Anytime Approximations of Classical Logic from Above

MARCELO FINGER, *Department of Computer Science, Institute of Mathematics and Statistics, University of São Paulo, São Paulo, Brazil. Email: mfinger@ime.usp.br*

RENATA WASSERMANN, *Department of Computer Science, Institute of Mathematics and Statistics, University of São Paulo, São Paulo, Brazil. Email: renata@ime.usp.br*

Abstract

In this article we present s_1, a family of logics that is useful to disprove propositional formulas by means of an anytime approximation process. The systems follows the paradigm of a parameterized family of logics established by Schaerf's and Cadoli's system S_1. We show that s_1 inherits several of the nice properties of S_1, while presenting several attractive new properties. The family s_1 deals with the full propositional language, has a complete tableau proof system which provides for incremental approximations; furthermore, it constitutes a full approximation of classical logic from above, with an approximation process with better relevance and locality properties than S_1. When applied to clausal inference, s_1 provides a strong simplification method. An application of s_1 to model-based diagnosis is presented, demonstrating how the solution to this problem can benefit from the use of s_1 approximations.

Keywords: Automated reasoning, approximate reasoning, theorem proving, satisfiability.

1 Introduction

Classical logic has been used to formalize knowledge representation and reasoning in several areas of Artificial Intelligence, such as planning, model-based diagnosis, belief revision and reasoning about actions. The main argument against the use of logic has always been the high computational complexity involved. Even if we stay within propositional logic, deciding satisfiability is NP-complete.

Recently, there have been several attempts to model resource-bounded reasoning through different logical formalisms. The idea which we follow in this article is that of approximate reasoning. Schaerf and Cadoli [18] proposed the following guidelines for approximations of classical logic:

(1) Approximations should have clear semantics.
(2) Approximate answers should be easier to compute than the original ones.
(3) Approximations should be incrementally refined, eventually converging to the right answer.
(4) We should be able to approximate from both ends, i.e. approximations should provide an upper bound for proving theorems and a lower bound for refuting non-theorems.

When approximating classical logic, we usually aim at giving two methods: one which is sound but incomplete, proving less theorems than classical logic and one which is complete

Published online 5 September 2006 doi:10.1093/logcom/exl018

but unsound, proving more theorems than classical logic. The former is called an approximation of classical logic *from below* and the latter is an approximation *from above*.

Approximating from below, using sound and incomplete methods, is a process reasonably well-understood. Several non-classical systems, such as paraconsistent or relevant logics, can be seen as approximations from below. One only has to restrict some axioms or derivation rules in order to obtain a subsystem of classical logic. Sound and incomplete methods are useful for proving theorems, for if a theorem is proven in a subsystem of classical logic, it must also be a classical theorem. An *approximation process* from below is a family of subclassical logics, each of which proves more theorems than the previous ones and in the limit it becomes classical logic.

On the other hand, approximating from above is useful for refutations, or for testing satisfiability. If a set of formulas is unsatisfiable in a system that approximates from above, then it is unsatisfiable in classical logic. However, defining such approximations is a harder task. A super-system of classical logic is usually classically inconsistent, and an *approximation process from above* has to avoid such pitfalls. In a family of logics that approximates classical logic from above, each super-classical logic must prove less theorems (i.e. disprove more classical non-theorems) than the previous ones, and in the limit it proves only the classical theorems.

In this article, we depart from the system S_1 proposed by Schaerf and Cadoli [18] to approximate classical logic from above. Their system was restricted to clausal formulas. We propose a similar system that can be extended to full propositional logic and show that our system truly approximates classical logic from above. The family s_1 allows for a *parametric approximation process from above*; the parameters are given by sets s of propositional letters such that for

$$\emptyset \subset s' \subset s'' \subset \ldots \subset \mathcal{P}$$

we have

$$\models^1_{\emptyset} \supset \models^1_{s'} \supset \models^1_{s''} \supset \ldots \supset \models^1_{\mathcal{P}} = \models_{\mathbf{CL}}$$

where $\mathcal{P}$ is the set of all proposition symbols, **CL** is classical logic and $\models^1_s$ is the entailment relation of the logic s_1 (s).

The system s_1 is first introduced by means of valuation semantics.[1] We then provide a tableau proof method for s_1, show how it can be used for performing step-by-step approximations of classical refutation and give an example of an application.

The article proceeds as follows: in the next section, we briefly review Cadoli and Schaerf's work on approximate entailment. In Section 3, we present the semantics for s_1 and in Section 4 we focus on the clausal fragment, so that our system can be compared to Schaerf and Cadoli's. We also present an example of application of s_1 to model-based diagnosis. In Section 5, we present an incremental tableau proof method for s_1 over full propositional logic. Finally, in Section 6, we review some logical systems which are related to ours and in Section 7 we point towards future work.

[1] A semantical presentation of s_1 was first introduced in [5].

Notation: Let $\mathcal{P}$ be a countable set of propositional letters. We concentrate on the classical propositional language $\mathcal{L}_\mathcal{C}$ formed by the usual boolean connectives $\rightarrow$ (implication), $\wedge$ (conjunction), $\vee$ (disjunction) and $\neg$ (negation).

Throughout the article, we use lowercase Latin letters to denote propositional letters, lowercase Greek letters to denote formulas, and uppercase letters (Greek or Latin) to denote sets of formulas.

We indicate by var(α) the set of propositional letters which occur in α. Let $S \subset \mathcal{P}$ be a finite set of propositional letters. We abuse notation and write that, for any formula $\alpha \in \mathcal{L}_\mathcal{C}$, $\alpha \in S$ if all its propositional letters are in S, i.e., var(α)$\subseteq S$. A *propositional valuation* v_p is a function $v_p : \mathcal{P} \rightarrow \{0, 1\}$.

If Γ and Δ are any sets, by $\|\Gamma - \Delta\|$ we mean the *symmetric difference* of two sets, namely $\|\Gamma - \Delta\| = (\Gamma - \Delta) \cup (\Delta - \Gamma)$.

2 The logical approximations

2.1 The generic notion of a logic approximation

We say that a logic $\mathbf{L}$ *is approximated by a family of logics* $\mathbf{L}_1, \mathbf{L}_2, \ldots$if

$$\|\mathbf{L} - \mathbf{L}_1\| \supseteq \|\mathbf{L} - \mathbf{L}_2\| \supseteq \ldots \supseteq \emptyset.$$

To complete this definition, we must specify what we mean by the logic **L**. In fact, there are several possibilities, leading to several forms of approximation:

(1) In a *right approximation*, the logic $\mathbf{L}$ is given by the *set of theorems* (or *theses*, or *validities*) of $\mathbf{L}$, $Th(\mathbf{L}) = \{\alpha | \models_\mathrm{L} \alpha\}$. The word 'right' refers to the right-hand side of the $\models_\mathrm{L}$ −symbol.
(2) In a *left approximation*, the logic $\mathbf{L}$ is given by the *set of antitheses* (or *inconsistencies*) of $\mathbf{L}$, $AntiTh(\mathbf{L}) = \{\beta | \beta \models_\mathrm{L} \bot\}$.
(3) In a *full approximation*, the logic $\mathbf{L}$ is given by its *entailment relation*, $\models_\mathrm{L} \subseteq 2^{\mathcal{L}_\mathrm{L}} \times \mathcal{L}_\mathrm{L}$, where $\mathcal{L}_\mathrm{L}$ is the set of formulas of $\mathbf{L}$. A full approximation is both a right and a left approximation.

There are two special cases of approximation worth noting. We say that the sequence, where $\mathbf{L}_1, \mathbf{L}_2, \ldots$ is an *approximation of* $\mathbf{L}$ *from below if*

$$\mathbf{L}_1 \subset \mathbf{L}_2 \subset \ldots \subset \mathbf{L}$$

Approximations from below are useful for theorem proving. Each logic $\mathbf{L}_i$ proves more elements (theses/antitheses/entailments) of the goal logic $\mathbf{L}$. On the other hand, we may have an *approximation* of $\mathbf{L}$ from above when,

$$\mathbf{L}_1 \supset \mathbf{L}_2 \supset \ldots \supset \mathbf{L}$$

Approximations from above are useful for disproving theorems, which in the case of propositional classical logic is equivalent to the well-known SAT problem. The SAT problem consists in, given a set of propositional formulas Γ, deciding whether Γ is satisfiable, i.e. whether a truth assignment can be found that makes each formula of Γ true.

An approximation is *finite* if there exists a natural number n such that $\mathbf{L}_n = \mathbf{L}$. And, we say that an approximation is *parameterized* if there exists a universe set U and parameter sets $S \subseteq U$ such that $\mathbf{L}(S)$ is defined for each set $S \subseteq U$. In this case, an approximation from above parameterized by sets of propositional letters would be a sequence such that whenever $\emptyset \subset S^1 \subset S^2 \subset \ldots \subset \mathcal{P}$ we have

$$\mathbf{L}(\emptyset) \supset \mathbf{L}(S^1) \supset \mathbf{L}(S^2) \supset \ldots \supset \mathbf{L}(\mathcal{P}) = \mathbf{L}.$$

Similarly, a parameterized approximation from below can be given. Given a parameterized approximation, an *anytime algorithm* is a procedure to decide if $\alpha \in \mathbf{L}$ (or, in case the goal is to refute, if $\alpha \notin \mathbf{L}$) that allows us to expand the parameter set S in such a way that, if the algorithm is stopped anytime, it produces a *partial answer*: either 'yes', or 'no up to $\mathbf{L}(S^i)$'. We say that this process of approximation is *incremental* if the algorithm for testing whether $\alpha \in \mathbf{L}(S^i)$ (or whether $\alpha \notin \mathbf{L}(S^i)$) can be started from the state where the proof that $\alpha \notin \mathbf{L}(S^{i-1})$ (or that $\alpha \in \mathbf{L}(S^{i-1})$) stopped.

2.2 *Approximate entailment*

In this section, we briefly present Schaerf and Cadoli's original family of logics S_1 and S_3 and comment their nice approximation properties and on some of their shortcomings. We then point to the specific limitations of the system S_1 which will be explored in further sections.

Schaerf and Cadoli proposed two approximations for classical propositional logic [18]. The idea of the systems was to work with formulas in clausal form and allow for non-classical valuation semantics. Both systems, S_1 and S_3, are parameterized by a context set S, which contains some propositional variables.

The systems are described via non-classical valuation semantics. In both systems, atoms inside S behave classically, i.e., if $p \in S$ we must have $v(p) = 1$ if and only if $v(\neg p) = 0$. In the system S_1, an atom outside S is assigned the value 0, and so is its negation ($v(p) = v(\neg p) = 0$). There is only one possibility for the pair $p, \neg p$, hence the name S_1. In S_3, there are three possibilities for such a pair, the two classical ones, where $v(p) = 0$ and $v(\neg p) = 1$or $v(p) = 1$ and $v(\neg p) = 0$, and a third possibility, where both p and $\neg p$ receive the value 1. Since we are only dealing with clauses (disjunctions of literals), we can easily extend S_1 and S_3 valuations to formulas using the classical rule for disjunction:

$$v(\alpha \vee \beta) = 0 \text{ iff } v(\alpha) = 0 \text{ and } v(\beta) = 0.$$

Given the above restrictions for the truth assignments, the notions of S_1 and S_3 entailment are defined in the usual way: we say that $B \models^1_S \alpha$ if and only if every S_1-valuation that satisfies all the formulas in B also satisfies α. In the same way, $B \models^3_S \alpha$ if and only if every S_3-valuation that satisfies all the formulas in B also satisfies α.

The idea of the approximations is to try to check whether for some set S, either $B \not\models^1_S \alpha$ or $B \models^3_S \alpha$. Since S_1-entailment is complete but unsound, if $B \not\models^1_S \alpha$, then we know that classically $B \not\models \alpha$. And since S_3-entailment is sound but incomplete, if $B \models^3_S \alpha$, then we know that classically $B \models \alpha$.

The following theorem shows that if we manage to obtain an answer for a small S, we stay within good complexity bounds.

THEOREM 2.1 [18]
There is an algorithm for deciding whether $B \models^{i}_{S} \alpha$ in $O(|B|.|\alpha|.2^{|S|})$ time, $i \in \{1, 3\}$.

Schaerf and Cadoli provide an incremental algorithm for approximating the test $B \models \alpha$ However, their systems have three major limitations:

(1) The system is restricted to $\rightarrow$-free formulas and in negation normal form. In [1], it is noted that the standard translation of formulas into clausal form does not preserve truth-values under the non-standard semantic of S_1 and S_3.
(2) The set S must be guessed at each step of the approximation; no method is given for the atoms to be added to S. Some heuristics for a specific application are presented in [21], but nothing is said about the general case.
(3) There is no incremental proof theory.

In [6], we have proposed solutions for these shortcomings concerning the system S_3. In this article, we concentrate on S_1.

LEMMA 2.2
According to our definitions above, S_3 constitutes a full approximation from below, and S_1 is simply a left approximation from above.

PROOF. The properties of S_3 have been treated elsewhere see [6]. To see that S_1 is a left approximation from above, suppose $S \subseteq S'$. We note that there is a one-to-many correspondence between $S_1(S)$- and $S_1(S')$- valuations; in fact, for every $v^1_{S'}$, let its correspondent v^1_S be such that v^1_S agrees with $v^1_{S'}$ on all atoms in S; v^1_S and $v^1_{S'}$ agree on all literals but those in $q \in S' \setminus S$, in which case $v^1_S(q) = 0$. So given $v^1_{S'}$, its corresponding v^1_S is fully determined, but not the other way round. It follows that if $S_1(S')$ falsifies a clause ψ with valuation $v^1_{S'}$, its corresponding $S_1(S)$-valuation also falsifies ψ, because v^1_S and $v^1_{S'}$ agree on all literals but those in $S' \setminus S$, but for any literal $q \in S' \setminus S$ we have $v^1_S(q) = 0$ so $v^1_S(\psi) = 0$. Now, suppose B is an $S_1(S')$-inconsistent finite set of clauses, which means that any $S_1(S)$-valuation falsifies some clause in B; but from the explained above, every $S_1(S)$-valuation corresponds to some $S_1(S')$-valuation and also falsifies a clause in B, so B is also an $S_1(S)$-inconsistent, which proves S_1 is a left approximation from above.

To see that it is not a right approximation from above, it suffices to consider the formula $p \vee \neg p$ for $p \notin S$; clearly $v^1_S(p \vee \neg p) = 0$, so S_1 fails to be a right approximation from above, and hence it is not a full approximation from above. ■

So the original system S_1 fails to be a full approximation from above of classical logic. We propose in this article a system which fully approximates classical logic from above, and which is not restricted to clausal form.

Another problem with Schaerf and Cadoli's S_1 is that in order to test whether $B \models^{1}_{S} \alpha$, the context set S must contain at least one atom from each clause in B, even clauses which are intuitively irrelevant for α. We will see in Section 3.2 that our system performs a little bit better with respect to relevance, but the problem of isolating what is relevant in a refutation is a hard one and we hope to be contributing towards a better understanding of that problem.

Section 3.2 presents a discussion on the notions of locality and relevance in the context of refutation, for both the S_1 family and the s_1-family which we describe next.

3 Semantics for s_1

The problem of creating a logic that approximates classical logic from above comes from the fact that any logic that is defined in terms of a binary valuation $v : \mathcal{L} \to \{0, 1\}$ that properly extends classical logic is inconsistent. Indeed, if it is a proper extension of classical logic, it will contradict a classical validity. Since it is an extension of classical logic, from this contradiction any formula is derivable.

Cadoli and Schaerf avoided this problem by taking a binary valuation that was not an extension of a classical one. Here, we take a different approach, for we want to construct an extension of classical entailment, and define a ternary valuation, that is, we define a valuation $v_s^1(\alpha) \subseteq \{0, 1\}$; later we show that $v_s^1(\alpha) \neq \emptyset$.

For that, consider the full language of classical logic based on a set of proposition symbols $\mathcal{P}$. We define the family of logics $\mathrm{s}_1(s)$, parameterized by the set $s \subseteq \mathcal{P}$. Let α be a formula and let *prop(α)* be the set of propositional symbols occurring in α. We say that $\alpha \in s$ iff $prop(\alpha) \subseteq s$.

Let v_p be a classical propositional valuation. Starting from v_p, we build an s_1-valuation $v_s^1 : \mathcal{L} \to 2^{\{0,1\}}$, by defining when $1 \in v_s^1(\alpha)$ and when $0 \in v_s^1(\alpha)$. This definition is parameterized by the set $s \subseteq \mathcal{P}$ in the following way. Initially, for propositional symbols, v_s^1 extends v_p:

$$0 \in v_s^1(p) \Leftrightarrow v_p(p) = 0$$
$$1 \in v_s^1(p) \Leftrightarrow v_p(p) = 1 \text{ or } p \notin s$$

That is, v_s^1 extends v_p but whenever we have an atom $p \notin s$, $1 \in v_s^1(p)$; if $p \notin s$ and $v_p(p) = 0$, we get $v_s^1(p) = \{0, 1\}$. The rest of the definition ofv_s^1 proceeds in the same spirit, as follows:

$$\begin{array}{lcll}
0 \in v_s^1(\neg\alpha) & \Leftrightarrow & 1 \in v_s^1(\alpha) & \\
0 \in v_s^1(\alpha \wedge \beta) & \Leftrightarrow & 0 \in v_s^1(\alpha) \text{ or } 0 \in v_s^1(\beta) & \\
0 \in v_s^1(\alpha \vee \beta) & \Leftrightarrow & 0 \in v_s^1(\alpha) \text{ and } 0 \in v_s^1(\beta) & \\
0 \in v_s^1(\alpha \to \beta) & \Leftrightarrow & 1 \in v_s^1(\alpha) \text{ and } 0 \in v_s^1(\beta) & \\
1 \in v_s^1(\neg\alpha) & \Leftrightarrow & 0 \in v_s^1(\alpha) & \text{or } \neg\alpha \notin s \\
1 \in v_s^1(\alpha \wedge \beta) & \Leftrightarrow & 1 \in v_s^1(\alpha) \text{ and } 1 \in v_s^1(\beta) & \text{or } \alpha \wedge \beta \notin s \\
1 \in v_s^1(\alpha \vee \beta) & \Leftrightarrow & 1 \in v_s^1(\alpha) \text{ or } 1 \in v_s^1(\beta) & \text{or } \alpha \vee \beta \notin s \\
1 \in v_s^1(\alpha \to \beta) & \Leftrightarrow & 0 \in v_s^1(\alpha) \text{ or } 1 \in v_s^1(\beta) & \text{or } \alpha \to \beta \notin s
\end{array}$$

This semantics extends classical logic but is not trivial. To see that, suppose we have a propositional valuation v_p such that $v_p(p) = 0$ and $v_p(q) = 0$. We compute $v_s^1(p \to q)$ for different values of s.

(1) Suppose $s = \emptyset$; then $p \to q \notin s$:

$$\left.\begin{array}{l} v_s^1(p) = \{0, 1\} \\ v_s^1(q) = \{0, 1\} \end{array}\right\} \Longrightarrow v_s^1(p \to q) = \{0, 1\}$$

(2) Suppose $s = \{q\}$; then $p \to q \notin s$

$$\left.\begin{array}{l} v_s^1(p) = \{0, 1\} \\ v_s^1(q) = \{0\} \end{array}\right\} \Longrightarrow v_s^1(p \to q) = \{0, 1\}$$

(3) Suppose $s = \{p\}$; still $p \to q \notin s$

$$\left.\begin{array}{l} v_s^1(p) = \{0\} \\ v_s^1(q) = \{0,1\} \end{array}\right\} \Longrightarrow v_s^1(p \to q) = \{1\}$$

Note that in this last case, even if $p \to q \notin s$, we have a 'classical' truth value for it, that is, $v_s^1(p \to q) \neq \{0,1\}$. In the case where $s = \{p, q\}$ we are in a totally classical setting, and we can easily obtain $v_s^1(p \to q) = \{0\}$ by making $v_p(p) = 1$ and $v_p(q) = 0$.

We now analyse the properties of s_1-valuations. We start pointing out two basic properties of v_s^1, namely that is a ternary relation and that $1 \in v_s^1(\alpha)$ whenever $\alpha \notin s$.

LEMMA 3.1
Let α be any formula. Then $v_s^1(\alpha) \neq \emptyset$. Furthermore, if $\alpha \notin s$ then $1 \in v_s^1(\alpha)$.

PROOF. We show that $v_s^1(\alpha) \neq \emptyset$, by structural induction on α. For the base case, note that for any propositional symbol, $v_p(p) \in v_s^1(p)$, so $v_s^1(p) \neq \emptyset$. For the inductive cases, note that the truth value of the formulas $\neg\alpha$, $\alpha \wedge \beta$, $\alpha \vee \beta$ and $\alpha \to \beta$ are not empty whenever, according to the semantic definition above, the truth value of their components are not empty.

If $\alpha \notin s$, a simple inspection on the definition above shows that $1 \in v_s^1(\alpha)$. ■

It is interesting to see that in one extreme, i.e. when $s = \emptyset$, s_1-valuations trivialize, assigning the value 1 to every formula in the language. When $s = \mathcal{P}$, s_1-valuations over the connectives correspond to Kleene's semantics for three-valued logics [8].

The next important property of v_s^1 is that it is an extension of classical logic in the following sense. Let v_s^1 be an s_1-valuation; its *underlying propositional valuation*, v_p is given by

$$\begin{array}{ll} v_p(p) = 0, & \text{if } 0 \in v_s^1(p) \\ v_p(p) = 1, & \text{if } 0 \notin v_s^1(p) \end{array}$$

as can be inspected from definition of v_s^1. Also note that v_p and s uniquely define v_s^1.

LEMMA 3.2
Let $v_c : \mathcal{L} \to \{0,1\}$ be a classical binary valuation extending v_p. Then, for every formula α, $v_c(\alpha) \in v_s^1(\alpha)$.

PROOF. By structural induction on α. It suffices to note that the property is valid for $p \in \mathcal{P}$. Then a simple inspection of the definition of v_s^1 gives us the inductive cases. ■

Just note that Lemma 3.2 implies that no formula α is such that $v_s^1(\alpha) = \emptyset$ as in Lemma 3.1. We can also say that if $\alpha \in s$, then v_s^1 behaves *classically* in the following sense.

LEMMA 3.3
Let v_p be a propositional valuation and let v_s^1 and v_c be, respectively, its $s_1(s)$ and classical extensions. If $\alpha \in s$,$v_s^1(\alpha) = \{v_c(\alpha)\}$.

PROOF. A simple inspection of the definition of v_s^1 shows that if $\alpha \in s$, $v_s^1 s$ behaves classically.

Finally, we compare s_1-valuations under expanding parameter sets s. ■

LEMMA 3.4
Suppose $s \subseteq s'$ and let $v_s^1(\alpha)$ and $v_{s'}^1(\alpha)$ extend the same propositional valuation. Then $v_s^1(\alpha) \supseteq v_{s'}^1(\alpha)$.

PROOF. If $\alpha \in s$, $v_s^1(\alpha)$ and $v_{s'}^1(\alpha)$ behave classically. If $\alpha \notin s$, then $1 \in v_s^1(\alpha)$ and we have to analyse what happens when $0 \in v_{s'}^1(\alpha)$. By structural induction on α, we show that $0 \in v_s^1(\alpha)$. For the base case, just note that v_s^1 and $v_{s'}^1$ have the same underlying propositional valuation.

Consider $0 \in v_{s'}^1(\neg\alpha)$, then $1 \in v_{s'}^1(\alpha)$. Since $\alpha \notin s$,$1 \in v_s^1(\alpha)$, so $0 \in v_s^1(\neg\alpha)$.

Consider $0 \in v_{s'}^1(\alpha \to \beta)$, then $1 \in v_{s'}^1(\alpha)$ and $0 \in v_{s'}^1(\beta)$. By the induction hypothesis, $0 \in v_s^1(\beta)$. If $\alpha \notin s$, $1 \in v_s^1(\alpha)$ and we are done. If $\alpha \in s$, then also $\alpha \in s'$, $v_{s'}^1(\alpha)$ and $v_{s'}^1(\alpha)$ behave classically and agree with each other, so $1 \in v_s^1(\alpha)$ and we are done.

The cases where $0 \in v_{s'}^1(\alpha \wedge \beta)$ and $0 \in v_{s'}^1(\alpha \vee \beta)$ are straightforward consequences of the induction hypothesis. ■

The next step is to define the notion of a s_1-entailment.

3.1 s_1*-entailment*

The idea is to define an entailment relation for s_1, $\models_s^1$, parameterized on the set $s \subseteq \mathcal{P}$ so as to extend for any s the classical entailment relation

$$B \models \alpha.$$

To achieve that, we have to make valuations applying on the left-hand side of $\models_s^1$ to be *stricter* than classical valuations, and the valuations that apply to the right handside of$\models_s^1$ to be more *relaxed* than classical valuations, for every $s \subseteq \mathcal{P}$. This motivates the following definitions.

DEFINITION 3.5
Let $\alpha \in \mathcal{L}$ and let v_s^1 be a s_1-valuation. Then:

(1) *If* $v_s^1(\alpha) = \{1\}$ *then we say that* α *is* strictly satisfied *by* v_s^1.
(2) *If* $1 \in v_s^1(\alpha)$ *then we say that* α *is* relaxedly satisfied *by* v_s^1.

That these definitions are the desired ones follows from the following.

LEMMA 3.6
Let $\alpha \in \mathcal{L}$ Then:

(1) α is strictly satisfiable implies that α is classically satisfiable.
(2) α is classically satisfiable implies that α is relaxedly satisfiable.

PROOF.

(a) Consider v_s^1 such that $v_s^1(\alpha) = \{1\}$. Let v_p be its underlying propositional valuation and let v_c be a classical valuation that extends v_p. Since $0 \notin v_s^1(\alpha)$, by Lemma 3.2 we have that $v_c(\alpha) \neq 0$, so $v_c(\alpha) = 1$.
(b) Consider a classical valuation v_c such that $v_c(\alpha) = 1$. Let v_p be its underlying propositional valuation. Then directly from Lemma 3.2 ,$1 \in v_s^1(\alpha)$. ■

We are now in a position to define the notion of s_1-entailment.

DEFINITION 3.7
We say that $\beta_1, \ldots, \beta_m \models_s^1 \alpha$ iff every s_1-valuation v_s^1 that strictly satisfies all β_i, $1 \leq i \leq n$, relaxedly satisfies α.

The following are important properties of s_1-entailment.

LEMMA 3.8

(1) $B \models^1_{\emptyset} \alpha$, for every $\alpha \in \mathcal{L}$.
(2) $\models^1_{\mathcal{P}} = \models_{\mathrm{CL}}$
(3) *If* $s \subseteq s'$, $\models^1_s \supseteq \models^1_{s'}$.

PROOF.

(1) By Lemma 3.1, $1 \in v^1_{\emptyset}(\alpha)$, for every $\alpha \in \mathcal{L}$.
(2) By Lemma 3.3, $v^1_{\mathcal{P}}$ is a classical valuation, and the notions of strict, relaxed and classical valuation coincide.
(3) Suppose $s \subseteq s'$, $B \models^1_{s'} \alpha$ but $B \not\models^1_s \alpha$. Then there exists v^1_s such that $v^1_s(\beta_i) = \{1\}$, for all $\beta_i \in B$ but $v^1_s(\alpha) = \{0\}$. Let $v^1_{s'}$ be the s_1-valuation generated by v^1_s underlying propositional valuation. From Lemma 3.4 we have that $v^1_{s'}(\beta_i) = \{1\}$, for all $\beta_i \in B$.
Since $B \models^1_{s'} \alpha$, we have that $1 \in v^1_{s'}(\alpha)$. Again by Lemma 3.4 we get $1 \in v^1_s(\alpha)$, which contradicts $v^1_s(\alpha) = \{0\}$. So $B \models^1_s \alpha$. ■

From what has been shown, it follows directly that this notion of entailment is the desired one.

THEOREM 3.9
The family of s_1-logics fully approximates classical logic from above, that is, for $\emptyset \subseteq s' \subseteq \ldots \subseteq s'^n \subseteq \mathcal{P}$:

$$\models^1_{\emptyset} \supseteq \models^1_{s'} \supseteq \ldots \supseteq \models^1_{s'^n} \supseteq \models^1_{\mathcal{P}} = \models_{\mathrm{CL}}$$

PROOF. Directly from Lemma 3.8. ■

Note that if v^1_s is a s_1-valuation falsifying $B \models^1_s \alpha$, we have a classical valuation v_c that falsifies $B \models \alpha$ built as an extension of the propositional valuation v_p such that $v_p(p) = 1 \iff v^1_s(p) = \{1\}$.

One interesting property that fails for s_1-entailment is the *deduction theorem*. One half of it is still true, namely that

$$B \models^1_s \alpha \Longrightarrow \models^1_s (\bigwedge B) \rightarrow \alpha.$$

However, the converse is not true. Here is a counterexample. Suppose $q \notin s$ and $p \in s$, so $q \rightarrow p \notin s$. Then $\models^1_s q \rightarrow p$; take a valuation that makes $v^1_s(q) = \{1\}$ and $v^1_s(p) = \{0\}$, hence $q \not\models^1_s p$.

We now examine s_1 and S_1 with respect to paraconsistency and paracompleteness.

DEFINITION 3.10
A logic L is paraconsistent if there are formulas α and β such that $\alpha, \neg\alpha \not\models_L \beta$.

DEFINITION. 3.11
A logic L is paraconsistent if there is a formula α such that $\not\models_L \alpha \vee \neg\alpha$.

Despite the fact that s_1 allows for truth values $\{0,1\}$ in which, apparently, a formula is both true and false, s_1 *is not a paraconsistent logic*. The reason behind this is quite simple. According to Theorem 3.9, s_1-entailment approximates classical logic from above, which implies that all classical theorems are s_1-theorems. In particular, the trivialization theorems all hold in s_1, that is, it is not possible to strictly satisfy both α and $\neg\alpha$; so s_1 cannot be paraconsistent.

By the same argument, we see that s_1 cannot be paracomplete, either. However, $S_1(S)$ is paracomplete when S does not contain all the propositional letters of the language, as shown later.

PROPOSITION 3.12
If S is a proper subset of $\mathcal{P}$, then the system S_1 is paracomplete.

PROOF. Let $p \notin S$ then in any S_1-valuation, both p and $\neg p$ are assigned the value 0. Hence, $\not\models^1_S p \vee \neg p$.

■

$S_1(S)$ is not paraconsistent either, for in S_1 it is not possible to satisfy both p and$\neg p$ simultaneously.

We next concentrate on the important property of locality.

3.2 *Locality and relevance*

We would like to have approximations that behave in a local way: formulas of B which are completely irrelevant to the formula being refuted should not play a role in the process. Intuitively, when trying to disprove $B \not\models \alpha$, we do not want the approximation process to include any more formulas from B than what is necessary for achieving this goal. In the approximation of a refutation, this is related to the size of the context set s needed to actually perform the refutation. The idea is that we do not want to include in s elements that are totally irrelevant to the refutation. Let us first define formally what we mean by 'relevant' and 'local'.

Let α be a formula and B a set of formulas. Let R be a binary relation between formulas defined as: $R(\beta_1, \beta_2)$ if and only if $\mathrm{var}(\beta_1) \cap \mathrm{var}(\beta_2) \neq \emptyset$; that is, $R(\beta_1, \beta_2)$ means that β_1 and β_2 are syntactically related by sharing some atom. Let R^* be the transitive closure of R and define B_α to be the set $\{\beta \in B | R^*(\alpha, \beta)\}$. Note that by definition, R^* is reflexive, symmetric and transitive. We say that B_α is the set of formulas in B that are *relevant* to α.

When trying to refute a formula, one should be able to restrict attention to the relevant formulas.

PROPOSITION 3.13
Let B be a consistent set of formulas and α a formula. then $B \not\models \alpha$ if and only if $B_\alpha \not\models \alpha$.

PROOF. One side of the implication, namely that if $B \not\models \alpha$ then $B_\alpha \not\models \alpha$ follows directly from the monotonicity of classical logic, since $B_\alpha \subseteq B$.

To show that $B_\alpha \not\models \alpha$ implies $B \not\models \alpha$, suppose that $B \models \alpha$. Let $(\beta_1, \beta_2, ..., \beta_n)$ be a proof for α, where each β_i is an axiom, an element of B or follows by Modus Ponens from β_{i_1} and β_{i_2}, with $i_1, i_2 < i$, and $\beta_n = \alpha$. Let k be the largest integer such that $1 \leq k \leq n$ and $\beta_k \notin B_\alpha$. We know that β_k is not used in an application of Modus Ponens, since it is not related through R^* to any β_l with $k < l$. This means that β_k can be deleted from the proof, and $(\beta_1, \beta_2, ..., \beta_{k-1}, \beta_{k+1}, ..., \beta_n)$ is still a proof for α. We can repeat this reasoning a finite number of steps until there is no more β_i in the proof which is not an element of B_α. Hence, $B_\alpha \vdash \alpha$ and therefore, $B_\alpha \models \alpha$. ■

Given B and α, we call a parameterized approximation *absolutely local* if the parameter set s is a subset of $\mathrm{var}(B_\alpha)$. Non-local approximations are thus undesirable because they force into the context set s atoms totally unrelated to the goal we want to achieve.

It is not hard to see that Schaerf and Cadoli's S_1 is non-local. In fact, in a $S_1(S)$ approximation, if B is a set of clauses and we want to refute $B \models \alpha$, then every clause in B must have at least one propositional letter in S. If all atoms in a clause c are outside S, then by the S_1 semantics, all valuations v^1_S will be such that $v^1_S(c) = 0$, and it will be impossible to satisfy B and falsify α. This fact has already been noted in the literature of applications of S_1 to diagnostic problems [21] (see Example 4.7). In s_1, the parameter set does not have to contain an atom of every clause, but only of those clause without positive literals.

Unfortunately, the s_1-method presented in this article is not absolutely local, as demonstrated subsequently.

LEMMA 3.14
Let B be a set of formulas and let $c \in B$ be a clause containing only negative literals. If $B \not\models^1_s \alpha$ then s must contain at least one atom in c.

PROOF. If $c = \neg q_1 \vee \ldots \vee \neg q_d$ and all $q_i \notin s$, $1 \leq i \leq d$, then for any v^1_s, $v^1_s(\neg q_i) = \{0, 1\}$ and $v^1_s(c) = \{0, 1\}$. Since $c \in B$, there is no s_1-valuation that strictly satisfies B, so it is not possible that $B \not\models^1_s \alpha$. ■

So, in absolute terms, neither S_1 nor s_1 are local. For s_1-approximations, clauses with negative literals only can be seen as global constraints that have to be met at every approximation step. However, this restriction to s_1-approximations is much less restrictive than that of S_1 approximations.

We can then introduce a notion of *relative locality* that compares the two methods by looking at the propositional letters that have to be considered and which are not in $\text{var}(B_\alpha)$:

DEFINITION 3.15
A parameterized approximation method M is more local than a method M' if and only if whenever $B \not\models_{M'(S')} \alpha$, $B \neq \emptyset$, there exists an S such that $S - var(B_\alpha) \subseteq S' - var(B_\alpha)$ and $B \not\models_{M(S)} \alpha$

As S_1 is an approximation of clausal logic, in the following we compare s_1 versus S_1 over the clausal fragment of propositional logic. The relative locality of the methods will be described in Corollary 4.5.

4 Clausal s_1

To better compare s_1 with S_1, let us study how s_1 behaves over clausal formulas. We are going to show, as in the S_1 case, s_1 allows for simplifications which reduce the size of the decision of $B \models^1_s \alpha$, with the advantage of being a more local procedure.

Let $s_1 - \text{simplify}\ (B, s)$ be the result of deleting from B all negative atoms outside s and all clauses that contain positive atoms outside s.

We first show a result reducing s_1-strict satisfiability to a smaller instance of classical satisfiability.

LEMMA 4.1
B is s_1-strictly satisfiable iff $s_1 - \text{simplify}(B, s)$ is classically satisfiable.

PROOF. We first note that $s_1 - \text{simplify}(B, s) \in s$ and behaves classically, so all we have to prove is that B is s_1-strictly satisfiable iff s_1-simplify (B, s) is s_1-strictly satisfiable.

Second, if $p \notin s$ is an atom, then $v_s^1(\neg p) = \{0, 1\}$:

(1) $p \notin s \Longrightarrow 1 \in v_s^1(p) \Longrightarrow 0 \in v_s^1(\neg p)$;
(2) $p \notin s \Longrightarrow \neg p \notin s \Longrightarrow 1 \in v_s^1(\neg p)$.

It follows that if $p \notin s$ then $v_s^1(\neg p \vee \beta) = \{1\}$ iff $v_s^1(\beta) = \{1\}$. So let B' , be the result of deleting from B all negative atoms outside s; it follows that B' is s_1-strictly satisfiable iff B is.

Now consider B'' obtained by deleting from B' all clauses that contain positive atoms outside s; clearly, $B'' = s_1$-simplify (B, s). Let $\beta \in B' - B''$ be a clause of the form $q \vee \gamma$ with $q \notin s$. Suppose B'' is s_1-strictly satisfiable with valuation v_s^1 ; now alter v_s^1 so as $v_s^1(q) = \{1\}$, which is an allowed s_1-valuation. Then $v_s^1(\beta) = \{1\}$, and B' is s_1-strictly satisfiable. Conversely, suppose B' is s_1-strictly satisfiable; since $B'' \subseteq B'$, it is immediate that B' is s_1-strictly satisfiable, finishing the proof. ■

We can thus reduce s_1-entailment to a smaller instance of classical entailment, a result analogous to [18, Theorems 5.2 and 5.4].

THEOREM 4.2
Let $\alpha \in s$ be a clause. Then $B \models_s^1 \alpha$ iff s_1-simplify $(B, s) \models \alpha$.

PROOF. Just note that if $\alpha \in s$, then it behaves classically and for any classical v_p generating v_s^1 then $v_p(\alpha) = 0$ iff $v_s^1(\alpha) = \{0\}$. This fact together with Lemma 4.1 gives us the result. ■

From the result above we can then show that, when restricted to formulas in clausal form, we obtain the same complexity upperbound for s_1 that was obtained for S_1 in [18, Theorem 4.3].

THEOREM 4.3
There is an algorithm for deciding whether $B \models_S^1 \alpha$ in $O(|B|.|\alpha|.2^{|s|})$ time.

PROOF. It suffices to note that after applying the simplification, the only atoms left for which a valuation must be found are those belonging to the set s. ■

This means that for a fixed s, the decision of $B \models_S^1 \alpha$ is done in time proportional to $|B|.|\alpha|$, as is the case for S_1.

Since s_1 does not have to consider all clauses, usually for a given pair B and α, if $B \not\models \alpha$ the refutation can be shown with a smaller context set than would be needed for S_1. We can show that the context set needed for s_1 will be at most as large as the one for S_1.

THEOREM 4.4
Let S be a set of propositional variables such that for given $B \neq \emptyset$ and α in clausal form, $B \not\models_S^1 \alpha$. Then, for s $=$S , $B \not\models_s^1 \alpha$.

PROOF. The result is proved by looking at the simplification methods for S_1 and s_1. Consider the S_1 simplification method *simplify-1* (B,S), $B \neq \emptyset$, in which for each clause in B all literals containing atoms outside of S are deleted. It is proved in [18] that *simplify*-1$(B, S) \not\models \alpha$ iff $B \not\models_S^1 \alpha$.

Now let us compare *simplify-1*(B,S) and s_1-simplify(B, s) for $S = s$. If a clause $c \in B$ contains a negative literal $l \notin S$, l will be deleted from c in both methods. Now suppose $c \in B$ contains a positive literal $l \notin S$. According to the definitions given earlier, l will be deleted from c in *simplify*-1(B, S), but the whole clause c will be deleted in s_1-simplify(B, s). As a consequence, s_1-simplify$(B, s) \subseteq$ *simplify*-1(B, S) and hence, from classical logic, if *simplify-1*$(B, S) \not\models \alpha$, then s_1-simplify$(B, s) \not\models \alpha$. ■

According to the Definition 3.15, from the theorem above we have the following corollary.

COROLLARY 4.5
s_1 is more local than S_1.

The converse does not hold, as can be seen in the Example 4.6.

4.1 Examples

Here we examine some examples and compare s_1 to Cadoli and Schaerf's S_1. We have already seen that, unlike S_1 entailment, s_1 entailment truly approximates classical entailment from above.

The following example was presented in [18]:

EXAMPLE 4.6
We want to check whether $B \not\models \alpha$, where $\alpha = \neg\texttt{child} \vee \texttt{pensioner}$ and

$$\begin{aligned} B = \{\ & \neg\texttt{person} \vee \texttt{child} \vee \texttt{young} \vee \texttt{adult} \vee \texttt{senior}, \\ & \neg\texttt{adult} \vee \texttt{student} \vee \texttt{worker} \vee \texttt{unemployed}, \\ & \neg\texttt{pensioner} \vee \texttt{senior}, \\ & \neg\texttt{young} \vee \texttt{student} \vee \texttt{worker}, \\ & \neg\texttt{senior} \vee \texttt{pensioner} \vee \texttt{worker}, \\ & \neg\texttt{pensioner} \vee \neg\texttt{student}, \\ & \neg\texttt{student} \vee \texttt{child} \vee \texttt{young} \vee \texttt{adult}, \\ & \neg\texttt{pensioner} \vee \neg\texttt{worker}\}. \end{aligned}$$

Cadoli and Schaerf show that for $S = \{\texttt{child}, \texttt{worker}, \texttt{pensioner}\}$, $B \not\models^1_S \alpha$ and hence $B \not\models \alpha$. The set S, in this case, cannot be smaller, for that would delete all atoms in a clause in B, thus falsifying B, making it impossible to refute α.

However, in s_1 we can choose a smaller $s = \{\texttt{child}, \texttt{pensioner}\}$, only adding the atoms of α, and apply the s_1-simplify(B, s), obtaining $\{\neg\texttt{pensioner}\}$, from which we can clearly see that s_1-simplify$(B, s) \not\models \alpha$, so$B \not\models^1_s \alpha$ and hence $B \not\models \alpha$.

In terms of s_1-valuations, this corresponds to taking a propositional valuation v_p such that $v_p(\textit{pensioner}) = 0$ and $v_p(p) = 1$ for p any other propositional letter; the s_1-valuation obtained from v_p strictly satisfies every formula in B but does not relaxedly satisfy α.

In S_1 we would not obtain the same result for $S = \{\texttt{child}, \texttt{pensioner}\}$, for in this case several clauses in B would have no atom in S and would thus be falsified, yielding $B \models^1_S \alpha$.

This example shows that we can obtain an answer to the question of whether $B \not\models \alpha$ with a set s smaller than the set S needed for S_1. We used the smallest s possible, only containing the atoms of α.

Another concern was the fact that S_1 did not allow for local reasoning. Consider the following example, borrowed from [1]:

EXAMPLE 4.7
The following represents beliefs about a young student, Hans.

$B = \{\texttt{student}, \neg\texttt{student} \vee \texttt{young}, \neg\texttt{young} \vee \neg\texttt{pensioner}, \texttt{worker}, \quad -\texttt{worker} \vee \neg\texttt{pensioner}, \texttt{blue-eyes}, \texttt{likes-dancing}, \texttt{six-feet-tall}\}$.

We want to know whether Hans is a pensioner ($\alpha = \texttt{pensioner}$).

We have seen that in order to use Cadoli and Schaerf's S_1, we had to start with a set S containing at least one atom of each clause. This means that when we build S, we have to take into account even clauses which are completely irrelevant to the query, as likes-dancing.

However, using s_1 we can have a truly local context set. If we choose the smallest possible s={pensioner}, then s_1-simplify (B,s) = {¬pensioner}, from which we conclude that $B \not\models^1_s \alpha$ and hence $B \not\models \alpha$.

In this example, $B_\alpha = \{\mathtt{student}, \neg\mathtt{student} \vee \mathtt{young}, \neg\mathtt{young} \vee \neg\mathtt{pensioner}, \mathtt{worker}, \neg\mathtt{worker} \vee \neg\mathtt{pensioner}\}$ and $\mathrm{var}(B_\alpha) = \{\mathtt{student}, \mathtt{young}, \mathtt{pensioner}, \mathtt{worker}\}$. We see that for S_1, the context set must contain atoms which are not in $\mathrm{var}(B_\alpha)$, while for s_1, we can have a context set which is a subset of the relevant atoms.

4.2 Approximating model-based diagnosis from above

In this section, we explore an application of both s_1 and S_1 to the area of model-based diagnosis [7]. Approximate entailment has been applied to diagnosis in [20, 21]. Now we want to show how our system compares to S_1 in this domain.

The problem of diagnosis consists in, given an observation of an abnormal behavior, finding the components of the system that may have caused the abnormality [17].

In the area known as model-based diagnosis [7], a model of the device to be diagnosed is given in some formal language. For the sake of simplicity, we will adapt the definitions given in [17] to only mention formulas in the propositional language $\mathcal{L}_\mathcal{C}$. We follow the definitions and the motivating example from [22].

The systems to be diagnosed will be described by a set of propositional formulas. For each component X of the system, we use a propositional variable of the form okX to indicate whether the component is working as it should. If there is no evidence that the system is not working, we can assume that variables of the form okX are true.

Definition 4.8
A system is a pair (SD,ASS), where:

(1) SD, the system description, is a finite set of formulas of L and
(2) ASS, the set of assumables, is a finite set of propositional variables of the form okX.

An *observation* is a formula of L. We will sometimes represent a system by (SD,ASS,OBS), where OBS is an observation for the system (SD,ASS).

The need for a diagnosis arises when an abnormal behavior is observed, i.e., when SD∪ASS∪{OBS} is inconsistent. A diagnosis is a minimal set of assumables that must be negated in order to restore consistency.

Definition 4.9
A diagnosis for (SD,ASS,OBS) is a minimal set $\Delta \subseteq$ ASS such that:

$\mathrm{SD} \cup \{\mathrm{OBS}\} \cup \mathrm{ASS} \setminus \Delta \cup \{\neg \mathrm{okX} | \mathrm{okX} \in \Delta\}$ is consistent.

A diagnosis for a system does not always exist:

Proposition 4.10 [17]
A diagnosis exists for (SD,ASS,OBS) if and only if SD∪{OBS} is consistent.

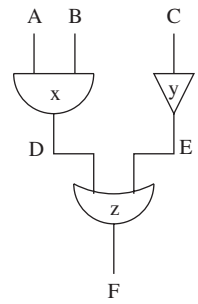

FIGURE 1. Circuit

Definition 4.9; can be simplified as follows:

PROPOSITION 4.11 [17]
The set $\Delta \subseteq$ ASS is a diagnosis for (SD,ASS,OBS) if and only if Δ is a minimal set such that SD $\cup$ { OBS} $\cup$ (ASS$\backslash\Delta$) is consistent.

Consider the circuit in Figure 1. The system description of this circuit is given by (SD,ASS), where[2]:

$$\begin{aligned}
\text{ASS} &= \{okX, okY, okZ\} \\
\text{SD} &= \{\neg A \vee \neg B \vee \neg okX \vee D, \\
&\quad A \vee \neg okX \vee \neg D, B \vee \neg okX \vee \neg D, \\
&\quad \neg C \vee \neg okY \vee \neg E, \\
&\quad C \vee \neg okY \vee E, \\
&\quad \neg D \vee \neg okZ \vee F, \\
&\quad \neg E \vee \neg okZ \vee F, \\
&\quad D \vee E \vee \neg okZ \vee \neg F\}
\end{aligned}$$

Suppose we have OBS $= \neg C \wedge \neg F$. This observation is inconsistent with SD $\cup$ ASS. According to Definition 4.9, there are two possible diagnoses: $\{okY\}$ and $\{okZ\}$, that is, either Y or Z is not working well.

Let us now examine the approximate results we can obtain applying S_1 or s_1. The idea is to try different sets S and s and apply simplify-1(SD,S) and s_1-simplify(SD,s) to check whether the simplified system description is consistent with the observation. If so, and if adding all the assumables gives rise to an inconsistency, then we can look for diagnoses.

Table 1 shows the result of applying simplify-1 to SD with different sets S. For the first seven options of S, the simplified SD is inconsistent, which means it is not possible to find a diagnosis. Note that in five of these cases the problem is that there are clauses with no atom is S. But in the third and seventh cases, even with at least one atom from each clause, the resulting set is inconsistent. In the eighth case, the system can find two possible diagnoses,

[2] The formulas in SD will be given clasual from so that S_1 can be used.

Table 1. S_1 diagnoses

S	Simplify-1(SD, S)	Diagnoses
$\emptyset$	$\{\bot\}$	None
$\{C, F\}$	$\{\bot, \neg C, C, F, \neg F\}$	None
$\{C, D, E, \mathrm{F}\}$	$\{D, \neg D, \neg \mathrm{C} \vee \neg E, C \vee E, \neg D \vee F, \neg E \vee F, D \vee E \vee \neg F\}$	None
$\{C, F, okZ\}$	$\{\bot, \neg C, C, \neg okZ \vee F, \neg okZ \vee \neg F\}$	None
$\{C, F, okY, okZ\}$	$\{\bot, \neg C \vee \neg okY, C \vee \neg okY, \neg okZ \vee F, \neg okZ \vee \neg F\}$	None
$\{C, E, F, okY, okZ\}$	$\{\bot, \neg C \vee \neg okY \vee \neg E, C \vee \neg okY \vee E, \neg okZ \vee F, \neg E \vee \neg okZ, E \vee \neg okZ \vee \neg F\}$	None
$\{C, D, E, F, okY, okZ\}$	$\{D, \neg D, \neg C \vee \neg okY \vee \neg E, C \vee \neg okY \vee E, \neg D \vee \neg okZ \vee F, \neg \mathrm{E} \vee \neg okZ \vee F, D \vee E \vee \neg okZ \vee \neg F\}$	None
$\{C, D, E, F, okX, okY, okZ\}$	$\{\neg okX \vee D, \neg okX \vee \neg D, \neg C \vee \neg okY \vee \neg E, C \vee \neg okY \vee E, \neg D \vee \neg okZ \vee F, \neg E \vee \neg okZ \vee F, D \vee E \vee \neg okZ \vee \neg F\}$	{okX, okY} and {okX, okZ}

Table 2. s_1 diagnosis

s	s_1-simplify (SD,s)	Diagnoses
$\emptyset$	$\{\bot\}$	None
$\{C,F\}$	$\{\neg C, F\}$	None
$\{C,D,E,F\}$	$\{D, \neg C \vee \neg E, C \vee E, \neg D \vee F, \neg E \vee F, D \vee E \vee \neg F\}$	None
$\{C,F,okZ\}$	$\{\neg C, \neg okZ \vee F\}$	{okZ}
$\{C,F,okY,okZ\}$	$\{\neg C \vee \neg okY, \neg okZ \vee F\}$	{okZ}
$\{C,E,F,okY,okZ\}$	$\{\neg C \vee \neg okY \vee \neg E, C \vee \neg okY \vee E, \neg okZ \vee F, \neg \mathrm{E} \vee \neg okZ \vee F\}$	{okZ}
$\{C,D,E,F,okY,okZ\}$	$\{D, \neg C \vee \neg okY \vee \neg E, C \vee \neg okY \vee E, \neg D \vee \neg okZ \vee F, \neg E \vee \neg okZ \vee F, D \vee E \vee \neg okZ \vee \neg F\}$	{okY} and {okZ}
$\{C,D,E,F,okX,okY,okZ\}$	$\{\neg okX \vee D, \neg okX \vee \neg D, \neg C \vee \neg okY \vee \neg E, C \vee \neg okY \vee E, \neg D \vee \neg okZ \vee F, \neg E \vee \neg okZ \vee F, D \vee \mathrm{E} \vee \neg okZ \vee \neg F\}$	{okY} and {okZ}

but it adds okX to them which is not needed in the classical case. In order to get the classical answer to the problem, we have to add A or B to S, but looking at the circuit in Figure 1 we see that both atoms are completely irrelevant for the problem.

Table 2 shows the results of applying s_1-simplify to SD using the same context sets which were used in Table 1. In the three first cases we cannot find any diagnoses. The first s yields and inconsistent simplified SD. In the second and third cases, the obtained set is consistent, but not consistent with the observation. In the fourth, fifth and sixth cases, we can find a partial answer and in the seventh case we already get the classical answer. Note that, in order to obtain the correct answer, we did not have to add to s any atom appearing only in the three first clauses of SD, which are intuitively irrelevant for the problem.

4.3 Refutations via s_1-approximations

We note that $B \not\models \alpha$ iff $B \wedge \neg\alpha$ is classically satisfiable. Then, Theorem 4.2 suggests an algorithm based on s_1-approximation for clausal theorem refutation, as a sequence of applications of (any) SAT algorithm to simpler problems:

ALGORITHM 4.12
Input: A set of clauses B and a clause α
Output: $\top$, if$B \not\models \alpha$; $\bot$, otherwise.
$s := atoms(\alpha)$;
while $s \neq letters(B)$
if SAT(s_1- simplify$(B, s) \wedge \neg\alpha) = \top$, return $\top$;
Expand s with atoms in B;
return $\bot$;

The expansion of s can be done with several heuristics. For instance, a conservative strategy would add atoms to s so as to include the smallest number of clauses. In the first iteration, this expansion must cope with the fact that a clause containing only negative literals must have at least one atom in s. Algorithm 4.12; becomes an approximation of SAT itself by making $\alpha = \bot$.

If we inspect with detail the nature of the approximation made in Algorithm 4.12;, we are dealing, in classical terms, with an approximation that fixes all atoms $p \notin s$ to a valuation $v(p) = 1$.

A dual approximation algorithm could then be obtained, by fixing $v(q) = 0$ for $q \notin s$. Let us call it the $\bar{\mathsf{s}}_1$ algorithm. Note that $\bar{\mathsf{s}}_1$ is also an approximation of classical logic from above. It restricts the number of possible valuations for formulas, thus allowing for more valid formulas. In this way, any classical theorem, which is satisfied by all classical valuations, will also be satisfied by $\bar{\mathsf{s}}_1$ -valuations which are a restriction of classical valuations.

Finally, a combination of s_1 and $\bar{\mathsf{s}}_1$ valuations can be constructed. In such a setting, the approximation parameter consists of a partition of the set of propositional atoms into three sets, $\langle s, s', s''\rangle$. As usual, the letters in s behave classically and accept any valuation. Atoms in s' have their valuations fixed to 1; similarly, atoms in s'' have their valuation fixed to 0. We start the approximation process with $s = \emptyset$. The approximation step consists of *moving* atoms from s' and or s'' into s. When $s' = s'' = \emptyset$ we are in classical logic. Since each step is a restriction on classical valuations, this system also provides an approximation of classical logic from above. This idea is close to work by Massacci [12], which will be described in Section 6.2.

Unfortunately, a complete investigation on such systems is outside the scope of this work. We return now to analyse the full propositional language of s_1, and present a sound and complete tableau-based proof system for it.

5 Tableaux for s_1

In this section, we turn back to the full propositional language and present a proof system for s_1-based on KE-tableaux, which will also provide a heuristic for approximations.

KE-tableaux were introduced by D'Agostino [2] as a principled computational improvement over Smullyan's Semantic Tableaux [19].

KE tableaux deal with T- and F-signed formulas: $T\ \alpha$ and $F\ \alpha$. For each connective, there are at least one T- and one F-linear expansion rules. Linear expansion rules always have a *main premise*, and may also have an auxiliary premise. They may have one

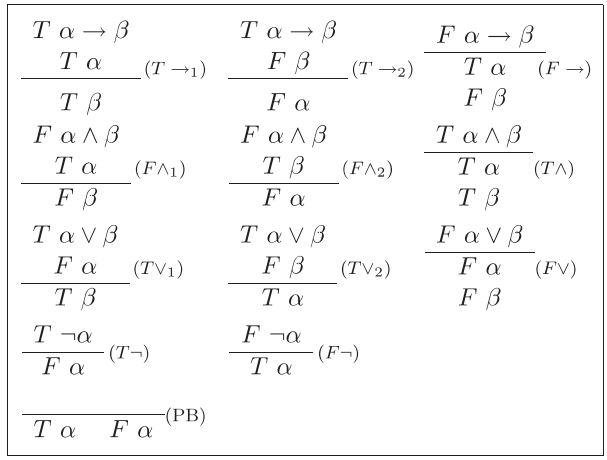

FIGURE 2. KE-rules for classical logic

or two consequences. The only branching rule is the *Principle of Bivalence*, stating that something cannot be true and false at the same time. Figure 2 shows KE-tableau expansion rules for classical logic.

KEs_1-Tableaux extend classical KE-tableaux. Besides employing signed formulas with the usual T and F signs, KEs_1-tableaux uses two new signs: 1, representing s_1-formulas whose valuations contain the value 1, and 0, representing s_1-formulas whose valuations contain the value 0.

The basic idea of the s_1 extension to KE-tableaux is that:

$$T\ \alpha \Longrightarrow v(A) = \{1\}$$
$$F\ \alpha \Longrightarrow v(A) = \{0\}$$
$$1\ \alpha \Longrightarrow 1 \in v(A)$$
$$0\ \alpha \Longrightarrow 0 \in v(A)$$

These four signs are not mutually exclusive, for any formula that receives the T-sign may also receive the 1-sign; and similarly, any formula that receives the F-sign may also receive the 0-sign.

The new collection of connective rules for KEs_1-tableau is illustrated in Figure 3. Note that some classical rules, such as $(T\to_{1,2})$, $(F\wedge_{1,2})$ and $(T\vee_{1,2})$ are missing, but can be derived from those in Figure 3. In this respect, KEs_1 is an extension of classical KE tableaux.

As in the classical KE rules, we have one-premised and two-premised rules. In one-premised rules, the premise is the rule's *main formula*; in two-premised rules, the top formula is the *main formula* and the other is the *auxiliary formula*. Main formulas are always compound signed formulas, but auxiliary formulas can be atomic.

In the KEs_1-rules of Figure 3, if we replace 1 by T and 0 by F we obtain the classical rules. In fact, such a replacement is allowed in a tableau when the formulas within the scope of the signs 1 and 0 behave classically. This motivates the addition to KEs_1-tableaux of the *Promotion Rules* in Figure 4.

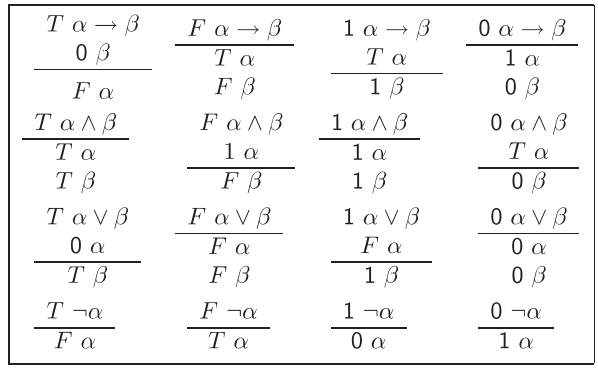

FIGURE 3. Connective rules for KEs_1-rules

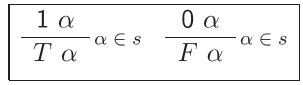

FIGURE 4. KEs_1 Promotion rules

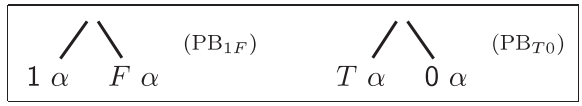

FIGURE 5. KEs_1 Branching rules based on the principle of Bivalence

Because KEs_1-tableau have four kinds of signs, we need actually to refine the branching rule corresponding to the *Principle of Bivalence*. We now have two rules to cope for that, (PB_{1F}) and (PB_{T0}), as illustrated in Figure 5.

Note that, as before, if we replace 1 by T and 0 by F the two branching rules collapse into the classical one.

Also, due to the presence of four kinds of formula signs, we have new ways of closing a tableau branch. Besides the usual contradictory pair $T\ \alpha$ and $F\ \alpha$, a KEs_1-branch can also *(strongly)* close with the pair $1\ \alpha$ and $F\ \alpha$, and with the pair $0\ \alpha$ and $T\ \alpha$:

$$\begin{array}{ccc} T\alpha & 1\alpha & 0\alpha \\ F\alpha & F\alpha & T\alpha \\ \times & \times & \times \end{array}$$

With this respect, we say that the formulas $T\ \alpha$ and $0\ \alpha$ are *opposite* formulas. Similarly, $F\ \alpha$ and $1\ \alpha$ are also *opposite* formulas.

There is yet another branch closure rule in KEs_1-tableaux. This rule differs from the previous ones in that it is *defeasible*. That is, for a particular logic $s_1(s)$, this rule closes a branch:

$$\begin{array}{c} F\ \alpha \quad \alpha \notin s \\ \hline \end{array}$$

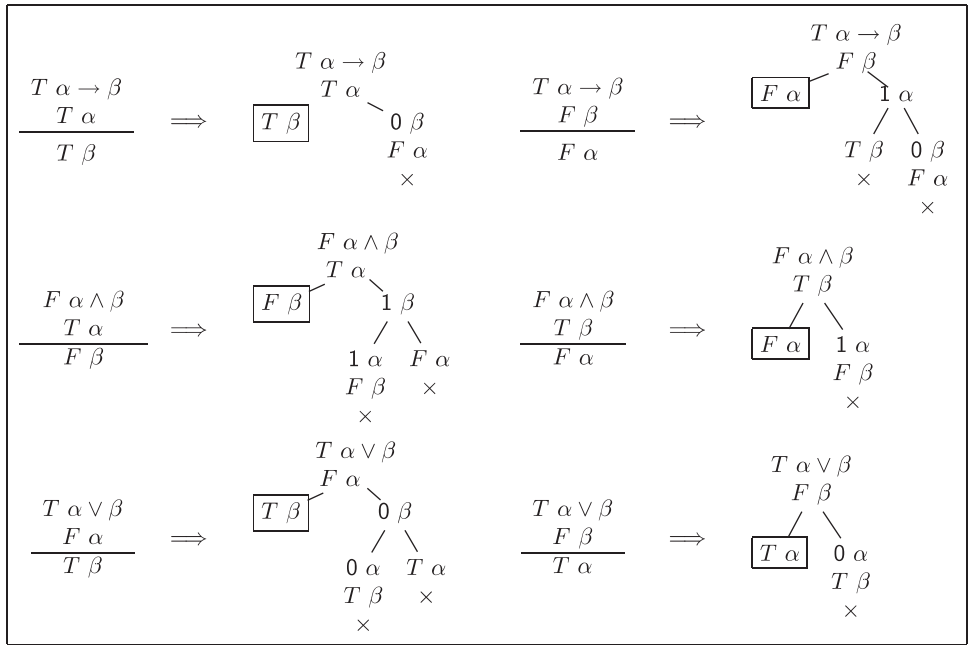

FIGURE 6. Derivation of classical KE rules in KEs_1

We use the symbol '$\times$' to represent strong closure and '$-$' to represent defeasible closure. It is sufficient for a branch to have a formula $F\,\alpha$ with $\alpha \notin s$ for it to defeasibly close. However, when we increment the size of s by adding new propositional symbols to it, we have shifted logics from $s_1(s)$ to $s_1(s')$, with $s \subset s'$. The process of enlarging s will reopen some branches, so in $s_1(s')$, if $\alpha \in s'$, the branch is not closed anymore by the *Defeasible Closure Rule*.

A KEs_1-tableau *closes* for a given set s if all its branches are strongly or defeasibly closed. It is not difficult to see that if all the branches are strongly closed, then the tableau also closes classically. Just like in classical closed branches, we do not apply any more rules to a defeasibly closed branch, at least until it is reopened; see discussion on reopening branches ahead. Thus, the closing rules should be applied before all others.

A KEs_1-tableau for the sequent $\alpha_1, \ldots, \alpha_n \vdash \beta$ is a tree whose initial nodes are $T\,\alpha_1, \ldots, T\,\alpha_n, F\beta$ and whose subsequent nodes are obtained using one the s_1-rules above. If a KEs_1-tableau closes for s we write $\alpha_1, \ldots, \alpha_n \vdash^1_s \beta$.

As an initial example of the uses of KEs_1, we derive in Figure 6 the classical KE-tableau rules. Note that none of the derivations in Figure 6 depends on the elements of the parameter set s.

5.1 *Anytime sequent refutations or the incrementality of KEs_1-approximations*

The KEs_1-tableau provides an incremental method for approximating classical logic from above. It is the defeasible closure rule that will guide the expansion of s, thus generating

an *incremental procedure* to disprove a sequent $\alpha_1, \ldots, \alpha_n \vdash^1_s \beta$. The procedure of moving from $s_1(s)$ to $s_1(s')$, with $s \subset s'$, in the construction of a KEs_1-tableau is the following:

(1) Expand the tableau for $\alpha_1, \ldots, \alpha_n \vdash^1_s \beta$ until it closes or there is a *saturated* open branch; the definition of saturation is given below in Section 5.2.
(2) If there is a saturated open branch that cannot be expanded, then the sequent has been disproved, i.e., $\alpha_1, \ldots, \alpha_n \nvdash \beta$.
(3) If the tableau is closed and there are no defeasibly closed branches, then tableau is classically closed and the sequent $\alpha_1, \ldots, \alpha_n \vdash \beta$ is classically valid.
(4) If there are defeasibly closed branches, choose one such branch such that it contains $F\,\alpha$ with $\alpha \notin s$.
(5) Expand s with the atoms of α. We have thus moved to a logic "closer" to classical logic, with a larger set of formulas that behave classically.
(6) Go to 1.

It may be that for $s \subset \mathcal{P}$ we already have an open branch of we have strongly closed the KEs_1-tableau. If no such situation occurs, eventually, all atoms in the sequent will be in s, so the tableau will be a classical one, and we are guaranteed that either the tableau will close or have an open saturated branch.

This process also provides an *anytime* refutation procedure. The process of approximation proceeds normally. If a solution is found, no problem. Otherwise, at some point we may decide to stop it, possibly because time or memory runs out. At this point, we are in logic $s_1(s)$, and we know that for all the logics up to $s_1(s)$, $\alpha_1, \ldots, \alpha_n \nvdash^1_s \beta$, which consists an *anytime answer* for the classical problem $\alpha_1, \ldots, \alpha_n \vdash^? \beta$. If more time or memory is provided, we may eventually find the exact, that is, classical answer.

With regards to the complexity of this method with respect to classical logic, the method above is as complex as a classical one in the worst case scenario, that is, the problem of deciding the invalidity of a sequent remains NP-complete. Note, however, that at each step, due to the defeasible closure of rules, the search space for a saturated open branch is potentially smaller in the method above, which may lead to faster proofs in some cases.

Here are some examples of this incremental method:

EXAMPLE 5.1
We want to check whether $p \rightarrow q, q \vdash^1_s p$.

1.	$T\ p \rightarrow q$	by hypothesis, $s = \emptyset$
2.	$T\ q$	by hypothesis
3.	$F\ p$	by hypothesis
4.	$-$	defeasible closure from 3.
	$/\ \backslash$	Reopen with $s = \{p\}$
5.	$0\ q \quad T\ q$	PB_{0T}
6.	$F\ p$	by rule $(0 \rightarrow)$ on 1 and 5

There is no KEs_1-rule which can be applied to the saturated branches, so both branches remain open. The tableau does not close, which means that $p \rightarrow q, q \nvdash p$.

As usual, an open branch gives us a valuation that refutes the initial sequent. Both branches give us $v^1_s(q) = \{1\}, v^1_s(p) = \{0\}$, which is a classical valuation.

EXAMPLE 5.2
We want to check whether $p \to q, q \to r \wedge t \vdash^1_s r$.

We will use the rule $(T\to_F)$ derived in Figure 6:

$$\frac{T\alpha \to \beta \quad F\,\beta}{F\,\alpha}$$

We have the following tableau:

	1. $Tp \to q$		by hypothesis, $s = \emptyset$
	2. $Tq \to r \wedge t$		by hypothesis
	3. Fr		by hypothesis
	$-$		reopen with s $= \{r\}$
$4'.$ $Tr \wedge t$			$4''.$ $0\ r \wedge t$ PB_{T0}
$5'.$ Tr	$(T\wedge)$ in $4'$		$5''.$ $F\ q$ $(T\to)$ in $2, 4''$
$\times$			$6''.$ $-$ reopen with $s = \{q, r\}$
			$7''.$ $F\ p$ $(T \to F)$ in $1, 5''$
			$8''.$ $-$ reopen with $s = \{p, q, r\}$

The right branch defeasibly closes three times. When it last reopens with an expansion of s, it is saturated, so we conclude that $p \to q, q \to r \wedge t \not\vdash r$. The open branch gives us the refuting valuation $v^1_s(p) = v^1_s(q) = v^1_s(r) = \{0\}$, which is also a classical valuation, that refutes the classical tableau. This tableau illustrates well the incrementality of the method of approximations, for the context set s had to be expanded three times.

In the following, we prove that KEs_1-tableaux are sound and complete with respect to the semantics of s_1.

5.2 Soundness

A branch in a KEs_1-tableau is *saturated* if:

(1) every possible linear rule has been applied; and
(2) for every main formula of a linear two-premised rule that cannot be the subject of a promotion, the branch contains either the corresponding auxiliary formula or an opposite of it.

A branch is *partly expanded* if it is not saturated. KEs_1-tableau is saturated if every branch is. A branch is open if it is neither strongly nor defeasibly closed.

A table is *saturated* if every branch is either closed or open saturated. To saturate a branch, it suffices to employ a *branching heuristics*:

> For each compound formula α such that $X\ \alpha$ occurs in a KEs_1-tableau and $X\ \alpha$ is the non-promotable main premise of a two-premised rule, one should branch over a formula $Y\beta$ such that it is the auxiliary premise for the two-premised rule.

It is clear that such a branching heuristics will lead to branches that either close or get saturated. Note that it does not mean that all instances of branching must be according to that rule. But its application guarantees that the tableau will eventually saturate.

Given a s_1-valuation v, we extend it to signed formulas so that $v(X\alpha)$ is either 0 or 1, in the following way:

$$\begin{array}{lll} v(T\alpha)=1 & \text{iff} & v(A)=\{1\} \\ v(F\alpha)=1 & \text{iff} & v(A)=\{0\} \\ v(1\alpha)=1 & \text{iff} & 1 \in v(A) \\ v(0\alpha)=1 & \text{iff} & 0 \in v(A) \end{array}$$

A branch Θ is satisfied if there exists a s_1-valuation v such that for every signed formula $X\alpha \in \Theta$, $v(X\alpha)=1$. In this case we write $v(\Theta)=1$.

LEMMA 5.3
With respect to the expansion rules in Figures 3 and 4 we have the following properties:

(1) For one-premised rules, if a s_1-valuation v satisfies its main formula then it satisfies its consequences.
(2) For two-premised rules, if the premises are satisfied by v so is the consequence.
(3) If the premise of a promotion rule is satisfied by v and the formula is in s, then the consequence is also satisfied.

PROOF. By inspection in all linear rules in Figures 3 and 4. Details omitted. ■

LEMMA 5.4
Let Θ_1 and Θ_2 be the expansions of branch Θ using one of the branching rules PB_{T0} or PB_{1F}. Let v be a s_1 valuation such that $v(\Theta)=1$. Then either $v(\Theta_1)=1$ or $v(\Theta_2)=1$.

PROOF. Simply note that PB_{T0} and PB_{1F} are mutually excludent, and for any formula α and any s_1-valuation v, v must satisfy either $T\,\alpha$ or $0\,\alpha$, and v must satisfy either $F\,\alpha$ or $1\,\alpha$. ■

THEOREM 5.5 (Soundness)
If $\alpha_1, \ldots, \alpha_n \vdash^1_s \beta$ then $\alpha_1, \ldots, \alpha_n \models^1_s \beta$.

PROOF. We show the contrapositive. Suppose $\alpha_1, \ldots, \alpha_n \not\models^1_s \beta$. Then there is a s_1-valuation v such that $v(\alpha_1)= \ldots = v(\alpha_n)=\{1\}$ but $v(\beta)=\{0\}$. That is, v satisfies the initial tableau for $T\alpha_1, \ldots, T\alpha_n, F\,\beta$. By Lemma 5.3, every linear expansion generates only signed formulas satisfied by v, and by Lemma 5.4 one of the branches remains satisfied by v after the application of PB. Therefore, we must always have at least one open branch in this tableau, for closed branches cannot be satisfied by any v. Therefore, $\alpha_1, \ldots, \alpha_n \not\vdash^1_s \beta$, as desired. ■

5.3 *Completeness*

We first start showing the converse of Lemma 5.3.

LEMMA 5.6
With respect to the expansion rules in Figures 3 and 4 we have the following properties:

(1) For one-premised rules, if a s_1-valuation v satisfies its consequences then it satisfies its main formula.
(2) For two-premised rules, if the opposite of an auxiliary premise is satisfied by v, so is the main premise.
(3) For two-premised rules, if the auxiliary premise and the consequence are satisfied by v, so is the main premise.

(4) If the consequence of a promotion rule is satisfied by v, so is the premise.

PROOF. By inspection in all linear rules in Figures 3 and 4. Details omitted. ■

Let Θ be a saturated open branch of a KE_1-tableau for the sequent $\alpha_1, \ldots, \alpha_n \vdash \beta$. Construct the *branch valuation* v_Θ in the following way. For each atomic formula $X\,p \in \Theta$:

(1) if $Tp \in \Theta$ then $v_\Theta(p)=\{1\}$.
(2) if $Fp \in \Theta$ then $v_\Theta(p)=\{0\}$.
(3) if $0p \in \Theta$ then
    in case $1p \notin \Theta$ and $p \in s$, $v_\Theta(p)=\{0\}$;
    otherwise $v_\Theta(p)=\{0,1\}$.
(4) if $1\ p \in \Theta$ and $0\ p \notin \Theta$, then $v_\Theta(p)=\{1\}$.

LEMMA 5.7

(1) The branch valuation v_Θ is a s_1-valuation.
(2) If Θ is a saturated open branch of a KE_1-tableau, then $v_\Theta\ (\Theta)=\{1\}$.

PROOF.

(1) Note that, since Θ is open, v_Θ cannot be contradictory, nor can it assign $\{0\}$ to a formula not in s. Also note that the rules of all conditions for a s_1-propositional valuation are met by the definition above.
(2) We show by induction on the structure of the formula α such that $X\alpha \in \Theta$, that $v_\Theta(X\,\alpha)=\{1\}$. If α is propositional, then $v_\Theta(X\,\alpha)=\{1\}$. Suppose α is a compound formula. Then we have three possibilities:
    α is the main formula of a one-premised rule. Then, by saturation, its consequences are in Θ and are of lower complexity so that, by the induction hypothesis, they are satisfied by v_Θ. So by Lemma 5.6(a) $v_\Theta(X\,\alpha)=\{1\}$.
    α is the main formula of a two-premised rule. By saturation, Θ contains either the auxiliary premise of a rule or the opposite of one. If the former case, by saturation we have that the consequence of the rule is in Θ. Since both the auxiliary premise and the consequence are of lower complexity, it follows from the induction hypothesis that they are satisfied by v_Θ. Lemma 5.6(c) then yields $v_\Theta(X\,\alpha)=\{1\}$. In the latter case, by induction hypothesis by, v_Θ satisfies the opposite of an auxiliary premise, so Lemma 5.6(b) we have that $v_\Theta(X\,\alpha)=\{1\}$.
    $X\,\alpha$ was promoted. In this case, let $Y\,\alpha$ be the result of its promotion. By the previous two cases, we must have $v_\Theta(Y\,\alpha)=\{1\}$, so by Lemma 5.6(d) we obtain $v_\Theta(X\,\alpha)=\{1\}$. ■

THEOREM 5.8 (Completeness)
If $\alpha_1, \ldots, \alpha_n \models_s^1 \beta$ then $\alpha_1, \ldots, \alpha_n \vdash_s^1 \beta$. In particular, all saturated KE_1-tableaux for $\alpha_1, \ldots, \alpha_n \vdash_s^1 \beta$ must close.

PROOF. Suppose $\alpha_1, \ldots, \alpha_n \models_s^1 \beta$. Then every valuation v such that $v(\alpha_1)= \ldots = v(\alpha_n)=\{1\}$ it must be the case that $1 \in v(\beta)$. Now suppose there is a saturated tableau for $T\,\alpha_1, \ldots, T\,\alpha_n, F\,\beta$ with an open branch. By Lemma 5.7, there is a valuation v such that $v(\alpha_1)= \ldots = v(\alpha_n)=\{1\}$ but $v(\beta) = \{0\}$, which is a contradiction. So we cannot have a saturated KE_1-tableaux with an open branch and therefore $\alpha_1, \ldots, \alpha_n \vdash_s^1 \beta$. ■

6 Related systems

In this section, we present three related systems and compare them to s_1.

6.1 Kleene 3-valued logic

Kleene [8] has introduced a three-valued logic, given by the truth tables in Figure 7.

The tables are an extension of the truth tables for classical logics, in the sense that when we delete the 'u' rows and columns, we get the classical tables. His main motivation was to extend classical logic in order to formalize a decision procedure for partial recursive predicates. The symbol 'u' stands for the value 'undefined'.

LEMMA 6.1
Fix a Kleene valuation v_K and let the parameter set s be such that $\mathcal{P} - s$ coincides with Kleene's and **u**-valued propositional letters. Consider a valuation v_s^1 such that for every $p \in \mathcal{P}$, $v_s^1(p) = \{1\}$ iff $v_K = \mathbf{t}$, $v_s^1(p) = \{0\}$ iff $v_K = \mathbf{f}$ and $v_s^1(p) = \{0, 1\}$ iff $v_K = \mathbf{u}$. Then the $s_1(s)$ valuation coincides with Kleene's logic under that same interpretation.

PROOF. This setting makes $v_s^1(p) = \{0, 1\}$ iff $p \notin s$. Then the proof is done by a simple inspection. For example, let us examine the rules for the $\vee$ connective; when A and B are both either **t**- or **f**-valued, both systems behave classically; when $v_s^1(A) = \{1\}$ $(= \mathbf{t})$, then $v_s^1(A \vee B) = \{1\}$ $(= \mathbf{t})$, and analogously for $v_s^1(B) = \{1\}$; finally, when $v_s^1(A) = \{0, 1\}$ $(= \mathbf{u})$ and $0 \in v_s^1(B)$ $(= \mathbf{u}$ or $= \mathbf{f})$, then $v_s^1(A \vee B) = \{0, 1\}$ $(= \mathbf{u})$, and analogously for $v_s^1(B) = \{0, 1\}$ $(= \mathbf{u})$ and $0 \in v_s^1(A)$ $(= \mathbf{u}$ or $= \mathbf{f})$, so all cases for $\vee$ are covered.

Details are omitted for the other connectives. ■

The converse of the Lemma above does not immediately hold, however. Indeed, suppose we take $s = \{q\}$ and a s_1-valuation v_s^1 such that $v_s^1(p) = \{1\}$ and $v_s^1(q) = \{0\}$. The corresponding Kleene valuation would be $v_K(p) = \mathbf{t}$ and $v_K(q) = \mathbf{f}$. However, since $p \wedge q \notin s$, we would have $v_s^1(p \wedge q) = \{0, 1\}$ but $v_K(p \wedge q) = f$.

If on the other hand, we accepted only s_1-valuations v_s^1 that for all $p \notin s$ $v_s^1(p) = \{0, 1\}$, then there would be a perfect match between s_1 and Kleene semantics, as can be seen by the same inspection as above.

6.2 Massacci's approximations

Perhaps the work in the literature that is closer to ours in aims and techniques is the work on Approximate Deduction by Fabio Massacci [10, 12] whose approach dealt not only with propositional logic by also with modal logic approximations [11].

$A \vee B$

	t	**f**	**u**
t	t	t	t
f	t	f	u
u	t	u	u

$A \wedge B$

	t	**f**	**u**
t	t	f	u
f	f	f	f
u	u	f	u

$A \rightarrow B$

	t	**f**	**u**
t	t	t	t
f	f	t	u
u	u	t	u

$\neg A$

t	f
f	t
u	u

FIGURE 7. Kleene's three-valued logic

In Massacci's approach, the set of propositional letters $\mathcal{P}$ was partitioned into three sets:

(1) The set of *interesting* propositions, int($\mathcal{P}$), which corresponds to our notion of context set s, i.e., the set of propositional letters whose behaviour is classical.
(2) The set of *unknown* propositions, unk($\mathcal{P}$), for which both the proposition and its negation are false; this set correspond, in Schaerf's and Cadoli's system $S_1(S)$, to the propositions in $\mathcal{P} - S$.
(3) The set of *incoherent* propositions, inc($\mathcal{P}$), for which both the proposition and its negation are true; this resembles (but is not an exact match) the set $\mathcal{P} - \mathcal{S}$ of Schaerf's and Cadoli's system $S_3(S)$.

Despite the fact that the set of propositional letters is partitioned in three, for the approximation process it was only considered the case where either $\text{unk}(\mathcal{P}) = \emptyset$ or $\text{inc}(\mathcal{P}) = \emptyset$. This latter case corresponds to an approximation useful for refuting theorems, so in what follows we will consider always $\text{inc}(\mathcal{P}) = \emptyset$. This corresponds to what Schaerf and Cadoli called an approximation that is *unsound but complete*, for not all theorems are classical ones, but all classical theorems are contemplated. In this sense, any approximation from above is unsound but complete.

Massacci's system starts considering a propositional interpretation, $\mathcal{I}$, which is simply what we called here a propositional valuation.[3] The semantics of the system is then developed for *signed formulas* only (not for pure formulas). If A is a propositional formula, there are two signed versions of it, $\mathtt{t}.A$ and $\mathtt{f}.A$, respectively the statement of A's truth and falsity. The semantics is worked up inductively, so if $p \in \mathcal{P}$ we have that:

$$\begin{array}{lll} \mathcal{I} \Vdash \mathtt{t}.p & \text{iff} & p \in \text{int}(\mathcal{P}) \text{ and } (p)I = 1 \\ \mathcal{I} \Vdash \mathtt{f}.p & \text{iff} & p \in \text{int}(\mathcal{P}) \text{ and } (p)I = 0 \end{array}$$

This means that, for $p \in \text{unk}(\mathcal{P})$, neither $\mathcal{I} \Vdash \mathtt{t}.\ p$ nor $\mathcal{I} \Vdash \mathtt{f}.\ p$. The signed formulas are then divided into α and β formulas, in the fashion of Smullyan [19]:

α	α_1	α_2
$\mathtt{t}.A \wedge B$	$\mathtt{t}.A$	$\mathtt{t}.B$
$\mathtt{f}.A \vee B$	$\mathtt{f}.A$	$\mathtt{f}.B$
$\mathtt{f}.A \rightarrow B$	$\mathtt{t}.A$	$\mathtt{f}.B$
$\mathtt{t}.\neg A$	$\mathtt{f}.A$	$\top$
$\mathtt{f}.\neg A$	$\mathtt{t}.A$	$\top$

β	β_1	β_2
$\mathtt{f}.A \wedge B$	$\mathtt{f}.A$	$\mathtt{f}.B$
$\mathtt{t}.A \vee B$	$\mathtt{t}.A$	$\mathtt{t}.B$
$\mathtt{t}.A \rightarrow B$	$\mathtt{f}.A$	$\mathtt{t}.B$

Then the semantics of signed formulas can be simply put as:

$$\begin{array}{lll} \mathcal{I} \Vdash \alpha & \text{iff} & \mathcal{I} \Vdash \alpha_1 \text{ and } \mathcal{I} \Vdash \alpha_2 \\ \mathcal{I} \Vdash \beta & \text{iff} & \mathcal{I} \Vdash \beta_1 \text{ or } \mathcal{I} \Vdash \beta_2 \end{array}$$

With this definition at hand, the notion of an unknown formula can be extended to all signed formulas. A signed propositional formula is unknown if the proposition is. An α

[3] We try to use Massacc's notation when describing his system.

formula is unknown if α_1 is unknown or if α_2 is unknown; a β formula is unknown if both β_1 and β_2 are unknown.

As a consequence, we have the following:

PROPOSITION 6.2
Let ψ be a signed formula. If all its propositional letters are unknown, then ψ is unknown.

It is interesting to note that in our system, it is enough that one propositional letter be outside s for the whole formula to be outside s.

In the notation of [12], B is an *approximate logical consequence* of U, $\models U \Rightarrow B$ where U is a set of formulas and B a formula, if for every interpretation $\mathcal{I}$ such that $\mathcal{I} \Vdash \mathtt{t}.A_i$ for each $A_i \in U$, then $\mathcal{I} \nVdash \mathtt{f}.B$.

With this semantics based not on plain formulas but on `f`- and `t`-marked formulas, the next obvious step is to define a tableau inference system for approximate reasoning. Such system is based on Smullyan's formulation of semantic tableaux [19] with `f`- and `t`-marked formulas, leading to a tableau system that is simpler than the KEs_1-tableau. It is notable that such tableau system contains a closing rule that is similar to our *defeasible* closure *rule*, which may be used as a basis for incremental heuristics in approximate reasoning, in a similar fashion to the incremental method of KEs_1-tableaux.

However, we have to point the following property of Massacci's formulation of the approximation process.

LEMMA 6.3
Consider a partition of $\mathcal{P}$ in interesting and unknown formulas and a classical sequent $A_1, \ldots, A_n \vdash B$ that we want to refute. Then the approximation process is non-local.

PROOF. This is a generalization of the argument that shows that $S_1(S)$ is non-local in Section 3.2. By Proposition 6.2, if some A_i has no interesting letters, $\mathcal{I} \nVdash \mathrm{t}.A_i$ for any $\mathcal{I}$. So the set $\mathrm{int}(\mathcal{P})$ must contain at least one propositional symbol of each A_i, even if A_i has no relevance to the refutation of B. ■

The fact that Massacci's approximations are non-local comes as no surprise, for Massacci claims that his system contains Schaerf's and Cadoli's systems as an instantiation. The fact that we could not extend S_1 and avoid the non-locality lies in the heart of our attempt to propose a different approximate system for refuting sequents.

6.3 *Probabilistic inference*

In this section we present an inference system based on probabilistic logic that also approximates classical logic from above. The difference, however, is that such system may approximate the refutation of a sequent in infinitely many steps, while s_1 reaches classical logic in finitely many approximation steps.

Several probabilistic logics have been proposed in the literature [13–16]. The current presentation is based on the probabilistic entailment of Knight [9].

Let $P : \mathcal{L} \rightarrow [0, 1]$ be a function that associates propositional formulas with real values in the interval [0,1]. Let $\models$ be classical entailment. We say that P is an *(admissible) probability function* over formulas if the following properties hold:

(1) if $\models \alpha$ then $P(\alpha) = 1$;
(2) if $\models \neg (\alpha \wedge \beta)$ then $P(\alpha \vee \beta) = P(\alpha) + P(\beta)$.

Several properties follow from this definition [15], among which:

(1) $P(\neg\,\alpha) = 1 - P(\alpha)$;
(2) if $\alpha \models \beta$ then $P(\alpha) \leq P(\beta)$;
(3) if $\models \alpha \leftrightarrow \beta$ then $P(\alpha) = P(\beta)$;
(4) $P(\alpha \vee \beta) = P(\alpha) + P(\beta) - P(\alpha \wedge \beta)$.

With this notion of probability function over formulas — as opposed to a valuation function—a new definition of probability entailment is possible, $\lhd_x$, parameterized by a probability threshold value $x \in [0,1]$:

We say that a formula $\alpha_1, \ldots, \alpha_m$ probably entails β with probability x, represented by $\alpha_1, \ldots, \alpha_m \lhd_x \beta$, if whenever $P(\alpha_1) = \ldots = P(\alpha_m) = 1$ then $P(\beta) \geq x$.

The original formulation of [Kni03] is more general. It defines a $\vec{y}, x$-entailment such that $\alpha_1, \ldots, \alpha_m \stackrel{\vec{y}}{\lhd}_x \beta$, if whenever $P(\alpha_i) \geq y_i$ for $1 \leq i \leq m$ then $P(\beta) \geq x$. Here we have fixed all $y_i = 1$, and thus operate on a restricted version of Knight's system. As a consequence, the inference system presented in [Kni03] does not correspond to $\lhd_x$.

It has been shown in [Kni03] that the probabilistic inference $\stackrel{\vec{y}}{\lhd}_x$, and thus $\lhd_x$, is effectively decidable; however, the decision procedure presented there has exponential complexity.

Some of the immediate properties of the $\lhd_x$-entailment are the following.

Proposition 6.4

(1) if $x = 1$, then $\lhd_x$ is classical entailment;
(2) if $x = 0$, then for any $\alpha_1, \ldots, \alpha_m, \beta$ we have that $\alpha_1, \ldots, \alpha_m \lhd_x \beta$.
(3) if $x_i \leq x_j$ then $\lhd_{x_i} \supseteq \lhd_{x_j}$.

From Proposition 6.4 above we obtain that if

$$0 \leq x_1 \leq \ldots \leq x_n \leq \ldots \leq 1$$

then,

$$\lhd_0 \supseteq \lhd_{x_1} \supseteq \ldots \supseteq \lhd_{x_n} \supseteq \ldots \supseteq \lhd_1 = \models$$

Since the sequence $0, x_1, \ldots, x_n, \ldots, 1$ is possibly infinite, then we have constructed a possibly infinite approximation from above of a classical entailment $\alpha_1, \ldots, \alpha_m \models \beta$. Note that, in theory, this approximation chain can be uncountable; however, for a fixed antecedent B and $x_i \neq x_j$, the sets $\{\beta | B \lhd_{x_i} \beta\}$ and $\{\gamma | B \lhd_{x_j} \gamma\}$ can differ only in countably many formulas, in such a way that the number of distinct approximation steps must be countable.

It is not known if there is a sound and complete inference system for each $\lhd_{x_i}$ nor if there is an efficient decision procedure for $\lhd_{x_i}$, $x_i \in [0,1]$. The possibility of an infinite approximation sequence is just a theoretical curiosity. In practice, one would, for example, divide the interval $[0,1]$ into k elements and proceed to a finite approximation $\lhd_{\frac{1}{k}} \supseteq \lhd_{\frac{2}{k}} \supseteq \ldots \supseteq \lhd_{\frac{k-1}{k}} \supseteq \lhd_1 = \models$.

7 Further work: approximating from both ends

The remaining challenge to the study of approximate inference systems is how to combine the approximations from above described here with approximations from below.

There are several proposals in the literature that corresponds to approximations from below. That is, there is a family of logics $\mathbf{L}_1, \ldots \mathbf{L}_n$ such that

$$\models_{\mathbf{L}_1} \subseteq \models_{\mathbf{L}_2} \subseteq \ldots \subseteq \models_{\mathbf{L}_n} = \models_{\mathbf{CL}} .$$

These systems are useful for proving theorems, in the following sense. We may try to establish $B \models_{\mathbf{L}_1} \alpha$; if it succeeds, than we know that classically $B \models \alpha$; otherwise we proceed to the next step in the approximation, until we either establish the result at some intermediary logic, or we reach classical logic.

There are several examples of systems that approximate classical logic from below:

(1) Schaerf's and Cadoli's system $S_3(S)$ perform approximations for clausal form formulas.
(2) Dalal's BCP-approximations [3], which present a family of approximate reasoners, each of which is tractable, correct and incomplete with respect to classical semantics.
(3) Massacci's approximation system [10, 12] for approximate reasoning, obtained by partitioning the set of propositional letters in $int(\mathcal{P})$ and $inc(\mathcal{P})$, thus forcing $unk(\mathcal{P}) = \emptyset$.
(4) Our own recent proposal of an extension of S_3 to the full propositional fragment, equipped with a tableau proof system, $KES_3(S)$, correct and complete with respect to each approximation step [4, 5].

The very goal of approximating a sequent $B \vdash \alpha$ from both ends is present in Massacci's PhD thesis [12], but no explicit methodology on how to perform it has been presented.

This view of approximating from both ends corresponds to, on the one hand, trying to prove the sequent in the approximations from below and, on the other hand, trying to falsify it in the approximations from above. Just one of this approximations can eventually succeed.

The challenge is to provide a method showing how these opposite processes feed each other. That is, how a failed tentative to falsify a sequent in one approximate step from above may provide a heuristic for the tentative to prove that sequent in an approximate step from below, and vice versa.

8 Conclusion

In this article we have investigated the idea of approximating classical logic 'from above', i.e. through intermediate logics which are complete but possibly unsound.

We have proposed the system s_1, which is parameterized by a context set s. The larger s gets, the closer we are to classical logic. Our system has a ternary valuation semantics which makes it an extension of classical propositional logic. We also presented a proof method based on tableaux which is sound and complete with respect to the semantics. The system was compared with others in the literature and we have shown an application in the area of model-based diagnosis. Future work includes implementing and testing the proof method and combining the results obtained with those for approximations from below.

Acknowledgements

Marcelo Finger is partly supported by the Brazilian Research Council (CNPq), grant PQ 300597/95-5. Renata Wassermann is partly supported by CNPq grant PQ 300196/01-6. This work has been supported by FAPESP project 03/00312-0.

References

[1] S. Chopra, R. Parikh and R. Wassermann. Approximate belief revision. *Logic Journal of the IGPL*, **9**, 755–768, 2001.

[2] M. D'Agostino. Are tableaux an improvement on truth-tables? — cut-free proofs and bivalence. *Journal of Logic, Language and Information*, **1**, 235–252, 1992.

[3] M. Dalal. Anytime families of tractable propositional reasoners. In *Proceedings of the International Symposium of Artificial Intelligence and Mathematics AI|MATH-96*, pp. 42–45, 1996.

[4] M. Finger and R. Wassermann. Expressivity and control in limited reasoning. In *Proceeding of the 15th European Conference on Artificial Intelligence (ECAI02)*, Frank van Harmelen, ed., pp. 272–276, Lyon, France, IOS Press Amsterdam, 2002.

[5] M. Finger and R. Wassermann. Logics for approximate reasoning: Approximating classical logic "from above". In *Proceedings of the XVI Brazilian Symposium on Artificial Intelligence (SBIA'02)*, Volume 2507, *Lecture Notes in Artificial Intelligence*, pp. 21–30. Springer-Verlag, Berlin, 2002.

[6] M. Finger and R. Wassermann. Approximate and limited reasoning: Semantics, proof theory, expressivity and control. *Journal of Logic and Computation*, **14**, 179–204, 2004.

[7] W. Console, L. Hamscher and J. de Kleer, eds, *Readings in Model-Based Diagnosis*. Morgan Kaufmann, San Francisco, CA, 1992.

[8] S. C. Kleene. On a notation for ordinal numbers. *Journal of Symbolic Logic*, **3**, 150–155, 1938.

[9] K. M. Knight. Probabilistic entailment and a non-probabilistic logic. *Logic Journal of the IGPL*, **11**, 353–365, 2003.

[10] F. Massacci. A proof theory for tractable approximations of propositional reasoning. In Maurizio Lenzerini, ed., *AI*IA-97*, Volume 1321, *LNAI*, pp. 219–230 SV, 1997.

[11] F. Massacci. Anytime approximate modal reasoning. In Jack Mostow and Charles Rich, eds, *AAAI-98*, pp. 274–279AAAIP.

[12] F. Massacci. *Efficient Approximate Deduction and an Application to Computer Security*. PhD thesis, Dottorato in Ingegneria Informatica, Università di Roma I "La Sapienza", Dipartimento di Informatica e Sistemistica, June 1998.

[13] N. J. Nilsson. Probabilistic logic. *Artificial Intelligence*, **28**, 71–87, 1986.

[14] N. J. Nilsson. Probabilistic logic revisited. *Artificial Intelligence*, **59**, 39–42, 1993.

[15] J. B. Paris. *The Uncertain Reasoner's Companion — a Mathematical Perspective*, Volume 39, *Cambridge Tracts in Theoretical Computer Science*, Cambridge University Press, Cambridge, 1994.

[16] J. B. Paris and A. Vencovsk. A proof theory for probabilistic uncertain reasoning. *Journal of Symbolic Logic*, **63**, 1007–1039, 1998.

[17] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, **32**, 57–95, 1987. Reprinted in [7].

[18] M. Schaerf and M. Cadoli. Tractable reasoning via approximation. *Artificial Intelligence*, **74**, 249–310, 1995.

[19] R. M. Smullyan. *First-Order Logic*. Springer-Verlag, Berlin, 1968.

[20] A. ten Teije and F. van Harmelen. Computing approximate diagnoses by using approximate entailment. In *Proceedings of KR'96*, Morgan Kaufmann, Cambridge, MA, 1996.

[21] A. ten Teije and F. van Harmelen. Exploiting domain knowledge for approximate diagnosis. In M. Pollack, ed., *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI'97)*, pp. 454–459, Nagoya, Japan, August 1997.

[22] R. Wassermann. Local diagnosis. *Journal of Applied Non-Classical Logics*, **11**, 107–129, 2001.