also states that rather than a negative result this should be interpreted that as an indication that a language as expressive as PTL allows more than one form of entailment. Entailment in this logic must of course be defeasible and non-monotonic, in the sense that conclusions can be retracted when new information is added.

5 ENTAILMENT FOR PTL

Two different forms of entailment are proposed for reasoning in PTL. Both are generalisations of rational closure. The first is LM-Entailment and the second is PT-Entailment. Entailment for PTL is the exact focus of the PTR project, as the project involves exploring the theory of and the implementation of an algorithm for entailment in PTL.

The advantages and disadvantages of these two forms of entailment are the different properties they satisfy. Both fulfil some but not all of the postulates proposed. LM-Entailment satisfies all of the postulates except for Strict Entailment. PT-Entailment on the other hand satisfies Strict Entailment but not the Single Model postulate and Conditional Rationality. For the definitions of these postulates and other postulates see Booth et al. [3].

The authors contend that the differences between these two forms of entailment mean that the context should determine which is appropriate. They mention the ideas of prototypical reasoning and presumptive reasoning. The distinction between these two is drawn in Lehmann [13]. However, the details are not important in the context of PTL.

The Tweety example details what we want from entailment for PTL. If we have a knowledge base $\mathcal{K} = \{\bullet b \rightarrow f, p \rightarrow b\}$ that says typical birds fly and penguins are birds. We would expect to conclude that typical penguins are typical birds, but when adding that $\bullet p \rightarrow \neg f$ (typical penguins do not fly) we would retract that conclusion. This is an example of what is required from entailment in PTL. Booth et al. [3] show that ranked entailment does not meet these requirements.

The idea of LM-Entailment is to apply the idea of rational closure construction to knowledge bases in PTL. The LM-Entailment algorithm uses the $\trianglelefteq_{\text{LM}}$ preference relation to return a ranked model of a knowledge base. Its form of entailment is defined such that a formula α is LM-entailed by $\mathcal{K}$ if and only if α is classically entailed by the ranked model returned by the algorithm.

PT-Entailment is based on a version of minimality derived from the characterisation of rational closure found in Giordano [9]. The idea of this form of entailment is to respect the presumption of typicality, from Lehmann [13]. This informally means we should assume every situation is as typical as possible.

PT-Entailment defines a new pre-order, $\trianglelefteq_{\text{PT}}$. To define this the authors Booth et al. first define a height function over a ranked interpretation. The height of a valuation in an interpretation corresponds to the number of the layer it is in, or to ∞. A lower height corresponds to a more typical valuation. Ordering two ranked interpretations, $\mathscr{R} \trianglelefteq_{\text{PT}} \mathscr{R}'$, then means that the height of every valuation in $\mathscr{R}$ is lower than the height of every valuation in $\mathscr{R}'$.

When working with the $\mid\sim$ operator, $\trianglelefteq_{\text{PT}}$ is equivalent to $\trianglelefteq_{\text{LM}}$ minimality defined for defeasible reasoning. However, using the typicality operator there is not one unique minimal model due to its expressivity. A number of minimal models can be given back with the PT pre-order notion of minimality. The definition of PT-Entailment is that a formula α is PT-entailed by $\mathcal{K}$ if and only if the PT-minimum of the models of $\mathcal{K}$ is a subset of the models of α.

The work on PTL is still ongoing with more investigation into the issues surrounding entailment to be done.

5.1 PTR Project Roles

In the PTR Project there are two components, one is investigating the different forms of reasoning and the other is implementing a reasoner for PTL. The project has been split up between the two forms of entailment, I will be investigating and implementing a reasoner for one of the PTL entailment algorithms, namely PT-Entailment. My partner, Guy Green, will be doing the same but focusing on LM-Entailment.

6 IMPLEMENTATION

The goal of the PTR project is to implement a reasoner for PTL. Specifically in my project I will be implementing the PT-Entailment form of reasoning. One possible approach to implementing the PT-Entailment algorithm is to use a SAT Solver.

SAT Solvers are algorithms that try to solve the Boolean satisfiability problem (SAT). This is determining if there is an interpretation that satisfies a given Boolean formula. SAT is an NP-Complete problem, meaning all problems in the complexity class NP are at most as difficult to solve as SAT. There are two different kinds of mainstream SAT solvers, according to A survey of SAT Solver, Gong and Zhou [11]. There are complete algorithms which search the entire solution space using backtracking, and incomplete algorithms. To use a SAT solver a Boolean formula must be converted into Conjunctive Normal Form, which means it is composed of a conjunction of disjunctions of literals. In other words, it is an AND of ORs.

Complete algorithms either find a satisfying interpretation or prove that a formula is unsatisfiable, i.e. that there is no interpretation where the formula evaluates as True. One of the most commonly used complete algorithms is the Davis-Putnam-Logemann-Loveland (DPLL) algorithm [8]. This has been used as the basis for most efficient SAT solvers. GRASP is a SAT Solver that used DPLL but also introduced the Conflict-Driven Clause Learning (CDCL) algorithm [19]. Based on these advancements a number of more efficient SAT algorithms were introduced such as MiniSAT [20], Chaff [15] and Lingeling [2] to name a few. For incomplete algorithms the local search algorithm is the most commonly used. GSAT [17] and its improvement WalkSAT [18] are local search algorithms for solving SAT.

In the implementation of the PTR project, a SAT Solver like the ones mentioned above could be used to develop a reasoner by reducing the PT-Entailment algorithm into a series of classical entailment checks. Another approach would be to use an ontology editor. One such ontology editor is Protege [16]. Either of these

approaches could be used in the project to develop a reasoner and implement PT-Entailment.

7 OTHER APPROACHES

There are also other approaches to defeasibility in logic. Casini [7] proposes an application of their own approach to rational closure in the field of Description Logics, which is an important knowledge representation formalism. Casini et al [6] introduces defeasibility into OWL Ontologies. Another approach to Typicality is presented in Giordano et al. [10] where a typicality operator **T** is extended to the description logic $\mathcal{ALC}$.

There are other fields of logic that deal with exceptions to rules in different ways besides defeasibility. For example, probabilistic logic. However, these approaches will not be dealt with in the PTR project.

8 CONCLUSIONS

To conclude, two forms of entailment for PTL have been proposed, these are LM-Entailment and PT-Entailment. Both have advantages and disadvantages from the properties they fulfil. An implementation of the two algorithms will be done in the PTR project in order to develop a reasoner. My own section of the project will focus on PT-Entailment. The theory of PTL builds upon previous work in the field of defeasible reasoning explored in this review.

The important concepts from propositional logic were discussed, including how sentences are built from propositional atoms connected by operators. An interpretation is a possible world where each atom is either true or false. A model is an interpretation that satisfies some other sentence. Reasoning in propositional logic comes down to computing classical entailment, in order to determine if a sentence follows from a knowledge base $\mathcal{K}$. The key result is that a sentence follows from $\mathcal{K}$ if every model for $\mathcal{K}$ is also a model for the sentence.

The $\bullet$ operator in PTL allows for more expressive statements that using the $|\sim$ operator. It is the expressivity of this language that allows for more than one form of entailment. Either operator is used in an extension to classical propositional logic. The defeasible nature of these operators means we can make statements about the typical case of a scenario. Entailment in either case had to be non-monotonic. To reason with the defeasible $|\sim$ conditional required the concept of rational closure, with an algorithm to determine if a statement follows from a given knowledge base. Another way of computing entailment is by using the construction of a minimal ranked model. Though there has been extensive work done in the field of defeasible reasoning, PTL is relatively new and unexplored in comparison.

Different approaches to implementing a reasoner for PTL exist, but the most likely is to use one of the SAT Solvers available. This will form the practical aspect of the PTR project.

REFERENCES

[1] Mordechai Ben-Ari. 2012. *Mathematical logic for computer science* (3 ed.). Springer Science & Business Media.

[2] Armin Biere. 2010. Lingeling, plingeling, picosat and precosat at sat race 2010. *FMV Report Series Technical Report* 10, 1 (2010).

[3] Richard Booth, Giovanni Casini, Thomas Meyer, and Ivan Varzinczak. 2015. On the Entailment Problem for a Logic of Typicality. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*. 2805–2811.

[4] Richard Booth, Thomas Meyer, and Ivan Varzinczak. 2012. PTL: A Propositional Typicality Logic. In *Proceedings of the 13th European Conference on Logics in Artificial Intelligence (JELIA) (LNCS)*, L. Fariñas del Cerro, A. Herzig, and J. Mengin (Eds.). Springer, 107–119.

[5] Richard Booth and Jeff B Paris. 1998. A Note on the Rational Closure of Knowledge Bases with Both Positive and Negative Knowledge. *Journal of Logic, Language and Information* 7, 2 (1998), 165–190.

[6] Giovanni Casini, Thomas Meyer, Kody Moodley, Uli Sattler, and Ivan Varzinczak. 2015. Introducing Defeasibility into OWL Ontologies. In *Proceedings of the 14th International Semantic Web Conference (ISWC) (LNCS)*, M. Arenas, O. Corcho, E. Simperl, M. Strohmaier, M. d'Aquin, K. Srinivas, P.T. Groth, M. Dumontier, J. Heflin, K. Thirunarayan, and S. Staab (Eds.). Springer, 409–426.

[7] Giovanni Casini and Umberto Straccia. 2010. Rational Closure for Defeasible Description Logics. In *Proceedings of the 12th European Conference on Logics in Artificial Intelligence (JELIA) (LNCS)*, T. Janhunen and I. Niemelä (Eds.). Springer-Verlag, 77–90.

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

[9] Laura Giordano, Valentina Gliozzi, Nicola Olivetti, and Gian Luca Pozzato. 2012. A minimal model semantics for nonmonotonic reasoning. In *Logics in Artificial Intelligence*. Springer, 228–241.

[10] Laura Giordano, Valentina Gliozzi, Nicola Olivetti, and Gian Luca Pozzato. 2013. A non-monotonic Description Logic for reasoning about typicality. *Artificial Intelligence* 195 (2013), 165–202.

[11] Weiwei Gong and Xu Zhou. 2017. A survey of SAT solver. *AIP Conference Proceedings* 1836, 1 (2017), 020059. https://doi.org/10.1063/1.4981999 arXiv:https://aip.scitation.org/doi/pdf/10.1063/1.4981999

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

[13] Daniel Lehmann. 1995. Another perspective on default reasoning. *Annals of Mathematics and Artificial Intelligence* 15, 1 (1995), 61–82.

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

[15] Matthew W Moskewicz, Conor F Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. 2001. Chaff: Engineering an efficient SAT solver. In *Proceedings of the 38th annual Design Automation Conference*. ACM, 530–535.

[16] Mark A Musen. 2015. The protégé project: a look back and a look forward. *AI matters* 1, 4 (2015), 4–12.

[17] Bart Selman and Henry Kautz. 1993. Domain-independent extensions to GSAT: Solving large structured satisfiability problems. In *IJCAI*, Vol. 93. Citeseer, 290–295.

[18] Bart Selman, Henry A Kautz, Bram Cohen, et al. 1993. Local search strategies for satisfiability testing. *Cliques, coloring, and satisfiability* 26 (1993), 521–532.

[19] João P. Marques Silva and Karem A. Sakallah. 1996. GRASP&Mdash;a New Search Algorithm for Satisfiability. In *Proceedings of the 1996 IEEE/ACM International Conference on Computer-aided Design (ICCAD '96)*. IEEE Computer Society, Washington, DC, USA, 220–227. http://dl.acm.org/citation.cfm?id=244522.244560

[20] Niklas Sorensson and Niklas Een. 2005. Minisat v1. 13-a sat solver with conflict-clause minimization. *SAT* 2005, 53 (2005), 1–2.