SCADR Literature Review

Daniel Park
University of Cape Town
Cape Town, South Africa
prkjoo001@myuct.ac.za

ABSTRACT

Knowledge representation and reasoning is one of the critical parts of artificial intelligence. A machine has its own representation of the world with given information, and it draws conclusions by reasoning with what it has. Humans want to implement AI in a way such that it will reason similar to human, but humans are not always straight forward like machines. It is certainly not preferable to keep human reasoning in the form of classical logic. Humans make assumptions that are not always true, or only partially true. This requires non-monotonic reasoning, where defeasible reasoning comes in part. We aim to design and implement algorithms that can handle defeasible reasoning, improving the scability of these algorithms.

CCS CONCEPTS

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

KEYWORDS

artificial intelligence, knowledge representation and reasoning, defeasible reasoning, KLM approach, Rational Closure, satisfiability solving

1 INTRODUCTION

Determining whether a given propositional logic formula is satisfiable or not is known as the canonical NP-complete Boolean satisfiability (SAT) problem [3]. Currently, there are many SAT solvers available to the public that are very efficient, but focusing mainly on classical propositional logic. These SAT solvers are capable of solving many reasoning problems, but there are very few that focuses on defeasible reasoning with efficiency. This review will focus on the aspects of classical propositional logic, defeasible logic, SAT solvers, and how we can use these to implement an algorithm for defeasible reasoning.

2 PROPOSITIONAL LOGIC

Propositional logic [2] is a study of logic with reasoning about knowledge or information, representing human language into a logical statement. We aim to combine or modify the propositional statements to form ideal statements, but these statements can be more complicated. Each statements have its own truth and the truth of the combined statements relies on the base statements that were used. We can then use this to formally analyse and reason about a statement or a set of statements, or draw a conclusion that was unknown by using our analysis.

2.1 Syntax

The propositional *atoms*, also known as variables or symbols, are used to build up the language of propositional logic [9, 19]. Statements are represented by propositional atoms. Each atom is assigned a truth-value, either True or False, denoted as T for True and F for False. Meta variables, denoted with lower case Latin alphabet letters: $p, r, ...$, are used to represent statements. For an example: a person is an atom, but we denote it with p, without any loss of interpretation. Once p is denoted for person, no other atoms other than person can be denoted as p. A finite set P is a set of all propositional atoms. The statements can be combined using connectives with the truth-values of the combined statements relying on the truth-value of each individual statements. Each logical statements consist of logical operators, where each statement has its own truth-values that are assigned depending on the statement [12].

All formulas that are element of the language $\mathcal{L}$, where $\alpha, \beta \in \mathcal{L}$, and $p \in P$ are recursively defined as follows: $\alpha ::= \top \mid \bot \mid p \mid \neg\alpha \mid \alpha \wedge \alpha \mid \alpha \vee \alpha \mid \alpha \rightarrow \alpha \mid \alpha \leftrightarrow \alpha$.

There are in fact more operators than the ones mentioned above, but we will only be using these operators since the composite of them can define all the other formulas.

The negation operator takes in single operand, while the other operators take in two operands [2]. For each operators:

- $\top$ is read as Top and means the statement is always True, and $\bot$ means the statement is always False.
- $\alpha \wedge \beta$ is a conjunction of two statements where if either of the statements is false, then the composite statement is also false.
- $\alpha \vee \beta$ is an inclusive disjunction such that if either of the statements is true, then the composite statement is also true.
- $\alpha \rightarrow \beta$ Is read as if α then β, meaning that whenever α is true, β is always True. The only case where this statement is False is when α is True, but β is False. For all other cases, this statement holds.
- $\alpha \leftrightarrow \beta$ Is read as α if and only if β. This statement only holds for when α is True then β has to be True and vice versa. For all other cases, the statement does not hold.

2.2 Semantics

For a statement consisting of an individual variable p only, p is assigned True, and for a statement consisting of an individual $\neg p$, p is assigned False. A statement is known as inconsistent if it consists of complementary atoms, such as both p and $\neg p$ in the same statement. Such a statement does exist, but this just means that p cannot exist according to the statement. Negation sign alternates the truth-value of atoms where $\neg p$ assigns False to p, but $\neg\neg p$ assigns True to p.

The following is a truth table of formulas for $p, q \in P$:

p	q	$\top$	$\bot$	p	$\neg p$	$p \wedge q$	$p \vee q$	$p \rightarrow q$	$p \leftrightarrow q$
T	T	T	F	T	F	T	T	T	T
T	F	T	F	T	T	F	T	F	F
F	T	T	F	F	F	F	T	T	F
F	F	T	F	F	T	F	F	T	T

Table 1: Truth table where the rows represent the interpretations

Sometimes, $\neg p$ is also written as $\overline{p}$, and this will be used to represent the valuations. For $p, q \in P$, referring to Table 1 from above, the set of all valuations are $\{pq, p\overline{q}, \overline{p}q, \overline{p}\overline{q}\}$. For example: the set of valuations for the statement $\mathcal{K} = \{p \wedge q\}$ is $\{pq\}$ and for $\mathcal{K} = \{p \rightarrow q\}$ is $\{pq, \overline{p}q, \overline{p}\overline{q}\}$, where $\mathcal{K}$ is the knowledge base.

2.3 Entailment

An entailment is a logical consequence from a statement or a set of statements. A statement or a set of statements logically follow from a single or composition of statements when the statement that is logically followed contains every truth of the base statement. Entailment is not a formula, but a concept, and it is denoted as $\models$. For the knowledge base $\mathcal{K}$, a set of formulas, and α, a statement, $\mathcal{K} \models \alpha$ is read as $\mathcal{K}$ entails α and this is true if and only if α contains every model of $\mathcal{K}$. It is possible for α to contain models that are not in $\mathcal{K}$, but the minimum requirement for $\mathcal{K}$ to entail α is for α to contain every model of $\mathcal{K}$. For an example: $\mathcal{K} \models (p \vee q)$, where $\mathcal{K} = \{p \wedge q\}$. By referring to Table 1, the statement $p \vee q$ has the valuations $pq, p\overline{q}, \overline{p}q$ and $p \wedge q$ has the valuations pq only. By inspection, it is clear that the statement $(p \vee q)$ contains every model of $\mathcal{K}$, thus $\mathcal{K}$ does entail $(p \vee q)$. A method of checking whether a knowledge base $\mathcal{K}$ entails a statement α is to see whether $\mathcal{K} \models \neg\alpha$. This is useful since it is not efficient to check every valuations when $\mathcal{K}$ actually does entail α. In a case of $\mathcal{K} \models \neg\alpha$, we will just have to find one valuation where it satisfies $\mathcal{K} \models \neg\alpha$. If $\mathcal{K}$ does not entail $\neg\alpha$, then $\mathcal{K} \models \alpha$. $\models \alpha$ means that α is valid if and only if α is True for all interpretations. $\not\models \alpha$ means α is invalid if and only if there is at least one interpretation that falsifies α. This entailment takes a crucial role as this review goes on.

3 LIMITATIONS OF CLASSICAL PROPOSITIONAL LOGIC

In a real-world scenario, it is not always straight forward to represent human language into a classical propositional logic language. Classical propositional logic is monotonic. Monotonic means that the conclusion you draw from all the information is fixed and adding new statements does not change this conclusion. However, think of a case where we have variables $p, q, f \in P$ and atoms penguin, bird and flies, where each atom is denoted by its corresponding variables. From here we can have $\mathcal{K} = \{p \rightarrow b, b \rightarrow f, p \rightarrow \neg f\}$. Which states that penguins are birds, birds fly, and penguins cannot fly. The above three statements are reasonable in a real-world case but putting them into classical propositional logic language results in the following: penguins are birds, penguins cannot fly, and birds fly, thus such penguins cannot exist. We know that penguins are birds that cannot fly, but the composite of the three statements is telling us that such element p cannot exists for classical propositional logic. What we actually meant was that birds typically fly. We know that not all birds can fly, such as penguins and ostriches, but these are still referred to as birds. So, we add a statement, $b \rightarrow \neg f$ to inform that there are birds that cannot fly. With monotonicity, this leads to a contradiction since we already concluded that birds fly. So, by adding new information on birds cannot fly, this will lead to inconsistency. This is where we represent non-monotonic KLM-style defeasible reasoning.

3.1 KLM-style Defeasible Approach

The Kraus, Lehmann and Magidor (KLM) approach [13] introduces a defeasible operator $\mid\sim$, where it expresses typicality. For $\alpha, \beta \in \mathcal{L}$, $\alpha \mid\sim \beta$ is read as: typically, if α, then β. This means that β is concluded from α unless there is contradictory information. Using the KLM approach, we can now have an atom penguin that can exist. By replacing $\rightarrow$ from the statement $b \rightarrow f$ to $b \mid\sim f$ and $p \rightarrow \neg f$ to $p \mid\sim \neg f$, we now have the following: $\mathcal{K} = \{p \rightarrow b, b \mid\sim f, p \mid\sim \neg f\}$. The composite of these statements reads as "penguins are birds, birds typically fly and penguins typically do not fly". This can entail that penguin is a bird that cannot fly.

The equivalent conditions come in handy when simplifying composite statements or when drawing conclusions. Lehmann and Magidor have provided the extensions of the KLM properties [15]. The following defeasible implication, with knowledge base $\mathcal{K}$, has equivalent conditions:

$$(Ref)\ \mathcal{K} \mid\approx \alpha \mid\sim \alpha \qquad (LLE)\ \frac{\top \models \alpha \leftrightarrow \beta, \mathcal{K} \mid\approx \alpha \mid\sim \gamma}{\mathcal{K} \mid\approx \beta \mid\sim \gamma}$$

$$(Or)\ \frac{\mathcal{K} \mid\approx \alpha \mid\sim \gamma, \mathcal{K} \mid\approx \beta \mid\sim \gamma}{\mathcal{K} \mid\approx \alpha \vee \beta \mid\sim \gamma} \qquad (And)\ \frac{\mathcal{K} \mid\approx \alpha \mid\sim \beta, \mathcal{K} \mid\approx \alpha \mid\sim \gamma}{\mathcal{K} \mid\approx \alpha \mid\sim \beta \wedge \gamma}$$

$$(RW)\ \frac{\mathcal{K} \mid\approx \alpha \mid\sim \beta, \beta \models \gamma}{\mathcal{K} \mid\approx \alpha \mid\sim \gamma} \qquad (CM)\ \frac{\mathcal{K} \mid\approx \alpha \mid\sim \beta, \mathcal{K} \mid\approx \alpha \mid\sim \gamma}{\mathcal{K} \mid\approx \alpha \wedge \beta \mid\sim \gamma}$$

$$(RM)\ \frac{\mathcal{K} \mid\approx \alpha \mid\sim \gamma, \mathcal{K} \not\mid\approx \alpha \mid\sim \neg\beta}{\mathcal{K} \mid\approx \alpha \wedge \beta \mid\sim \gamma}$$

The defeasible entailment, denoted as $\mid\approx$, is used when a knowledge base $\mathcal{K}$ entails a defeasible statement.

3.2 Ranked Interpretation

In KLM approach, due to typicality, it is not always clear to draw a conclusion from a defeasible entailment. The semantics of KLM-style rational defeasible implications use ranked interpretation [15] for this purpose.

Definition 1. *A ranked interpretation $\mathcal{R}$ is a function from $\mathcal{U}$ to $\mathcal{N} \cup \{\infty\}$, satisfying the following convexity property: for every $i \in \mathcal{N}$,*

if there exists a $u \in \mathcal{U}$ such that $\mathcal{R}(u) = i$, then there is a $u \in \mathcal{U}$ for which $\mathcal{R}(u) = j$ for every j such that $0 \leq j < i$.

The ranked interpretation assigns a rank to every possible valuations of P, where P is the set of propositional atoms. The structure of ranking interpretation starts from rank 0 to rank $\mathcal{N}$, and an extra rank ∞. Each rank consists of at least one valuation, meaning that there cannot be an empty rank. A valuation with lower rank is more typical than the valuations that have greater ranks. The valuation at rank 0 is most preferred and the valuations at rank ∞ are impossible. The valuations at rank $\mathcal{N}$ can still occur, but it is least preferred. $\mathcal{R} \Vdash \alpha$ is read as $\mathcal{R}$ satisfies α and this holds if α is true in all ranks of $\mathcal{R}$, excluding the rank ∞. $\mathcal{R} \Vdash \alpha \mid\sim \beta$ if, in the lowest rank of where α holds, β also holds. $\mathcal{R}$ generates the ordering $\preceq_{\mathcal{R}}$ on $\mathcal{U}$ as follows: $\mathcal{R}_1 \preceq_{\mathcal{K}} \mathcal{R}_2$ if for every $u \in \mathcal{U}, \mathcal{R}_1(u) \leq \mathcal{R}_2$ [5]. By Giordano et al, there is an unique minimal interpretation with respect to $\preceq_{\mathcal{K}}$[10].

For an example, using the penguin example mentioned earlier, with $\mathcal{K} = \{p \rightarrow b, b \mid\sim f, p \mid\sim \neg f\}$ and $P = \{p, f, g\}$ [5] we can have the following graphical representation of ranked interpretation:

∞	$p\overline{bf}$	$p\bar{b}f$	
2	pbf		
1	$\bar{p}b\bar{f}$	$\bar{p}b\overline{f}$	
0	$\overline{pbf}$	$p\bar{b}\bar{f}$	$\bar{p}bf$

From the above figure, it shows that the rank of $p\overline{bf}$ and $p\bar{b}f$ is equal to ∞, meaning that these two valuations are not possible, and the valuations $\overline{pbf}, \overline{p\bar{b}f}, \bar{p}bf$ are the ones that are most preferred since they have rank 0. The valuations that are in rank 0 represents the most typical world. This does not mean that the valuations that are not in rank 0 are impossible (excluding the valuations in rank ∞), but they are less preferable as their rank increases.

3.3 Rational Closure

Rational Closure [15] and Lexicographic Closure [14] are two of the multiple formalisations of defeasible entailment. Rational closure was suggested by Lehmann and Magidor and Lexicographic closure was proposed by Lehmann. Both rational closure and lexicographic closure satisfies LM-rationality [15].

Ranked Interpretation Entailment. $\mathcal{R}_{\mathcal{K}}$ is a unique minimal ranked interpretation with respect to $\preceq_{\mathcal{K}}$ [10]. $\mathcal{K} \mid\approx \alpha \mid\sim \beta$ if $\mathcal{R}_{\mathcal{K}} \Vdash \alpha \mid\sim \beta$.

Base Rank Algorithm Entailment. The *Base Rank Algorithm* will be used for checking defeasible entailment due to the above algorithm missing the approach to find the unique minimal ranked interpretation. The ranking of all the statements in the knowledge base $\mathcal{K}$ is constructed with the following algorithm:

(1) Define $\mathcal{K}_C$ as all the classical statements in $\mathcal{K}$

(2) Define $\mathcal{K}_D$ as all the defeasible statements in $\mathcal{K}$, and convert all the defeasible statements to classical statements, where the operator $\mid\sim$ becomes $\rightarrow$

(3) $\Sigma_0 = \mathcal{K}_D$

(4) $\Sigma_i = \{\alpha \rightarrow \beta \mid \alpha \rightarrow \beta \in \mathcal{K}_C \cup \Sigma_{i-1} \models \neg\alpha\}$

(5) i starts from 1 to where $\Sigma_i = \Sigma_{i-1}$ or Σ_i is empty, and $i \in \mathbb{N}$

(6) $\mathcal{R}_\infty = \mathcal{K}_C$

(7) $\mathcal{R}_j = \Sigma_j - \Sigma_{j+1}, 0 \leq j < i, j \in \mathbb{N}$

The result of this algorithm is the ranking of $\mathcal{K}$, where all the statements are in the form of classical logic.

For example, let the set of atoms $P = \{b, f, p, w, r\}$, and the knowledge base $\mathcal{K} = \{b \mid\sim f, p \rightarrow b, p \mid\sim \neg f, r \rightarrow b, b \mid\sim w\}$. By applying the algorithm from above, we have the following:

- $\mathcal{K}_C = \{r \rightarrow b, p \rightarrow b\}$
- $\mathcal{K}_D = \{b \mid\sim f, p \mid\sim \neg f, b \mid\sim w\}$ and this get converted into $\{b \rightarrow f, p \rightarrow \neg f, b \rightarrow w\}$
- $\Sigma_0 = \{b \rightarrow f, p \rightarrow \neg f, b \rightarrow w\}$
- $\Sigma_1 = \{p \rightarrow \neg f\}$ since $\Sigma_0 \cup \mathcal{K}_C \models \neg p$, but does not entail $\neg b$
- $\Sigma_2 = \emptyset$. We stop at Σ_1 since Σ_2 is empty
- $\mathcal{R}_\infty = \{r \rightarrow b, p \rightarrow b\}$
- $\mathcal{R}_1 = \Sigma_1 \backslash \Sigma_2 = \{p \rightarrow \neg f\}$
- $\mathcal{R}_0 = \Sigma_2 \backslash \Sigma_1 = \{b \rightarrow f, b \rightarrow w\}$

The visual representation of the ranking:

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

Statements with higher ranking have more typicality than the statements that have lower ranking. For the entailment checking, check the following:

- For $\mathcal{K} \mid\approx \alpha$, if every statement in the rank ∞ entails α, then $\mathcal{K} \mid\approx \alpha$ holds.
- For $\mathcal{K} \mid\approx \alpha \mid\sim \beta$, if every statement in $\mathcal{R}$ entails $\neg\alpha$ then this means that there is a conflict. In this case remove the ranking of the highest level and repeat this step of checking whether $\mathcal{R}$ entails $\neg\alpha$. Repeat this procedure until the bottom rank is the only rank remaining. If the statements in the bottom rank entails $\neg\alpha$, then conclude that $\mathcal{K} \mid\approx \alpha \mid\sim \beta$.

The above entailment checking returns $\mathcal{K} \mid\approx \alpha \mid\sim \beta$ if it holds by the definition of ranked interpretation [7].

4 SAT SOLVER

With many formulas and variables, it can be challenging to figure out whether a certain formula, or a composite of formulas, is either satisfiable or not. This is where SAT solvers come in handy.

A SAT solver is a satisfiability solving algorithm that determines whether a given input is satisfiable or not. The inputs are a formula or a composite of formulas. For the SAT solver to produce satisfiable, it is required to have at least one interpretation that satisfies the input. If there is not a single interpretation that can satisfy the input, the SAT solver will determine the input to be unsatisfiable. These SAT solvers are known to be NP-complete (Non-deterministic Polynomial time) [6, 16, 18, 21], but the efficiency varies for different solvers.

4.1 Algorithms

The Semantic Tableaux [2] is a satisfiability algorithm for propositional logic. Currently, there are many SAT solvers publicly available, such as GRASP [16], POSIT [6], WalkSAT [17], and many of them are designed and implemented employing Davis–Putnam–Logemann–Loveland (DPLL) algorithm [1, 6, 16–18, 21] and heuristic local search [11]. The general form of the input for SAT solvers is CNF since it is simple and useful [18].

CNF stands for Conjunctive Normal Form, where it contains a clause or conjunction of clauses. Clauses are connected by logical operator $\wedge$. A clause is a disjunction of literals where each literal is an atom or its negation [20]. An example of CNF formula is: $(a \vee b) \wedge (\neg a \vee c)$. This formula consists of two clauses with four atoms. A formula $(a \rightarrow b) \vee (a \wedge c)$ is not in conjunctive normal form since it consists of logical operator $\rightarrow$. For the SAT solvers that require the formula to be in CNF, the classical propositional logic statements must first be converted to CNF formula since CNF formula does not accept logical operators other than $\vee$ and $\wedge$.

The conversion to CNF requires the following rule:
Double Negation:
1. $A \leftrightarrow \neg(\neg A)$
De Morgan's Laws:
2. $\neg(A \vee B) \leftrightarrow (\neg A) \wedge (\neg B)$
3. $\neg(A \wedge B) \leftrightarrow (\neg A) \vee (\neg B)$
Distributive Law:
4. $(A \vee (B \wedge C)) \leftrightarrow (A \vee B) \wedge (A \vee C)$

A CNF formula is unsatisfied by any interpretation that fails at least one clause or more.

4.2 Heuristic Local Search

Heuristic local search [11] techniques are not guaranteed to be complete, meaning that it is not guaranteed for the algorithm to return satisfiable if one exists or prove satisfiability [11]. For this reason, most complete SAT solvers are based on DPLL search algorithm [18].

4.3 DPLL

DPLL, sometimes just referred as Davis-Putnam (DP)[21], is a complete, backtracking-based search algorithm for checking whether a formula is satisfiable or not. An algorithm is complete when it returns every right answer for the right input, but it might return a right answer for a wrong input. The formulas for the DPLL algorithm are in conjunctive normal form (CNF).

DPLL search algorithm employs backtrack searching, where a variable and a propositional value are selected for branching purposes at each step. Two values, either 0 or 1, can be assigned to a variable at each branching step. Branching is assigning either 0 or 1 to the selected variable. After that, the logical consequences of each branching step is evaluated. Each time a conflict is found, the backtracking is executed.

Backtracking is undoing branching steps until a branch that has only been assigned a single value is reached. When both values have been assigned to the selected variable at a branching step, backtracking will undo this branching step. If both values have been considered for the first branching step, and backtracking undoes this first branching step, then the given CNF formula is considered to be unsatisfiable. This kind of backtracking is called chronological backtracking [21].

4.4 Efficiency

There are many SAT solving competitions going on around the world every year and each year participants produce algorithms that are more optimized and faster than previous years. The Kissat SAT solver won first place in the main track of the SAT competition 2020 and unsatisfiable instances [8].

Kissat SAT solver is an improved reimplementation of CaDiCaL, where CaDiCaL is a efficient version of Conflict-Driven Clause Learning (CDCL) solver [4]. CDCL SAT solvers are widely used across the world for their efficiency. The CDCL SAT solver is primary inspired by DPLL solvers [4].

5 CONCLUSION

Unlike machines, humans are sometimes quite difficult to understand, and modeling human reasoning is not straight forward. Classical reasoning is not sufficient enough to model human language, and this is where defeasible logic comes in handy. With *non-monotonicity* of defeasible logic, we can model human language better than we would have done by using classical logic. With *Rational Closure* [15], the defeasible part of the knowledge base can be represented as *classical propositional logic*. We can use this to improve the scability of defeasible entailment algorithms into SAT solvers. For the SAT solving algorithm, the *DPLL algorithm* [21] and *CDCL* [4] algorithm is used widely and CDCL is very efficient compared to other SAT solvers out there [8]. With *Rational Closure* and CDCL, this is a step forward to developing an efficient system for defeasible reasoning.

REFERENCES

[1] Roberto J Bayardo Jr and Robert Schrag. Using csp look-back techniques to solve real-world sat instances. In *Aaai/iaai*, pages 203–208. Providence, RI, 1997.

[2] Mordechai Ben-Ari. *Mathematical logic for computer science*. Springer Science & Business Media, 2012.

[3] Armin Biere, Marijn Heule, and Hans van Maaren. *Handbook of satisfiability*, volume 185. IOS press, 2009.

[4] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh. Conflict-driven clause learning sat solvers. *Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications*, pages 131–153, 2009.

[5] Giovanni Casini, Thomas Meyer, Kodylan Moodley, and Ivan Varzinczak. Towards practical defeasible reasoning for description logics. 2013.

[6] Jon William Freeman. *Improvements to propositional satisfiability search algorithms*. PhD thesis, Citeseer, 1995.

[7] Michael Freund. Preferential reasoning in the perspective of poole default logic. *Artificial Intelligence*, 98(1-2):209–235, 1998.

[8] SAT2020. 2020. SAT Competition 2020. Retrieved from https://satcompetition.github.io/2020/results.html.

[9] Jean H Gallier. *Logic for computer science: foundations of automatic theorem proving*. Courier Dover Publications, 2015.

[10] Laura Giordano, Valentina Gliozzi, Nicola Olivetti, and Gian Luca Pozzato. Semantic characterization of rational closure: From propositional logic to description logics. *Artificial Intelligence*, 226:1–33, 2015.

[11] Holger H Hoos and Thomas Stützle. Local search algorithms for sat: An empirical evaluation. *Journal of Automated Reasoning*, 24(4):421–481, 2000.

[12] Adam Kaliski. An overview of klm-style defeasible entailment. Master's thesis, Faculty of Science, 2020.

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

[14] Daniel Lehmann. Another perspective on default reasoning. *Annals of mathematics and artificial intelligence*, 15(1):61–82, 1995.

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

[16] Joao P Marques-Silva and Karem A Sakallah. Grasp: A search algorithm for propositional satisfiability. *IEEE Transactions on Computers*, 48(5):506–521, 1999.

[17] David McAllester, Bart Selman, and Henry Kautz. Evidence for invariants in local search. In *AAAI/IAAI*, pages 321–326. Rhode Island, USA, 1997.

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

[19] Stuart Russell and Peter Norvig. Artificial intelligence: a modern approach. 2002.

[20] Frank Van Harmelen, Vladimir Lifschitz, and Bruce Porter. *Handbook of knowledge representation*. Elsevier, 2008.

[21] Hantao Zhang and Mark Stickel. Implementing the davis–putnam method. *Journal of Automated Reasoning*, 24(1):277–296, 2000.