SCADRV2 Literature Review

Dhiresh Thakor Vallabh
University of Cape Town
Cape Town, South Africa
THKDHI001@myuct.ac.za

ABSTRACT

Knowledge representation and reasoning is a concept in artificial intelligence (AI). This concept involves storing some information about the world or a topic into a machine and it creates conclusions based on the knowledge given. Defeasible reasoning is a non-classical form of reasoning that expands upon the classical form by being able to reason about new and/or conflicting information. This is done by giving information that is non-monotonic and accepting information that is typically true. This project aims to improve upon the implementation of defeasible entailment algorithms and improve the scalability to handle larger knowledge frameworks.

CCS CONCEPTS

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

KEYWORDS

artificial intelligence, knowledge representation and reasoning, defeasible reasoning, Boolean satisfiability solving

1 INTRODUCTION

Knowledge representation has been an important subject in artificial intelligence (AI) as it has many practical capabilities [8]. Knowledge representation uses symbols and variables to represent information about the world and attempts to reason about this information to create conclusions based on the information it has [11]. By using different algorithms and techniques to make reasoning about information faster, we can create more reliable and efficient machines.

This literature review will begin by explaining the syntax and semantics of propositional logic. We will then discuss the motivation behind defeasible reasoning, the preferential approach, and the KLM approach. Following this, we will discuss rational closure and its algorithm. We will also describe the Boolean satisfiability problem. We then conclude by discussing satisfiability (SAT) solvers, their definitions, notation, algorithms for computation, and how it can be used for defeasible entailment.

2 PROPOSITIONAL LOGIC

Propositional logic is a formal reasoning framework that converts information represented in a spoken language into logical statements [7]. These statements are created by simplifying complicated pieces of information into propositional atoms. These atoms can then be assigned truth values (true or false value) which create a logical statement about the world. These statements can then be combined, using binary connectives, with their truth values to form new conclusions. This framework, defined by [1], uses specific syntax and semantics which will now be introduced.

2.1 Syntax

The language $\mathcal{L}$ of propositional logic is made up of propositional *atoms* $\mathcal{P}$. Each propositional atom is assigned a truth value which is denoted as either *True* (T) or *False* (F). Atoms are denoted using Greek symbols (e.g., α, β, etc.) or abbreviated versions of a noun (e.g., *animal*, a, *carnivores*, c, etc.). Propositional formulas are then created by combining propositional atoms with Boolean operators known as *connectives*. The classical form has many connectives but this project will focus on the connectives listed below:

Symbol	Name
$\neg$	Negation
$\wedge$	Conjunction
$\vee$	Disjunction
$\rightarrow$	Implication
$\leftrightarrow$	Equivalence

Table 1: Connectives and their Symbol Representations

The negation connective requires a single operand (*unary*) whilst the remaining connectives require two operands (*binary*). Propositional formulas can now be defined as the set of $p \in \mathcal{P}$ and $\alpha, \beta \in \mathcal{L}$, $\{p,\ \neg\alpha,\ \alpha \wedge \beta,\ \alpha \vee \beta,\ \alpha \rightarrow \beta, \alpha \leftrightarrow \beta\}$ that are defined in $\mathcal{L}$. There are also two constants included in $\mathcal{L}$ known as Top ($\top$) and bottom ($\bot$). These constants represent always true and always false, respectively.

2.2 Semantics

2.2.1 Formulas. Each atom can be associated with a truth value and different combinations can derive different truth values based on the connectives. These variations can be mapped out by creating a *truth table*.

A truth table can be formally defined as the set of all evaluations $\mathcal{V}$ where a single evaluation E is denoted as $E : P \mapsto \{T, F\}$. If $E = p$ then p is true and $\bar{p}$ is false. If $E = p$ then E *satisfies* p. *Satisfaction* can be applied to more complex formulas since connectives return a singular truth value.

A truth table can also include the complimentary atoms of p and $\neg p$ in the same statement but p can never exist due to the contradiction. The table below shows how variables and connectives interact to generate new truth values based on existing Boolean atoms:

An evaluation E that can satisfy some formula $\alpha \in \mathcal{L}$ is known as a *model* of that formula, with the set of all models of α denoted as γ. α is considered satisfiable when γ has at least one model. α is considered unsatisfiable when γ is empty. This represents a

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

Table 2: Truth Table of Variables p and q

contradiction (e.g., $p \wedge \neg p$ as mentioned before). If $\gamma = V$, then the $\top$ constant applies for every evaluation (e.g., $p \vee \neg p$).

2.2.2 Knowledge Bases. A *knowledge base* is a finite set of propositional formulas. A knowledge base $\mathcal{K}$ is satisfied by an evaluation E iff for every $\beta \in \mathcal{K}$, E satisfies β. The set of models of $\mathcal{K}$ is the set of all models of $E \in V$ that satisfy $\mathcal{K}$.

2.2.3 Entailment. Entailment is when some formula α is the logical consequence of a knowledge base $\mathcal{K}$. α is considered a logical consequence when the set of models in $\mathcal{K}$ is satisfied by α. The notation for entailment is $\mathcal{K} \models \alpha$ ($\mathcal{K}$ entails α). Later sections will further explain entailment in defeasible reasoning and satisfiability.

3 DEFEASIBLE REASONING

3.1 Motivation

In reality, the human mind is more complicated and the classical forms of reasoning, such as propositional logic, can only depict a portion of human reasoning. Moodley [13] explains that classical reasoning cannot accept exceptions. This is because all logical statements and previous conclusions in the knowledge base $\mathcal{K}$ are infallible. This is done so that $\mathcal{K}$ is able to work with incomplete information. This concept is known as monotonicity. This means that any new information must be reconciled with existing information. This reconciliation may work for some statements but can easily lead to contradictions and therefore incorrect conclusions.

For example, the statements *"students do not pay taxes"*, *"employed people pay taxes"*, *"person is a student"*, *"person is employed"* all represent infallible logical statements that can be added to $\mathcal{K}$. However, the only way for $\mathcal{K}$ to reason about this information is to conclude that there are no employed students even if there may be employed students. $\mathcal{K}$ will represent this by having no models. This is because $\mathcal{K}$ entails that the person pays taxes and does not pay taxes which leads to a contradiction [13]. This problem makes knowledge bases unreliable and unlike a human's mind which can reconcile these contradictions.

The solution is to use systems that are nonmonotonic. There many nonmonotonic systems in existence that have been explained by Kaliski [7]. These systems include Default Logic, Circumscription, Belief Revision, and more. We will focus on the Preferential Approach which is defined by the KLM framework. This is because the Preferential Approach has more generality and has a more clearly defined structure than the other approaches.

3.2 The Preferential Approach

As mentioned in section 3.1, *preferential approach*, and by extension the KLM framework, is better than other non-classical systems. This is because the framework defines certain properties of defeasible entailment that both solve the problem of classical reasoning and creates an ordered structure that is easier to follow.

3.2.1 Consequence Relations. The KLM framework begins by defining new type of relation between formulas known as *Consequence relations* [7]. Consequence relations create the ability to handle exceptions by describing a relation as "typically true". This is denoted by $\mid\sim$. For example, people $\mid\sim$ pay taxes means *"people typically pay taxes"*. Since this is between logical formulas and not atoms, the $\mid\sim$ does not represent a new connective. The $\mid\sim$ only represents the relationship between the formulas.

3.2.2 Preferential Interpretations. The next property that is defined is that of *preferential interpretations*. This property states that if there are two evaluations E and V and E is more *typical* than V, then E will be preferred more than V.

There are *preferential semantics* when it comes to creating these preferential interpretations. Each evaluation is mapped onto a *state/s*. These states can be infinite depending on the interpretations. These states are then partially ordered based on preferences. A state $s \in S$, where S is the set of states, is said to be *minimal* there is no $s' \in S$ where $s' \prec s$ ($\prec$ is used to denote preceding).

A preferential interpretation, $\mathcal{P}$, is well-founded iff the states and partial order is well-founded. The states and order are only well-founded iff there is a minimal state in some subset of S. $\mathcal{P}$ is also considered a well-founded and finite interpretation or *"smooth"* iff for any formula $\alpha \in \mathcal{L}$, $[[\alpha]]^{\mathcal{P}}$ has a minimal state s. $[[\alpha]]^{\mathcal{P}}$ is the set of all states in a given preferential interpretation such that each evaluation in a state/s satisfies α. Smoothness and well-foundedness ensures that there is no infinite sets of states.

Finally, $\mathcal{P}$ is sound when there is a preferential consequence relation (or defeasible implication) ($\mid\sim_{\mathcal{P}}$). This relation is satisfied iff for any $s \in [[\alpha]]^{\mathcal{P}}$ and $\alpha, \beta \in \mathcal{L}$, $s \Vdash \beta$. Soundness essentially removes scenarios that are very unlikely to happen which gives more preference to more typical scenarios [7].

3.2.3 Ranked Interpretations. Ranked interpretations [7] are formally defined as a function $\mathcal{R} : \mathcal{V} \mapsto \mathcal{N} \cup \{\infty\}$ such that if there is some $I, J \in \mathcal{V}$ then for some $\mathcal{R}(I) = i$ there will always be some $\mathcal{R}(J) = j$ where $0 < j < i$.

Ranked interpretations are based upon preferential interpretations but make certain changes. The biggest difference is that evaluations can no longer belong to many states. This is to reduce the number of duplicated states. The second difference is that any states are ordered into a ranked system based on how typical a state is. If any states have the same rank, they are either as preferred as one another or are incomparable from one another. When there are duplicate states in the rankings, the most minimal state is kept and the other states are removed. If the duplicate states are the same rank, then all but any one state is removed from the rank.

By using the soundness rule of preferential interpretations, if $\mathcal{R}$ satisfies some preferential consequence relation then we can describe $\mathcal{R}$ as a *ranked model* of the relation.

3.2.4 Ranked Entailment. Ranked entailment [7] is similar to propositional entailment in that with a knowledge base $\mathcal{K}$, and a defeasible implication $\mathcal{K} \mid\approx_{\mathcal{R}} \alpha \mid\sim \beta$ iff for every ranked interpretation, $\mathcal{R}$, such that $\mathcal{R} \Vdash \mathcal{K}$, then $\mathcal{R} \Vdash \alpha \mid\sim \beta$. This form of entailment is

still considered monotonic and therefore *defeasible entailment* is needed.

3.3 KLM Properties for Defeasible Entailment

Lehmann and Magidor [10] devised a set of properties that needed to be followed to achieve any form of defeasible entailment. When all of the properties are met, the defeasible entailment method is described as *LM-rational*. These properties were created to isolate as many sensible entailment relations as possible. These properties for a knowledge base $\mathcal{K}$ and propositional formulas α, β, γ are defined as follows:

(Ref) $\mathcal{K} \mid\!\approx \alpha \mid\!\sim \alpha$ And $\frac{\mathcal{K} \mid\!\approx \alpha \mid\!\sim \beta, \mathcal{K} \mid\!\approx \alpha \mid\!\sim \gamma}{\mathcal{K} \mid\!\approx \alpha \mid\!\sim \beta \wedge \gamma}$

(LLE) $\frac{\mathcal{K} \mid\!\approx \alpha \leftrightarrow \beta, \mathcal{K} \mid\!\approx \alpha \mid\!\sim \gamma}{\mathcal{K} \mid\!\approx \beta \mid\!\sim \gamma}$ Or $\frac{\mathcal{K} \mid\!\approx \alpha \mid\!\sim \gamma, \mathcal{K} \mid\!\approx \beta \mid\!\sim \gamma}{\mathcal{K} \mid\!\approx \alpha \vee \beta \mid\!\sim \gamma}$

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

3.4 Rational Closure

Lexicographic closure was proposed by Lehmann [9] and *Rational closure* was proposed by Lehmann and Magidor [5] as methods for defeasible entailment. Both methods are considered LM-rational. To fully explain rational closure, there are certain principles and methods to understand.

3.4.1 Minimal Ranked Entailment. *Minimal ranked entailment* begins creating a partial order of ranked interpretations for a knowledge base $\mathcal{K}$ and ranks the interpretations by how likely the interpretation is. The higher the rank (lower the number) means that the interpretation is more likely and vice versa. Giordano et al. [5] noted that eventually there is a *minimal element*. This minimal element that satisfies $\mathcal{K}$ will be used to create a an entailment relation between the knowledge base and any defeasible implication that is satisfied by the minimal element.

3.4.2 Materialisation. *Materialisation* [2] converts a defeasible implication $\alpha \mid\!\sim \beta$ into a material implication $\alpha \rightarrow \beta$. This material implication is also known as the *material counterpart* of the defeasible implication [7]. The materialisation of a knowledge base $\overrightarrow{\mathcal{K}}$ is the set of all material counterparts of defeasible implications within a knowledge base $\mathcal{K}$.

3.4.3 Base Rank Algorithm. The rational closure method uses the *base rank algorithm* [2] to determine if a formula is in the rational closure of a knowledge base $\mathcal{K}$. The layout of the algorithm is shown by the following:

(1) Separate the classical statements $\mathcal{K}_C$ from the defeasible statements $\mathcal{K}_\mathcal{D}$.

(2) Materialise $\mathcal{K}_\mathcal{D}$ into $\overrightarrow{\mathcal{K}}_\mathcal{D}$.

(3) Create a rank $\mathcal{R}^\mathcal{K}_\infty$ that holds all of $\mathcal{K}_C$.

(4) Create a set of *exceptional* subsets $\mathcal{E}^K_0, \mathcal{E}^K_1, \mathcal{E}^K_2, \ldots, \mathcal{E}^K_{n-1}$ where $\mathcal{E}^K_0 = \overrightarrow{\mathcal{K}}_\mathcal{D}$.

(5) Let $\mathcal{E}^\mathcal{K}_i = \{\alpha \rightarrow \beta \mid (\alpha \rightarrow \beta) \in \mathcal{K}_C \cup \mathcal{E}^\mathcal{K}_{i-1} \models \neg\alpha\}$ where $i > 0$. This will separate the statements such that $\mathcal{E}^\mathcal{K}_i$ only holds statements in $\mathcal{E}^\mathcal{K}_{i-1}$ such that α is false.

(6) Let the ranks $\mathcal{R}^\mathcal{K}_i = \mathcal{E}^\mathcal{K}_{i-1} \setminus \mathcal{E}^\mathcal{K}_i$.

(7) Stop when $\mathcal{E}^\mathcal{K}_i = \mathcal{E}^\mathcal{K}_{i-1}$ or when $\mathcal{E}^\mathcal{K}_{i-1} = \emptyset$.

This will create ranks where the more likely statements will be in higher ranks. This will also make all ranks hold only classical statements.

3.4.4 Defeasible Entailment Using Rational Closure. By using rational closure for some $\mathcal{K}$, it is possible to determine if $\mathcal{K}$ entails some defeasible implication $\alpha \mid\!\sim \beta$.

(1) Determine if $\mathcal{R}^\mathcal{K}_\infty$ entails $\neg\alpha$.

(2) Determine if each rank in $\mathcal{R}^\mathcal{K}_n$ entails $\neg\alpha$ starting from $i = 0$ where $i \in n$. If $\mathcal{R}^\mathcal{K}_i$ does entail $\neg\alpha$ then the rank set is removed and move on to $\mathcal{R}^\mathcal{K}_{i+1}$ (i.e. a lower rank). This is done until there is a statement in a rank that does not entail $\neg\alpha$. The current statement, rank, statements in the current rank, and the remaining statements and ranks are kept.

(3) If there are no remaining ranks then $\mathcal{K}$ does not entail $\alpha \mid\!\sim \beta$.

(4) If there is at least one rank then determine if the remaining statements and ranks entail $\alpha \rightarrow \beta$.

The output for this will be true if $\mathcal{K} \mid\!\approx \alpha \mid\!\sim \beta$ and false otherwise [7]. To demonstrate these algorithms and KLM defeasible entailment, the classic birds and penguins example will be given. Consider the knowledge base where $\mathcal{K} = \{b \mid\!\sim f, p \rightarrow b, p \mid\!\sim \neg f, b \mid\!\sim w\}$ where b = bird, p = penguin, f = flies, and w = wings [2]. Also consider the defeasible query $p \mid\!\sim \neg f$.

(1) The base rank algorithm will make $\mathcal{K}_C = \{p \rightarrow b\}$ and $\overrightarrow{\mathcal{K}}_\mathcal{D} = \{b \rightarrow f, p \rightarrow \neg f, b \rightarrow w\}$.

(2) In $\overrightarrow{\mathcal{K}}_\mathcal{D}$, p is the only variable that is less typical. Therefore it will have its own rank.

(3) $\mathcal{K}_C$ will form $\mathcal{R}^\mathcal{K}_0$.

The algorithm will create the following result:

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

Table 3: Ranks Using Base Rank Algorithm

The rational closure algorithm will then check if the query is entailed by the knowledge base. The algorithm will check if $\mathcal{R}^\mathcal{K}_\infty \mid\!\approx \neg p$. This is not true and the algorithm will check the other ranks. We then check if $\mathcal{R}^\mathcal{K}_0 \mid\!\approx \neg p$. This is true and $\mathcal{R}^\mathcal{K}_0$ is removed from the ranks. The new ranks will look like the following:

∞	$p \rightarrow b$
1	$p \rightarrow \neg f$

Table 4: Ranks After Removing $\mathcal{R}_0^{\mathcal{K}}$

We then check if $\mathcal{R}_1^{\mathcal{K}} \mid\approx \neg p$. This is not true and so we check if $\mathcal{R}_1^{\mathcal{K}} \mid\approx p \rightarrow \neg f$. This will return true and so $\mathcal{K} \mid\approx p \rightarrow \neg f$.

4 SAT SOLVERS

4.1 The Boolean Satisfiability Problem

The Boolean Satisfiability Problem (*SAT*) involves determining whether there exists some formula has a satisfying truth value assignment [12].

This problem is considered a canonically NP-complete problem and there have been many algorithms created to improve efficiency [12]. SAT has been researched as it has many use cases for subjects such as AI planning, software verification, and, of course, knowledge representation and reasoning. The algorithms that have been created are known as *SAT solvers*. These SAT solvers have seen actual success and often produce satisfying assignments, if they exist, depending on the algorithm.

4.2 SAT Solver Definitions and Notation

Conjunctive normal form (*CNF*) describes the format that propositional formulas are converted to for SAT solvers. This format is used because it eases the work that SAT solvers must perform and in no way limits the SAT solver at all. This format is formed by combining *clauses* by using the $\wedge$ (and) connective. Clauses are formed by combining literals by using the $\vee$ (or) connective. These literals can either be the variable itself or the negation ($\neg$) of a variable. These literals are then assigned truth values to determine satisfiability [6]. An example of a formula in CNF format is (a $\vee$ b) $\wedge$ (c $\vee$ d) $\wedge$ ($\neg$ b $\vee$ c).

Gomes et al. [6] outlines two methods for SAT solver algorithms. The first method is a *complete* algorithm. Complete algorithms determine if there is a satisfying truth value assignment or if there is no assignment at all. This method has been researched for decades but the solution is still relatively slow. The other method is an *incomplete* algorithm. Incomplete algorithms can neither always return a satisfying truth value assignment nor can it prove that a formula has no solution at all. Many incomplete algorithms are based on *stochastic local search* and are generally much faster than complete algorithms.

4.3 Semantic Tableaux Algorithm for Propositional Logic

SAT solvers can be used in propositional logic however there can be very complex formulas that have hundreds of literals. Therefore there needs to be an algorithm that can efficiently search through these formulas for satisfiability.

This is where the *semantic tableaux* algorithm becomes useful. Ben-Ari [1] explains that this algorithm decomposes formulas into sets of literals. It then checks if any of the sets include both an atom and its negation. If the set does include this pair, then the set is unsatisfiable. The formula itself only requires that at least one set is satisfiable for the formula to be satisfiable.

The decomposition of the formula is done by creating a tree structure where the formula is the parent and the possible outcomes are the children. The tree ends when it cannot be decomposed anymore. Other methods outside of the tableaux method may uses matrices to depict a similar representation [14].

4.4 The DPLL Algorithm

The (*DPLL*) algorithm, sometimes referred to as the Davis-Putnam method [15], is a complete, backtracking algorithm that was created by Davis, Logemann, and Loveland [3] and is based off previous work by Davis and Putnam [4]. DPLL was one of the first algorithms created for SAT solving and has sparked new improvements on different SAT solving procedures such as conflict-drive learning, backjumping, etc [12]. However, the work on DPLL is still the basis of most modern complete SAT solvers due to its own improvements and its foundational research.

The original algorithm created by Davis and Putnam often resulted in high memory usage for larger formulae [16]. This was because they were using a type of elimination process [12]. This was later refined into a backtracking search process. This process involves taking a CNF formula as an input and breaking it down into the sets of clauses and literals. An unassigned literal will then be selected and assigned a temporary truth value. This creates a *branch* and the algorithm will then use this branch to see if there is any combination of values that will make the clause satisfiable. If the branch is not satisfiable, then the algorithm will remove the branch (i.e. *backtrack*) and create a new one based on a different literal. If the algorithm is not satisfied by any of the branches, then it will continue backtracking branching until it cannot anymore. In which case, the algorithm will return that the formula is a unsatisfiable.

4.5 SAT Solvers and Entailment

By using SAT solvers on defeasible reasoning algorithms, we can check if a statement α in propositional logic is entailed by a knowledge base $\mathcal{K}$ through satisfiability. This is done to reduce the complexity of solving entailment to solving satisfiability.

Given a formula β, the SAT solver will do this by checking if the models of $\mathcal{K}$ satisfy $\mathcal{K} \cup \{\neg\beta\}$. If the SAT solver is satisfied, then there is an evaluation which is a model of $\mathcal{K}$ that evaluates $\neg\beta = T$. Therefore $\mathcal{K} \not\models \beta$. If the SAT solver is unsatisfied, then there are no models that evaluates $\neg\beta = T$. This means that $\mathcal{K} \models \beta$.

To demonstrate the usefulness of this, we will use the previous tax example. Lets consider a knowledge base $\mathcal{K}$ that states *"students do not pay taxes, "employed people pay taxes", "person is a student", "person is employed"* which is shortened to $\mathcal{K} = \{s \rightarrow \neg t, e \rightarrow t, p \rightarrow s, p \rightarrow e\}$ and we queried the statement *People do not pay taxes* ($p \rightarrow \neg t$). The SAT solver will check if $\mathcal{K} \cup \{\neg(p \rightarrow \neg t)\}$ is satisfiable. Therefore the SAT solver will be satisfied and $\mathcal{K} \not\models p \rightarrow \neg t$.

5 CONCLUSIONS

In this review, we have seen that classical propositional logic sets a foundation for the syntax and semantics. However, it has also shown

that it is limited due to to its inability to handle contradictions or create exceptional cases due to its monotonicity.

Defeasible reasoning has been shown to solve this problem by using preferential logic principles. The KLM framework also forms a foundation for different methods of solving the monotonicity problem. We can also conclude that defeasible reasoning can offer a structured format for defeasible reasoning by outlining the rational closure algorithm.

Finally, we have seen the importance of the SAT problem, the inner workings of SAT solvers, and how SAT solvers are vital to reducing the complexity of entailment through satisfiability.

REFERENCES

[1] Mordechai Ben-Ari. 2012. *Propositional Logic: Formulas, Models, Tableaux.* Springer London, London, 1, 7–47.

[2] Giovanni Casini, Thomas Meyer, and Ivan Varzinczak. 2019. Taking defeasible entailment beyond rational closure. In *European Conference on Logics in Artificial Intelligence*. Springer, 182–197.

[3] Martin Davis, George Logemann, and Donald Loveland. 1962. A machine program for theorem-proving. *Commun. ACM* 5, 7 (1962), 394–397.

[4] Martin Davis and Hilary Putnam. 1960. A computing procedure for quantification theory. *Journal of the ACM (JACM)* 7, 3 (1960), 201–215.

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

[6] Carla P Gomes, Henry Kautz, Ashish Sabharwal, and Bart Selman. 2008. Satisfiability solvers. *Foundations of Artificial Intelligence* 3 (2008), 89–134.

[7] Adam Kaliski. 2020. *An Overview of KLM-Style Defeasible Entailment.* Master's thesis. Faculty of Science, University of Cape Town, Rondebosch, Cape Town, 7700.

[8] Gerhard Lakemeyer and Bernhard Nebel. 1994. Foundations of Knowledge representation and Reasoning. *Foundations of knowledge representation and reasoning* (1994), 1–12.

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

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

[11] Hector J Levesque. 1986. Knowledge representation and reasoning. *Annual review of computer science* 1, 1 (1986), 255–287.

[12] Filip Marić and Predrag Janičić. 2010. Formal correctness proof for DPLL procedure. *Informatica* 21, 1 (2010), 57–78.

[13] Kody Moodley. 2016. *Practical Reasoning for Defeasible Description Logics.* Ph.D. Dissertation. School of Mathematics, Statistics and Computer Science, University of KwaZulu-Natal, Durban, South Africa.

[14] Andrei Voronkov. 1998. Herbrand's theorem, automated reasoning and semantic tableaux. In *Proceedings. Thirteenth Annual IEEE Symposium on Logic in Computer Science (Cat. No. 98CB36226)*. IEEE, 252–263.

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

[16] Lintao Zhang and Sharad Malik. 2002. The quest for efficient boolean satisfiability solvers. In *International conference on computer aided verification*. Springer, 17–36.