Scalable Defeasible Reasoning Literature Review

Aidan Bailey
University of Cape Town
Cape Town, South Africa
BLYAID001@myuct.ac.za

ABSTRACT

Knowledge representation and reasoning is a subset of artificial intelligence wherein varieties of basic logical principles harnessed by humans are formalised. This is usually done through collating information into a knowledge base with which a system can infer conclusions. While it's possible to represent many logical conclusions successfully within the classical variant of this field, there exists a critical flaw in that of its monotonicity. That is, as more information is collected, it becomes increasingly likely a contradiction is introduced. Classical reasoning algorithms do not have the scope to effectively deal with contradictions and as such, fail horribly when presented with them. In order to account for this, nonmonotonic reasoning is currently a prolific area of study, wherein logical systems are constructed such that information can be retracted from the knowledge base in the case of a contradiction, detailing a rational way of reasoning around it. This provides a style of logical reasoning much more akin to that of humans as we are constantly evaluating exceptions to otherwise rational conclusions. In the project, we plan to implement a variety of algorithms derived from one such nonmonontic framework, as well as implement a knowledge base generation tool with which testing data can be generated to assist with the algorithms' development and prove their correctness.

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

The problem of logical reasoning is at its core one of philosophy and work has been done within the field in order to formulize it [22]. This has lead to the understanding that the notion of logical entailment can be effectively reduced to the problem of boolean satisfiability. Luckily, this is an NP-complete problem and a well studied section of computation [34]. Propositional logic is a type of classical reasoning which defines syntax and semantics with respect to boolean satisfiability, offering succinct and intuitive ways of representing logical propositions, e.g, *"all birds fly"* can be written as $bird \rightarrow flies$. The simplicity of proposositional logic has led to it being the basis of many other variants of logical reasoning [2]. While propositional logic is monotonic, it encapsulates the qualities of decidability and expressiveness making it a good candidate on which nonmonotonic theory can be built upon [19].

With propositional logic in hand, a framework for defeasible reasoning that uses propositional logic at it's core will be investigated, first conceived by Kraus, Lehmann and Magnidor (KLM) in their 1998 paper entitled *Nonmonotonic Reasoning, Preferential Models and Cumulative Logics* [19]. For the project, two defeasible entailment algorithms based on the KLM approach are to be implemented, along with a tool for generating testing data. The first of these algorithms, *rational closure*, was defined by Lehmann and Magnidor in their 1992 paper "What does a conditional knowledge base entail?" [21]; and the second, *lexicographic closure*, was published by Lehmann in the 1995 paper "Another perspective on Default Reasoning" [20]. Only rational closure will be investigated in this review as it represents a good example of KLM-style defeasible entailment.

After this work has been investigated, the boolean satisifiability problem (SAT) will be briefly outlined as well as the basics of a tools used to solve SAT (sat-solvers).

2 PROPOSITIONAL LOGIC

2.1 Motivation

A proposition can be defined as a claim made with regards to to the world inhabited (e.g., 'one plus one equals two', 'Earth is farther from the sun than Venus') [2]. Intuitively, a proposition can be resolved to a boolean value based on its accuracy.

This encaptulates the essence of propositional logic, wherein English propositions are abstracted to symbolic formulas and logical entailment is reduced to boolean satisfiability [22]. Propositional logic constructs these symbolic representations (formulas) through combining propositional atoms using binary connectives. Reasoning procedures are then defined with respect to these formulas.

Propositional logic can also be viewed as an alternative nomenclature for the philosophical field of truth-function logic (when under the assumption that all sentences are propositions) [22]. It follows then that the purpose of propositional logic is to represent the fundamental way in which humans can arrive at logical conclusions. Owing to this, proposositional logic is used as a foundation to many other variants of logical reasoning. The semantics and syntax of propositional logic will now be outlined formally.

2.2 Syntax

The *propositional language* $\mathcal{L}$ generated by a finite set of propositional *atoms* $\mathcal{P}$ describes a set of propositional *formulas*. A propositional formula is defined inductively as either a single atom or a set of formulas combined using propositional *connectives*.

An atom encodes a propositional statement and is represented as either a lower case greek letter (e.g., $\alpha, \beta, \gamma \ldots$) or an abbreviation of the statement it represents (e.g, $bird$, b, $flies$, f, $Sparrow \ldots$).

Each atom in $\mathcal{P}$ has an associated *truth value* which will be denoted using T for *true* and F for *false*.

Each propositional connective symbolizes a different logical relation (boolean operator). There are 16 different connectives in propositional logic. We will however restrict this to the classical propositional connectives [19]. These connectives are defined in the table below:

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

So the propositional language $\mathcal{L}$ generated by a finite set of atoms $\mathcal{P}$ can be defined inductively as the set of formulas where for some $p \in \mathcal{P}$ and $\alpha, \beta \in \mathcal{L}$, $\{p,\ \neg\alpha,\ \alpha \wedge \beta,\ \alpha \vee \beta,\ \alpha \rightarrow \beta, \alpha \leftrightarrow \beta\}$ are also formulas in $\mathcal{L}$, along with two additional symbols, top $\top$ - a tautology (statement that is always true) - and bottom $\bot$ - a contradiction (a statement that is always false) [18]. Thus we are now able to desribe the syntax of propositional but we have yet to explore the process of logical entailment using these formulas. This will be investigated through the semantics of propositional logic as summarized by Kaliski [18].

2.3 Semantics

A formula can be evaluated using a mapping, known as a *valuation*, that associates each atom with a boolean truth value. A valuation u is formally defined as a function such that $u : P \mapsto \{T, F\}$. We say the valutation $u = \{p\bar{q}\}$ assigns atom p a truth value of true and atom q a truth value of false. We formally say u *satisfies* p (denoted $u \Vdash p$) since p is true under u. The set of all possible valuations a formula can take is denoted with $\mathcal{U}$. When $\mathcal{P} = \{p, q\}$, $\mathcal{U}$ takes the form $\{\bar{p}\bar{q}, \bar{p}q, p\bar{q}, pq\}$ as this signifies each possible combination of truth values for the atoms in $\mathcal{P}$.

This notion of *satisfaction* extends to formulas as well since the boolean operator connectives take in a number of operends and return singular boolean results. To provide meaning to such operators, we use truth tables [22] which stipulate both the truth values of the atoms (left) and the resulting evalutations of some formula(s) (right). The table below depicts this for each connective:

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

Any valuation u that satisfies some formula $\alpha \in \mathcal{L}$ is said to be a *model* of that formula. The set of all models of α is denoted by $\hat{\alpha}$. If $\hat{\alpha}$ contains at least one model, then α is *satisfiable*. If $\hat{\alpha}$ is empty, α is *unsatisfiable* (a *contradiction*). If $\hat{\alpha} = \mathcal{U}$, then α as a *tautology* as it is true for every possible valuation. For example, $p \wedge \neg p$ is a contradiction while $p \vee \neg p$ is a tautology.

The previous notions define the basics of *model theory* [2] in which we will go into futher detail. For some formulas $\alpha, \beta \in \mathcal{L}$, β is a *logically entailed* by α, written $\alpha \models \beta$, iff $\forall u \in \mathcal{U}$ such that $u \Vdash \alpha$, then $u \Vdash \beta$. This can also be described in set notation, i.e., $\hat{\alpha} \subseteq \hat{\beta}$. We can now say α and β are *logically equivalent*, written $\alpha \equiv \beta$, iff $\alpha \models \beta$ and $\beta \models \alpha$, i.e., iff $\hat{\alpha} \subseteq \hat{\beta}$ and $\hat{\beta} \subseteq \hat{\alpha}$.

We now focus our attention on finite sets of propositional formulas known as *knowledge bases*. For some knowledge base $\mathcal{K}$, a valuation u satisfies $\mathcal{K}$, written $u \Vdash \mathcal{K}$, iff $\forall \alpha \in \mathcal{K}$, $u \Vdash \alpha$. The models of $\mathcal{K}$, denoted by $\hat{\mathcal{K}}$, is the set containing the models of every formula within $\mathcal{K}$. Logical entailment and equivalence with regards to knowledge bases is defined in an analogous manner to formulas, so it will be gone over briefly. For some knowledge bases $\mathcal{K}_1, \mathcal{K}_2$: $\mathcal{K}_2$ is a logically entailed by $\mathcal{K}_1$ iff $\hat{\mathcal{K}}_1 \subseteq \hat{\mathcal{K}}_2$; and is logically equivalent to $\mathcal{K}_2$ iff $\hat{\mathcal{K}}_1 = \hat{\mathcal{K}}_2$. It is said that some formula $\beta \in \mathcal{L}$ is *logically entailed* by $\mathcal{K}$ iff $\forall \alpha \in \mathcal{K}$, $\alpha \models \beta$.

Logical entailment has thus been defined with respect to propositional logic and as such, the notion of defeasible reasoning will now be investigated.

3 DEFEASIBLE REASONING

3.1 Motivation

Classical reasoning systems, such as proposositional logic, are unable to accurately depict real-life human logical reasoning owing to their monotonicity. This is explained by McDermott and Doyle [24] through the assertion we are always working with incomplete knowledge and thus perception and "common sense" are utilized by humans to make up for it by temporarily dismissing commonly held beliefs when presented with new contradictory information.

Propositional logic assumes each proposition contained in knowledge base is absolute and thus, conclusions entailed by adding propositions to the knowledge base are never retracted. For example, the propositions *"birds fly"*, *"penguins are birds"*, *"penguins don't fly"* all represent well founded propositions brought on by the world we inhabit and can be encoded using the propositional knowledge base $\mathcal{K} = \{b \rightarrow f, p \rightarrow b, p \rightarrow \neg f\}$. When attempts are made to reason with $\mathcal{K}$, we find it has no models. This is because $\mathcal{K}$ entails that penguins fly and penguins don't fly, a contradiction. Thus this knowledge base has been rendered useless other than exemplifying the problem brought on by monotonicity.

Nonmonotonic systems offer a solution to this and as such many different systems for nonmonotonic reasoning have already been constructed [29]. The KLM framework for defeasible reasoning [19] defines a preferential approach to building defeasible systems using a combination of conditional logic [10] and a variant of preferential logic [30] as its foundation. It has been found that the defeasible reasoning algorithms produced by this approach are both inherently computable and at least as efficient as classical reasoning algorithms [20], making it an attractive choice for practical defeasible reasoning tools.

3.2 A Preferential Approach

As stated before, the KLM approach employs a preferential approach to defeasible reasoning which in turn has a basis in propositional

logic [30]. *Consequence relations*, denoted by $\vert\sim$, are used to express *defeasible implication*. For instance, $bird \vert\sim flies$ reads as *"birds typically fly"*. It is important to note this does not represent a new propositional connective, but rather a pairing of propositional formulas.

The semantics of KLM-style defeasible entailment require *ranked interpretation* to be defined [12]. Ranked interpretations refer to a set of valuations (referred to as *states*) with some partial order applied to the states such that the most preferred states are found at the bottom and the least preferred states are found at the top.

We can define a ranked interpretation more formally as the function $\mathcal{R} : \mathcal{U} \mapsto \mathcal{N} \cup \{\infty\}$ such that $\mathcal{R}(u) = 0$ for some $u \in \mathcal{U}$, and $\forall i \in \mathcal{N}$ if $\mathcal{R}(v) = i$ then $\forall j \in \mathcal{N}$ such that $0 \leq j < i$, there $\exists u \in \mathcal{U}$ for which $\mathcal{R}(u) = j$ [12]. The symbol $\prec$ denotes the ordering of the states in $\mathcal{R}$.

With this formalization, we define $[\![\alpha]\!]^{\mathcal{P}}$ as the set of all states $s \in \mathcal{R}$ such that $s \Vdash \alpha$. A state s is said to be *minimal* with respect to $\mathcal{R}$ if there exists no state $s' \in \mathcal{R}$ such that $s' \prec s$. The set of all minimum states in $\mathcal{R}$ is described with $\min_{\prec}(\mathcal{P})$.

A ranked interpretation $\mathcal{R}$ satisfies some defeasible implication $\alpha \vert\sim \beta$ if for any $s \in [\![\alpha]\!]^{\mathcal{R}} \cap \min_{\prec}(\mathcal{P})$, $s \Vdash \beta$ as well. $\mathcal{R}$ would then be described as a *ranked model* of $\alpha \vert\sim \beta$. When considering the knowledge base from the prior example, $p \vert\sim \neg f$ would be satisfied by the ranked interpretation depicted in the table below since the minimum state for p, i.e., $pb\bar{f}$, satisfies $\neg f$.

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

We can even check classical propositional satisfaction using this method. That is for some $\alpha \in \mathcal{L}$, $\mathcal{R}$ satisfies α if $\mathcal{R} \Vdash \neg\alpha \vert\sim \bot$ [12]. This, along with the example above, shows us this preferencial approach is more expressive than propositional logic (an unsurprising conclusion).

Ranked preferential entailment functions analogously to its propositional counterpart. For a knowledge base $\mathcal{K}$ and a defeasible implication $\alpha \vert\sim \beta$, $\mathcal{K} \mid\approx_{\mathcal{R}} \alpha \vert\sim \beta$ iff for every ranked interpretation $\mathcal{R}$ such that $\mathcal{R} \Vdash \mathcal{K}$, $\mathcal{R} \Vdash \alpha \vert\sim \beta$. This implies ranked entailment is still monotonic, and thus we move our attention to the idea of *defeasible entailment* [18].

3.3 KLM-Style Defeasible Entailment

Unlike propositional logic, there is no fixed way to go about defeasible entailment in the KLM approach to defeasible reasoning. Rather, we agree the goal is to find whether a defeasible implication is entailed by a knowledge base using a method that adheres to several *rationality properties* proposed by Lehmann and Magnidor [21]. Giovanni et al. [12] proposed *LM-rational* as a term to describe any defeasible entailment procedure that satisfies all of these properties.

The rationality properties, defined by [18], for all knowledge bases $\mathcal{K}$ and propositional formulas α, β, γ as follows:

The first defeasible entailment algorithm for the KLM approach also proposed by Lehmann and Magnidor [21] will now be defined.

(Ref) $\mathcal{K} \mid\approx \alpha \vert\sim \alpha$

(LLE) $\dfrac{\mathcal{K} \mid\approx \alpha \leftrightarrow \beta, \mathcal{K} \mid\approx \alpha \vert\sim \gamma}{\mathcal{K} \mid\approx \beta \vert\sim \gamma}$

(RW) $\dfrac{\mathcal{K} \mid\approx \alpha \rightarrow \beta, \mathcal{K} \mid\approx \gamma \vert\sim \alpha}{\mathcal{K} \mid\approx \gamma \vert\sim \beta}$

And $\dfrac{\mathcal{K} \mid\approx \alpha \vert\sim \beta, \mathcal{K} \mid\approx \alpha \vert\sim \gamma}{\mathcal{K} \mid\approx \alpha \vert\sim \beta \wedge \gamma}$

Or $\dfrac{\mathcal{K} \mid\approx \alpha \vert\sim \gamma, \mathcal{K} \mid\approx \beta \vert\sim \gamma}{\mathcal{K} \mid\approx \alpha \vee \beta \vert\sim \gamma}$

(CM) $\dfrac{\mathcal{K} \mid\approx \alpha \vert\sim \gamma, \mathcal{K} \mid\approx \alpha \vert\sim \beta}{\mathcal{K} \mid\approx \alpha \wedge \beta \vert\sim \gamma}$

3.4 Rational Closure

Rational closure is an LM-rational method of defeasible entailment. In order to define it, we must first gain an idea of *minimal ranked entailment* [18]. This involves defining a partial order for all ranked models of a knowledge base $\mathcal{K}$, denoted with $\preceq_{\mathcal{K}}$. This follows the same principle of ranked interpretations, i.e., the lower the ranked model, the more likely it is (and vice versa). Giordano et al. [11] found there exists a *minimum ranked model* within this ordering. It is around finding this minimum ranked model that the rational closure of a knowledge base $\mathcal{K}$ is based [12]. This is because for $\mathcal{K}$ to entail some defeasible implication, the minimum ranked model must satisfy it.

Before detailing the the initial step of rational closure, we must first know the relationship between a knowledge base $\mathcal{K}$ and its *material counterpart* $\overrightarrow{\mathcal{K}}$. This is quite simple as $\overrightarrow{\mathcal{K}}$ is identical to $\mathcal{K}$, just with all defeasible implications replaced with material implications, e.g., $bird \vert\sim flies$ is replaced with $bird \rightarrow flies$.

Now, the initial step of applying rational closure to a knowledge base $\mathcal{K}$ is to perform the *BaseRank* algorithm on $\mathcal{K}$. This can be defined in a similar manner to a ranked interpretation, as BaseRank maps each formula in $\overrightarrow{\mathcal{K}}$ to a rank in $\mathcal{N} \cup \{\infty\}$ which represents an *exceptional subset* of $\mathcal{K}$. The set of all such exceptional subsets is denoted $\mathcal{E}_n^{\mathcal{K}}$. BaseRank is used to find the minimum ranked model.

An overview of BaseRank can be described by first separating $\mathcal{K}$ into its classical and defeasible components, denoted with K_C and $\mathcal{K}_D$ respectively. $\overrightarrow{\mathcal{K}_D}$ is placed in $\mathcal{E}_0^{\mathcal{K}}$ while K_C is kept aside. For each $\alpha \rightarrow \beta \in \mathcal{E}_0^{\mathcal{K}}$, we check whether $\mathcal{E}_0^{\mathcal{K}} \cup K_C \models \neg\alpha$. If this is the case, the antecedent α is *exceptional* and all formulas within $\mathcal{E}_0^{\mathcal{K}}$ that have α as an antecedent are copied to $\mathcal{E}_1^{\mathcal{K}}$. After each antecedent has been checked, rank $\mathcal{R}_0^{\mathcal{K}}$ is assigned the formulas $\mathcal{E}_0^{K} \backslash \mathcal{E}_1^{\mathcal{K}}$ and we repeat the process for the next subset, i.e., $\mathcal{E}_1^{\mathcal{K}}$. When a subset is reached in which there are no formulas requiring demotion, we have reached the end of the iteration and we add a final rank $\mathcal{R}_{\infty}^{K}$ to house K_C.

With $\mathcal{R}^K$ in hand, we can find out whether $\mathcal{K}$ entails some defeasible query $\alpha \vert\sim \beta$ using the *RationalClosure algorithm*. RationalClosure boils defeasible entailment checking down to the classical propositional entailment that was outlined earlier in the review. It returns `true` iff $\mathcal{K} \mid\approx \alpha \vert\sim \beta$, i.e., iff $\alpha \vert\sim \beta$ is in the rational closure of $\mathcal{K}$ [9].

A overview for RationalClosure will now be described. Given a knowledge base $\mathcal{K}$, the result from BaseRank $\mathcal{R}^K$, and some defeasible implication query $\alpha \vert\sim \beta$.

(1) We check if $\overrightarrow{\mathcal{K}}$ entails $\neg\alpha$.
(2) If it doesn't, α is *compatible* with $\mathcal{K}$ and the result will be based on whether $\overrightarrow{\mathcal{K}}$ entails $\alpha \rightarrow \beta$.
(3) If it does, α is incompatible with $\mathcal{K}$ and the most preferred rank is retracted from $\mathcal{R}^{\mathcal{K}}$, the resulting knowledge base we'll refer to as $\mathcal{K}'$.

- If $\mathcal{R}^{\mathcal{K}'}$ is an empty set, $\mathcal{K} \not\approx \alpha \mid\sim \beta$.
- If $\mathcal{R}^{\mathcal{K}'}$ contains at least one rank, we return to (1) with $\mathcal{K}'$.

In order to finalize this investigation into KLM-style defeasible entailment, rational closure will be exhibited using an extension of the birds and penguins example from before. For this extension, we'll add the statements "*Robins are birds*" ($R \rightarrow b$), and *"birds typically have wings"* ($b \mid\sim w$). We capture all this with the knowledge base $\mathcal{K} = \{b \mid\sim f, p \rightarrow b, p \mid\sim \neg f, R \rightarrow b, b \mid\sim w\}$.

When applying BaseRank to this knowledge base, $K_C = \{p \rightarrow b, R \rightarrow b\}$ and $K_D = \{b \rightarrow f, p \rightarrow \neg f, b \rightarrow w\}$. The only exceptional antecedent when ranking is p and is found in $\mathcal{E}_0^{\mathcal{K}}$. The result of BaseRank $\mathcal{R}^{\mathcal{K}}$ is:

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

We will now use RationalClosure to see if $\mathcal{K}$ defeasibly entails the query $p \rightarrow \neg f$, or *"penguins don't fly"*. We start by checking whether $\overrightarrow{\mathcal{K}} \models \neg p$. This is the case as there is no model u of $\overrightarrow{\mathcal{K}}$ such that $u \Vdash p$. Because of this, we remove the most preferred rank from $\mathcal{R}^{\mathcal{K}}$, i.e., $\mathcal{R}_0^{\mathcal{K}}$, producing:

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

We check again whether $\overrightarrow{\mathcal{K}} \models \neg p$ and find that it does not since there exists a model u of $\overrightarrow{\mathcal{K}}$ such that $u \Vdash p$, e.g, $\{pbr\bar{f}\}$. We now check whether $\overrightarrow{\mathcal{K}}$ entails $p \rightarrow \neg f$ which it clearly does since $p \rightarrow \neg f$ is a formula within $\overrightarrow{\mathcal{K}}$ and as such, any model of $\overrightarrow{\mathcal{K}}$ is also a model of $p \rightarrow \neg f$. We can therefore assert $\mathcal{K} \mid\approx p \rightarrow \neg f$ and with that, the investigation into KLM-style defeasible entailment is concluded.

4 SAT-SOLVERS

4.1 The Problem of Boolean Satisifiability

The problem of boolean satisfiability (*SAT*) is one of generic combinatorial reasoning [14]. It represents the decision whether there exists some interpretation which satisfies a given boolean formula. It has been proven to be NP-complete [4] and while it relates fundamentally to the field of automated theorem-proving, it can also be found in the logical reasoning procedures for knowledge representation and reasoning, as well as a variety of other important problems such as software testing [33] and hardware verification [32] (to name only a few).

This shows SAT to be a very important problem that requires an efficient solution. Boolean satisfiability solvers (*sat-solvers* [14]) attempt to solve SAT efficiently with many different optimisations [15, 28]. There is an international competition in order to find the best sat-solvers [17].

4.2 Sat-solver Basics

Generally, sat-solvers deal with propositional formulas written in *conjunctive normal form* (CNF) [26]. This describes the language made up of *clauses* joined with $\wedge$ connectives. Clauses are constructed from atomic *literals* (the literals for atom p are $\{p, \neg p\}$) and $\vee$ connectives. Algorithms exist that allow easy conversion from a propostional formula to CNF. We can *assign* an atom *true* (T) or *false* (F). An example of a propositional formula in CNF is $(p \vee q) \wedge (p \vee \neg q)$. If we assign p to *true*, we get $(T \vee q) \wedge (T \vee \neg q)$.

There are two main types of sat-solver algorithms [14]. The first of which, *complete algorithms*, either find a solution which satisifes the formula, or prove that the formula is unsatisfiable. The second type, *incomplete algorithms*, take a non-deterministic approach, often employing methods such as stochastic local search [16] to search through all possible atomic assignments in order to find a solution to the formula. Both types have their pros and cons. Complete algorithms deterministically return either a solution to the formula or a proof that it is unsatisfiable, but are regularly quite slow. Incomplete algorithms can implement a vast array of optimisations to improve their search efficiency, but how long they will take to terminate is never known for sure (along with the fact they provide little explanation for their findings). Because of this, incomplete algorithms are generally used when unsatisfiability is not expected (e.g., [27]).

4.3 Basic Sat-solving in Propositional Logic

By definition, satsolving speaks directly to the satisfiability of propositional formulas and can be formulised as such. That is, given some formula $\alpha \in \mathcal{L}$, if there exists a valuation $u \in \mathcal{U}$ such that $u \Vdash \alpha$, i.e., $\hat{\alpha} \neq \emptyset$, then α is satisfiable.

An obvious way of assertaining this in practice consitutes an incomplete algorithm that implements brute force search. That is, for some formula $\alpha \in \mathcal{L}$, each valuation $u \in \mathcal{U}$ is tested individually. This can be very time-consuming and inefficient.

One alternative to brute force, defined in [2], is a semantic tableaux based complete algorithm. A tableau is constructed by decomposing a formula into sets of atomic literals. The construction is complete when no further decomposition is possible. A *clash*, i.e., when both literals for an atom are found within the same subset, represents a contradiction and thus the initial formula is found to be unsatisfiable. The completed tableau acts as a proof for the formula's satisfiability. Other kinds of tableaux algorithms for propositional logic can be found in [7].

4.4 DPLL

The DPLL published by Davis, Logemann, and Loveland in 1960 [5] (based on a preliminary algorithm by Davis and Putnam [6]) is a historic sat-solver algorithm on which which many other sat-solvers are based (e.g., [8], [31]). Silva and Shakallah [23] and Bayardo and Schrag [1] proposed changes to DPLL which greatly improved its proficiency. This lead to further sat-solvers being constructed around it (e.g., [25], [13]) which expanded the optimisations on various aspects of DPLL. It's clear from this that DPLL is regarded as a very influential sat-solver and for this reason its basic operation will be outlined.

DPLL at its core is a backtracking algorithm. It takes in a propositional formula in CNF α (actually a set of clauses but this makes more sense for an explanation), returning *true* if α is satisfiable and *false* if α is unsatisfiable. The algorithm starts by executing a *branching* procedure, i.e., some atom in α is assigned a random truth value. We continue branching until we find an assignment that satisifes α (in which case we return true), or α is false (in which case we *backtrack*). Backtracking involves retracting the most recent branching assignment and branching again with a different assignment (unless there are no new assignments to branch to, in which we backtrack again). If there exists no new branches to be taken and we can't backtrack any further, α is unsatisfiable and *false* is returned.

We'll exhibit this algorithm with a simple example. Given

$$(a \vee \neg b) \wedge (\neg a \vee b)$$

we execute DPLL:

1 `Branch:` $a := T$
  $(T \vee \neg b) \wedge (F \vee b)$
2 `Branch:` $b := F$
  $(T \vee F) \wedge (F \vee F)$ (unsatisfied)
3 `Backtrack:` $b := F$
  $(T \vee \neg b) \wedge (F \vee b)$
4 `Branch:` $b := T$
  $(T \vee F) \wedge (F \vee T)$ (satisfied!)
5 `Return:` *true*

It's clear to see there are many optimisations that can be applied to DPLL [26], but it still provides a good framework for sat-solving that has stood the test of time. CaDiCaL [3], a sat-solver with a DPLL foundation, was a top performer in the recent 2020 SAT competition affiliated with *"23rd International Conference on Theory and Applications of Satisfiability Testing"*. Not bad for a 60 year old algorithm. And with that, the brief investigation into sat-solvers is concluded.

5 CONCLUSIONS

In this review, the syntax and semantics relating to propositional logic was outlined. It was shown that introducing contradictory information into a knowledge base renders classical propositional entailment useless owing to its monotonicity.

It was found that the KLM approach to defeasible reasoning offers a solution to this based on preferential logic. KLM-style defeasible reasoning does not conform to a single method of defeasible entailment. The KLM approach was also shown to be more expressive than that of propositional logic owing to its preferential base. Each step of the rational closure algorithm was outlined and shown to effectively deal with the problem that classical proposositional logic initially struggled with. It was found that since rational closure reduces defeasible entailment to classical propositional entailment, it can be implemented very efficiently with sat-solvers.

Finally, we investigated the basics of sat-solvers, identifying the importance of the SAT problem, its relation to propositional logic, and a brief description of an important historical sat-solver DPLL.

REFERENCES

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

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

[3] Armin Biere, Katalin Fazekas, Mathias Fleury, and Maximillian Heisinger. 2020. CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling Entering the SAT Competition 2020. In *Proc. of SAT Competition 2020 – Solver and Benchmark Descriptions (Department of Computer Science Report Series B, Vol. B-2020-1)*, Tomas Balyo, Nils Froleyks, Marijn Heule, Markus Iser, Matti Järvisalo, and Martin Suda (Eds.). University of Helsinki, 51–53.

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

[5] 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

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

[7] Marcello D'Agostino. 1999. *Tableau Methods for Classical Propositional Logic*. 45–123. https://doi.org/10.1007/978-94-017-1754-0_2

[8] Jon William Freeman. 1995. *Improvements to propositional satisfiability search algorithms*. Ph.D. Dissertation. Citeseer.

[9] Michael Freund. 1998. Preferential reasoning in the perspective of Poole default logic. *Artificial Intelligence* 98, 1 (1998), 209–235. https://doi.org/10.1016/S0004-3702(97)00053-2

[10] D. M. Gabbay. 1985. Theoretical Foundations for Non-Monotonic Reasoning in Expert Systems. In *Logics and Models of Concurrent Systems*, Krzysztof R. Apt (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 439–457.

[11] L. Giordano, V. Gliozzi, N. Olivetti, and G.L. Pozzato. 2015. Semantic characterization of rational closure: From propositional logic to description logics. *Artificial Intelligence* 226 (2015), 1–33. https://doi.org/10.1016/j.artint.2015.05.001

[12] Ivan Varzinczak Giovanni Casini, Thomas Meyer. 2018. Defeasible Entailment: from Rational Closure to Lexicographic Closure and Beyond. In *17th International Workshop on Non-Monotonic Reasoning (NMR)*. Arizona, USA, 109–118.

[13] Eugene Goldberg and Yakov Novikov. 2007. BerkMin: A fast and robust SAT-solver. *Discrete Applied Mathematics* 155, 12 (2007), 1549–1561.

[14] Carla P. Gomes, Henry Kautz, Ashish Sabharwal, and Bart Selman. 2007. Satisfiability solvers.

[15] Weiwei Gong and Xu Zhou. 2017. A survey of SAT solver. In *AIP Conference Proceedings*, Vol. 1836. AIP Publishing LLC, 020059.

[16] Holger H. Hoos and Thomas Stützle. 1999. Towards a characterisation of the behaviour of stochastic local search algorithms for SAT. *Artificial Intelligence* 112, 1 (1999), 213–232. https://doi.org/10.1016/S0004-3702(99)00048-X

[17] Matti Järvisalo, Daniel Le Berre, Olivier Roussel, and Laurent Simon. 2012. The international SAT solver competitions. *Ai Magazine* 33, 1 (2012), 89–92.

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

[19] Sarit Kraus, Daniel Lehmann, and Menachem Magidor. 1990. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence* 44, 1-7 (1990). https://doi.org/10.1016/0004-3702(90)90101-5

[20] Daniel Lehmann. 1999. Another perspective on Default Reasoning. *Annals of Mathematics and Artificial Intelligence* 15 (11 1999). https://doi.org/10.1007/BF01535841

[21] Daniel Lehmann and Menachem Magidor. 1992. What does a conditional knowledge base entail? *Artificial Intelligence* 55, 1 (1992), 1–60. https://doi.org/10.1016/0004-3702(92)90041-U

[22] P. D. Magnus, Tim Button, Aaron Thomas-Bolduc, Richard Zach, and Robert Trueman. 2020. *Forall X: Calgary. An Introduction to Formal Logic*. Open Logic Project. 1–101,378 pages.

[23] J.P. Marques Silva and K.A. Sakallah. 1996. Conflict analysis in search algorithms for satisfiability. In *Proceedings Eighth IEEE International Conference on Tools with Artificial Intelligence*. 467–469. https://doi.org/10.1109/TAI.1996.560789

[24] Drew McDermott and Jon Doyle. 1980. Non-monotonic logic I. *Artificial Intelligence* 13, 1 (1980), 41–72. https://doi.org/10.1016/0004-3702(80)90012-0 Special Issue on Non-Monotonic Logic.

[25] 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*. 530–535.

[26] Alexander Nadel. 2009. *Understanding and improving a modern SAT solver*. Ph.D. Dissertation. Tel Aviv University.

[27] Gi-Joon Nam, Karem A. Sakallah, and Rob A. Rutenbar. 1999. Satisfiability-Based Layout Revisited: Detailed Routing of Complex FPGAs via Search-Based Boolean SAT. In *Proceedings of the 1999 ACM/SIGDA Seventh International Symposium on Field Programmable Gate Arrays* (Monterey, California, USA) *(FPGA '99)*. Association for Computing Machinery, New York, NY, USA, 167–175. https://doi.org/10.1145/296399.296450

[28] Mukul R Prasad, Armin Biere, and Aarti Gupta. 2005. A survey of recent advances in SAT-based formal verification. *International Journal on Software Tools for Technology Transfer* 7, 2 (2005), 156–173.

[29] Raymond Reiter. 1988. Chapter 12 - Nonmonotonic Reasoning. In *Exploring Artificial Intelligence*, Howard E. Shrobe and the American Association for Artificial Intelligence (Eds.). Morgan Kaufmann, 439–481. https://doi.org/10.1016/B978-0-934613-67-5.50016-2

[30] Y. Shoham. 1987. A semantical approach to nonmonotonic logics. In *LICS 1987*.

[31] Allen Van Gelder and Yumi Tsuji. 1995. *Satisfiability testing with more reasoning and less guessing*. Computer Research Laboratory [University of California, Santa Cruz].

[32] Miroslav N Velev and Randal E Bryant. 2003. Effective use of Boolean satisfiability procedures in the formal verification of superscalar and VLIW microprocessors. *Journal of Symbolic Computation* 35, 2 (2003), 73–106. https://doi.org/10.1016/S0747-7171(02)00091-3

[33] Akihisa Yamada, Takashi Kitamura, Cyrille Artho, Eun-Hye Choi, Yutaka Oiwa, and Armin Biere. 2015. Optimization of Combinatorial Testing by Incremental SAT Solving. In *2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST)*. 1–10. https://doi.org/10.1109/ICST.2015.7102599

[34] Lintao Zhang and Sharad Malik. 2002. The Quest for Efficient Boolean Satisfiability Solvers. In *Computer Aided Verification*, Ed Brinksma and Kim Guldstrand Larsen (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1–2.