Tableaux for Approximate Reasoning

Marcelo Finger and **Renata Wassermann**
{mfinger,renata}@ime.usp.br
Department of Computer Science
Institute of Mathematics and Statistics
University of São Paulo, Brazil

Abstract

In this paper, we show that inconsistency can be fruitfully used to approximate classical theorem proving. We extend Cadoli and Schaerf's Approximate Entailment, originally defined only for formulas in clausal form, to full classical propositional logic. To this end, we provide approximations to classical logic via a family of logics solidly based on formal semantics and a tableaux proof system. Soundness and completeness are shown for the tableaux calculus with respect to the given semantics.

The tableaux system is then shown to provide a useful heuristics for the incremental approximation of classical logic, a feature that was lacking in existing proposals for approximate reasoning.

By means of such incremental method, we can move from one logic to the next one in the family, aiming to show a classical theorem. Incrementality means that we can proceed with the proof in the latter logic from the point where it stopped in the former one, without doing any recomputation.

1 Introduction

It has been an effort of logicians for quite some time to make *inconsistency respectable* [Gabbay and Hunter, 1991]. The aim of such line of research was to show that some inconsistent states may be tolerated without one ever having to resolve the inconsistency. This paper, however, focuses in another way of making inconsistency respectable, namely by showing that inconsistency can be fruitfully used to approximate classical theorem proving. In fact, we are taking advantage of a fact frequently pointed out in the literature, that to maintain a complete and consistent picture of the world the world is computationally expensive. We may ignore inconsistencies in irrelevant parts of the world while trying to prove a classical result.

The high price of theorem proving is well known. Logic has been used in several areas of Artificial Intelligence as a tool for representing knowledge as well as a tool for problem solving. One of the main criticism to the use of logic as a tool for automatic problem solving refers to the computational complexity of logical problems. Even if we restrict ourselves to classical propositional logic, deciding whether a set of formulas logically implies a certain formula is an NP-complete problem [Garey and Johnson, 1979].

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 [Schaerf and Cadoli, 1995]. 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 [Schaerf and Cadoli, 1995], 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.

In this paper, we generalize Cadoli and Schaerf's semantics to deal with full propositional logic. This generalization is needed, since the usual translation of formulas into clausal form is not sound under Cadoli and Schaerf's non-standard semantics. We present a tableaux system for the extended logic which is sound and complete with respect to the semantic. 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. Our contribution is thus, besides the generalization of approximate entailment, a constructive method for calculating the approximations.

The paper proceeds as follows: in Section 3, we briefly describe Cadoli and Schaerf's approximate entailment. In Section 4, we extend their semantic to full propositional logic. In Section 5, we present the tableaux method for the extended logic. In Section 6, we illustrate the use of the tableaux and show some of their properties. In Section 7 we proof soundness and completeness of the method. Finally, in Section 8, we discuss the method and point toward future work.

2 Preliminaries

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 $v_p : \mathcal{P} \rightarrow \{0, 1\}$.

3 Approximate Entailment

We briefly present here the notion of *approximate entailment* and summarize the main results obtained in [Schaerf and Cadoli, 1995].

Schaerf and Cadoli define two approximations of classical entailment: $\models^1_S$ which is complete but not sound, and $\models^3_S$ which is sound and incomplete. These approximations are carried out over a set of atoms $S \subseteq \mathcal{P}$ which determines their closeness to classical entailment. In the trivial extreme of approximate entailment, i.e., when $S = \mathcal{P}$, classical entailment is obtained. At the other extreme, when $S = \varnothing$, $\models^1_S$ holds for any two formulas (i.e., for all α,β, we have $\alpha \models^1_S \beta$) and $\models^3_S$ corresponds to Levesque's logic for explicit beliefs [Levesque, 1984], which bears a connection to relevance logics such as those of Anderson and Belnap [Anderson and Belnap, 1975].

In an S_1 assignment, if $p \in S$, then p and $\neg p$ are given opposite truth values; if $p \notin S$, then p and $\neg p$ both get the value 0. 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 names S_1 and S_3 come from the possible truth assignments for literals outside S. If $p \notin S$, there is only one S_1 assignment for p and $\neg p$, the one which makes them both false. 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. Satisfiability, entailment, and validity are defined in the usual way.

The following examples illustrate 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 $B \models \alpha$. On the other hand, $\models^1_S$ is unsound but complete, and can be used for approximating $\not\models$, i.e., if for some S we have that $B \not\models^1_S \alpha$, then $B \not\models \alpha$.

Example 1 ([Schaerf and Cadoli, 1995]) We want to check whether $B \models \alpha$, where $\alpha = \neg$`cow` $\vee$ `molar-teeth` and

$$\begin{aligned} B = \; \{ & \neg\texttt{cow} \vee \texttt{grass-eater}, \\ & \neg\texttt{dog} \vee \texttt{carnivore}, \\ & \neg\texttt{grass-eater} \vee \neg\texttt{canine-teeth}, \\ & \neg\texttt{carnivore} \vee \texttt{mammal}, \\ & \neg\texttt{mammal} \vee \texttt{canine-teeth} \vee \texttt{molar-teeth}, \\ & \neg\texttt{grass-eater} \vee \texttt{mammal}, \\ & \neg\texttt{mammal} \vee \texttt{vertebrate}, \\ & \neg\texttt{vertebrate} \vee \texttt{animal}\}. \end{aligned}$$

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

Example 2 ([Schaerf and Cadoli, 1995]) We want to check whether $B \not\models \beta$, where β=$\neg$`child` $\vee$ `pensioner` and

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

For $S = \{$`child`, `worker`, `pensioner`$\}$, we have that $B \not\models^1_S \beta$, and hence $B \not\models \beta$.

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

Lemma 3.1 ([Schaerf and Cadoli, 1995]) *Let simplify-1(B, S) be the result of deleting all literals of B which mention atoms outside S. B is S_1-satisfiable if and only if simplify-1(B, S) is classically satisfiable.*

Theorem 3.2 ([Schaerf and Cadoli, 1995]) *Let α be $\alpha_S \vee \alpha_{\overline{S}}$, where letters$(\alpha_S) \subseteq S$ and letters$(\alpha_{\overline{S}}) \cap S = \varnothing$. Then $B \models^1_S \alpha$ iff $B \cup \{\neg\alpha_S\}$ is not S_1 satisfiable.*[1]

This means that, in order to test whether $B \not\models^1_S \alpha$, for every literal of α of the form p, where $p \in S$, we add the clause $\neg p$ to B and for every literal of α of the form $\neg p$, where $p \in S$, we add the clause p to B. Let B' be this expanded set of clauses. We now must check whether B' is S_1 satisfiable. Using Lemma 3.1, we can reduce this problem to testing the classical satisfatibility of a simplified set of clauses.

Lemma 3.3 ([Schaerf and Cadoli, 1995]) *Let simplify-3(B, S) be the result of deleting all clauses of B which contain an atom outside S. Then B is*

[1]This can only be done because α_S behaves classically and we can compute its negation in clausal form (as a set of clauses).

S_3-satisfiable if and only if simplify-3(B, S) is classically satisfiable.

Theorem 3.4 ([Schaerf and Cadoli, 1995]) *Let letters$(\alpha) \subseteq S$. Then $B \models^3_S \alpha$ iff $B \cup \{\neg\alpha\}$ is not S_3 satisfiable.*

As in the case of S_1, Lemma 3.3 and Theorem 3.4 together provide a construcive method for testing S_3 entailment. Consider the Example 1, where $S = \{\texttt{grass-eater}, \texttt{mammal}, \texttt{canine-teeth}\}$ and we want to test whether $B \models^3_S \neg\texttt{cow} \vee \texttt{molar-teeth}$. In order to use Theorem 3.4 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 3.3 to simplify the expanded base, obtaining:

$$\begin{array}{ll} B' = & \{\neg\texttt{cow} \vee \texttt{grass-eater}, \\ & \neg\texttt{grass-eater} \vee \neg\texttt{canine-teeth}, \\ & \neg\texttt{mammal} \vee \texttt{canine-teeth} \vee \texttt{molar-teeth}, \\ & \neg\texttt{grass-eater} \vee \texttt{mammal}, \\ & \texttt{cow}, \neg\ \texttt{molar-teeth}\} \end{array}$$

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

Theorem 3.5 ([Schaerf and Cadoli, 1995]) *There exists an algorithm for deciding if $B \models^3_S \alpha$ and deciding $B \models^1_S \alpha$ which runs in $O(|B|.|\alpha|.2^{|S|})$ time.*

The result above depends on a polynomial time satisfiability algorithm for belief bases and formulas in clausal form alone. This result has been extended in [Cadoli and Schaerf, 1995] for formulas in negation normal form, but is not extendable to formulas in arbitrary forms [Cadoli and Schaerf, 1996].

The good point of Schaerf and Cadoli's systems is that they present an incremental algorithm to test for S_i 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 [Chopra *et al.*, 2000] 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 [ten Teije and van Harmelen, 1997], but nothing is said about the general case.

In this paper, we will concentrate on S_3 entailment. In the following sections we will extend S_3 to full propositional logic, present a proof method for the extended system and show how this proof method helps us to incrementally construct the set S.

4 Semantics

In this section, we extend S_3 to full propositional logic. We present a binary semantics for the full fragment of S_3, for that reason we call it an $S_{3.2}$ semantics [2].

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

Definition 4.1 An *$S_{3.2}$-valuation* $v^{3.2}_S$ is a function, $v^{3.2}_S : \mathcal{L}_C \rightarrow \{0, 1\}$, that extends a propositional valuation v_p (i.e., $v^{3.2}_S(p) = v_p(p)$), satisfying the following restrictions:

$$\begin{array}{llll} (i) & v^{3.2}_S(\alpha \wedge \beta) = 1 & \Leftrightarrow & v^{3.2}_S(\alpha) = v^{3.2}_S(\beta) = 1 \\ (ii) & v^{3.2}_S(\alpha \vee \beta) = 0 & \Leftrightarrow & v^{3.2}_S(\alpha) = v^{3.2}_S(\beta) = 0 \\ (iii) & v^{3.2}_S(\alpha \rightarrow \beta) = 0 & \Leftrightarrow & v^{3.2}_S(\alpha) = 1 \text{ and} \\ & & & v^{3.2}_S(\beta) = 0 \\ (iv) & v^{3.2}_S(\neg\alpha) = 0 & \Rightarrow & v^{3.2}_S(\alpha) = 1 \\ (v) & v^{3.2}_S(\neg\alpha) = 1, \alpha \in S & \Rightarrow & v^{3.2}_S(\alpha) = 0 \end{array}$$

Rules (i)–(iii) are exactly those of classical logic. Rules (iv) and (v) restrict the semantics of negation: rule (iv) states that if $v^{3.2}_S(\neg\alpha) = 0$, then negation behaves classically and forces $v^{3.2}_S(\alpha) = 1$; rule (v) states that if $v^{3.2}_S(\neg\alpha) = 1$, 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 $S_{3.2}$-valuation is not uniquely defined by the propositional valuation it extends. This is due to the fact that if $\alpha \notin S$ and $v^{3.2}_S(\alpha) = 1$, the value of $v^{3.2}_S(\neg\alpha)$ can be either 0 (in which case α has a classical behaviour) or 1 (in which case α behaves paraconsistently).

Lemma 4.2 *The valuation $v^{3.2}_S$ is determined by its value on the set $\mathcal{P} \cup \{\neg\alpha | \alpha \notin S\}$.*

Proof: For the connectives $\wedge$, $\vee$ and $\rightarrow$, it is immediate that the value of $v^{3.2}_S$ is determined by that of the subformulas. If $v^{3.2}_S(\alpha) = 0$, then by rule (iv) $v^{3.2}_S(\neg\alpha) = 1$. If $\alpha \in S$ the value of $v^{3.2}_S(\neg\alpha)$ is the opposite of that of $v^{3.2}_S(\alpha)$. We are left only with the formulas in $\mathcal{P} \cup \{\neg\alpha | \alpha \notin S\}$ to determine $v^{3.2}_S$. $\square$

We define a formula α to be *S-valid* in $S_{3.2}$ if $v^{3.2}_S(\alpha) = 1$ for any $S_{3.2}$-valuation. A formula is *S-satisfiable* in $S_{3.2}$ if there is one $v^{3.2}_S$ such that $v^{3.2}_S(\alpha) = 1$. The *$S_{3.2}$-entailment* relationship between a set of formulas Γ and a formula α is represented as

$$\Gamma \models^{3.2}_S \alpha$$

and holds if every valuation $v^{3.2}_S$ that simultaneously satisfies all formulas in Γ also satisfies α. A formula is S-valid if it is entailed by $\varnothing$, represented as $\models^{3.2}_S \alpha$.

Lemma 4.2 suggests a translation between a formula in S_3 and one in classical logic, such that every formula

[2] A three-valued semantics, called $S_{3.3}$ semantics, can be provided to cover the $\rightarrow$-free fragment; in such a fragment, $S_{3.2}$ and $S_{3.3}$ coincide.

of the form $\neg\alpha$ with $\alpha \notin S$ is mapped into a new propositional symbol $p_{\neg\alpha}$. Let α^{*S} be the translation of α, defined as:

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

Lemma 4.3 *A formula α is S-satisfiable in $S_{3.2}$ iff $(\neg\alpha)^{*S}$ is classically satisfiable*

Lemma 4.4 *Let $Prop(\alpha)$ represent the propositional symbols in α. Every formula α with $Prop(\alpha) \cap S = \varnothing$ is satisfiable in $S_{3.2}$.*

Proof: Since $Prop(\alpha) \cap S = \varnothing$, the translation $(\alpha)^{*S}$ leads us into the $\neg$-free fragment of classical logic (without falsity, $\bot$), and any formula is such fragment is satisfiable (just make all atoms 1), so Lemma 4.3 makes α satisfiable in $S_{3.2}$. $\square$

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.

Lemma 4.5 *$S_{3.2}$-validity is subclassical. That is, any S-valid formula in $S_{3.2}$ is classically valid, for any S.*

Proof: It suffices to notice that any classical valuation satisfies the $S_{3.2}$-restrictions for any S. Therefore, the set of all classical valuations is contained in the set of all $S_{3.2}$-valuations (but there are $S_{3.2}$-valuations that are not classic). So if a formula is satisfied by all $S_{3.2}$-valuations, it will be satisfied by all classical ones. $\square$

The deduction theorem holds for the $S_{3.2}$ semantics, as can be seen directly from the definitions.

Lemma 4.6 (Deduction Theorem) *Let Γ be a finite set of formulas. Then*

$$\Gamma \models_S^{3.2} \alpha \textit{ iff } \models_S^{3.2} \bigwedge \Gamma \rightarrow \alpha.$$

Now we examine a few examples of $S_{3.2}$ entailment.

Example 3 Consider the formula $\alpha \vee \neg\alpha$. We show that it is a valid formula in $S_{3.2}$ 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 $v_S^{3.2}$ be a valuation. If $v_S^{3.2}(\neg\alpha) = 1$, then $\alpha \vee \neg\alpha$ clearly is true. If, however, $v_S^{3.2}(\neg\alpha) = 0$, by rule (iv) above $v_S^{3.2}(\alpha) = 1$, so $\alpha \vee \neg\alpha$ is also true.

Example 4 We now show that the $S_{3.2}$ semantics is paraconsistent. For that, consider the two propositions p and q and suppose that $p \notin S$; take a valuation $v_S^{3.2}$ such that $v_S^{3.2}(p) = 1$ and $v_S^{3.2}(q) = 0$, and consider the formula $(p \wedge \neg p) \rightarrow q$. By Lemma 4.2, the value of $v_S^{3.2}$ is not fully determined, so we fix $v_S^{3.2}(\neg p) = 1$. It is simple to verify now that the valuation thus constructed is such that $v_S^{3.2}((p \wedge \neg p) \rightarrow q) = 0$, that is the logic does not trivialize in the presence of inconsistency.

Example 5 We now analyse the validity of Modus Ponens in $S_{3.2}$. The usual formulation of Modus Ponens, $\alpha \rightarrow \beta, \alpha \models_S^{3.2} \beta$, is valid in $S_{3.2}$; indeed, if $v_S^{3.2}$ 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 [Schaerf and Cadoli, 1995]), 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 $\neg\alpha \vee \beta, \alpha \not\models_S^{3.2} \beta$.

5 KE-Tableaux for S_3

We develop an inference system for the full logic S_3 based on the KE-tableau methodology. KE-tableaux were introduced by D'Agostino [D'Agostino, 1992] as a principled computational improvement over Smullyan's Semantic Tableaux [Smullyan, 1968], and have since been successfully applied to a variety of logics [D'Agostino and Gabbay, 1994; Broda and Finger, 1995; Broda *et al.*, 1999].

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. 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 something cannot be true and false at the same time. Figure 1 shows KE-tableau expansion rules for classical logic.

In Figure 1 we see that each of the binary connectives $\rightarrow$, $\wedge$ and $\vee$ are associated to two two-premised rules and one one-premised rule. The two-premised rules have a *main antecedent* and an *auxiliary antecedent*; the one-premised rules have two consequences. Classical negation is associated to two one premised rules, each with a single conclusion. The final line presents the *Principle of Bivalence* (PB), stating that any formula α is either true of 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 premise for a two-premised 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

$$
\frac{\begin{array}{c}T\ \alpha\to\beta\\ T\ \alpha\end{array}}{T\ \beta}\,(T\to_1)
\qquad
\frac{\begin{array}{c}T\ \alpha\to\beta\\ F\ \beta\end{array}}{F\ \alpha}\,(T\to_2)
\qquad
\frac{F\ \alpha\to\beta}{\begin{array}{c}T\ \alpha\\ F\ \beta\end{array}}\,(F\to)
$$

$$
\frac{\begin{array}{c}F\ \alpha\wedge\beta\\ T\ \alpha\end{array}}{F\ \beta}\,(F\wedge_1)
\qquad
\frac{\begin{array}{c}F\ \alpha\wedge\beta\\ T\ \beta\end{array}}{F\ \alpha}\,(F\wedge_2)
\qquad
\frac{T\ \alpha\wedge\beta}{\begin{array}{c}T\ \alpha\\ T\ \beta\end{array}}\,(T\wedge)
$$

$$
\frac{\begin{array}{c}T\ \alpha\vee\beta\\ F\ \alpha\end{array}}{T\ \beta}\,(T\vee_1)
\qquad
\frac{\begin{array}{c}T\ \alpha\vee\beta\\ F\ \beta\end{array}}{T\ \alpha}\,(T\vee_2)
\qquad
\frac{F\ \alpha\vee\beta}{\begin{array}{c}F\ \alpha\\ F\ \beta\end{array}}\,(F\vee)
$$

$$
\frac{T\ \neg\alpha}{F\ \alpha}\,(T\neg)
\qquad
\frac{F\ \neg\alpha}{T\ \alpha}\,(F\neg)
$$

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

Figure 1: KE-rules for classical logic

some previously occurring formula.

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 1. 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 6 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 2, where the boxed formulas indicate the closure of condition for each branch. The left tableau shows $\alpha \to \beta \vdash \neg\alpha \vee \beta$ and the right one shows $\neg\alpha \vee \beta \vdash \alpha \to \beta$.

$$
\begin{array}{llc|llc}
1. & T\ \alpha\to\beta & & 1. & T\ \neg\alpha\vee\beta & \\
2. & F\ \neg\alpha\vee\beta & & 2. & F\ \alpha\to\beta & \\
3. & F\ \neg\alpha & \text{from 2} & 3. & \boxed{T\ \alpha} & \text{from 2}\\
4. & \boxed{F\ \beta} & \text{from 2} & 4. & F\ \beta & \text{from 2}\\
5. & T\ \alpha & \text{from 3} & 5. & T\ \neg\alpha & \text{from 1,4}\\
6. & \boxed{T\ \beta} & \text{from 1,5} & 6. & \boxed{F\ \alpha} & \text{from 5}\\
 & \times & & & \times &
\end{array}
$$

Figure 2: The classical equivalence of $\alpha \to \beta$ and $\neg\alpha \vee \beta$

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 [D'Agostino, 1992].

Example 7 To show the use of PB, we present a KE-tableau now showing that $\neg\neg\alpha \vee \neg\neg\beta, \neg\alpha \vdash \beta$.

$$
\begin{array}{c}
T\ \neg\neg\alpha \vee \neg\neg\beta\\
\boxed{T\ \neg\alpha}\\
\boxed{\boxed{F\ \beta}}\\
F\ \alpha\\
\begin{array}{c|c}
T\ \neg\neg\alpha & F\ \neg\neg\alpha\\
\boxed{F\ \neg\alpha} & T\ \neg\neg\beta\\
\times & F\ \neg\beta\\
 & \boxed{\boxed{T\ \beta}}\\
 & \times
\end{array}
\end{array}
$$

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 premise 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.

5.1 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:

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

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 4 will be presented in Section 7. First, let us examine a few examples.

Example 8 We first show that $\to$ is no longer definable in terms of $\vee$ and $\wedge$, by redoing the tableaux of Figure 2 in Figure 3.

In Figure 3 we are assuming that $S = \varnothing$. Note that the left tableau for $\alpha \to \beta \vdash \neg\alpha \vee \beta$ is exactly the same as for classical logic.

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 3: $\rightarrow$ is not definable in terms of $\vee$ and $\neg$ in KES_3

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 3 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.

6 The Incrementality of the Method

The idea of approximate reasoning found in [Schaerf and Cadoli, 1995] 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 [Schaerf and Cadoli, 1995] 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. $S := \varnothing$.
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 $S = \varnothing$. Note that in step 6 above, there may be more then one stuck point in the tableau, so we need to chose 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 chose the first one, S becomes $\{p\}$ and the tableau closes; if we chose 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 6.1 (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.*

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, it's corresponding KES_3-tableau, when it becomes classical, will eventually close. $\square$

Example 9 The following example illustrates the use of the system KES_3. Consider the problem of Example 1, where we want to know whether $\neg\texttt{cow} \vee \texttt{molar-teeth}$ 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 4 shows the complete tableau (g-e stands for grass-eater, c-t for canine-teeth, and m-t for molar-teeth).

$$
\begin{aligned}
B = \{ & \neg\texttt{cow} \vee \texttt{grass-eater},\\
& \neg\texttt{dog}\vee \texttt{carnivore},\\
& \neg\texttt{grass-eater} \vee \neg\texttt{canine-teeth},\\
& \neg\texttt{carnivore} \vee \texttt{mammal},\\
& \neg\texttt{mammal} \vee \texttt{canine-teeth} \vee \texttt{molar-teeth},\\
& \neg\texttt{grass-eater} \vee \texttt{mammal},\\
& \neg\texttt{mammal} \vee \texttt{vertebrate},\\
& \neg\texttt{vertebrate} \vee \texttt{animal}\}.
\end{aligned}
$$

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

Figure 4: KES_3 tableau for clauses

We start with $S = \varnothing$. 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 $B \models^3_S \neg\texttt{cow} \vee \texttt{molar-teeth}$ for $S=\{\texttt{grass-eater}, \texttt{canine-teeth}, \texttt{mammal}\}$.

Example 10 This example shows that formulas that are classically equivalent may have different behaviour under S_3. We transform the set B from Example 1 into a set B' which is classically equivalent to B but is not in clausal form. Then we try to check whether $B' \models^3_S \texttt{cow} \rightarrow \texttt{molar-teeth}$. Figure 5 shows the complete tableau. Note that this time, we can close the tableau adding only one atom to S, i.e., $B' \models^3_S \texttt{cow} \rightarrow \texttt{molar-teeth}$ for $S=\{\texttt{canine-teeth}\}$.

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

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

Figure 5: KES_3 tableau with implication

7 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 5 with respect to the $S_{3.2}$ two-valued semantics of Section 4.

First, we need to define the notions of soundness and completeness. So KES_3 is *sound with respect to the $S_{3.2}$ semantics* if whenever a tableau closes for an input sequent, then the sequent's antecedent formulas entail its consequent in $S_{3.2}$. Conversely, the KES_3-tableau method is *complete with respect to the $S_{3.2}$ semantics* if for all sequents such that the the antecedent entails the consequent in $S_{3.2}$, all KES_3-tableaux close.

We extend the valuation to signed formulas in the obvious way, that is, $v_S^{3.2}(T\alpha) = 1$ iff $v_S^{3.2}(\alpha) = 1$ and $v_S^{3.2}(F\alpha) = 1$ iff $v_S^{3.2}(\alpha) = 0$. A valuation satisfy a branch in a tableau if it simultaneously satisfy all the signed formulas in the branch.

7.1 Soundness

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

Lemma 7.1 *If the antecedents of the KES$_3$ linear expansion rules are S-satisfied in S_3 by $v_S^{3.2}$ so are its conclusions.*

Proof: A simple inspection of the rules in Figure 1 with the modification in $(T\ \neg)$ for KES_3 shows the result. □

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

Lemma 7.2 *If a branch is satisfied by a valuation $v_S^{3.2}$ prior to the application of PB, then at least one of the two branches generated is satisfied by a valuation $v_S^{3.2}$ after the application of PB.*

Proof: Suppose the branching occurs over the formula α. Because $v_S^{3.2}$ is a function onto $\{0, 1\}$, we have that $v_S^{3.2}(T\ \alpha) = 1$ or $v_S^{3.2}(F\ \alpha) = 1$, so $v_S^{3.2}$ satisfies one of the two branches generated by the application of PB. □

Theorem 7.3 (Soundness) *Suppose a tableau for $\alpha_1, \ldots, \alpha_n \vdash \beta$ closes. Then $\alpha_1, \ldots, \alpha_n \models_S^{3.2} \beta$.*

Proof: We show the contrapositive. So suppose $\alpha_1, \ldots, \alpha_n \not\models_S^{3.2} \beta$, so there is a valuation $v_S^{3.2}$ such that $v_S^{3.2}(\alpha_1) = \ldots = v_S^{3.2}(\alpha_1) = 1$ and $v_S^{3.2}(\beta) = 0$. 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 $v_S^{3.2}$.

By Lemmas 7.1 and 7.2, 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. □

7.2 Completeness

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

Lemma 7.4 *An open complete branch in a KES$_3$-tableau is S-satisfiable in $S_{3.2}$.*

Proof: Given an open complete branch B, we construct the following valuation $v_S^{3.2}$, based on the propositional and negated formulas in B:

$$\begin{array}{lll} v_S^{3.2}(p) = 1 & \text{iff} & T\ p \in B \\ v_S^{3.2}(p) = 0 & \text{iff} & F\ p \in B \\ v_S^{3.2}(\neg\alpha) = 1 & \text{iff} & T\ \neg\alpha \in B, \text{and}\ \alpha \notin S \end{array}$$

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 4.2, we need to define the value of $v_S^{3.2}$ for propositions and formulas of the form $\neg\alpha$ not occurring in B; in fact, the value of $v_S^{3.2}$ for such formulas can be any, but to be deterministic let us make them all false.

A simple structural induction on the signed formulas in B shows that $v_S^{3.2}$ satisfies the branch. □

Theorem 7.5 (Completeness) *If $\alpha_1, \ldots, \alpha_n \models_S^{3.2} \beta$ 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 7.4 there is a $S_{3.2}$ valuation that satisfies B, which includes $T\ \alpha_1, \ldots, T\ \alpha_n, F\ \beta$, contradicting $\alpha_1, \ldots, \alpha_n \models_S^{3.2} \beta$. □

8 Discussion

We have extended Cadoli and Schaerf's Approximate Entailment to full propositional logic, providing a semantics, and a sound and complete proof method.

The proof method is an adaptation of a tableaux system which was shown to be computationally more efficient than semantic tableaux. The method is incremental: the tableau is built for a given context set S and when new atoms are added to S we can continue from where we stopped. The construction of the tableaux makes clear which atoms must be added to S.

A proof method based on tableaux for paraconsistent logic has been proposed in [Carnielli and Lima-Marques, 1992]. The idea behind their system is very similar to ours: the only difference from tableaux for classical logic is an extra condition for the rules involving negation. However, as in the calculus C_1 [da Costa, 1963], the condition that a formula behaves classically is part of the language. As a result, the tableau construction sometimes loops.

Future work includes studying the formal relation between the extended version of S_3 and da Costa's C_1, the complexity of the proposed proof method, and a system of tableaux for the extended version of S_1.

Acknowledgments: Marcelo Finger is partly supported by the Brazilian Research Council (CNPq), grant PQ 300597/95-5. Renata Wassermann was supported by FAPESP through grant 99/11602-6. This work was developed under the CNPq project APQ 468765/00-0.

References

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

[Broda and Finger, 1995] 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.

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

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

[Cadoli and Schaerf, 1996] Marco Cadoli and Marco Schaerf. The complexity of entailment in propositional multivalued logics. *Annals of Mathematics and Artificial Intelligence*, 18(1):29–50, 1996.

[Carnielli and Lima-Marques, 1992] Walter A. Carnielli and Mamede Lima-Marques. Reasoning under inconsistent knowledge. *Journal of Applied Non-Classical Logics*, 2(1):49–79, 1992.

[Chopra *et al.*, 2000] Samir Chopra, Rohit Parikh, and Renata Wassermann. Approximate belief revision. In *Proceedings of Workshop on Language, Logic and Information (WoLLIC)*, 2000.

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

[D'Agostino and Gabbay, 1994] 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.

[D'Agostino, 1992] 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.

[Gabbay and Hunter, 1991] Dov Gabbay and Anthony Hunter. Making inconsistency respectable, part 1. In *Proceedings of Fundamental of Artificial Intelligence Research (FAIR '91)*, Vol 535 of LNAI, pages 19–32. Springer-Verlag, 1991.

[Garey and Johnson, 1979] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1979.

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

[Schaerf and Cadoli, 1995] Marco Schaerf and Marco Cadoli. Tractable reasoning via approximation. *Artificial Intelligence*, 74(2):249–310, 1995.

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

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