

The Universe of Approximations

Marcelo Finger^{1,2}

*Departamento de Ciência da Computação
Universidade de São Paulo
São Paulo, Brasil*

Renata Wassermann³

*Departamento de Ciência da Computação
Universidade de São Paulo
São Paulo, Brasil*

Abstract

The idea of *approximate entailment* has been proposed in [13] as a way of modeling the reasoning of an agent with limited resources. They proposed a system in which a family of logics, parameterized by a set of propositional letters, approximates classical logic as the size of the set increases.

In this paper, we take the idea further, extending two of their systems to deal with full propositional logic, giving them semantics and sound and complete proof methods based on tableaux. We then present a more general system of which the two previous systems are particular cases and show how it can be used to formalize heuristics used in theorem proving.

Keywords: Automated Reasoning, Deductive Systems, Approximate Reasoning, Non-classical Logics, Knowledge Representation.

1 Introduction

Logic has been used in several areas of Artificial Intelligence as a tool for representing knowledge as well as a tool for modeling agents' reasoning. Ideal agents know all the logical consequences of their beliefs. However, real agents are *limited* in their capabilities. Due to these limitations, a real rational agent must devise some strategy to make good use of the available resources.

¹ Marcelo Finger is partly supported by the Brazilian Research Council (CNPq), grant PQ 300597/95-5. Renata Wassermann is partly supported by CNPq grant 68.0099/01-8.

² Email: mfinger@ime.usp.br

³ Email: renata@ime.usp.br

In this work we propose a general framework for modeling limited reasoning, and show two systems which are special cases of the more general framework. Our system is based on Cadoli and Schaerf's *approximate entailment* [13]. 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 [13], 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.

Our starting point here is their family of logics S_3 . S_3 is 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 $S^0 \subseteq S^1 \subseteq \dots \subseteq \mathcal{P}$; let $Th(L)$ indicate the set of theorems of a logic. Then, by means of successive approximations:

$$Th(S_3(\emptyset)) \subseteq Th(S_3(S^0)) \subseteq Th(S_3(S^1)) \subseteq \dots \subseteq Th(S_3(\mathcal{P}))$$

where $Th(S_3(\mathcal{P})) = Th(CL)$ is the set of classical theorems. From this property, we see that it suffices to prove a result in some S_3 -approximation to have a classically valid theorem.

Approximate entailment has been used to formalize approximate diagnosis [15] and belief revision [5]. However, the knowledge had to be encoded in clausal form. It happens that each approximation step is characterized by a formal logic. The final step of the approximations is classical logic, in which every formula is equivalent to one in clausal form. However, in *none* of the intermediate systems such equivalence holds.

The original system has been extended to deal with full propositional logic in [10]. In this paper, we extend another system, S_3^* , which was introduced in [4]. We provide a proof method based on tableaux for extended S_3^* and then show that both S_3 and S_3^* are particular cases of a system that we call S_e . We provide semantics and a tableaux method for S_e . We then show how this general system can be used to formalize different heuristics used in theorem proving.

Most proofs are omitted due to space limitations.

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 The Family S_3

In this section, we first present Cadoli and Schaerf's system S_3 and then the extended version that deals with full propositional logic.

Schaerf and Cadoli [13] define two approximations of classical entailment: \models_S^1 which is complete but not sound, and \models_S^3 which is classically sound but incomplete. These approximations are carried out over a set of atoms $S \subseteq \mathcal{P}$ which determines their closeness to classical entailment. Here we will concentrate only in the latter, namely the S_3 family of logics.

In the trivial extreme of S_3 approximate entailment, i.e., when $S = \mathcal{P}$, classical entailment is obtained. At the other extreme, \models_\emptyset^3 corresponds to Levesque's logic for explicit beliefs [12], 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 possible truth assignments for literals outside S . If $p \notin S$, there are three possible S_3 assignments, the two classical ones, assigning p and $\neg p$ opposite truth values, and an extra one, making them both true. The set of formulas for which we are testing entailments is assumed to be in clausal form.

Formally, the semantics of the logic $S_3(S)$ over clauses is constructed by defining an S_3 -valuation of literals into $\{0, 1\}$ such that:

- $v_S^3(\neg p) = 1$ iff $v_S^3(p) = 0$, if $p \in S$.
- if $p \notin S$, we can have one of 3 possibilities:
 - $v_S^3(\neg p) = 1$ and $v_S^3(p) = 0$
 - $v_S^3(\neg p) = 0$ and $v_S^3(p) = 1$
 - $v_S^3(\neg p) = v_S^3(p) = 1$

This valuation can be generalized simply to clauses. By varying S , we generate a family of logics. Also, satisfiability, validity and entailment are defined in the usual way.

Although in classical logic any formula is equivalent to one in clausal form, the usual transformation does not preserve truth-values under the non-standard S_3 semantics. The S_3 family of logics has been extended to propositional formula in [10], where a sound and complete incremental proof system for it was also provided.

The generalized semantics for S_3 is the following:

Definition 2.1 An S_3 -valuation v_S^3 is a function, $v_S^3 : \mathcal{L}_C \rightarrow \{0, 1\}$, that extends a propositional valuation v_p (i.e., $v_S^3(p) = v_p(p)$), satisfying the following

restrictions:

- (i) $v_S^3(\alpha \wedge \beta) = 1 \iff v_S^3(\alpha) = v_S^3(\beta) = 1$
- (ii) $v_S^3(\alpha \vee \beta) = 0 \iff v_S^3(\alpha) = v_S^3(\beta) = 0$
- (iii) $v_S^3(\alpha \rightarrow \beta) = 0 \iff v_S^3(\alpha) = 1$ and
 $v_S^3(\beta) = 0$
- (iv) $v_S^3(\neg\alpha) = 0 \implies v_S^3(\alpha) = 1$
- (v) $v_S^3(\neg\alpha) = 1, \alpha \in S \implies v_S^3(\alpha) = 0$

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 behavior, or 1, in which case α behaves paraconsistently. A comparison between S_3 semantics and axiomatization and da Costa’s Paraconsistent Logic C_1 was done in [9].

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 *S₃-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 $\models_S^3 \alpha$.

An inference system for the full logic S_3 based on the KE-tableau methodology was developed in [10] and further developed in [11]. KE-tableaux were introduced by D’Agostino [7] as a principled computational improvement over Smullyan’s Semantic Tableaux [14], and have since been successfully applied to a variety of logics [6,2,3].

KE-tableaux deal with *T*- and *F*-signed formulas. An expansion of a tableau is allowed when the premises 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 premises 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 premise, and may also have an auxiliary premise. They may have one or two consequences. The only branching rule is the *Principle of Bivalence*, stating that a formula has to be either true or false. Figure 1 shows KE-tableau expansion rules for the family

S_3 .

$\frac{T \alpha \rightarrow \beta}{T \alpha} (T \rightarrow_1)$	$\frac{T \alpha \rightarrow \beta}{F \beta} (T \rightarrow_2)$	$\frac{F \alpha \rightarrow \beta}{T \alpha} (F \rightarrow)$
$T \beta$	$F \alpha$	$F \beta$
$\frac{F \alpha \wedge \beta}{T \alpha} (F \wedge_1)$	$\frac{F \alpha \wedge \beta}{T \beta} (F \wedge_2)$	$\frac{T \alpha \wedge \beta}{T \alpha} (T \wedge)$
$F \beta$	$F \alpha$	$T \beta$
$\frac{T \alpha \vee \beta}{F \alpha} (T \vee_1)$	$\frac{T \alpha \vee \beta}{F \beta} (T \vee_2)$	$\frac{F \alpha \vee \beta}{F \alpha} (F \vee)$
$T \beta$	$T \alpha$	$F \beta$
$\frac{T \neg \alpha}{F \alpha} (T \neg)$	$\frac{F \neg \alpha}{T \alpha} (F \neg)$	
$\frac{}{T \alpha \quad F \alpha} (PB)$		

 Fig. 1. KE-rules for S_3

The only way in which such a tableau system differs from a classical one is in the $(T \neg)$ rule, which comes with a proviso:

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

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 $KE S_3$.

This makes our system immediately subclassical, for any tableau that closes for $KE S_3$ also closes for classical logic. So any theorems we prove in $KE S_3$ are also classical theorems.

So $KE S_3$ is correct and incomplete with respect to classical logic. In fact, $KE S_3$ is complete and correct with respect to the semantics above.

Theorem 2.2 ([10]) $\alpha_1, \dots, \alpha_n \models_S^3 \beta$ iff any possible $KE S_3$ tableau for $\alpha_1, \dots, \alpha_n \vdash \beta$ closes. Furthermore, if one S_3 tableau for $\alpha_1, \dots, \alpha_n \vdash \beta$ closes, any such tableau closes.

3 The Dual Family S_3^*

Cadoli and Schaerf in a subsequent work [4] have proposed a dual family of logics which they called S_3^* . An S_3^* -valuation of literals into $\{0, 1\}$ such that:

- $v_S^{3*}(\neg p) = 1$ iff $v_S^{3*}(p) = 0$, if $p \in S$.
- if $p \notin S$, we can have one of 3 possibilities:
 - $v_S^{3*}(\neg p) = 1$ and $v_S^{3*}(p) = 0$
 - $v_S^{3*}(\neg p) = 0$ and $v_S^{3*}(p) = 1$
 - $v_S^{3*}(\neg p) = v_S^{3*}(p) = 0$

Only this last line differs from the previous S_3 family, in that for an atom $p \notin S$, both p and $\neg p$ may be false. As a result, in such a logic, the formula $p \vee \neg p$ is not valid for $p \notin S$, which characterizes such logics as *paracomplete*. This logic was presented in [4] with the same setting as S_3 was presented: formulas in clausal form only (in fact, negation normal form was also accepted); no extension to full logic; no proof theory.

In an analogous way to our extension of S_3 , we extend here S_3^* to full propositional logic.

Definition 3.1 An S_3^* -valuation v_S^{3*} is a function, $v_S^{3*} : \mathcal{L}_C \rightarrow \{0, 1\}$, that extends a propositional valuation v_p (i.e., $v_S^{3*}(p) = v_p(p)$), satisfying the following restrictions:

- (i) $v_S^{3*}(\alpha \wedge \beta) = 1 \iff v_S^{3*}(\alpha) = v_S^{3*}(\beta) = 1$
- (ii) $v_S^{3*}(\alpha \vee \beta) = 0 \iff v_S^{3*}(\alpha) = v_S^{3*}(\beta) = 0$
- (iii) $v_S^{3*}(\alpha \rightarrow \beta) = 0 \iff v_S^{3*}(\alpha) = 1$ and $v_S^{3*}(\beta) = 0$
- (iv) $v_S^{3*}(\neg\alpha) = 0, \alpha \in S \Rightarrow v_S^{3*}(\alpha) = 1$
- (v) $v_S^{3*}(\neg\alpha) = 1 \Rightarrow v_S^{3*}(\alpha) = 0$

The definition of S_3^* -logical consequence, \models_S^{3*} is totally analogous to that of \models_S^3 .

Also, in an analogous way, we define a KE-tableau proof system for the S -parameterized family of logics S_3^* . The rules for the connectives \rightarrow , \wedge and \vee are the same as in S_3 (which are the same as the classical rules). The rules for negation are now:

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

The rule $(T \neg)$ is the classical one, while $(F \neg)$ comes with a proviso. This is dual to the situation in S_3 .

As an example we illustrate two tableaux. The first one is the principle of contradiction, $p, \neg p \vdash q$, which was not a theorem in paraconsistent S_3 but is now a theorem in S_3^* . The second example is the principle of excluded middle $\vdash p \vee \neg p$, which was a theorem in S_3 , but which is not a theorem in paracomplete S_3^* .

$$\begin{array}{cc}
 T p & F p \vee \neg p \\
 T \neg p & F p \\
 F q & F \neg p \\
 F p & - \\
 \times &
 \end{array}$$

In the first tableau, we simply apply $(T \neg)$ to the second line to close the tableau. In the second tableau, we would want to apply $(F \neg)$, but since we consider $S = \emptyset$, the proviso precludes us from doing that, and the tableau remains open.

We have the following soundness and completeness result for $KE S_3^*$ tableaux.

Theorem 3.2 $\alpha_1, \dots, \alpha_n \models_S^{\beta} \beta$ iff any possible $KE S_3^*$ tableau for $\alpha_1, \dots, \alpha_n \vdash \beta$ closes. Furthermore, if one S_3^* tableau for $\alpha_1, \dots, \alpha_n \vdash \beta$ closes, any such tableau closes.

From the way $KE S_3^*$ was built, it is clear that it is also an approximation of classical logic from below. Also, it appears that S_3 and S_3^* are incompatible families, due to the following properties, that come straight from the definitions of v_S^3 and v_S^{3*} .

Lemma 3.3 For $\alpha \notin S$:

- (a) In S_3 , if $v_S^3(\alpha) = 1$, then $v_S^3(\neg\alpha)$ may be either 0 or 1.
- (b) In S_3^* , if $v_S^{3*}(\alpha) = 1$, then $v_S^{3*}(\neg\alpha) = 0$.
- (c) In S_3^* , if $v_S^3(\alpha) = 0$, then $v_S^3(\neg\alpha)$ may be either 0 or 1.
- (d) In S_3 , if $v_S^{3*}(\alpha) = 0$, then $v_S^3(\neg\alpha) = 1$.

However, as we are going to see, this does not consist in any kind of incompatibility, and we may have systems that obey both rules.

If we concentrate on both tableau methods that approximate classical logic from below, we see that both consist of a restriction of one rule in a classical tableaux.

It turns out that constructing a proof theoretical approximation of classical logic is a trivial task!

Creating a Family that Approximates Classical Logic. It suffices that one restricts the use of one (or a set of) rules of one's favorite proof method to a limited set a formulas, in a way that eventually the rule will be applicable to all classical formulas.

The hard bit, however, is to find a corresponding semantics for such a system. In the following, we show all the possibilities of approximating classical logic from below by restricting the KE-tableau method, and provide it with a generic semantics as well.

4 A Generalized Approximation Inference

We have seen how to restrict $(T\neg)$ in KES_3 and how to restrict the use of $(F\neg)$ in KES_3^* . We assumed that the parameter set S governing both approximations was the same, but since there was no interactions between the S_3 -rule and the S_3^* rule, this assumption has no consequences. Now, we are going for greater generality, and we assume different context sets for each rule. This gives us a system where every connective behaves classically only for formulas which belong to the corresponding context set: $S_\wedge^T, S_\wedge^F, S_\vee^T, S_\vee^F, S_\rightarrow^T, S_\rightarrow^F, S_\neg^T$ and S_\neg^F .

An initial step towards this generalization was given in in [11], with a system that dealt with S_\neg^T and S_\rightarrow^T .

We present a generalization of the KE-tableaux, that we call KES_e , that deals with context sets for all the tableau rules. The system is obtained by adding restrictions to each expansion rule, as illustrated in Figure 2.

$T \alpha \rightarrow \beta$	$T \alpha \rightarrow \beta$	$F \alpha \rightarrow \beta$
$\frac{T \alpha}{T \alpha}$	$\frac{F \beta}{F \beta}$	$\frac{F \alpha \rightarrow \beta}{T \alpha \text{ if } \alpha \in S_\rightarrow^F}$
$T \beta \text{ if } \alpha \in S_\rightarrow^T$	$F \alpha \text{ if } \beta \in S_\rightarrow^T$	$F \beta \text{ if } \beta \in S_\rightarrow^F$
$F \alpha \wedge \beta$	$F \alpha \wedge \beta$	$T \alpha \wedge \beta$
$\frac{T \alpha}{T \alpha}$	$\frac{T \beta}{T \beta}$	$\frac{T \alpha \wedge \beta}{T \alpha \text{ if } \alpha \in S_\wedge^T}$
$F \beta \text{ if } \alpha \in S_\wedge^F$	$F \alpha \text{ if } \beta \in S_\wedge^F$	$T \beta \text{ if } \beta \in S_\wedge^T$
$T \alpha \vee \beta$	$T \alpha \vee \beta$	$F \alpha \vee \beta$
$\frac{F \alpha}{F \alpha}$	$\frac{F \beta}{F \beta}$	$\frac{F \alpha \vee \beta}{F \alpha \text{ if } \alpha \in S_\vee^F}$
$T \beta \text{ if } \alpha \in S_\vee^T$	$T \alpha \text{ if } \beta \in S_\vee^T$	$F \beta \text{ if } \beta \in S_\vee^F$
$T \neg \alpha$	$F \neg \alpha$	
$\frac{T \neg \alpha}{F \alpha \text{ if } \alpha \in S_\neg^T}$	$\frac{F \neg \alpha}{T \alpha \text{ if } \alpha \in S_\neg^F}$	
		$(F\neg)$
$\frac{}{} \text{---} (PB)$		
$T \alpha$	$F \alpha$	

Fig. 2. KE-rules for the generalized system

Lemma 4.1 *KES_e can simulate the dynamic evolution of both KES_3 and KES_3^* .*

Proof. To see that KES_e can simulate the dynamic evolution of KES_3 , it suffices to set $S = S_-^T$ 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 $(T\rightarrow)$ rule. Similarly for KES_3^* . \square

As usual, we want our system to be based on a sound and complete sub-classical semantics

4.1 Semantics for Generalized Approximate Inference

Definition 4.2 An S_e -valuation v_S^e is a function, $v_S^e : \mathcal{L}_C \rightarrow \{0, 1\}$, that extends a propositional valuation v_p (i.e., $v_S^e(p) = v_p(p)$), satisfying the following restrictions:

$$\begin{aligned}
 (\wedge_1) \quad & v_S^e(\alpha \wedge \beta) = 1, \alpha \in S_\wedge^T && \Rightarrow v_S^e(\alpha) = 1 \\
 (\wedge_2) \quad & v_S^e(\alpha \wedge \beta) = 1, \beta \in S_\wedge^T && \Rightarrow v_S^e(\beta) = 1 \\
 (\wedge_3) \quad & v_S^e(\alpha \wedge \beta) = 0, v_S^e(\alpha) = 1, \alpha \in S_\wedge^F && \Rightarrow v_S^e(\beta) = 0 \\
 (\wedge_4) \quad & v_S^e(\alpha \wedge \beta) = 0, v_S^e(\beta) = 1, \beta \in S_\wedge^F && \Rightarrow v_S^e(\alpha) = 0 \\
 (\vee_1) \quad & v_S^e(\alpha \vee \beta) = 0, \alpha \in S_\vee^F && \Rightarrow v_S^e(\alpha) = 0 \\
 (\vee_2) \quad & v_S^e(\alpha \vee \beta) = 0, \beta \in S_\vee^F && \Rightarrow v_S^e(\beta) = 0 \\
 (\vee_3) \quad & v_S^e(\alpha \vee \beta) = 1, v_S^e(\alpha) = 0, \alpha \in S_\vee^T && \Rightarrow v_S^e(\beta) = 1 \\
 (\vee_4) \quad & v_S^e(\alpha \vee \beta) = 1, v_S^e(\beta) = 0, \beta \in S_\vee^T && \Rightarrow v_S^e(\alpha) = 1 \\
 (\rightarrow_1) \quad & v_S^e(\alpha \rightarrow \beta) = 0, \alpha \in S_\rightarrow^F && \Rightarrow v_S^e(\alpha) = 1 \\
 (\rightarrow_2) \quad & v_S^e(\alpha \rightarrow \beta) = 0, \beta \in S_\rightarrow^F && \Rightarrow v_S^e(\beta) = 0 \\
 (\rightarrow_3) \quad & v_S^e(\alpha \rightarrow \beta) = 1, v_S^e(\alpha) = 1, \alpha \in S_\rightarrow^T && \Rightarrow v_S^e(\beta) = 1 \\
 (\rightarrow_4) \quad & v_S^e(\alpha \rightarrow \beta) = 1, v_S^e(\beta) = 0, \beta \in S_\rightarrow^T && \Rightarrow v_S^e(\alpha) = 0 \\
 (\neg_1) \quad & v_S^e(\neg\alpha) = 0, \alpha \in S_-^F && \Rightarrow v_S^e(\alpha) = 1 \\
 (\neg_2) \quad & v_S^e(\neg\alpha) = 1, \alpha \in S_-^T && \Rightarrow v_S^e(\alpha) = 0
 \end{aligned}$$

It is easy to see that the semantics of S_3 is a particular case of the system above, where the sets S_\wedge^T , S_\wedge^F , S_\vee^T , S_\vee^F , S_\rightarrow^T , S_\rightarrow^F , and S_-^F contain all the propositional letters of the language and $S = S_-^T$. Similarly, the semantics of S_3^* corresponds to $S = S_-^F$ and $S_\wedge^T = S_\wedge^F = S_\vee^T = S_\vee^F = S_\rightarrow^T = S_\rightarrow^F = S_-^T = \mathcal{P}$.

4.2 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 antecedent entails the consequent in S_e , all KES_e -tableaux close.

We extend an S_e -valuation to signed formulas making $v_S^e(T\alpha) = 1$ iff $v_S^e(\alpha) = 1$ and $v_S^e(F\alpha) = 1$ iff $v_S^e(\alpha) = 0$. 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 4.3 *If the antecedents of the KES_e linear expansion rules are S -satisfied in S_e by v_S^e so are its conclusions.*

Lemma 4.4 *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.*

Theorem 4.5 (Soundness) *Suppose a tableau for $\alpha_1, \dots, \alpha_n \vdash \beta$ closes. Then $\alpha_1, \dots, \alpha_n \models_S^e \beta$.*

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

By Lemmas 4.3 and 4.4, 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. \square

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

Lemma 4.6 *An open complete branch in a KES_e -tableau is S -satisfiable in S_e .*

Theorem 4.7 (Completeness) *If $\alpha_1, \dots, \alpha_n \models_S^e \beta$ then any possible KES_e tableau for $\alpha_1, \dots, \alpha_n \vdash \beta$ closes.*

Proof. Suppose for contradiction that there is a tableau for $\alpha_1, \dots, \alpha_n \vdash \beta$ with an open complete branch B . Then by Lemma 4.6 there is an S_e valuation that satisfies B , which includes $T \alpha_1, \dots, T \alpha_n, F \beta$, contradicting $\alpha_1, \dots, \alpha_n \models_S^e \beta$. \square

4.3 Applications of S_e

We examine here the use of S_e as a formalization of proof strategies using KE-tableaux. In a tableau expansion, more than one rule may be applicable at a time, and the choice of which rule to use may have dramatic effects, for a short proof may exist but the wrong choice of rule application may lead to an explosion in the number of branches.

Let $X \in \{T, F\}$ and $* \in \{\wedge, \vee, \rightarrow, \neg\}$. The use of an S_*^X context set in KES_e -tableaux may lead to a delay in using the rule $(X*)$. This works as

follows: suppose the rule $(X*)$ is classically applicable at one point in the branch expansion, but the corresponding proviso, $\alpha \in S_*^X$, is not met at that point. The use of the rule $(X*)$ is then blocked. All other applicable rules would take precedence, and will be applied. After their application, there are two possibilities: either all branches passing through that $(X*)$ -blocked point are closed, in which case there is nothing to be done, or there is at least one open branch. In the latter case, the formula α is inserted in S_*^X , the logic is changed to one “closer” to classical logic, from $S_e(S_*^X)$ to $S_e(S_*^X \cup \{\alpha\})$, so that the proviso is now met. The expansion of the tableau can then proceed incrementally in the new logic, without having to restart from square 1.

In the systems S_3 and S_3^* we have seen that most of the sets S_*^X were equal to \mathcal{P} . So the choice of which set S_*^X to be chosen to be different from \mathcal{P} has to do with which rule application we want to postpone.

Clearly, we do not want to postpone the application of one-premised linear rules. These rules, which correspond to the α -rules in Smullyan’s analytic tableaux, never generate a new branch and are all commutative, for the application of one rule does not invalidate the application of another. This means that for theorem proving purposes we would want to have

$$S_{\rightarrow}^F = S_{\wedge}^T = S_{\vee}^F = S_{\neg}^T = S_{\neg}^F = \mathcal{P}$$

Note that the context sets of S_3 and S_3^* are included in the sets above that we want to maintain fixed in \mathcal{P} . In fact, this is in accordance to some experiments made in [8] with the implementation of $KE S_3$ tableaux, in which a decrease in performance was noted from the use of $KE S_3$ strategy in respect to classical KE in which the one-premised rules were given application precedence, as above.

The two-premised rules are normally associated with the branching process, which is the important point to concentrate on when trying to reduce the size of a proof. There are two premises in those rules. The main premise is the *main formula*, which in Smullyan’s analytic tableaux are associated to β -rules and the branching process. The other premise is the *auxiliary formula*, which is associated with the *KE branching heuristics*. According to such heuristics, given a main formula of a two-premised rule where the auxiliary formula is absent, one should branch using PB so as to generate, in one of the branches, the missing auxiliary formula.

As a result, a strategy for tableau branching based on S_e should keep track of the formulas in the context sets S_{\wedge}^F , S_{\vee}^T and S_{\neg}^T .

The resulting strategy goes in accordance with the general intuition of tableau expansion: first expand the formulas that do not generate new branches, and only then expand the branching rules. Furthermore, our new strategy now places further restrictions on the branching rule, for we are giving preference to branch over a formula that is already on one of the sets S_{\wedge}^F , S_{\vee}^T and S_{\neg}^T . That is, our strategy gives preference for branching over subformulas of formulas over which there has already occurred a branching operation higher up

in the tableau.

Example 4.8 In this example we will consider $S = S_{\wedge}^F = S_{\vee}^T = S_{\rightarrow}^T$. That is, if there is a branching over some formula, subsequent branches over subformulas of it will be privileged. This strategy is good if there are irrelevant formulas, for it helps to avoid using them for the branching heuristics.

To see that, consider the sequent

$$A, (A \rightarrow B) \vee (A \rightarrow C), ((A \wedge B \vee C) \rightarrow D) \vee (A \wedge (B \vee C) \rightarrow E), (G \wedge F) \rightarrow (B \vee C) \vdash D \vee E$$

which generates, with initially $S = \emptyset$, the initial tableau

1. $T A$
2. $T (A \rightarrow B) \vee (A \rightarrow C)$
3. $T (A \wedge (B \vee C) \rightarrow D) \vee (A \wedge (B \vee C) \rightarrow E)$
4. $T (G \wedge F) \rightarrow (B \vee C)$
5. $F D \vee E$

Note that line 4 is totally irrelevant to the proof, and we want to avoid using it. After the first expansion of line 5 into

6. $F D (F \vee) 5$
7. $F E (F \vee) 5$

we have a choice of lines 2, 3 and 4 over which to apply the branching heuristics. As all those lines are T -marked, we choose a formula that has some atoms in common with the F -marked formulas in tableau; this justifiable, for the F -marked formulas are those we are trying to prove, and so we chose formulas that are relevant to the goal.

This choice leads to a branch over 3, with branching formulas $F A \wedge (B \vee C) \rightarrow D$ and $T A \wedge (B \vee C) \rightarrow D$. The left-hand branch develops as follows:

- 8a. $F A \wedge (B \vee C) \rightarrow D$
 - 9a. $T A \wedge (B \vee C) \rightarrow E \quad (T \vee) 3, 8a \quad S := \{A, B, C, D\}$
 - 10a. $T A \wedge (B \vee C) \quad (F \rightarrow) 8a$
 - 11a. $T E \quad (T \rightarrow) 9a, 10a$
- ×

In line 9a, the use of $(T \vee)$ using 3 as main premise and 8a as auxiliary premise forces the insertion of all atoms of 8a into S . Since $S = S_{\wedge}^F = S_{\vee}^T = S_{\rightarrow}^T$, this new S allows the use of $(T \rightarrow)$ to obtain line 12a, which closes the branch with line 7.

On the right-hand branch, we obtain the following expansion:

$$\begin{array}{ll}
 8b. & T A \wedge (B \vee C) \rightarrow D \\
 9b. & F A \wedge (B \vee C) \quad (T \rightarrow) 8b, 6 \\
 10b. & F (B \vee C) \quad (F \wedge) 9b, 1 \\
 11b. & F B \quad (F \vee) 10b \\
 12b. & F C \quad (F \vee) 10b
 \end{array}$$

The fact that $B, C \in S$ licenses the use on $(T \rightarrow)$ in line 9b; similarly, $A \in S$ licenses $(F \wedge)$ in line 10b. At this point we have to branch over lines 2 or 4. But 4 is blocked, for some of its atoms are outside A , which does not occur with line 2. So the expansion proceeds branching over $A \rightarrow B$.

$$\begin{array}{ll|ll}
 13ba. & T A \rightarrow B & & 13bb. & F A \rightarrow B \\
 14ba. & T B & (T \rightarrow) 13ba, 1 & 14bb. & T A \rightarrow C \quad (T \vee) 2, 13bb \\
 & \times & & 15bb. & T C \quad (T \rightarrow) 14bb, 1 \\
 & & & & \times
 \end{array}$$

As all branches are closed, the tableau is proved in S_e and also in classical logic.

5 Conclusions and Future Work

In this paper, we have extended the system S_3^* [4] to deal with full propositional logic, obtaining a family of paracomplete logics which is dual to the family of paraconsistent logics S_3 . Comparing the semantics and proof methods of both systems, we noted that the idea behind those systems, namely restricting the application of a rule, could be further generalized. This generalization gave us the system S_e , for which we gave a semantic and a sound and complete proof method. We then showed how S_e can be used to formalize different heuristics used for theorem proving.

Future work includes extending the implementation of the theorem prover for KE and $KE S_3$ to $KE S_e$ and testing it extensively with different context sets.

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 4th 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(3):283–318, 1999.
- [4] Marco Cadoli and Marco Schaerf. The complexity of entailment in propositional multivalued logics. *Annals of Mathematics and Artificial Intelligence*, 18(1):29–50, 1996.
- [5] Samir Chopra, Rohit Parikh, and Renata Wassermann. Approximate belief revision. *Logic Journal of the IGPL*, 9(6):755–768, 2001.
- [6] 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.
- [7] Marcello D’Agostino. Are tableaux an improvement on truth-tables? — cut-free proofs and bivalence. *Journal of Logic, Language and Information*, 1:235–252, 1992.
- [8] Wagner Dias. Tableaux implementation for approximate reasoning (in portuguese). Master’s thesis, Dept. of Computer Science, University of São Paulo, 2002.
- [9] Marcelo Finger and Renata Wassermann. Approximate reasoning and paraconsistency. In *8th Workshop on Logic, Language, Information and Computation (WoLLIC’2001)*, pages 77–86, July 31–August 3 2001.
- [10] Marcelo Finger and Renata Wassermann. Tableaux for approximate reasoning. In Leopoldo Bertossi and Jan Chomicki, editors, *IJCAI-2001 Workshop on Inconsistency in Data and Knowledge*, pages 71–79, Seattle, August 6–10 2001.
- [11] Marcelo Finger and Renata Wassermann. Expressivity and control in limited reasoning. In Frank van Harmelen, editor, *15th European Conference on Artificial Intelligence (ECAI02)*, pages 272–276, Lyon, France, 2002. IOS Press.
- [12] Hector Levesque. A logic of implicit and explicit belief. In *Proceedings of AAAI-84*, 1984.
- [13] Marco Schaerf and Marco Cadoli. Tractable reasoning via approximation. *Artificial Intelligence*, 74(2):249–310, 1995.
- [14] Raymond M. Smullyan. *First-Order Logic*. Springer-Verlag, 1968.
- [15] Annette ten Teije and Frank van Harmelen. Computing approximate diagnoses by using approximate entailment. In *Proceedings of KR’96*, 1996.