Approximate and Limited Reasoning: Semantics, Proof Theory, Expressivity and Control

MARCELO FINGER and RENATA WASSERMAN, *Department of Computer Science, Institute of Mathematics and Statistics, University of São Paulo, Brazil.*

E-mail: {mfinger,renata}@ime.usp.br

Abstract

Real agents (natural or artificial) are limited in their reasoning capabilities. In this paper, we present a general framework for modelling limited reasoning based on approximate reasoning and discuss its properties. We start from Cadoli and Schaerf's approximate entailment. We first extend their system to deal with the full language of propositional logic. A tableau inference system is proposed for the extended system together with a subclassical semantics; it is shown that this new approximate reasoning system is sound and complete with respect to this semantics. We show how this system can be incrementally used to move from one approximation to the next until the reasoning limitation is reached. We also present a sound and complete axiomatization of the extended system. We note that although the extension is more expressive than the original system, it offers less control over the approximation process. We then propose a more general system and show that it keeps the increased expressivity and recovers the control. A sound and complete formulation for this new system is given and its expressivity and control advantages are formally proved.

Keywords: Automated reasoning, deductive systems, approximate reasoning, limited reasoning, non-classical logics, knowledge representation.

1 Introduction

Ideal agents know all the consequences of their beliefs. However, real agents are *limited* in their capabilities. In this paper, we present a general framework for modelling limited reasoning based on approximate reasoning and discuss its properties.

Cadoli and Schaerf have proposed the use of *approximate entailment* as a way of reaching at least partial results when solving a problem completely would be too expensive. Their method consists in defining different logics for which satisfiability is easier to compute than classical logic and treat these logics as upper and lower bounds for the classical problem. In [14], these approximate logics are defined by means of valuation semantics and algorithms for testing satisfiability. The language they use is restricted to that of clauses, i.e. negation appears only in the scope of atoms and there is no implication.

The approximations are based on the idea of a context set S of atoms. The atoms in S are the only ones whose consistency is taken into account in the process of verifying whether a given formula is entailed by a set of formulas. As we increase the size of the context set S, we get closer to classical entailment, but the computational complexity also increases.

One of the systems proposed in [14] is the system S_3. S_3 is in fact a family of logics parameterized by a set S of *relevant propositions*. These logics approximate classical logic (CL) in the following sense. Let $\mathcal{P}$ be a set of propositions and $S^0 \subset S^1 \subset \ldots \subset \mathcal{P}$; let $Th(L)$ indicate the set of theorems of a logic. Then:

$$Th(S_3(\emptyset)) \subset Th(S_3(S^0)) \subset Th(S_3(S^1)) \subset \ldots \subset Th(S_3(\mathcal{P})) = Th(CL).$$

In this work, we study Cadoli and Schaerf's family of logics S_3, which we call here CSS_3. Their system only deals with formulas in negation normal form. In this paper, we generalize Cadoli and Schaerf's semantics to deal with full propositional logic. We present a tableaux system, which we call KES_3, for the extended logic which is sound and complete with respect to the semantics. The main feature of our system is that the tableaux method gives a clear way of constructing the context set S in order to obtain the classical answer.

In S_3, atoms which are not part of the context set S may have a paraconsistent behaviour. Paraconsistent logics were introduced by da Costa in the end of the 1950s to avoid a well-known problem of classical logic, that of trivialization in the presence of inconsistency [8]. If we happen to have both a formula and its negation in a set, classical logic can infer from this set just any formula of the language. This is not very reasonable if we think of realistic systems storing information. Suppose we update, by mistake, a database with inconsistent information about a product in a shop. It does not seem reasonable to say that now all information about other products is 'contaminated' by the inconsistency. The inconsistency damages the database only locally, i.e. only for reasoning about that particular product.

Da Costa's system C_1 avoids trivialization by changing the behaviour of negation. Some formulas are said to be *well behaved* and for them, negation behaves classically. For formulas which are not well behaved, it may happen that both the formula and its negation are assigned the value true.

In this paper, we relate a formalism developed by computer scientists aiming at computational efficiency (Cadoli and Schaerf's system) and another developed by logicians aiming to deal with inconsistency (da Costa's C_1).

As a result, we can adapt the proof method that we have developed for S_3 to use with C_1. We also give a sound and complete axiomatization for S_3, based on the axiomatization of the calculus C_1. In Cadoli and Schaerf's work, the treatment of S_3 is purely semantical. An axiomatization was still missing.

In [14], approximate logics are defined by means of valuation semantics and algorithms for testing satisfiability. Here we consider CSS_3 equipped with a resolution-style inference system and KES_3 equipped with a KE-tableaux system [11]. The set S plays the role of the *limitation in reasoning capabilities*. In such a setting, two questions naturally arise:

- Given S, what is $Th(S_3(S))$? This is what we call the *expressivity* of $S_3(S)$.
- How do we expand S to $S' \supset S$ in trying to prove a theorem? In other words, how do we *control* theorem proving before we exhaust our limited resources in reasoning?

The first property is a *static* property of the system, and the second is a *dynamic* one. The dynamic issue is to have an *anytime* method: one that can be stopped at any time, whenever we reach a limit for S-expansion.

We are going to show that, on the static side, the expressivity of CS$S_3(S)$ is smaller than that of KE$S_3(S)$. As a compensation, we are going to show that CS$S_3(S)$ offers more control than KE$S_3(S)$ even when operating over clausal form formulas. We then are going to extend KE$S_3(S)$ to recover the controlling capability. We end up constructing an inference system KES_e with much finer control than both systems, and which still has a sound and complete semantics.

To summarize, the main contributions of the present paper are: an extension of Schaerf and Cadoli's semantics to full propositional logic, a sound and complete proof method based on tableaux, heuristics for increasing the context set S, a sound and complete axiomatization, a comparison to da

Costa's system for paraconsistent logic and a formal analysis of the trade-off between expressivity and control in the approximation process.

It is interesting to note that the tableaux system proposed here can also be restricted to the clausal fragment, keeping the good complexity results of Schaerf and Cadoli and at the same time, providing heuristics that guide the approximation.

The paper proceeds as follows: in the next section, we introduce Cadoli and Schaerf's Approximate Entailment and our extension of their semantics to full propositional logic. A sound and complete axiomatization of the full S_3 is given in Section 3, where the system is compared to da Costa's Paraconsistent Logics [8]. We then turn to a more computational proof method based on KE-tableaux, presented in Section 4. In Section 5, we discuss the gains and losses of the new system with respect to the original formulation of S_3. Finally, in Section 5.3, we present a generalization of the underlying ideas of S_3, ending with a system that allows for much more control in the approximation process.

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

Throughout the paper, 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.

Let $S \subset \mathcal{P}$ be a finite set of propositional letters. We abuse notation and write that, for any formula $\alpha \in \mathcal{L}_C$, $\alpha \in S$ if all its propositional letters are in S. A *propositional valuation* v_p is a function $v_p : \mathcal{P} \rightarrow \{0, 1\}$.

2 Approximate inference

In this section, we present Cadoli and Schaerf's system and extend it to deal with full propositional logic.

2.1 Cadoli and Schaerf's proposal

We briefly present here the notion of *approximate entailment* and summarize the main results obtained in [14].

Schaerf and Cadoli define two approximations of classical entailment: $\models^1_S$ which is complete but not sound, and $\models^3_S$ which is classically sound but incomplete. Here we deal only with the latter. In the trivial extreme of approximate entailment, i.e. when $S = \mathcal{P}$, classical entailment is obtained. At the other extreme, when $S = \emptyset$, $\models^3_S$ corresponds to Levesque's logic for explicit beliefs [13], which bears a connection to relevance logics such as those of Anderson and Belnap [1].

In an S_3 assignment, if $p \in S$, then p and $\neg p$ get opposite truth values, while if $p \notin S$, p and $\neg p$ do not both get 0, but may both get 1. The name S_3 comes from the three possible truth assignments for pairs p, $\neg p$ outside S. The set of formulas for which we are testing entailment is assumed to be in clausal form. Satisfiability, entailment, and validity are defined in the usual way.

The following example illustrates the use of approximate entailment. Since $\models^3_S$ is sound but incomplete, it can be used to approximate $\models$, i.e. if for some S we have that $B \models^3_S \alpha$, then $B \models \alpha$.

EXAMPLE 2.1 ([14])
We want to check whether $B \models \alpha$, where α = $\neg$`cow` $\vee$ `molar-teeth` and

$B =$ $\{\neg$`cow` $\vee$ `grass-eater`, $\neg$`dog`$\vee$ `carnivore`,
$\neg$`grass-eater` $\vee$ $\neg$`canine-teeth`, $\neg$`carnivore` $\vee$ `mammal`,
$\neg$`mammal` $\vee$ `canine-teeth` $\vee$ `molar-teeth`, $\neg$`grass-eater` $\vee$ `mammal`,
$\neg$`mammal` $\vee$ `vertebrate`, $\neg$`vertebrate` $\vee$ `animal`$\}$.

For $S = \{$`grass-eater`, `mammal`, `canine-teeth`$\}$, we have that $B \models^3_S \alpha$, hence $B \models \alpha$.

Note that in the example above, S is a small part of the language. The approximation of classical inference is made via a simplification of the set of clauses B as follows (for a given conclusion α and context set S):

LEMMA 2.2 ([14])
Let *simplify-3*(B, S) be the result of deleting all clauses of B which contain an atom outside S. Then B is S_3-satisfiable if and only if *simplify-3*(B, S) is classically satisfiable.

THEOREM 2.3 ([14])
Let $\alpha \in S$. Then $B \models^3_S \alpha$ iff $B \cup \{\neg\alpha\}$ is not S_3 satisfiable.

Lemma 2.2 and Theorem 2.3 together provide a constructive method for testing S_3 entailment. Consider the Example 2.1, where $S = \{$`grass-eater`, `mammal`, `canine-teeth`$\}$ and we want to test whether $B \models^3_S \neg$`cow` $\vee$ `molar-teeth`. In order to use Theorem 2.3 we must add `cow` and `molar-teeth` to S, then add the clauses `cow` and $\neg$`molar-teeth` to B. We can then use Lemma 2.2 to simplify the expanded base, obtaining:

$B' =$ $\{\neg$`cow` $\vee$ `grass-eater`,
$\neg$`grass-eater` $\vee$ $\neg$`canine-teeth`,
$\neg$`mammal` $\vee$ `canine-teeth` $\vee$ `molar-teeth`,
$\neg$`grass-eater` $\vee$ `mammal`,
`cow`, $\neg$ `molar-teeth`$\}$

This set is classically unsatisfiable, thus, $B \models^3_S \neg$`cow` $\vee$ `molar-teeth`.
Schaerf and Cadoli then obtain the following results for approximate inference:

THEOREM 2.4 ([14])
There is an algorithm for deciding whether $B \models^3_S \alpha$ in $O(|B|.|\alpha|.2^{|S|})$ time.[1]

This algorithm can be seen as a resolution method applied only to clauses where all literals are in S.

The good point of Schaerf and Cadoli's system is that they present an incremental algorithm to test for S_3 entailment as new elements are added to S. But there are two major limitations in their results:

1. The system is restricted to $\rightarrow$-free formulas and in negation normal form. In [6] it is noted that the standard translation of formulas into clausal form does not preserve truth-values under the non-standard semantic of 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 [16], but nothing is said about the general case.

In this paper, we extend their system to full propositional logic, giving semantics, a sound and complete axiomatization and a sound and complete tableaux system. The tableaux system gives us some heuristics for increasing the size of the set S.

[1] The result above depends on a polynomial time satisfiability algorithm for formulas in clausal form. This result has been extended in [4] for formulas in negation normal form, but is not extendable to formulas in arbitrary forms [5].

2.2 Extending approximate inference

In this section, we extend S_3 to full propositional logic. We present a binary semantics for the full fragment of S_3.

The two-valued semantics for S_3 is based on a propositional valuation, as defined below.

DEFINITION 2.5
An *S_3-valuation* v_S^3 is a function, $v_S^3 : \mathcal{L}_C \to \{0,1\}$, that extends a propositional valuation v_p (i.e. $v_S^3(p) = v_p(p)$), satisfying the following restrictions:

$$\begin{array}{lllll}
(i) & v_S^3(\alpha \wedge \beta) = 1 & \Leftrightarrow & v_S^3(\alpha) = v_S^3(\beta) = 1 \\
(ii) & v_S^3(\alpha \vee \beta) = 0 & \Leftrightarrow & v_S^3(\alpha) = v_S^3(\beta) = 0 \\
(iii) & v_S^3(\alpha \to \beta) = 0 & \Leftrightarrow & v_S^3(\alpha) = 1 \text{ and} \\
 & & & v_S^3(\beta) = 0 \\
(iv) & v_S^3(\neg\alpha) = 0 & \Rightarrow & v_S^3(\alpha) = 1 \\
(v) & v_S^3(\neg\alpha) = 1, \alpha \in S & \Rightarrow & v_S^3(\alpha) = 0
\end{array}$$

Rules (i)–(iii) are exactly those of classical logic. Rules (iv) and (v) restrict the semantics of negation: rule (iv) states that if $v_S^3(\neg\alpha) = 0$, then negation behaves classically and forces $v_S^3(\alpha) = 1$; rule (v) states that if $v_S^3(\neg\alpha) = 1$, negation must behave classically only if $\alpha \in S$. Formulas outside S may behave classically or paraconsistently, i.e. both the formula and its negation may be assigned the truth value 1.

Note that an S_3-valuation is not uniquely defined by the propositional valuation it extends. This is due to the fact that if $\alpha \notin S$ and $v_S^3(\alpha) = 1$, the value of $v_S^3(\neg\alpha)$ can be either 0 (in which case α has a classical behaviour) or 1 (in which case α behaves paraconsistently).

LEMMA 2.6
The valuation v_S^3 is determined by its value on the set $\mathcal{P} \cup \{\neg\alpha | \alpha \notin S \wedge v_S^3(\alpha) = 0\}$.

PROOF. For the connectives $\wedge$, $\vee$ and $\to$, it is immediate that the value of v_S^3 is determined by that of the subformulas. If $v_S^3(\alpha) = 0$, then by rule (iv) $v_S^3(\neg\alpha) = 1$. If $\alpha \in S$ the value of $v_S^3(\neg\alpha)$ is the opposite to that of $v_S^3(\alpha)$. If $\alpha \notin S$ but $v_S^3(\alpha) = 0$, then necessarily $v_S^3(\neg\alpha) = 1$. We are left with only the formulas in $\mathcal{P} \cup \{\neg\alpha | \alpha \notin S \wedge v_S^3(\alpha) = 1\}$ to determine v_S^3. ∎

We define a formula α to be *S-valid* in S_3 if $v_S^3(\alpha) = 1$ for any S_3-valuation. A formula is *S-satisfiable* in S_3 if there is at least one v_S^3 such that $v_S^3(\alpha) = 1$. The *S_3-entailment* relationship between a set of formulas Γ and a formula α is represented as

$$\Gamma \models_S^3 \alpha$$

and holds if every valuation v_S^3 that simultaneously satisfies all formulas in Γ also satisfies α. A formula is S-valid if it is entailed by $\emptyset$, represented as $\models_S^3 \alpha$.

Lemma 2.6 suggests a translation between a formula in S_3 and one in classical logic, such that every formula of the form $\neg\alpha$ with $\alpha \notin S$ is mapped into a formula $\alpha^{*S} \to p_{\neg\alpha}$, where $p_{\neg\alpha}$ is a new propositional symbol. Let α^{*S} be the translation of α, defined as:

$$\begin{array}{rcl}
p^{*S} & = & p \\
(\alpha \circ \beta)^{*S} & = & \alpha^{*S} \circ \beta^{*S}, \quad \circ \in \{\wedge, \vee, \to\} \\
(\neg\alpha)^{*S} & = & \begin{cases} \neg(\alpha^{*S}) & , \text{ if } \alpha \in S \\ \alpha^{*S} \to p_{\neg\alpha}, & \text{if } \alpha \notin S. \end{cases}
\end{array}$$

LEMMA 2.7
A formula α is S-satisfiable in S_3 iff $(\alpha)^{*S}$ is classically satisfiable.

PROOF. As seen in the proof of Lemma 2.6, we want to assure that if $\alpha \notin S$ but $v^3_S(\alpha) = 0$, then necessarily $v^3_S(\neg\alpha) = 1$. This is taken care of by the last line of the translation. The rest of the translation keeps the classical properties of the other connectives and of formulas in S. ∎

COROLLARY 2.8
For any $S \subseteq \mathcal{P}$, the complexity of $S_3(S)$-satisfiability is in NP.

PROOF. Since the size of α^{*S} is linear with the size of α, by Lemma 2.6 the complexity of $S_3(S)$-satisfiability is upper-bounded by the complexity of classical satisfiability. ∎

Corollary 2.8 shows that the complexity of an approximation step is never more complex than the complexity of propositional classical logic. However, there are classes of formulas whose complexity is much simpler.

LEMMA 2.9
Let $Prop(\alpha)$ represent the propositional symbols in α. Every formula α with $Prop(\alpha) \cap S = \emptyset$ is satisfiable in S_3.

PROOF. Since $Prop(\alpha) \cap S = \emptyset$, the translation $(\alpha)^{*S}$ leads into the $\neg$-free fragment of classical logic (without falsity, $\bot$), and any formula in such a fragment is satisfiable (just make all atoms 1), so Lemma 2.7 makes α satisfiable in S_3. ∎

Note that for Cadoli and Schaerf's S_3, the condition $Prop(\alpha) \not\subseteq S$ is sufficient, since they only deal with clauses. If any literal of a clause is not in S, it can be assigned the value 1, making the whole clause (a disjunction of literals) satisfiable.

We now show that family of logics S_3 is indeed an approximation of classical logic, as defined in the introduction.

LEMMA 2.10 (Approximation of Classical Logic)
For any $S \subset \mathcal{P}$:

(i) $S_3(S)$-validity is subclassical; that is, any $S_3(S)$-valid formula is classically valid.

(ii) For $S \subset S'$, $Th(S_3(S)) \subset Th(S_3(S'))$.

(iii) $S_3(\mathcal{P})$ is classical logic.

PROOF.

(i) Any classical valuation satisfies the S_3-restrictions for any S. Therefore, the set of all classical valuations is contained in the set of all S_3-valuations (but there are S_3-valuations that are not classic). So if a formula is satisfied by all S_3-valuations, it will be satisfied by all classical ones.

(ii) If $p \in S' - S$, the formula $p \rightarrow \neg\neg p$ is a theorem of $S_3(S')$ (classically) but not in $S_3(S)$, by making $v^3_S(p) = v^3_S(\neg p) = 1$ and $v^3_S(\neg\neg p) = 0$.

(iii) Trivial.

∎

The deduction theorem holds for the S_3 semantics, as can be seen directly from the definitions.

LEMMA 2.11 (Deduction Theorem)
Let Γ be a finite set of formulas. Then

$$\Gamma \models^3_S \alpha \textit{ iff } \models^3_S \bigwedge \Gamma \rightarrow \alpha.$$

Now we examine a few examples of S_3 entailment.

EXAMPLE 2.12
Consider the formula $\alpha \vee \neg\alpha$. We show that it is a valid formula in S_3 for any S.

Indeed, if $\alpha \in S$, we are in a classical setting, so any valuation makes $\alpha \vee \neg\alpha$ true.

If $\alpha \notin S$, let v_S^3 be a valuation. If $v_S^3(\neg\alpha) = 1$, then $\alpha \vee \neg\alpha$ clearly is true. If, however, $v_S^3(\neg\alpha) = 0$, by rule (iv) above $v_S^3(\alpha) = 1$, so $\alpha \vee \neg\alpha$ is also true.

EXAMPLE 2.13
We now show that the S_3 semantics is paraconsistent. For that, consider the two propositions p and q and suppose that $p \notin S$; take a valuation v_S^3 such that $v_S^3(p) = 1$ and $v_S^3(q) = 0$, and consider the formula $(p \wedge \neg p) \rightarrow q$. By Lemma 2.6, the value of v_S^3 is not fully determined, so we fix $v_S^3(\neg p) = 1$. It is simple to verify now that the valuation thus constructed is such that $v_S^3((p \wedge \neg p) \rightarrow q) = 0$, that is the logic does not always trivialize in the presence of inconsistency.

EXAMPLE 2.14
We now analyse the validity of Modus Ponens in S_3. The usual formulation of Modus Ponens, $\alpha \rightarrow \beta, \alpha \models_S^3 \beta$, is valid in S_3; indeed, if v_S^3 satisfies α, the only possible way that it also satisfies $\alpha \rightarrow \beta$ is that it satisfies β, thus proving the entailment. Note that since no $\neg$-formula was involved, the reasoning is totally classical.

However, if we consider the version of Modus Ponens consisting of the translation of $\alpha \rightarrow \beta$ into $\neg\alpha \vee \beta$ (the only possible version of Modus Ponens in [14]), the situation changes completely if $\alpha \notin S$, for then we can have a valuation that satisfies both α and $\neg\alpha$ (and thus $\neg\alpha \vee \beta$), but that falsifies β, so that $\neg\alpha \vee \beta, \alpha \not\models_S^3 \beta$.

A sound and complete axiomatization of the full S_3 is given in Section 3, where it is also compared with da Costa's Paraconsistent Logics [8]. We then turn to a more computational proof method based on KE-tableaux.

3 Approximate Inference and Paraconsistency

In this section, we present a sound and complete axiomatization for S_3, based on the axiomatization of the paraconsistent logic C_1 [8]. We then compare the systems S_3 and C_1.

3.1 An axiomatization for S_3

The following is an axiomatization for S_3, based on the axiomatization given in [8]:

Consider the axioms in Figure 1. The *Positive Axioms* are precisely the classical axioms for the connectives $\rightarrow$, $\wedge$ and $\vee$; the rule of Modus Ponens is presented as $(\rightarrow_3)$. Our system differs from classical logic in the *Negation Axioms*.

In fact, without the proviso in axioms **(neg$_1$)** and **(neg$_2$)**, the negation axioms are precisely the classical $\neg$-axioms. The $\neg$-introduction of axiom **(neg$_1$)** is restricted to the consequences of α in S. The trivialization of axiom **(neg$_2$)** is restricted to the members of S, that is, non-members of S can behave paraconsistently. Axiom **(neg$_3$)** tells us that the excluded middle is accepted unconditionally in our logic.

THEOREM 3.1
The axiomatization of Figure 1 is sound with respect to the S_3 semantics. That is, all formulas inferred from the S_3 axiomatization are true in all S_3-valuations.

Positive Axioms:

($\rightarrow_1$) $\alpha \rightarrow (\beta \rightarrow \alpha)$
($\rightarrow_2$) $(\alpha \rightarrow \beta) \rightarrow (\alpha \rightarrow (\beta \rightarrow \gamma)) \rightarrow (\alpha \rightarrow \gamma)$
($\wedge_1$) $\alpha \wedge \beta \rightarrow \alpha$
($\wedge_2$) $\alpha \wedge \beta \rightarrow \beta$
($\wedge_3$) $\alpha \rightarrow (\beta \rightarrow \alpha \wedge \beta)$
($\vee_1$) $\alpha \rightarrow \alpha \vee \beta$
($\vee_2$) $\beta \rightarrow \alpha \vee \beta$
($\vee_3$) $(\alpha \rightarrow \gamma) \rightarrow ((\beta \rightarrow \gamma) \rightarrow (\alpha \vee \beta \rightarrow \gamma))$

Negation Axioms:

(neg$_1$) $(\alpha \rightarrow \beta) \rightarrow ((\alpha \rightarrow \neg\beta) \rightarrow \neg\alpha)$, provided $\beta \in S$
(neg$_2$) $(\alpha \wedge \neg\alpha) \rightarrow \beta$, provided $\alpha \in S$
(neg$_3$) $\alpha \vee \neg\alpha$

Inference Rule:

(MP) $\alpha, \alpha \rightarrow \beta / \beta$

FIGURE 1. An axiomatization for S_3

PROOF. By a straightforward, but tedious, verification of the validity of the axioms. The positive axioms are dealt with by the classical part of the S_3 semantics. Axiom **(neg$_3$)** is dealt with in Example 2.13, and Modus Ponens is dealt with in Example 2.14. The validity of axioms **(neg $_1$)** and **(neg$_2$)** are easily verified. ∎

A formula α is *S-inconsistent* if it is provable that $\alpha \rightarrow (\beta \wedge \neg\beta)$ for some $\beta \in S$. Similarly, a set X of formulas is *S-inconsistent* if there are formulas $\chi_1, \ldots, \chi_n \in X$ such that $\bigwedge \chi_i \rightarrow (\beta \wedge \neg\beta)$ for some $\beta \in S$. A formula or set is *S-consistent* if it is not S-inconsistent.

The axiomatization above is *S-complete* iff for any S-consistent formula α there is an S_3 valuation v_S^3 such that $v_S^3(\alpha) = 1$. The proof of completeness follows a Lindenbaum construction.

A *maximal S-consistent set* (MSCS) is a set of formulas X such that:

- X is S-consistent; and
- there is no MSCS $X' \supset X$.

LEMMA 3.2
For every S-consistent formula α there is a MSCS X such that $\alpha \in X$.

PROOF. Consider an enumeration of the formulas $\beta_0, \beta_1, \ldots, \beta_i, \ldots$. Construct a sequence of sets X_i such that

$$X_0 = \alpha$$
$$X_{i+1} = \begin{cases} X_i \cup \{\beta_i\}, & \text{if } X_i \cup \{\beta_i\} \text{ is } S\text{-consistent} \\ X_i, & \text{otherwise.} \end{cases}$$

Let $X = \bigcup_{i=0}^{\infty} X_i$. We claim that X is a MSCS, for the following reasons:

- X is S-consistent. Otherwise let X_{k+1} be the first element in the sequence that is S-inconsistent. Then we must have $X_{k+1} \neq X_k$, so $X_{k+1} = X_k \cup \{\beta_k\}$, but this is only possible if $X_k \cup \{\beta_k\}$ is S-consistent, contradicting the fact that X_{k+1} is S-inconsistent.

- X is maximal. Otherwise there is a MSCS $X' \supset X$. In this case there is a $\beta_k \in X' - X$. Consider the set $X_k \cup \{\beta_k\}$; if it is S-consistent, then $X_k \cup \{\beta_k\} = X_{k+1} \supset X$, which contradicts $\beta_k \notin X$. If $X_k \cup \{\beta_k\}$ is S-inconsistent, then X' is inconsistent because $X_k \supset X \supset X'$ and $\beta_k \in X'$, which contradicts the consistency of X'. Hence X must be maximal.

It is obvious that $\alpha \in X$, so the proof is finished. ∎

LEMMA 3.3
Let X be a MSCS. Let $v : \mathcal{L}_C \longrightarrow \{0, 1\}$ such that $v(\alpha) = 1$ iff $\alpha \in X$. Then v is a S_3 valuation.

PROOF. We have to show that v satisfies the restrictions of Definition 2.5. Conditions (i)–(iii) are classical semantical conditions and are accounted for by the positive axioms, which are the classical ones.

We then show that: $(*) \quad v(\alpha) = 0 \Longrightarrow v(\neg\alpha) = 1$. For that, suppose $v(\alpha) = 0$. Then $\alpha \notin X$. Due to the maximality of X, there must be $\chi_1, \ldots, \chi_n \in X$ such that it is provable that $((\bigwedge \chi_i) \wedge \alpha) \rightarrow (\beta \wedge \neg\beta)$ for some $\beta \in S$. Therefore:

$$\vdash ((\bigwedge \chi_i) \wedge \alpha) \rightarrow \beta \Longrightarrow \vdash (\bigwedge \chi_i) \rightarrow (\alpha \rightarrow \beta) \Longrightarrow \alpha \rightarrow \beta \in X$$
$$\vdash ((\bigwedge \chi_i) \wedge \alpha) \rightarrow \neg\beta \Longrightarrow \vdash (\bigwedge \chi_i) \rightarrow (\alpha \rightarrow \neg\beta) \Longrightarrow \alpha \rightarrow \neg\beta \in X.$$

It follows that both $\alpha \rightarrow \beta \in X$ and $\alpha \rightarrow \neg\beta \in X$. From Axiom **(neg$_1$)**, it follows that $\neg\alpha \in X$, so $v(\neg\alpha) = 1$, proving $(*)$.

From (*) it follows that v obeys property (iv) of Definition 2.5. Indeed, suppose that $v(\neg\alpha) = 0$; if $v(\alpha) = 0$, then by $(*)$ we have that $v(\neg\alpha) = 1$, a contradiction. So we must have $v(\alpha) = 1$, satisfying (iv).

We then show that: $(**) \quad v(\alpha) = 1, \alpha \in S \Longrightarrow v(\neg\alpha) = 0$. For that, suppose $v(\alpha) = 1$ and $\alpha \in S$. Then $\alpha \in X$. Suppose now, for contradiction that $\neg\alpha \in X$. From Axiom **(neg $_2$)** we have that any $\beta \in X$, contradicting the consistency of X. So $\neg\alpha \notin X$ and $v(\neg\alpha) = 0$, thus proving $(**)$.

From $(**)$ we see that v obeys property (v) of Definition 2.5. Indeed, suppose that $v(\neg\alpha) = 1$ and $\alpha \in S$; if $v(\alpha) = 1$, by $(**)$ we have that $v(\neg\alpha) = 0$, a contradiction. So we must have $v(\alpha) = 0$, satisfying (v). ∎

THEOREM 3.4
The axiomatization of Figure 1 is complete with respect to the S_3 semantics.

PROOF. By Lemma 3.2 there is a MSCS X with $\alpha \in X$. Then, by Lemma 3.3, there is a S_3-valuation v^3_S such that $v^3_S(\beta) = 1$ iff $\beta \in X$. In particular, $v^3_S(\alpha) = 1$. ∎

3.2 Relating S_3 to Da Costa's C_1

In this section, we introduce Da Costa's calculus C_1 by means of axioms and valuation semantics. We then show how our system relates to this calculus.

DEFINITION 3.5
A formula α is said to be *well behaved* if the principle of non-contradiction holds for α, i.e. if $\neg(\alpha \wedge \neg\alpha)$ holds. We use α^o to denote $\neg(\alpha \wedge \neg\alpha)$.

Da Costa's calculus [8] was introduced in order to deal with possible inconsistencies that should not damage reasoning by trivializing it. The idea is to block derivations from formulas which are not well behaved, isolating inconsistencies.

The following is an axiomatization of C_1:

($\rightarrow_1$) $\alpha \rightarrow (\beta \rightarrow \alpha)$
($\rightarrow_2$) $(\alpha \rightarrow \beta) \rightarrow (\alpha \rightarrow (\beta \rightarrow \gamma)) \rightarrow (\alpha \rightarrow \gamma)$
($\wedge_1$) $\alpha \wedge \beta \rightarrow \alpha$
($\wedge_2$) $\alpha \wedge \beta \rightarrow \beta$
($\wedge_3$) $\alpha \rightarrow (\beta \rightarrow \alpha \wedge \beta)$
($\vee_1$) $\alpha \rightarrow \alpha \vee \beta$
($\vee_2$) $\beta \rightarrow \alpha \vee \beta$
($\vee_3$) $(\alpha \rightarrow \gamma) \rightarrow ((\beta \rightarrow \gamma) \rightarrow (\alpha \vee \beta \rightarrow \gamma))$
($\neg_1$) $\beta^o \rightarrow (\alpha \rightarrow \beta) \rightarrow ((\alpha \rightarrow \neg\beta) \rightarrow \neg\alpha)$
($\neg_2$) $\alpha^o \wedge \beta^o \rightarrow ((\alpha \rightarrow \beta)^o \wedge (\alpha \wedge \beta)^o \wedge (\alpha \vee \beta)^o)$
($\neg_3$) $\alpha \vee \neg\alpha$
($\neg_4$) $\neg\neg\alpha \rightarrow \alpha$[1em]
Inference Rule:

(MP) $\alpha, \alpha \rightarrow \beta / \beta$

A semantic for this system was given in [7]:

DEFINITION 3.6
An $\mathcal{N}$*-valuation* $v_{\mathcal{N}}$ is a function, $v_{\mathcal{N}} : \mathcal{L}_C \rightarrow \{0,1\}$, that extends a propositional valuation v_p (i.e. $v_{\mathcal{N}}(p) = v_p(p)$), satisfying the following restrictions:

(i)	$v_{\mathcal{N}}(\alpha \wedge \beta) = 1$	$\Leftrightarrow$	$v_{\mathcal{N}}(\alpha) = v_{\mathcal{N}}(\beta) = 1$
(ii)	$v_{\mathcal{N}}(\alpha \vee \beta) = 0$	$\Leftrightarrow$	$v_{\mathcal{N}}(\alpha) = v_{\mathcal{N}}(\beta) = 0$
(iii)	$v_{\mathcal{N}}(\alpha \rightarrow \beta) = 0$	$\Leftrightarrow$	$v_{\mathcal{N}}(\alpha) = 1$ and $v_{\mathcal{N}}(\beta) = 0$
(iv)	$v_{\mathcal{N}}(\alpha) = 0$	$\Rightarrow$	$v_{\mathcal{N}}(\neg\alpha) = 1$
(v)	$v_{\mathcal{N}}(\neg\neg\alpha) = 1$	$\Rightarrow$	$v_{\mathcal{N}}(\alpha) = 1$
(vi)	$v_{\mathcal{N}}(\beta^o) = v_{\mathcal{N}}(\alpha \rightarrow \beta) = v_{\mathcal{N}}(\alpha \rightarrow \neg\beta) = 1$	$\Rightarrow$	$v_{\mathcal{N}}(\alpha) = 0$
(vii)	$v_{\mathcal{N}}(\alpha^o) = v_{\mathcal{N}}(\beta^o) = 1$	$\Rightarrow$	$v_{\mathcal{N}}((\alpha * \beta)^o) = 1, * \in \{\rightarrow, \wedge, \vee\}$

An $\mathcal{N}$-valuation has the following properties:

LEMMA 3.7
1. $v_{\mathcal{N}}(\alpha) = 1 \Leftrightarrow v_{\mathcal{N}}(\neg\alpha \wedge \alpha^o) = 0$.
2. $v_{\mathcal{N}}(\alpha^o) = 0 \Leftrightarrow v_{\mathcal{N}}(\alpha) = v_{\mathcal{N}}(\neg\alpha)$.

It is not hard to see that the systems are very similar, although not equivalent. There are two main differences. First, the set of formulas for which $v_{\mathcal{N}}(\alpha^o) = 1$ does not correspond exactly to S:

LEMMA 3.8
If $\alpha \in S$, then $v_{\mathcal{N}}(\alpha^o) = 1$. The converse does not always hold, i.e. we may have an $\mathcal{N}$-valuation v such that $v(\alpha^o) = 1$ and $\alpha \notin S$.

PROOF. Follows directly from the definitions. $\alpha \in S$ means that α and $\neg\alpha$ must have opposite truth values, but $\alpha \notin S$ does not necessarily mean that α and $\neg\alpha$ have the same truth values. ∎

The second difference between the two systems is in the behaviour of double negation. In C_1, double negation may be eliminated (but not introduced), while in S_3, α does not follow from $\neg\neg\alpha$.

Since we have sound and complete semantics for both systems, we can show how the systems relate to each other by checking the semantics.

PROPOSITION 3.9
Every $\mathcal{N}$-valuation is an S_3-valuation.

PROOF. Conditions (i), (ii), (iii), and (iv) are the same. We only have to check whether $v^3_S(\neg\alpha) = 1, \alpha \in S \Rightarrow v^3_S(\alpha) = 0$ follows from the definition of $\mathcal{N}$-valuation. We know that $\alpha \in S$ implies $v^3_S(\alpha^o) = 1$. And from $v^3_S(\neg\alpha) = v^3_S(\alpha^o) = 1$ it follows by part (2) of Lemma 3.7 that $v^3_S(\alpha) = 0$. ∎

PROPOSITION 3.10
Not every S_3-valuation is an $\mathcal{N}$-valuation.

PROOF. It suffices to see that we may have an S_3-valuation v so that for some formula α such that $\alpha \notin S$, we have $v(\neg\neg\alpha) = v(\neg\alpha) = 1$ and $v(\alpha) = 0$. This valuation fails to satisfy item (v) of the definition of $\mathcal{N}$-valuation. ∎

The two propositions show that S_3 is actually more general than C_1.

COROLLARY 3.11
Let S be a fixed set of formulas closed under formula construction and let C_1^S be C_1 with an added axiom α^o for each $\alpha \in S$. Then Theorems(S_3) $\subseteq$ Theorems(C_1^S) $\subseteq$ Theorems($\mathcal{L}_C$).

4 Tableaux for approximate inference

We develop an inference system for the full logic S_3 based on the KE-tableau methodology. KE-tableaux were introduced by D'Agostino [10, 12, 11] as a principled computational improvement over Smullyan's Semantic Tableaux [15], and have since been successfully applied to a variety of logics [9, 2, 3].

KE-tableaux deal with T- and F-signed formulas. So if α is a formula, $T\ \alpha$ and $F\ \alpha$ are signed formulas. $T\ \alpha$ is the *conjugate formula* of $F\ \alpha$, and vice versa. In classical logic it is possible to have an unsigned version of tableaux [15], but in non-classical logics in general, and in KES_3 in particular, the use of signed formula is crucial.[2] An expansion of a tableau is allowed when the premisses of an expansion rule are present in a branch; the expansion consists of adding the conclusions of the rule to the end of all branches passing through the set of all premisses of that rule.

For each connective, there are at least one T- and one F-linear expansion rules. Linear expansion rules always have a main premiss, and may also have an auxiliary premiss. They may have one or two consequences. The only branching rule is the *Principle of Bivalence*, stating that something must be either true or false. Figure 2 shows KE-tableau expansion rules for classical logic.

In Figure 2 we see that each of the binary connectives $\rightarrow, \wedge$ and $\vee$ are associated to two two-premissed rules and one one-premissed rule. The two-premissed rules have a *main premiss* and an *auxiliary premiss*; the one-premissed rules have two consequences. Classical negation is associated to two one premissed rules, each with a single conclusion. The final line presents the *Principle of Bivalence* (PB), stating that any formula α is either true or false. The application of PB transforms a single branch into two branches with the same prefix, differing only by the final formula, each new branch getting one of the two conjugates.

PB is used according to a *branching rule*: PB is used to generate the auxiliary premiss for a two-premissed rule; this guarantees that PB is only used over subformulas of some complex formula occurring in the tableau. This also guarantees the *subformula property*, i.e. an expansion always introduces in the tableau a subformula of some previously occurring formula.

[2] In unsigned tableaux, F-marks are replaced by the connective $\neg$; this artifice only works for classical-like negation, but S_3 has a non-classical negation semantics, and F-marks and $\neg$-marks are not interchangeable.

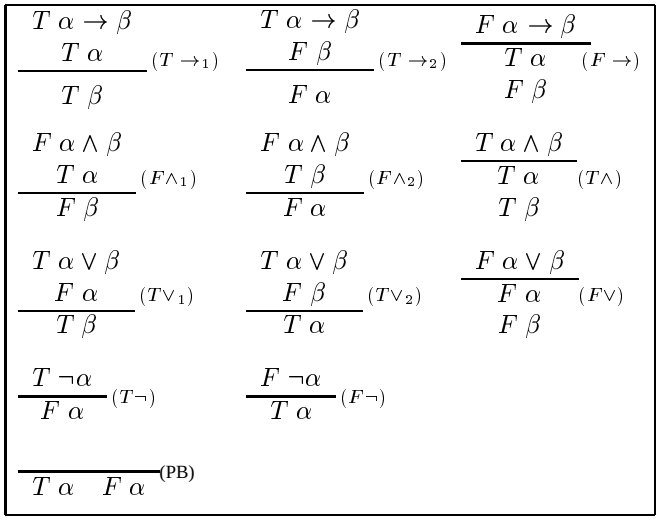


FIGURE 2. KE-rules for classical logic

As in semantic tableaux, to show that $\alpha_1, \ldots, \alpha_n \vdash \beta$ we start with the initial tableau

$$\begin{array}{c} T\ \alpha_1 \\ \vdots \\ T\ \alpha_n \\ F\ \beta \end{array}$$

and develop the tableau by applying the expansion rules in Figure 2. A branch is closed if it contains both $F\ \alpha$ and $T\ \alpha$, for some formula α. The sequent above is shown if we can *close* all branches in the tableau, in which case the tableau is said to be closed.

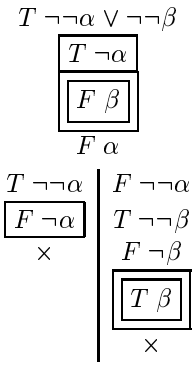
EXAMPLE 4.1
We know that classically, $\alpha \to \beta$ is equivalent to $\neg\alpha \vee \beta$. This is shown by means of the two KE-tableaux in Figure 3, where the boxed formulas indicate the closure condition for each branch. The left tableau shows $\alpha \to \beta \vdash \neg\alpha \vee \beta$ and the right one shows $\neg\alpha \vee \beta \vdash \alpha \to \beta$.

Note that both tableaux would branch in a semantic tableau version of this proof. The fact that KE-tableaux do not branch is an indication that they are more efficient than traditional semantic tableaux. In fact, KE-tableaux can p-simulate semantic tableaux, but the converse is not true [10].

1.	$T\ \alpha \to \beta$		1.	$T\ \neg\alpha \vee \beta$	
2.	$F\ \neg\alpha \vee \beta$		2.	$F\ \alpha \to \beta$	
3.	$F\ \neg\alpha$	from 2	3.	$\boxed{T\ \alpha}$	from 2
4.	$\boxed{F\ \beta}$	from 2	4.	$F\ \beta$	from 2
5.	$T\ \alpha$	from 3	5.	$T\ \neg\alpha$	from 1,4
6.	$\boxed{T\ \beta}$	from 1,5	6.	$\boxed{F\ \alpha}$	from 5
	$\times$			$\times$	

FIGURE 3. The classical equivalence of $\alpha \to \beta$ and $\neg\alpha \vee \beta$

EXAMPLE 4.2
To show the use of PB, we present a KE-tableau now showing that $\neg\neg\alpha \vee \neg\neg\beta, \neg\alpha \vdash \beta$.



After expanding $T\ \neg\alpha$, no more linear expansion rules are applicable, so we branch over $\neg\neg\alpha$ in $T\ \neg\neg\alpha \vee \neg\neg\beta$ according to the branching rule, so that the negative branch on the right can be used as an auxiliary premiss to $T\ \neg\neg\alpha \vee \neg\neg\beta$ generating $T\ \neg\neg\beta$. On the left branch, a single expansion of $T\ \neg\neg\alpha$ leads us to $F\ \neg\alpha$, which closes that branch and the tableau.

4.1 Tableaux for S_3

In order to construct a KE-tableau system for S_3, we keep almost all the classical rules, changing only the rule $(T\ \neg)$, by adding a side condition. The old rule $(T\ \neg)$ is removed and its S_3 version becomes:

$$\frac{T\ \neg\alpha}{F\ \alpha}\ \text{provided that } \alpha \in S.$$

The meaning of this rule is that the expansion of a branch is only allowed if it contains the rule's antecedent *and the proviso is satisfied*, that is, the formula in question belongs to S. This rule is actually a restriction of the classical rule, stating that if $\alpha \notin S$ the $(T\ \neg)$-rule cannot be applied. Let us call the system thus obtained KES_3.

This makes our system immediately subclassical, for any tableau that closes for KES_3 also closes for classical logic. So any theorems we prove in KES_3 are also classical theorems. The converse

is not the case, as the examples below will show that there are classical theorems which are not KES_3-theorems.

So KES_3 is correct and incomplete with respect to classical logic. The actual proof of correctness and completeness of KES_3 with respect to the semantics presented in Section 2.2 will be presented in Section 4.3. First, let us examine a few examples.

EXAMPLE 4.3
We first show that $\rightarrow$ is no longer definable in terms of $\vee$ and $\wedge$, by redoing the tableaux of Figure 3 in Figure 4.

1.	$T\ \alpha \rightarrow \beta$	
2.	$F\ \neg\alpha \vee \beta$	
3.	$F\ \neg\alpha$	from 2
4.	$\boxed{F\ \beta}$	from 2
5.	$T\ \alpha$	from 3
6.	$\boxed{T\ \beta}$	from 1,5
	$\times$	

1.	$T\ \neg\alpha \vee \beta$	
2.	$F\ \alpha \rightarrow \beta$	
3.	$T\ \alpha$	from 2
4.	$F\ \beta$	from 2
5.	$T\ \neg\alpha$	from 1,4
	?	

FIGURE 4. $\rightarrow$ is not definable in terms of $\vee$ and $\neg$ in KES_3

In Figure 4 we are assuming that $S = \emptyset$. Note that the left tableau for $\alpha \rightarrow \beta \vdash \neg\alpha \vee \beta$ is exactly the same as for classical logic.

However, the tableau on the right for $\neg\alpha \vee \beta \vdash \alpha \rightarrow \beta$ cannot be closed for the rule on $T\ \neg\alpha$ cannot be applied for $\alpha \notin S$. We get stuck, as there are no further rules to be applied, meaning that the input sequent is not provable.

This shows that $\rightarrow$ can no longer be defined in terms of $\vee$ and $\neg$ in S_3.

One important feature of the open tableau in Figure 4 is that if, at the point that it gets stuck, we insert the propositional letters of α in the set S, the tableau expansion can proceed as in classical logic. In fact, the tableau then closes after a single step. This shows that the sequent $\neg\alpha \vee \beta \vdash \alpha \rightarrow \beta$ is deducible if $\alpha \in S$ (and nothing needs to be said about β).

What we have actually done is to change the logic we are operating with during the KE-tableau expansion by adding a formula to S. That formula was chosen so that a stuck tableau could proceed classically. This actually makes us move one step closer to classical logic. Classical logic is reached when all atoms are in S.

This simple procedure suggests an incremental way of doing approximate theorem proving.

4.2 The incrementality of the method

The idea of approximate reasoning found in [14] consists in trying to prove a classical formula in S_3 for increasingly large sets S. Apart from considering an $\rightarrow$-free fragment, the work in [14] did not provide a way of choosing S or how to increment it. Our theorem proving method for a given input sequent is intended to fill this gap. It is summarized in the following:

1. $S := \emptyset$.
2. Transform the input sequent in an initial KES_3-tableau.

3. Expand the tableau until it is closed or stuck.
4. If the tableau is closed, terminate with success.
5. If the tableau contains a branch that cannot be classically expanded, terminate with failure.
6. If the tableau is stuck due to a formula $T\neg\alpha$, make $S := S \cup \{\alpha\}$ and go back to 3.

By $S := S \cup \{\alpha\}$ we mean that all atoms in α are added to S. It is clear that if an atom does not appear inside the scope of a negation, it will not be inserted in S. However, that does not mean that if it appears inside a negation it will end up in S, as the tableau for $\vdash p \vee \neg p$ shows:

$$\begin{array}{c} F\ p \vee \neg p \\ F\ p \\ F\ \neg p \\ T\ p \\ \times \end{array}$$

which shows that $\vdash p \vee \neg p$ is S_3-valid for $S = \emptyset$. Note that in step 6 above, there may be more then one stuck point in the tableau, so we need to choose one formula to proceed. If there are two stuck formulas in the same branch of the tableau, the contents of S may differ according to the choice of formula we make at step 6. This is illustrated by the following tableau for $p \wedge q \vdash \neg\neg p \vee \neg\neg q$:

$$\begin{array}{c} T\ p \wedge q \\ F\ \neg\neg p \vee \neg\neg q \\ T\ p \\ T\ q \\ F\ \neg\neg p \\ F\ \neg\neg q \\ T\ \neg p \\ T\ \neg q \end{array}$$

At this point we have both $T\ \neg p$ and $T\ \neg q$ blocking the branch development. If we choose the first one, S becomes $\{p\}$ and the tableau closes; if we choose the second one, S becomes $\{q\}$ and the tableau also closes.

An interesting fact is that we can only prove $\neg(p \wedge \neg p)$ in KES_3 for $p \in S$, which shows the paraconsistency of the system:

$$\begin{array}{c} F\ \neg(p \wedge \neg p) \\ T\ p \wedge \neg p \\ \boxed{\text{T p}} \\ T\ \neg p \\ \boxed{\text{F p}}\ \text{if } p \in S \\ \times \end{array}$$

We now show the correctness of the proposed incremental method.

THEOREM 4.4 (Correctness of the Method)
Given an input sequent $\alpha_1, \ldots, \alpha_n \vdash \beta$ then the method above always terminates. It terminates with success iff the sequent is classically valid.

1.	T ¬cow ∨ g-e				
2.	T ¬dog∨ carnivore				
3.	T ¬g-e ∨ ¬c-t				
4.	T ¬carnivore ∨ mammal				
5.	T ¬mammal ∨ c-t ∨ m-t				
6.	T ¬g-e ∨ mammal				
7.	T ¬mammal ∨ vertebrate				
8.	T ¬vertebrate ∨ animal				
9.	F ¬cow ∨ m-t				
10.	F ¬cow	from 9			
11.	F m-t	from 9			
12.	T g-e	from 1,10			
13.	F ¬g-e		13′.	T ¬g-e	PB
14.	T ¬c-t	from 3,13	14′	F g-e	g-e $\in S$
15.	T mammal	from 6,13	15′	×	
16.	T ¬mammal ∨ c-t	from 5,11			
17.	F c-t	from 14, c-t $\in S$			
18.	T ¬mammal	from 16,17			
19.	F mammal	from 18, mammal $\in S$			
20.	×				

FIGURE 5. KES_3 tableau for clauses

PROOF. Note that the set S can only increase at each cycle, so if the tableau does not close, S will eventually contain all propositional symbols in the input sequent that occur within a T-marked negation, in which case the tableau will be a classical one; by the termination property of KE-tableaux, it will terminate. Because any closing KES_3 tableau also classically closes, a successful terminating tableau must be classically valid. On the other hand, if a sequent is classically valid, its corresponding KES_3-tableau, when it becomes classical, will eventually close. ∎

EXAMPLE 4.5

The following example illustrates the use of the system KES_3. Consider the problem of Example 2.1, where we want to know whether ¬cow ∨ molar-teeth follows from a set of clauses B. We start by labelling the initial clauses with T (lines 1-8) and the formula we want to refute with F (line 9). Figure 5 shows the complete tableau (g-e stands for grass-eater, c-t for canine-teeth, and m-t for molar-teeth).

$B =$ {¬cow ∨ grass-eater,
¬dog∨ carnivore,
¬grass-eater ∨ ¬canine-teeth,
¬carnivore ∨ mammal,
¬mammal ∨ canine-teeth ∨ molar-teeth,
¬grass-eater ∨ mammal,
¬mammal ∨ vertebrate,
¬vertebrate ∨ animal}.

We start with $S = \emptyset$. When we get to line 14′, we need to add grass-eater to S in order to

close the right branch. We can then proceed applying rules until line 17, where one more atom, `canine-teeth`, must be added to S. What happens is that the tableau does not close if these atoms are not in S. During the development of the tableau we get clues about which atoms must be in S. When the atom `mammal` is added to S, the tableau closes and we have that $B \models^3_S \neg$`cow` $\vee$ `molar-teeth` for S= {`grass-eater`, `canine-teeth`, `mammal`}.

EXAMPLE 4.6
This example shows that formulas that are classically equivalent may have different behaviour under S_3. We transform the set B from Example 2.1 into a set B' which is classically equivalent to B but is not in clausal form. Then we try to check whether $B' \models^3_S$ `cow` $\rightarrow$ `molar-teeth`. Figure 6 shows the complete tableau. Note that this time, we can close the tableau adding only one atom to S, i.e. $B' \models^3_S$ `cow` $\rightarrow$ `molar-teeth` for S={`canine-teeth`}.

B' = {`cow` $\rightarrow$ `grass-eater`,
`dog`$\rightarrow$ `carnivore`,
`grass-eater` $\rightarrow \neg$`canine-teeth`,
`carnivore` $\rightarrow$ `mammal`,
`mammal` $\rightarrow$ `canine-teeth` $\vee$ `molar-teeth`,
`grass-eater` $\rightarrow$ `mammal`,
`mammal` $\rightarrow$ `vertebrate`,
`vertebrate` $\rightarrow$ `animal`}.

1.	T `cow` $\rightarrow$ `g-e`	
2.	T `dog`$\rightarrow$ `carnivore`	
3.	T `g-e` $\rightarrow \neg$`c-t`	
4.	T `carnivore` $\rightarrow$ `mammal`	
5.	T `mammal` $\rightarrow$ `c-t` $\vee$ `m-t`	
6.	T `g-e` $\rightarrow$ `mammal`	
7.	T `mammal` $\rightarrow$ `vertebrate`	
8.	T `vertebrate` $\rightarrow$ `animal`	
9.	F `cow` $\rightarrow$ `m-t`	
10.	T `cow`	from 9
11.	F `m-t`	from 9
12.	T `g-e`	from 1,10
13.	$T \neg$`c-t`	from 3,12
14.	T `mammal`	from 6,12
15.	T `c-t` $\vee$ `m-t`	from 5,14
16.	T `c-t`	from 11,15
17.	F `c-t`	from 13, `c-t` $\in S$
18.	$\times$	

FIGURE 6. KES_3 tableau with implication

4.3 Soundness and completeness of KES_3

It is very important to note that we are not proposing a simple *ad hoc* modification of a KE-tableau for doing theorem proving, but we are building a mechanism for approximate reasoning with a solid

logical basis. To sustain such a claim, we have to prove the soundness and completeness of the KES_3 tableau method of Section 4 with respect to the S_3 two-valued semantics of Section 2.2.

First, we need to define the notions of soundness and completeness. So KES_3 is *sound with respect to the S_3 semantics* if whenever a tableau closes for an input sequent, then the sequent's antecedent formulas entail its consequent in S_3. Conversely, the KES_3-tableau method is *complete with respect to the S_3 semantics* if for all sequents such that the antecedent entails the consequent in S_3, all KES_3-tableaux close.

We extend the valuation to signed formulas in the obvious way, that is, $v^3_S(T\alpha) = 1$ iff $v^3_S(\alpha) = 1$ and $v^3_S(F\alpha) = 1$ iff $v^3_S(\alpha) = 0$. A valuation satisfies a branch in a tableau if it simultaneously satisfies all the signed formulas in the branch.

To prove soundness, we first show the correctness of all linear expansion rules of KES_3.

LEMMA 4.7
If the antecedents of the KES_3 linear expansion rules are S-satisfied in S_3 by v^3_S so are its conclusions.

PROOF. A simple inspection of the rules in Figure 2 with the modification in $(T\ \neg)$ for KES_3 shows the result. ∎

We now show that the branching rule PB also preserve satisfiability.

LEMMA 4.8
If a branch is satisfied by a valuation v^3_S prior to the application of PB, then at least one of the two branches generated is satisfied by a valuation v^3_S after the application of PB.

PROOF. Suppose the branching occurs over the formula α. Because v^3_S is a function onto $\{0, 1\}$, we have that $v^3_S(T\ \alpha) = 1$ or $v^3_S(F\ \alpha) = 1$, so v^3_S satisfies one of the two branches generated by the application of PB. ∎

THEOREM 4.9 (Soundness)
Suppose a tableau for $\alpha_1, \ldots, \alpha_n \vdash \beta$ closes. Then $\alpha_1, \ldots, \alpha_n \models^3_S \beta$.

PROOF. We show the contrapositive. So suppose $\alpha_1, \ldots, \alpha_n \not\models^3_S \beta$, so there is a valuation v^3_S such that $v^3_S(\alpha_1) = \ldots = v^3_S(\alpha_1) = 1$ and $v^3_S(\beta) = 0$. In this case, the initial tableau for $\alpha_1, \ldots, \alpha_n \vdash \beta$ is such that all formulas $T\ \alpha_1, \ldots, T\ \alpha_n, F\ \beta$ are satisfied by v^3_S.

By Lemmas 4.7 and 4.8, we see that each application of an expansion rule preserves at least one satisfiable branch. As closed branches are not satisfiable, at least one branch remains open and the tableau cannot close. ∎

We say that a branch of a tableau is complete if there are no more applicable expansion rules.

LEMMA 4.10
An open complete branch in a KES_3-tableau is S-satisfiable in S_3.

PROOF. Given an open complete branch B, we construct the following valuation v^3_S, based on the propositional and negated formulas in B:

$$\begin{array}{lll} v^3_S(p) = 1 & \text{iff} & T\ p \in B \\ v^3_S(p) = 0 & \text{iff} & F\ p \in B \\ v^3_S(\neg\alpha) = 1 & \text{iff} & T\ \neg\alpha \in B, \text{and } \alpha \notin S. \end{array}$$

Since the tableau is open, we do not have that for the same atom q, both $T\ q$ and $F\ q$ are in B, so the valuation above is a partial function. To obtain a complete function, according to Lemma 2.6,

we need to define the value of v_S^3 for propositions and formulas of the form $\neg\alpha$ not occurring in B and not in S. We can set them all to true, respecting the semantics.

A simple structural induction on the signed formulas in B shows that v_S^3 satisfies the branch. ∎

THEOREM 4.11 (Completeness)
If $\alpha_1, \ldots, \alpha_n \models_S^3 \beta$ then any possible KES_3 tableau for $\alpha_1, \ldots, \alpha_n \vdash \beta$ closes.

PROOF. Suppose for contradiction that there is a tableau for $\alpha_1, \ldots, \alpha_n \vdash \beta$ with an open complete branch B. Then by Lemma 4.10 there is a S_3 valuation that satisfies B, which includes $T\ \alpha_1, \ldots, T\ \alpha_n, F\ \beta$, contradicting $\alpha_1, \ldots, \alpha_n \models_S^3 \beta$. ∎

We can adapt the tableaux system proposed here for extended S_3 to build a proof method for C_1. In the rule where we had the proviso $\alpha \in S$ we now have α^o, and we have to add a rule to deal with double negation. The study of this tableaux system is left for future work.

5 The dynamics of approximations: expressivity versus control

In this section we compare KES_3 with the Cadoli-Schaerf (CSS_3) method with regards to *expressivity* (i.e. the theorems proved for the same set S) and with the *control* that the set S exerts over the proof development.

5.1 Dynamic properties

A property that tells us in which direction to expand our limited resource to achieve a goal is a *dynamic* property of the method. KES_3 provides a method for expanding S in trying to prove a theorem, namely: *'If a branch is closed due to a blocked use of $(T\neg)$, add the blocking formula α to S, so as to unblock that branch.'*

Cadoli and Schaerf did not provide a dynamic extension for their system. However, to be honest with their excellent work, such a dynamic behaviour can easily be provided in analogy to ours. In CSS_3, we can resolve $\alpha \vee l$ with $\neg l \vee \beta$ only if $l \in S$, which gives us the dynamic rule: *'If resolution is blocked due to the absence of resolvents in S, add a potential resolvent l to S, so as to unblock resolution.'*

With that formulation, we compare the dynamics of KES_3 and CSS_3 for conjunctive normal form formulas. We note that both methods are highly non-deterministic in their behaviour of choosing branch expansion rules and resolvents.

Suppose the size of a CSS_3 proof is measured by the number of resolution steps, and the size of a KES_3 is measured by the number of expansion rules applied.

THEOREM 5.1
Let B, α be a set of clauses and a clause. In a proof of $B \vdash \alpha$, KES_3 can linearly simulate the dynamics of CSS_3, generating the same S.

PROOF. Every clause $\beta \in B$ in CSS_3 is associated with $T\ \beta$ in KES_3 and α is associated with $F\ \alpha$. In [14] every atom in α was implicitly considered part of S, so in order to compare both systems, we have to start by putting those atoms in S. This is important for KES_3 to simulate CSS_3, for suppose $\neg p$ is a disjunct in α and the final step of a CSS_3 is to resolve $\neg p$ with p. This would correspond to an expansion involving $T\ \neg p$ and $T\ p$, and the tableau can only close if $p \in S$, with an intermediate step taking $T\ \neg p$ into $F\ p$, which closes with $T\ p$.

Having that in mind, it suffices to show that the three possible CSS_3 resolution steps can be

simulated in KES_3; in classical presentations, there is only a single resolution rule; in non-classical ones, all non-equivalent formats need to be considered:

$$\frac{\alpha \vee l \quad \neg l}{\alpha\ (l \in S)} \qquad \frac{\neg l \vee \alpha \quad l}{\alpha\ (l \in S)} \qquad \frac{\neg l \vee \alpha \quad l \vee \beta}{\alpha \vee \beta\ (l \in S)}$$

can be simulated in KES_3 as:

$$\frac{\begin{array}{c} T\ \alpha \vee l \\ T\ \neg l \\ F\ \alpha \end{array}}{\begin{array}{c} F\ l\ (l \in S) \\ T\ \alpha \\ \times \end{array}} \qquad \frac{\begin{array}{c} T\ \neg l \vee \alpha \\ T\ l \\ F\ \alpha \end{array}}{\begin{array}{c} T\ \neg l \\ F\ l\ (l \in S) \\ \times \end{array}} \qquad \frac{\begin{array}{c} T\ \neg l \vee \alpha \\ T\ l \vee \beta \\ F\ \alpha \vee \beta \end{array}}{\begin{array}{c} F\ \alpha \\ F\ \beta \\ T\ \neg l \\ F\ l\ (l \in S) \\ T\ l \\ \times \end{array}}$$

Only a constant number of steps was used, independent of the form of α and β (so as to achieve linear simulation), without multiplying the number of branches and by adding exactly the same formulas to S in each step. ∎

The proof above shows that every possible approximation $S^0 \subset \ldots \subset S^k$ in CSS_3 is also possible in KES_3. However, because KES_3 deals with a larger language, several transformational tricks can be used in KES_3 to improve its *static expressivity* which cannot be simulated by CSS_3.

5.2 Static expressivity

The *static expressivity* of a method is the set of theorems it can prove with a fixed limited resource. In our case, we are going to compare the *set of theorems that can be proved* with a given S.

The idea is to use the larger language of KES_3 to rewrite $\neg l \vee \alpha$ as $l \rightarrow \alpha$. Since α may be a large disjunction, we may not know *a priori* which negative literal to transform, so this transformation is assumed to be applied 'on the fly' during theorem proving.

We can now show that for a fixed S, and formulas in $\rightarrow$-clausal form, KES_3 can prove more theorems.

THEOREM 5.2
Let B, α be a set of formulas and a formula in CNF. Suppose CSS_3 proves $B \vdash \alpha$ with S. Suppose KE applies the transformation above to clauses with one or more negative literal. Then KES_3 proves $B \vdash \alpha$ in time linear w.r.t. the time needed by CSS_3, with an $S' \subseteq S$; it is possible that $S' \subset S$.

PROOF. It suffices to note that, in the simulation of CSS_3-resolution, because $l \rightarrow \alpha$ contains no negated literal and need not be converted to $\neg l \vee \alpha$, one need not always add l to S' (see Figure 7). ∎

This does not only mean that the static expressivity of KES_3 is higher, but also that CSS_3 may not simulate any expansion $S^0 \subset \ldots \subset S^k$ in KES_3: KES_3 may proceed without the addition of new elements to S at points where CSS_3 is surely blocked and needs S to be expanded.

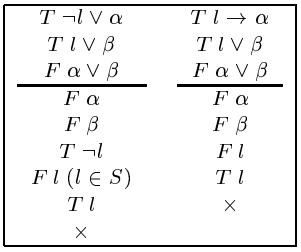


FIGURE 7. Simulation of CSS_3 by KES_3

As an example for this result, consider the tableaux in Figure 5 and Figure 6. The $\vee$-clauses have been *a priori* transformed to $\rightarrow$, but the same could have been done on-the-fly. The tableau shown in Figure 6 ends up with $S = \{\texttt{canine-teeth}\}$, a subset from the S computed in Example 2.1.

What was the price payed for such an increase of static expressivity? The answer is: *loss of control* in the deduction process.

The *sensitivity* of a proof method depends on the set of new theorems ΔT we get when we move from S to $S \cup \Delta S$. Proof method 1 has *more control* than method 2 if it has more sensitivity, that is, if for the same ΔS, $\Delta T_1 \subseteq \Delta T_2$. Note that sensitivity and control are also dynamic properties.

In CSS_3, the set S has an effect over (i.e. controls) the set of atoms over which resolution can be applied. In KES_3, the set S controls the formulas over which $(T\neg)$ can be applied; by applying the transformation above, we eliminate $\neg$-formulas and thus reduce the control of S on KES_3 proofs. If we add to S an atom that only occurs non-negated in $B \vdash \alpha$, no new theorems are obtained in KES_3. We have thus shown the following:

THEOREM 5.3

- KES_3 is more expressive than CSS_3.
- CSS_3 has more control than KES_3.

5.3 Recovering control

We have seen that although the extension proposed to S_3 allows for more expressivity, we end up losing control over the resources used. Cadoli and Schaerf use resolution as the only inference rule and the set S determines the set of atoms over which resolution may be applied. In our system, modus ponens is valid even if S is empty, i.e. $\alpha \rightarrow \beta, \beta \vdash_{\mathrm{KE}S_3} \beta$.

If we want to regain control, we can add a restriction to the application of modus ponens, so that we always need part of the formulas to be in S. We end up with rules like these:

$$\frac{T\ \alpha \rightarrow \beta \quad T\ \alpha}{T\ \beta \text{ if } \alpha \in S^T_{\rightarrow}} \qquad \frac{T\ \neg\alpha}{F\ \alpha \text{ if } \alpha \in S^T_{\neg}}$$

where $S = S^T_{\neg} \cup S^T_{\rightarrow}$. This blocks the use of transformation rule in Theorem 5.2, and the two systems can clearly simulate each other.

EXAMPLE 5.4

If we apply the new rule $(T\ \rightarrow)$ to the tableau in Figure 6, every line in which $(T\ \rightarrow)$ was applied would cause an expansion in $S^T_{\rightarrow}$ at each such line, namely:

- `cow` is added at line 12
- `grass-eater` is added at line 13
- `mammal` is added at line 15
- `canine-teeth` is added at line 17

And since in [14] every atom in α was implicitly considered part of S, we end up with the same S-set as in Example 2.1.

We should not stop here at gaining control over the approximation process. It is clear that we can get a deduction system with a much finer control over deductions, finer then either KES_3 or CSS_3.

6 A generalized approximation inference

We have moved from restricting $(T\neg)$ in KES_3, to restricting the use of both $(T\neg)$ and $(T\ \rightarrow)$. If we are willing to have the size of the set S as an indication of the inference rules allowed, we could go even further and add restrictions to every rule of our system. And still further: we may have different context sets for each rule. This would give us a system where every connective behaves classically only for formulas which belong to the corresponding context set: $S^T_\wedge$, $S^F_\wedge$, $S^T_\vee$, $S^F_\vee$, $S^T_\rightarrow$, $S^F_\rightarrow$, $S^T_\neg$ and $S^F_\neg$.

We propose here a generalization of the KES_3 tableaux, that we call KES_e. The system is obtained by adding restrictions to each expansion rule, as illustrated in Figure 8.

$$\frac{T\ \alpha \rightarrow \beta \quad T\ \alpha}{T\ \beta \text{ if } \alpha \in S^T_\rightarrow}\ (T\rightarrow_1) \qquad \frac{T\ \alpha \rightarrow \beta \quad F\ \beta}{F\ \alpha \text{ if } \beta \in S^T_\rightarrow}\ (T\rightarrow_2) \qquad \frac{F\ \alpha \rightarrow \beta}{\begin{array}{l}T\ \alpha \text{ if } \alpha \in S^F_\rightarrow \\ F\ \beta \text{ if } \beta \in S^F_\rightarrow\end{array}}\ (F\rightarrow)$$

$$\frac{F\ \alpha \wedge \beta \quad T\ \alpha}{F\ \beta \text{ if } \alpha \in S^F_\wedge}\ (F\wedge_1) \qquad \frac{F\ \alpha \wedge \beta \quad T\ \beta}{F\ \alpha \text{ if } \beta \in S^F_\wedge}\ (F\wedge_2) \qquad \frac{T\ \alpha \wedge \beta}{\begin{array}{l}T\ \alpha \text{ if } \alpha \in S^T_\wedge \\ T\ \beta \text{ if } \beta \in S^T_\wedge\end{array}}\ (T\wedge)$$

$$\frac{T\ \alpha \vee \beta \quad F\ \alpha}{T\ \beta \text{ if } \alpha \in S^T_\vee}\ (T\vee_1) \qquad \frac{T\ \alpha \vee \beta \quad F\ \beta}{T\ \alpha \text{ if } \beta \in S^T_\vee}\ (T\vee_2) \qquad \frac{F\ \alpha \vee \beta}{\begin{array}{l}F\ \alpha \text{ if } \alpha \in S^F_\vee \\ F\ \beta \text{ if } \beta \in S^F_\vee\end{array}}\ (F\vee)$$

$$\frac{T\ \neg\alpha}{F\ \alpha \text{ if } \alpha \in S^T_\neg}\ (T\neg) \qquad \frac{F\ \neg\alpha}{T\ \alpha \text{ if } \alpha \in S^F_\neg}\ (F\neg)$$

$$\frac{}{T\ \alpha \quad F\ \alpha}\ (\text{PB})$$

FIGURE 8. KE-rules for the generalized system

THEOREM 6.1
KES_e can simulate the dynamic evolution of both CSS_3 and KES_3. Also, for a given S, the expressivity of KE$S_3(S)$ can be simulated by a suitable instantiation of the S-parameters of KES_e.

PROOF. To see that KES_e can simulate the dynamic evolution of KES_3, it suffices to set $S = S^T_\neg$ and all other S parameters to the full set of propositional letters. In practice, this amounts to lifting the proviso of all rules except for the $(T\neg)$ rule. Since KES_3 simulates the dynamics of CSS_3, we have that KES_e can simulate the dynamic evolution of both CSS_3 and KES_3.

For a fixed S, we can simulate the expressivity of KE$S_3(S)$ by setting $S^T_\neg = S$ and all other S parameters to the full set of propositional letters (actually, it is enough to take the set of propositional letters that appear in the initial tableau). ∎

As expected, there is a tradeoff between expressivity and control. KES_e gives us a very fine control, but a limited expressivity for a given collection of S^*_*-sets. Basically, we may need to add to some S^*_*-set at almost every expansion step in the tableau. As usual, we want our system to be based on a sound and complete subclassical semantics

6.1 Semantics for generalized approximate inference

DEFINITION 6.2
An S_e-*valuation* v^e_S is a function, $v^e_S : \mathcal{L}_C \to \{0,1\}$, that extends a propositional valuation v_p (i.e. $v^e_S(p) = v_p(p)$), satisfying the following restrictions:

$$
\begin{array}{lllll}
(\wedge_1) & v^e_S(\alpha \wedge \beta) = 1, \alpha \in S^T_\wedge & \Rightarrow & v^e_S(\alpha) = 1 \\
(\wedge_2) & v^e_S(\alpha \wedge \beta) = 1, \beta \in S^T_\wedge & \Rightarrow & v^e_S(\beta) = 1 \\
(\wedge_3) & v^e_S(\alpha \wedge \beta) = 0, v^e_S(\alpha) = 1, \alpha \in S^F_\wedge & \Rightarrow & v^e_S(\beta) = 0 \\
(\wedge_4) & v^e_S(\alpha \wedge \beta) = 0, v^e_S(\beta) = 1, \beta \in S^F_\wedge & \Rightarrow & v^e_S(\alpha) = 0 \\
(\vee_1) & v^e_S(\alpha \vee \beta) = 0, \alpha \in S^F_\vee & \Rightarrow & v^e_S(\alpha) = 0 \\
(\vee_2) & v^e_S(\alpha \vee \beta) = 0, \beta \in S^F_\vee & \Rightarrow & v^e_S(\beta) = 0 \\
(\vee_3) & v^e_S(\alpha \vee \beta) = 1, v^e_S(\alpha) = 0, \alpha \in S^T_\vee & \Rightarrow & v^e_S(\beta) = 1 \\
(\vee_4) & v^e_S(\alpha \vee \beta) = 1, v^e_S(\beta) = 0, \beta \in S^T_\vee & \Rightarrow & v^e_S(\alpha) = 1 \\
(\rightarrow_1) & v^e_S(\alpha \rightarrow \beta) = 0, \alpha \in S^F_\rightarrow & \Rightarrow & v^e_S(\alpha) = 1 \\
(\rightarrow_2) & v^e_S(\alpha \rightarrow \beta) = 0, \beta \in S^F_\rightarrow & \Rightarrow & v^e_S(\beta) = 0 \\
(\rightarrow_3) & v^e_S(\alpha \rightarrow \beta) = 1, v^e_S(\alpha) = 1, \alpha \in S^T_\rightarrow & \Rightarrow & v^e_S(\beta) = 1 \\
(\rightarrow_4) & v^e_S(\alpha \rightarrow \beta) = 1, v^e_S(\beta) = 0, \beta \in S^T_\rightarrow & \Rightarrow & v^e_S(\alpha) = 0 \\
(\neg_1) & v^e_S(\neg\alpha) = 0, \alpha \in S^F_\neg & \Rightarrow & v^e_S(\alpha) = 1 \\
(\neg_2) & v^e_S(\neg\alpha) = 1, \alpha \in S^T_\neg & \Rightarrow & v^e_S(\alpha) = 0
\end{array}
$$

It is easy to see that the extension of S_3 given in Sections 2.2 and 4 is a particular case of the system above, where the sets $S^T_\wedge$, $S^F_\wedge$, $S^T_\vee$, $S^F_\vee$, $S^T_\rightarrow$, $S^F_\rightarrow$, and $S^F_\neg$ contain all the propositional letters of the language and $S = S^T_\neg$.

6.2 Soundness and completeness

We say that KES_e is *sound with respect to the S_e semantics* if whenever a tableau closes for an input sequent, then the sequent's antecedent formulas entail its consequent in S_e. Conversely, the KES_e-tableau method is *complete with respect to the S_e semantics* if for all sequents such that the the antecedent entails the consequent in S_e, all KES_e-tableaux close.

We extend an S_e-valuation to signed formulas making $v^e_S(T\alpha) = 1$ iff $v^e_S(\alpha) = 1$ and $v^e_S(F\alpha) = 1$ iff $v^e_S(\alpha) = 0$. A valuation satisfies a branch in a tableau if it simultaneously satisfies all the signed formulas in the branch.

To prove soundness, we first show the correctness of all linear expansion rules of KES_e.

LEMMA 6.3
If the antecedents of the KES_e linear expansion rules are S-satisfied in S_e by v^e_S so are its conclusions.

PROOF. A simple inspection of the rules in Figure 8 shows the result. ∎

We now show that the branching rule PB also preserve satisfiability.

LEMMA 6.4
If a branch is satisfied by a valuation v_S^e prior to the application of PB, then at least one of the two branches generated is satisfied by a valuation v_S^e after the application of PB.

PROOF. Suppose the branching occurs over the formula α. Because v_S^e is a function onto $\{0,1\}$, we have that $v_S^e(T\ \alpha) = 1$ or $v_S^e(F\ \alpha) = 1$, so v_S^e satisfies one of the two branches generated by the application of PB. ∎

THEOREM 6.5 (Soundness)
Suppose a tableau for $\alpha_1, \ldots, \alpha_n \vdash \beta$ closes. Then $\alpha_1, \ldots, \alpha_n \models_S^e \beta$.

PROOF. We show the contrapositive. So suppose $\alpha_1, \ldots, \alpha_n \not\models_S^e \beta$, so there is a valuation v_S^e such that $v_S^e(\alpha_1) = \ldots = v_S^e(\alpha_1) = 1$ and $v_S^e(\beta) = 0$. In this case, the initial tableau for $\alpha_1, \ldots, \alpha_n \vdash \beta$ is such that all formulas $T\ \alpha_1, \ldots, T\ \alpha_n, F\ \beta$ are satisfied by v_S^e.

By Lemmas 6.3 and 6.4, we see that each application of an expansion rule preserves at least one satisfiable branch. As closed branches are not satisfiable, at least one branch remains open and the tableau cannot close. ∎

We say that a branch of a tableau is complete if there are no more applicable expansion rules.

LEMMA 6.6
An open complete branch in a KES_e-tableau is S-satisfiable in S_e.

PROOF. Let B be the set of formulas that occur in the open complete branch. We have to build an S_e valuation that satisfies it.

We start the construction of v by setting $v(\alpha) = 1$ for every α such that $T\ \alpha \in B$ and $v(\alpha) = 0$ for every α such that $F\ \alpha \in B$. Since the branch is open and complete, there is no formula α for which both $T\ \alpha$ and $F\ \alpha$, hence v is a partial function that satisfies the branch. We have to (i) extend v to a total function and (ii) show that v is an S_e-valuation.

(i) We can extend v by combining it with any classical propositional valuation. Let v_p be a propositional valuation. If $v(p)$ is undefined, then set $v(p)$ to $v_p(p)$. We then extend v by combining it with a classical extension of v_p. Suppose $\gamma = \alpha \wedge \beta$ and $v(\gamma)$ undefined. We set $v(\gamma)$ to 1 iff $v(\alpha) = v(\beta) = 1$. Analogously for the other connectives.

(ii) To show that v is an S_e-valuation, we must show that it satisfies the properties in Definition 6.2. For example, take property $(\wedge_1)$:

$$v_S^e(\alpha \wedge \beta) = 1, \beta \in S_\wedge^T \Rightarrow v_S^e(\alpha) = 1.$$

If $T\ \alpha \wedge \beta$ is in B, then since $\alpha \in S_\wedge^T$, by the rule $(T_\wedge)$, $T\ \alpha$ is also in B and hence, $v(\alpha) = 1$. If $T\ \alpha \wedge \beta$ is not in B, then $v(\alpha \wedge \beta) = 1$ iff $v(\alpha) = 1$ and $v(\beta) = 1$. Consider rule $(\neg_1)$:

$$v_S^e(\neg\alpha) = 0, \alpha \in S_\neg^F \Rightarrow v_S^e(\alpha) = 1.$$

If $F\ \neg\alpha$ is in B and $\alpha \in S_\neg^F$, then by the rule $(F_\neg)$, $T\ \alpha$ is in B and hence, $v(\alpha) = 1$. If $F\ \neg\alpha$ is not in B, then $v(\neg\alpha) = 0$ iff $v(\alpha) = 1$.

In an analogous way, we can show that the valuation v satisfies the other twelve properties in Definition 6.2. ∎

THEOREM 6.7 (Completeness)
If $\alpha_1, \ldots, \alpha_n \models^e_S \beta$ then any possible KES_e tableau for $\alpha_1, \ldots, \alpha_n \vdash \beta$ closes.

PROOF. Suppose for contradiction that there is a tableau for $\alpha_1, \ldots, \alpha_n \vdash \beta$ with an open complete branch B. Then by Lemma 6.6 there is an S_e valuation that satisfies B, which includes $T\ \alpha_1, \ldots, T\ \alpha_n, F\ \beta$, contradicting $\alpha_1, \ldots, \alpha_n \models^e_S \beta$. ∎

7 Conclusions and future work

We have extended Schaerf and Cadoli's system S_3 of approximate entailment to deal with full propositional logic. We extended their semantics, provided a sound and complete axiomatization and a sound and complete proof method based on a computationally efficient version of semantic tableaux.

Extending the system from the clausal fragment to full propositional logic does not preserve the good complexity bounds provided by Schaerf and Cadoli. But as can easily be seen from the proof method, KES_3 is to classical logic as S_3 was to the clausal fragment, i.e. the approximation proceeds in an incremental way and in the worst case a proof is as hard as in classical logic. Moreover, the tableaux system can be easily restricted to the clausal fragment, putting together the complexity upper bound provided by Schaerf and Cadoli and the heuristics given here for augmenting the set S.

We have analysed our extension and the gains and losses with respect to the original system. While being more expressive than Schaerf and Cadoli's, our system allows for less control in the approximation process.

We have then developed a general framework for modelling limited reasoning that extends the idea of having a context set even further. Instead of a single set, we have now several contexts that have to be set in order to determine the logic in which we are working. Having all these different context sets for different inference rules gives us the possibility of setting limits to each sort of inference. The application of each rule of inference may have different costs.

What is missing now is a work of logical engineering: for particular applications, determine the costs and parameter sets for each inference rule.

The implementation of a theorem prover based on KES_3 has been completed. Ongoing work consists of parameter and strategy testing of this prover, as well as an adaptation of the prover to KES_e.

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 was developed under the CNPq project APQ 468765/00-0.

References

[1] A.R Anderson and N.D Belnap. *Entailment: The Logic of Relevance and Necessity, Vol. 1*. Princeton University Press, 1975.

[2] K. Broda and M. Finger. KE-tableaux for a fragment of linear logic. In *Proceedings of the Fourth International Workshop on Analytic Tableaux and Related Methods*, Koblenz, May 1995.

[3] K. Broda, M. Finger, and A. Russo. Labelled natural deduction for substructural logics. *Logic Journal of the IGPL*, **7**, 283–318, 1999.

[4] M. Cadoli and M. Schaerf. Approximate inference in default logic and circumscription. *Fundamenta Informaticae*, **23**, 123–143, 1995.

[5] M. Cadoli and M. Schaerf. The complexity of entailment in propositional multivalued logics. *Annals of Mathematics and Artificial Intelligence*, **18**, 29–50, 1996.

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

[7] N. C. A. da Costa and E. H. Alves. A semantical analysis of the calculi C_n. *Notre Dame Journal of Formal Logic*, **16**, 621–630, 1977.

[8] N. C.A. da Costa. Calculs propositionnels pour les systémes formels inconsistants. *Comptes Rendus d'Academie des Sciences de Paris*, **257**, 3790–3792, 1963.

[9] M. D'Agostino and D. Gabbay. A generalization of analytic deduction via labelled tableaux, part I: basic substructural logics. *Journal of Automated Reasoning*, **13**, 243–281, 1994.

[10] 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.

[11] M. D'Agostino. Tableau methods for classical propositional logic. In *Handbook of Tableau Methods*, M. D'Agostino, D. Gabbay, R. Haehnle, and J. Posegga, eds, pp. 45–124. Kluwer, 1999.

[12] M. D'Agostino and M. Mondadori. The taming of the cut: Classical refutations with analytic cut. *Journal of Logic and Computation*, **4**, 285–319, 1994.

[13] H. Levesque. A logic of implicit and explicit belief. In *Proceedings of AAAI-84*, 1984.

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

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

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

Received 11 March 2002