Propositional Typicality Reasoning (PTR)

A review on Literature Relating to Propositional Typicality Logic and Reasoning

Guy Green
University of Cape Town
Department of Computer Science
guygreen1995@gmail.com

ABSTRACT

Classical Propositional Logic is the basis of most other logics. While it is relatively simple compared to other Logics, it has always had the problem of being less expressive and capable of handling exceptions than other languages.

There has been a move to try and make additions to Classical Propositional Logic to attempt to make it more expressive and better at handling errors.

The suggestions discussed in this paper are the addition of Defeasibility to Propositional Logic with the defeasibility operator and Propositional Typicality Logic (PTL) which is the addition of a Typicality operator to Propositional Logic. The idea of entailment is also discussed in each of these.

There is a particular focus on PTL and reasoning ad this is what the project is on. There are two types of entailment in PTL: PT-Entailment and LM-Entailment. There is a particular focus LM-Entailment and how it deals with the impossibility result that reasoning in PTL has.

KEYWORDS

Propositional Logic, Defeasibility, Typicality, Entailment

1 INTRODUCTION

Propositional Typicality Logic (PTL) is a (recently) proposed addition to the classical propositional logic. It suggests adding a typicality operator (•). It allows for additional expressibility and exception handling. With the new proposed language, come challenges with the entailment and reasoning.

This Literature review will focus on the related topics to PTL and reasoning. The topics investigated and discussed will be Classical Propositional Logic, Defeasibility and Propositional Typicality Logic:

Classical Propositional Logic is the basis of PTL and so understanding how the language and the different concepts (for example: entailment) work in the basis language is incredibly crucial in understanding PTL and reasoning for PTL.

Defeasibility adds on the idea of exception handling to Classical Propositional Logic but is not as expressive as PTL. It is created by adding a defeasibility operator ($\mid\!\sim$). With Defeasibility added to Propositional Logic, it has been found that reasoning is well-understood and behaves well, which unfortunately is not the case when it comes to PTL.

PTL is an extension on Defeasibility. PTL is created by adding the typicality operator to Classical Propositional Logic. Defeasibility in Propositional Logic is the equivalent to having the typicality operator just on the antecedent. The difference between the defeasibility operator and the typicality operator is that one of them is a binary operator and the other is an unary operator respectively.

Two algorithms have been defined to calculate the entailment for PTL. The two algorithms are LM-entailment and PT-entailment. The project proposed, Propositional Typicality Reasoning (PTR) is to focus on implementing these two algorithms for entailment of PTL, The algorithm that will be focused on in this paper is LM-Entailment and PT-Entailment will be focused on by my project partner.

2 CLASSICAL PROPOSITIONAL LOGIC

Classical Propositional Logic [1] is a logic system that is considered to be the basis of all the other logic systems [1]. Understanding Propositional Logic is, therefore, crucial in understanding the other Logic Systems

2.1 Concept

In Propositional Logic, everything can be reduced down to sentences which are made up of expressions which are known as atomic propositions (atoms) and then operators in-between them. Whatever is on the left-hand side of the operator is called the antecedent and whatever is on the right hand side of the operator is called the consequent. These sentences make up a Knowledge Base from which reasoning can be used to determine whether or not other sentences are true. The atomic propositions are expressions which only have a value of True or False depending on an interpretation hence the name. Atoms are referred to using lower-case letters. Sentences in which all the atoms within the sentence have been assigned truth values of either true or false are called interpretations.

In Propositional logic there are many different operators, but there are five bases operators from which you can make the rest. This paper will focus on these five operators. They are negation ($\neg$), conjunction ($\wedge$), disjunction ($\vee$), implication ($\rightarrow$) and equivalence ($\leftrightarrow$). Each of these operators can be used in combination with atoms to make sentences. All of the operators are binary except for negation, which is unary.

[1]Everything in this section was taken from the Ben-Ari textbook, Chapter Two [1]

The negation operator ($\neg$) is used before an atom and just means not or the negation of what ever it is in front of. So for example, $\neg p$, just means that this sentence is only true if p is false. The negation operator has an interesting property. The interesting property is that you can put it in front of any valid sentence in the language and that will still make a valid sentence in the language.

The conjunction operator ($\wedge$) is used between two atoms and just means that the sentence is only true if both the antecedent and the consequent are true. So for example, $p \wedge q$, as a sentence will be true only if both p and q are true and if either of them are false or both of them are false, the sentence would be considered false.

The disjunction operator ($\vee$) is also used between two atoms and just means that the sentence is on only true if one of the atoms are true. So for example, $p \vee q$, as a sentence will only be true if either of p or q are true or both and the only condition that would make the sentence false is if both p and q are false.

The implication operator ($\rightarrow$) is also used between two atoms and just means that if the antecedent is true, then the consequent must be true. For example, $p \rightarrow q$, means that if p is true then q must be true. It also follows from that statement that $\neg q \rightarrow \neg p$.

The equivalence operator ($\leftrightarrow$) is also used between two atoms and just means that if the antecedent is true, then the consequent must be true and vice versa. So that means that the two atoms will have an equivalent value. For example, $p \leftrightarrow q$, means that the sentence is only true if p and q are both true or both false.

The rest of the operators that are used in Classical Propositional Logic can be made by using combinations of these five bases operators and so they are not needed to help define Propositional Logic and, therefore, are also not needed in PTL.

So for the *tweety* example (which is used in many papers as an example), the sentences would look like this

$$b \rightarrow f \tag{1}$$
$$p \rightarrow b \tag{2}$$
$$p \rightarrow \neg f \tag{3}$$

where b means birds, f means fly and p means penguins. For this example there are eight interpretations, with each atom either having a value of 1 or 0 to denote true or false respectively

Propositional Logic is a relatively simple and easy to compute logic system, but its downfalls come in terms of exception handling and expressibility. Because Propositional Logic is relatively binary and rigid, it struggles with exception handling and expressibility, which is what we need to describe and model the real world. Therefore, something needs to be introduced into propositional logic to make it more applicable to describe the real world.

2.2 Entailment

Entailment is the idea of reasoning from a knowledge base. So considering a knowledge base, K , it is a just a set of sentences, which can be considered the "rules" of the knowledge base. Testing whether or not another sentence follows from those rules is the process of entailment or reasoning. It produces an answer of either True or False. Either it entails from a knowledge base or it doesn't.

To test whether a sentence, A entails from a knowledge base, we look at the all the different ways that values can be assigned to the atom in the knowledge base or in other words we look at all the interpretations (defined in section 2.1) of the knowledge base. We consider every interpretation that makes the Knowledge Base, K, true. The set of interpretations that make a knowledge base true is called a model. If for every interpretation that makes the knowledge base, K, true, it also makes the sentence, A true, then we say the sentence is entailed from the knowledge base. The notation for entailment is $K \models A$. For an example in terms of the actual atoms and sentences: if we have the sentence, $p \rightarrow q$, we look at all the interpretations that make the knowledge base true and if in every interpretation that p is true, q is also true then it follows that the sentence entails from the sentence. And if it doesn't, the sentence does not entail from the knowledge base.

Building on this idea of whether or not a sentence entails from a knowledge base, there comes a nice and well-behaved definition for entailment for Classical Propositional Logic. A sentence, α, entails from a Knowledge Base, K, if and only if the set of models of the knowledge base, K is a subset of the the set of models of the sentence, α. The notation for this is given as $K \models \alpha$ *iff* $m(K) \subseteq m(\alpha)$. Propositional logic is a monotonic language. This means that if you add sentences to the knowledge, you don't lose any conclusions from the knowledge base itself.

Looking at entailment using the method and definition shown above for the it tweety example shown in section 2.1, first thing that is needed is to look at the interpretations of the knowledge base, K. The interpretations of the knowledge base are the set $\{111, 110, 100, 011, 001, 000, 101, 010\}$ where the first number represents the truth value for the atom Bird, the second number represents the truth value for the atom Penguin and the third number represents the truth value for Fly. Now we have to take out the interpretations that make all the statements not true for the knowledge base. The set that is left that make all the statements in the knowledge base true is $\{000, 101, 001\}$. This set is the model of that knowledge base.

For this example, two sentences are going to be checked if they entail from the knowledge base. The two sentences are $p \rightarrow f$ and $\neg p$.

For $p \rightarrow f$ (Sentence A), it has the same possible interpretations as the knowledge base. But the set of interpretations that make the sentence true is different to the model of the knowledge base. The set of interpretations that makes Sentence A true is as follows: $\{000, 111, 100, 011, 001, 101\}$. This is the model for the Sentence A. As can be seen, $m(K) \subseteq m(A)$ is true and so it can be said that $K \models A$.

For $\neg p$ (Sentence B), It also has the same possible interpretations as the knowledge base. But the set of interpretations that make the sentence true is different to both the model of the knowledge base and of Sentence A. The set of interpretations that makes Sentence B true is as follows: $\{100, 001, 000, 101\}$. This is the model for Sentence B. As can be seen, $m(K) \subseteq m(B)$ is true and so it can be said that $K \models B$.

Intuitively, this says that Penguins can fly and there are no Penguins which is not what is trying to be said or what correlates to what happens in the real world. The additions to Classical Propositional Logic are to try to help and deal with the concept of exception handling and expressibility. These additions help Propositional Logic deal with situations like Penguins not being able to fly but still being a bird.

3 DEFEASIBILITY

To try deal with the problem Classical Propositional Logic has of lack expressibility and not being able to handle exceptions, the idea of defeasibility has been suggested as an addition to Classical Propositional Logic. Adding Defeasibility to Propositional Logic helps with dealing with examples like the *tweety* example. Defeasibility was first investigated by Kraus et al [13]. It was called the KLM-Approach. With the idea of defeasibility, do come challenges with entailment and reasoning, but there have been shown to be well-understood and well-behaved algorithms to solve this. The concept of entailment was with Defeasibility and Conditionals was first investigated in Lehmann and Magidor [14].

3.1 Concept

Defeasibility in propositional logic just includes the addition of one binary operator which is called the Defeasibility Operator ($\mid\!\sim$). The idea of the Defeasibility Operator is to attempt to deal with exceptionality and, therefore, to try and make Propositional Logic more expressive.

The Defeasibility Operator ($\mid\!\sim$) is a binary operator and, therefore, is between two atoms or sentences. It just means that if the antecedent is true then, typically, the consequent is true but not always. So for example, $p \mid\!\sim q$ means that if p is true, then typically q is true but there are some exceptional cases where q is not true.

So for the *tweety* example, the sentences with Defeasibility added would look like this

$$b \mid\!\sim f \tag{4}$$

$$p \rightarrow b \tag{5}$$

$$p \mid\!\sim \neg f \tag{6}$$

Each of the atoms mean the same thing as in the previous section and there are the same number of interpretations.

This introduces the idea of exception handling, but still has its own problems in terms of expressibility. It also creates some problems with entailment and makes the algorithm you use to entail integral in how much information you can entail from it.

Before starting with understanding entailment for Propositional Logic with the Defeasibility, a new structure called Ranked Interpretations. Ranked interpretations were first investigated in Lehmann and Magidor (1992) [14]. The concept of Ranked Interpretation has been adapted slightly since then with the papers Booth and Paris (1998) [6] and Giordano et al (2012) [9]. Ranked Interpretations divides the different interpretations into different levels of typicality and ranks them. These levels are called partitions. The lower the partition is, the more typical the interpretations that are contained in the partition. From that it follows that the higher the partition is, the more exceptional the interpretations are that are contained in the partition. Partitions are generally denoted by $L_0, L_1 ..., L_{n-1}$ where n is the number of partitions.

3.2 Entailment

Entailment brings new challenges when dealing with defeasibility. The idea of entailment in Propositional logic is no longer good enough to properly reason with the idea of defeasibility. It is also important to note that with adding defeasibility to Propositional Logic comes, the property of monotonicity falls away when reasoning from a knowledge base.

The idea of a Rational Closure (Lehmann and Magidor (1992)) [14] is introduced to attempt to solve the problem of Entailment with Defeasibility. There are two methods using the rational closure to attempt to solve the entailment problem with defeasibility. The two methods are the Rational Closure algorithm and the Rational Closure Minimality Model solution.

The Rational Closure algorithm creates a ranked structure of the sentences in a knowledge base. The ranked structure is a similar concept to ranked interpretations but using sentences from the knowledge base. But instead of having the lowest partition be the most typical interpretations, the lowest partition are the most exceptional sentences in the knowledge base. And the highest partition is made up of the most typical sentences. The most typical sentences are generally the Classical Propositional Logic sentences that do not contain the defeasibility operator. The sentences that do contain the defeasibility operator are ranked based on the exceptionality of the antecedent. Once the ranked structure is completed the idea is to convert all the defeasible sentences to classical sentences. And then from that if we trying to reason whether a certain sentence is true or entails from the knowledge base, We check if the antecedent of that sentence follows from the knowledge base. If it does, then we can just use normal entailment process as Classical Propositional Logic. If not, then we "throw away" the most exceptional or the lowest partition of the ranked structure and check again if the antecedent follows. This process is repeated until either a True or False result is produced. An example of the Ranked sentence structure is shown in Figure 1. it is important to note that not much could be entailed from this particular example but if we added the sentence $b \mid\!\sim w$ (birds typically have wings). This sentence would be in the partition L_2 and we would be able to entail $p \mid\!\sim w$ from the structure.

To use the Rational Closure Model Minimality Solution, a ranked

L_∞ :	$p \rightarrow b$
L_2 :	$b \mid\sim f$
L_1 :	$p \mid\sim \neg f$

Figure 1: A ranked Sentence Structure for the *tweety* example

interpretation structure needs to be created for the knowledge base. To derive the ranked interpretation structure, the first thing that needs to be done is to look at all the interpretations of the knowledge base. From this set of interpretations, we eliminate or remove any interpretations that violate any of the Classical Propositional Logic sentences without the defeasibility set. With this reduced set, we place all the interpretations on the partition L_0. We then look at all the defeasible sentences and focus on the one in which the antecedent is the least exceptional. We then only look at the typical interpretations of the least exceptional antecedent. All of those interpretations stay on the partition, L_0, and rest of the interpretations move up a partition to L_1. The same process is the continued with regards to the next least exceptional antecedent of the defensible sentences and the most exceptional interpretations with regards the antecedent moving up a partition. This process is repeated until no interpretations are needed to move up a partition.

It has been shown that once the ranked structure of interpretations is derived, that this is the minimal model that can be created from the knowledge base.The LM-preference relation ($\trianglelefteq_{\text{LM}}$) is the notation to show the minimal model. From this ranked interpretations structure, checking whether a sentence with the defeasibility operator entails from the knowledge base is relatively simple concept. All that is needed is to check the most typical partition or lowest partition that the antecedent of the sentence appears in and if the classical sentence (replace the defeasibility operator with the implication operator) is shown to be true on that partition, it entails otherwise it doesn't. Checking classical statements, the classical statement must be true for every ranked partition. Entailment in PTL extends this idea of ranked interpretations and this algorithm

To show how to create the ranked interpretation structure for the Rational Closure Minimality Model Solution works (as this is what is being extended in for entailment in PTL), the *Tweety* Example shall be used. The first step is to look at all the interpretations of the Knowledge base. The set of all interpretations, as shown above, is the set $\{111, 110, 100, 011, 001, 000, 101, 010\}$. Following the algorithm, we first remove all the interpretations that violate the classical sentence which is $p \rightarrow b$. There are only two interpretations that violate this, which are 011 and 010. Now we have a set of six interpretations. These six interpretations are placed into the partition, L_0 for now. We now look at the most typical antecedent of all of the sentences with a defeasible operator in them. The most typical antecedent is b in this case. Only the most typical interpretations of b stay in L_0 and the rest are moved up to L_1. So what remains in the L_0 is the set $\{000, 001, 101\}$. Now we look at the next most typical antecedent, which in this case is p and we look at the most typical cases of p. The most typical cases of p remain in L_1 and the rest move up to L_2. This is our last step as there will be only one interpretation in L_2 which means it is the most typical interpretation in that partition. The structure that is created is shown in Figure 2 .

L_2 :		111	
L_1 :	100		110
L_0 :	000	001	101

Figure 2: A ranked interpretation Structure for the *tweety* example

4 PROPOSITIONAL TYPICALITY LOGIC (PTL)

Propositional Typicality Logic (PTL) is a recently suggested answer to attempt to make Propositional Logic more expressive. Propositional Typicality Logic was first investigated in Booth et al [5]. With PTL being more expressive, come new challenges with entailment. Entailment in PTL was first investigated in Booth et al [3]. PTL is a relatively new addition and so there is not much literature on it.

4.1 Concept

PTL is just the addition of the typicality operator (•) to Classical Propositional Logic. Originally the typicality operator was suggested to be a bar (for example: $\bar{a}$) [5] but was changed to the typicality operator above in later literature. The typicality operator is a similar concept to the Defeasibility operator except the typicality operator is a unary operator as opposed to the Defeasibility operator which is a binary operator. This allows typicality to be used both on the side of antecedent as well as the consequent.

The typicality operator (•) only takes one operand. It just means that typically or most of time the operand is true. For example, $\bullet p$ means that typically p is true. And as an extension, $\bullet p \rightarrow \bullet q$ means typically p implies typically q. As an example to show the similarity between Defeasibility and PTL: the sentence $\bullet p \rightarrow q$ in PTL has exactly the same meaning as the sentence $p \mid\sim q$ with Defeasibility in Classical Propositional Logic.

So for the *tweety* example, the sentences would look like this:

$$\bullet b \rightarrow f \tag{7}$$

$$p \rightarrow b \tag{8}$$

$$\bullet p \rightarrow \neg f \tag{9}$$

This doesn't seem like it has added much expressibility, but if you add a sentence like, $\bullet b \rightarrow \bullet w$ (Typical birds have typical wings), you get an idea on how expressive PTL can be.

The typicality operator being unary instead of binary makes PTL more expressive than Defeasibility in Propositional Logic. PTL is very expressive and good at exception handling. With this in mind, it is the best form of Propositional Logic (at the moment) to try to describe and model situations in the real world. PTL also still has the property of being non-monotonic

4.2 Entailment

With the expressibility of PTL, it seems natural to expect difficulty with entailment. And PTL doesn't disappoint.

Entailment for Defeasibility presents challenges when used for PTL. Booth et al [3] puts forward ten postulates that the consequence operator needs to meet simultaneously for the algorithms in Entailment for Defeasibility to produce well-behaved and proper answers. The ten postulates are inclusion, cumulativity, ampliativeness, defeasibility, Conditional Rationality, Single Model, Extends Rational Closure, Strict Entailment, Conditional Strict Entailment and Classical Entailment, and Typical Entailment. Each of these concepts are defined in Booth et al [3]. In Booth et al [4] put forward 9 postulates with most of the postulates being the same with a couple of them being changed slightly. In both papers, they show an impossibility result, showing that entailment in PTL cannot meet all the conditions simultaneously. Instead of taking this as a negative result, the authors of this paper interpreted this result as there can be multiple methods of Entailment for PTL of which each method satisfies a certain subset of these postulates [3]. Two new methods of entailment are suggested for PTL. They are LM-Entailment and PT-Entailment.

Both of the methods are extensions of the Rational Closure Model Minimality Solution with different adaptions to attempt to try to minimise the number of postulates that the entailment does not . Each of the methods have their own advantages and their own disadvantages.

PT-Entailment satisfies Strict Entailment but does not satisfy the Single Model Postulate and Conditional Rationality, while LM-Entailment satisfies the Single Model Postulate and Conditional Rationality, but does not satisfy Strict entailment.

For LM-Entailment, in Booth et al [3], they try to adapt the idea of a Rational Closure construction to deal with arbitrary knowledge bases in PTL rather than just Conditional Knowledge Bases.

For LM-Entailment, to start, you do exactly the same as you would starting the Rational Closure Model Minimality Solution. The idea is to still make a Minimal Ranked Interpretation model ($\trianglelefteq_{\mathrm{LM}}$). There is just a slight adaption to account for the properties of PTL.So you first remove all the interpretations that violate any of the classical sentences and the remaining interpretations are all put into the first partition L_0. You look at all the interpretations that violate the most typical (or best) interpretations of the knowledge base and the those interpretations all move up to the next partition (L_1). You the look at the next most typical or best interpretations of the knowledge base for that model and the interpretations that violate that model, you move up to the next partition. You continue this process until either there are no interpretations that violate the model or you reach a fixed point.

If there are no interpretations that violate the model, then you have a Minimal model and that is the final ranked structure you are left with.

The impossibility result for PTL introduces the idea that you can reach a fixed point. A fixed point is when you are moving up the all of the interpretations in the partition as they violate the most typical thing on that specific partition. The idea in LM-entailment is to just throw away those interpretations and return the remaining ranked interpretation.

Therefore, the implementation of LM-Entailment in the *tweety* example shown in the previous example would actually produce the same ranked structure as Figure 2.

For more information on how the algorithms work, refer to Booth et al [3].

The main advantage for PT-Entailment is that it allows for all of the expressibility of PTL. While for LM-Entailment, the main advantage is that it satisfies all the conditions needed except Strict Entailment.This allows Entailment for PTL similarly to Entailment for Defeasibility in Propositional Logic, but does take away from the expressibility of PTL.

5 IMPLEMENTATIONS

The focus of PTR is to create implementations of both of PT-Entailment and LM-Entailment. The separation of the project is outlined in the introduction. This section will focus on different algorithms and software that have been used to implement reasoning and entailment with similar languages to PTL.

SAT-solvers take up the majority of reasoning with Propositional Logic and some of the similar languages. When adding the notion of typicality, SAT-solvers are still considered to be a good way of implementing reasoning and entailment.

SAT-solvers were created to solve the Boolean Satisfiability Problem. The Boolean Satisfiability Problem is just to determine whether a formula or sentence is satisfiable in Propositional Logic. This has been shown to be a NP-complete problem by Cook[7].

SAT-solvers are classified into two types of algorithms. The two types are called complete and incomplete. [11] Complete SAT-solvers searches the whole solution space for a given satisfiability problem while incomplete SAT-solvers searches randomly in the assignment space of the given variable. It is important to note that Propositional Logic has been shown to be P-Space complete [18].

The most commonly known and used complete SAT-solver algorithm is called the DPLL algorithm [8]. Many advancements and adaptions have been made on the DPLL algorithm to create more efficient SAT-Solvers like CryptoMiniSat [12], MiniSat [17], Lingeling [2] and many more.Any of the more recent advancements would be able to be used for the project.

The most commonly known and used incomplete SAT-solver algorithm is the local search algorithm. There have been many advancements and adaptions on this algorithm as well. Some possible example algorithms GSAT [16] and NSAT [15]. Any of the more recent advancements would be able to be used for the project.

It is important to note that reasoning using Ontology reasoners has been tried before using similar languages [10] and this is also a possibility for our project. We would use Protege for the reasoning. Another method of reasoning that's been used on similar languages is probabilistic reasoning. We will not be using this in our project.

6 CONCLUSIONS

Propositional Typicality Logic (PTL), with the typicality operator to Classical Propositional Logic, has been shown to be more expressive and better at handling errors than both Classical Propositional Logic and Defeasibility, but brings through challenges with entailment and reasoning.

This Literature review has focused on the related topics to PTL and reasoning for PTL. The topics that have been discussed include Classical Propositional Logic, Defeasibility and Propositional Typicality Logic and reasoning in each of them.

Propositional Logic is the basis of the other languages. Entailment is the crux of reasoning in Propositional Logic and it has been shown to show nice results in terms of a definition and algorithm of entailment and the process has been shown to be well-behaved. The definition of entailment in Propositional Logic is given as $K \models \alpha$ *iff* $m(K) \subseteq m(\alpha)$.

Defeasibility as an addition to Classical Propositional Logic is more expressive and better at exception handling than Classical Propositional Logic. The process of Entailment has been shown to be well-behaved and well-understood with Defeasibility. The literature also focuses on the non-monotonic reasoning in general. For Entailment in Defeasibility there are two suggested algorithms using a Rational Closure construction. The two methods are the Rational Closure Algorithm and Rational Closure Model Minimality Solution. The Rational Model Minimality Solution is what is extended in entailment for PTL.

While PTL has been shown to be more expressive and better at exception handling than the other two languages. There are challenges with entailment. There has been shown to be an impossibility result to show that the Rational Closure for Deafeasible reasoning is not good enough for entailment. This result has been interpreted as saying that there is more than one algorithm for entailment for PTL.

There have been two suggested algorithms for Entailment in PTL. They are PT-Entailment and LM-entailment. The project proposed is focusing on the theory and implementations of these two methods, with my Project Partner focusing on PT-entailment and my focusing on LM-entailment. LM-Entailment deals with the impossibility result by disregarding some interpretations.

For implementation of these algorithms of entailment, different SAT-Solvers have been investigated with both the possibility of Complete and incomplete algorithms being used. The idea of an ontology reasoner is also a possibility for an implementation of entailment for PTL. An important result is that propositional logic has been shown to P-Space complete.

REFERENCES

[1] M. Ben-Ari. 2012. *Mathematical Logic for Computer Science* (3 ed.). Springer-Verlag.

[2] A Biere. 2014. Lingeling Essentials, A Tutorial on Design and Implementation Aspects of the the SAT Solver Lingeling. *POS@ SAT* 88 (2014).

[3] R. Booth, G. Casini, T. Meyer, and I. 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] R. Booth, G. Casini, T. Meyer, and I. Varzinczak. 2015. What Does Entailment for PTL Mean?. In *Proceedings of the 12th International Symposium on Logical Formalizations of Commonsense Reasoning*.

[5] R. Booth, T. Meyer, and I. 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.

[6] R. Booth and J.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.

[7] SA Cook. 1971. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*. ACM, 151–158.

[8] M Davis, G Logemann, and D Loveland. 1962. A machine program for theorem-proving. *Commun. ACM* 5, 7 (1962), 394–397.

[9] L. Giordano, V. Gliozzi, N. Olivetti, and G.L. Pozzato. 2012. A Minimal Model Semantics for Nonmonotonic Reasoning. 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, 228–241.

[10] L Giordano, V Gliozzi, GL Pozzato, and R Renzulli. 2017. RAT-OWL: Reasoning with rational closure in description logics of typicality. In *CEUR WORKSHOP PROCEEDINGS*, Vol. 1949. CEUR-WS, 306–320.

[11] W Gong and X Zhou. 2017. A survey of SAT solver. In *AIP Conference Proceedings*, Vol. 1836. AIP Publishing, 020059.

[12] W Kehui, W Tao, Z Xinjie, and L Huiying. 2011. Cryptominisat solver based algebraic side-channel attack on present. In *Instrumentation, Measurement, Computer, Communication and Control, 2011 First International Conference on*. IEEE, 561–565.

[13] S. Kraus, D. Lehmann, and M. Magidor. 1990. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence* 44 (1990), 167–207.

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

[15] D McAllester, B Selman, and H Kautz. 1997. Evidence for invariants in local search. In *AAAI/IAAI*. Rhode Island, USA, 321–326.

[16] B Selman, HJ Levesque, and DG et al. Mitchell. 1992. A New Method for Solving Hard Satisfiability Problems.. In *AAAI*, Vol. 92. 440–446.

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

[18] R Statman. 1979. Intuitionistic propositional logic is polynomial-space complete. *Theoretical Computer Science* 9, 1 (1979), 67–72.