Chapter 5

Belnap's four valued logic

5.1 Introduction

The study of first-degree entailment occupies a special position in the field of relevance logics: it can be seen either as the study of the validity of formulae of the form $A \rightarrow B$, where $\rightarrow$ is Anderson and Belnap's relevant implication and A, B are implication-free formulae, or as the study of the notion of *relevant deducibility* between standard formulae built-up from the usual connectives. In the latter interpretation it is associated with the problem — well-known to computer scientists who work in the area of Automated Deduction — of obtaining sound information from possibly inconsistent databases[1]. An interesting semantic characterization of first-degree entailment was given by Belnap in [Bel77] who also emphasized its connections with the problem of 'how a computer should think'[Bel76]. In [Dun76] Dunn presented a tableau system based on a modification of Jeffrey's method of 'coupled trees' [Jef81]. In this chapter we study the consequence relation associated with Belnap's semantics and produce two different calculi which are sound and complete for it. Unlike Dunn's 'coupled tree' calculus, our calculi use *one* tree only. This simplification allows us to exploit fully the formal analogy with the corresponding classical calculi, and to obtain simple extensions to a first order version of Belnap's logic (neither Jeffrey nor Dunn explain how the method of 'coupled trees' can be extended to deal with quantifiers).

In sections 5.2 and 5.3 we briefly discuss the background problem and illustrate Belnap's semantics. In section 5.4 we formulate a tableau method which produces 'single' instead of 'coupled' tableaux, and prove it sound and complete for (the first-order version of) the consequence relation associated

[1]For recent contributions in this area and in the related one of logic programming, see [DCHLS90] and [BS89].

with Belnap's semantics. Then, in section 5.5, we define another calculus based on the classical systems **KE** which has been studied in Chapter 3.

5.2 'How a computer should think'

Deductive Reasoning is often described as a process of revealing 'hidden' information from explicit data and, as such, it is a basic *tool* in the area of 'intelligent' database management or question-answering systems. Unfortunately, the most time-honoured and well-developed framework for deductive reasoning — classical logic — is unsuitable to this purpose. The reason is that databases, especially large ones, have a great propensity to become *inconsistent*: first, the information stored is usually obtained from different sources which might conflict with each other; second, the information obtained from each source, even if it is not *obviously* inconsistent, may 'hide' contradictions. But it is well-known that classical two-valued logic is oversensitive to contradictions: if Γ is an inconsistent set of sentences, then — according to classical logic — *any sentence* follows from Γ. This does not imply, of course, that classical logic is *incorrect*, but only suggests that there are circumstances in which it is highly recommendable to abandon it and use another. A radical solution to this problem would be to require any database to be consistent *before* starting deductive processing. But, for practical reasons, this 'solution' is no better than the original problem: first, contradictions do not always lie on the surface, and the only way to detect such implicit contradictions is to apply deductive reasoning itself; second, even explicit contradictions may not be removable because they originate in conflicting data fed into the computer by different and equally reliable sources.

But if classical logic is not to be recommended for application in deductive database management, what kind of logic is the one with which a computer should 'think'?

5.3 Belnap's four-valued model

In this chapter we shall make use of the approach developed by Belnap [Bel77, Bel76] on the basis of a work of Dunn [Dun76]. Let **4** denote the set $\{$**T**, **F**, **Both**, **None**$\}$. The elements of **4** are called *truth-values*. Belnap calls them 'told values' to emphasize their epistemic character. We may think of them as the four possible ways in which an atomic sentence P can belong to the 'present state of information' : (1) the computer is told that P is true (and is not told that P is false); (2) the computer is told that P

is false (and is not told that P is true); (3) the computer is told that P is both true and false (perhaps from different sources, or in different instants of time); (4) the computer is not told anything about the truth value of P.

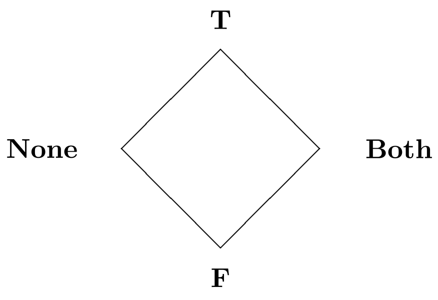
The values of complex sentences are obtained by Belnap by means of monotony considerations, based on Scott's approximation lattices, resulting in the following tables:

	None	**F**	**T**	**Both**
$\neg$	**None**	**T**	**F**	**Both**

$\wedge$	**None**	**F**	**T**	**Both**
None	**None**	**F**	**None**	**F**
F	**F**	**F**	**F**	**F**
T	**None**	**F**	**T**	**Both**
Both	**F**	**F**	**Both**	**Both**

$\vee$	**None**	**F**	**T**	**Both**
None	**None**	**None**	**T**	**T**
F	**None**	**F**	**T**	**Both**
T	**T**	**T**	**T**	**T**
Both	**T**	**Both**	**T**	**Both**

These tables represent a lattice which is called **L4**:



We define a *set-up* as a mapping of atomic formulae into **4**. Using the truth-tables given above, every set-up can be extended to a mapping of *all formulae* into **4** in the usual inductive way. We shall call such an extended mapping a **4**-*valuation* .

Definition 5.3.1 We say that A *entails* B, and write $A \rightarrow B$, if for all **4**-valuations v, $v(A) \preceq v(B)$, where $\preceq$ is the partial ordering associated with the lattice **L4**. We also say that a *non empty set* of formulae Γ *entails* A, and write $\Gamma \vdash A$, if the conjunction of all formulae in Γ entails A.

Notice that the relation $\vdash$ mentioned in Def. 5.3.1 is a *monotonic* consequence relation.

The logic characterized by Belnap's semantics corresponds to the logic of first-degree entailment (see [AB75, section 15.2]). This system admits of the following Hilbert-style formulation:

Axioms:

(5.1) $A \wedge B \rightarrow A$

(5.2) $A \wedge B \rightarrow B$

(5.3) $A \rightarrow A \vee B$

(5.4) $B \rightarrow A \vee B$

(5.5) $A \wedge (C \vee B) \rightarrow (A \wedge B) \vee C$

(5.6) $A \rightarrow \neg\neg A$

(5.7) $\neg\neg A \rightarrow A$

Rules:

(5.8) $A \rightarrow B, B \rightarrow C \;\vdash\; A \rightarrow C$

(5.9) $A \rightarrow B, A \rightarrow C \;\vdash\; A \rightarrow B \wedge C$

(5.10) $A \rightarrow C, B \rightarrow C \;\vdash\; A \vee B \rightarrow C$

(5.11) $A \rightarrow B \;\vdash\; \neg B \rightarrow \neg A$

Let us now introduce some useful terminology:

Definition 5.3.2 Let us say, given a **4**-valuation v, that a formula A is:

1. *at least true* under v if $v(A) = \mathbf{T}$ or $v(A) = \mathbf{Both}$.
2. *non-true* under v if $v(A) = \mathbf{F}$ or $v(A) = \mathbf{None}$.
3. *at least false* under v if $v(A) = \mathbf{F}$ or $v(A) = \mathbf{Both}$.

4. *non-false* under v if $v(A) = \mathbf{T}$ or $v(A) = \mathbf{None}$.

It is not difficult to see that the definition of $\Gamma \vdash A$ given in Definition 5.3.1 is equivalent to the following one:

Definition 5.3.3 $\Gamma \vdash A$ if and only if for every **4**-valuation v, (i) if all the elements of Γ are at least true under v, then A is at least true under v *and* (ii) if all the elements of Γ are non-false under v, then A is non-false under v.

5.4 Semantic tableaux for Belnap's logic

In this section we suggest a tableau method which, unlike Dunn's one, uses only one tree. The resulting tableaux are *binary* trees and are identical to classical tableaux of *signed* formulae except that they contain four types of s-formulae instead of two.

5.4.1 Propositional tableaux

Signed Formulae. We introduce the symbols t, f, t*, f* and define a *signed* formula as an expression of the form t(A), f(A), t$^*(A)$ or f$^*(A)$, where A is an unsigned formula. Intuitively we interpret 't(A)' as 'A is at least true', 'f(A)' as 'A is non-true', 't$^*(A)$' as 'A is non-false' and 'f$^*(A)$' as 'A is at least false'.

The *conjugate* of an s-formula s(A) is:

f(A)	if s = t
t(A)	if s = f
f$^*(A)$	if s = t*
t$^*(A)$	if s = f*

The *converse* of an s-formula s(A) is:

t$^*(A)$	if s = t
t(A)	if s = t*
f$^*(A)$	if s = f
f(A)	if s = f*

We are now in a position to formulate our tableau rules which are given in Table 5.1. The reader will notice the formal analogy with the classical rules.

$\dfrac{\mathrm{t}(A \wedge B)}{\begin{array}{c}\mathrm{t}(A)\\ \mathrm{t}(B)\end{array}}$	$\dfrac{\mathrm{t}^*(A \wedge B)}{\begin{array}{c}\mathrm{t}^*(A)\\ \mathrm{t}^*(B)\end{array}}$	$\dfrac{\mathrm{f}(A \vee B)}{\begin{array}{c}\mathrm{f}(A)\\ \mathrm{f}(B)\end{array}}$	$\dfrac{\mathrm{f}^*(A \vee B)}{\begin{array}{c}\mathrm{f}^*(A)\\ \mathrm{f}^*(B)\end{array}}$
$\dfrac{\mathrm{f}(A \wedge B)}{\mathrm{f}(A) \mid \mathrm{f}(B)}$	$\dfrac{\mathrm{f}^*(A \wedge B)}{\mathrm{f}^*(A) \mid \mathrm{f}^*(B)}$	$\dfrac{\mathrm{t}(A \vee B)}{\mathrm{t}(A) \mid \mathrm{t}(B)}$	$\dfrac{\mathrm{t}^*(A \vee B)}{\mathrm{t}^*(A) \mid \mathrm{t}^*(B)}$
$\dfrac{\mathrm{t}(\neg A)}{\mathrm{f}^*(A)}$	$\dfrac{\mathrm{t}^*(\neg A)}{\mathrm{f}(A)}$	$\dfrac{\mathrm{f}(\neg A)}{\mathrm{t}^*(A)}$	$\dfrac{\mathrm{f}^*(\neg A)}{\mathrm{t}(A)}$

Table 5.1: Propositional tableau rules for Belnap's four valued logic.

α	α_1	α_2
$\mathrm{t}(A \wedge B)$	$\mathrm{t}(A)$	$\mathrm{t}(B)$
$\mathrm{t}^*(A \wedge B)$	$\mathrm{t}^*(A)$	$\mathrm{t}^*(B)$
$\mathrm{f}(A \vee B)$	$\mathrm{f}(A)$	$\mathrm{f}(B)$
$\mathrm{f}^*(A \vee B)$	$\mathrm{f}^*(A)$	$\mathrm{f}^*(B)$
$\mathrm{t}(\neg A)$	$\mathrm{f}^*(A)$	$\mathrm{f}^*(A)$
$\mathrm{t}^*(\neg A)$	$\mathrm{f}(A)$	$\mathrm{f}(A)$
$\mathrm{f}(\neg A)$	$\mathrm{t}^*(A)$	$\mathrm{t}^*(A)$
$\mathrm{f}^*(\neg A)$	$\mathrm{t}(A)$	$\mathrm{t}(A)$

Table 5.2: Formulae of type α.

It may be useful to extend Smullyan's unifying notation to cover the new types of s-formulae which occur in our language.

We use the letter 'α' to stand for any s-formula of one of the forms: $\mathrm{t}(A \wedge B)$, $\mathrm{t}^*(A \wedge B)$, $\mathrm{f}(A \vee B)$, $\mathrm{f}^*(A \vee B)$, $\mathrm{t}(\neg A)$, $\mathrm{t}^*(\neg A)$, $\mathrm{f}(\neg A)$ and $\mathrm{f}^*(\neg A)$. For every such formula α, its *components* α_1 and α_2 are defined as in Table 5.2. We use 'β' to stand for any formula of one of the forms: $\mathrm{f}(A \wedge B)$, $\mathrm{f}^*(A \wedge B)$, $\mathrm{t}(A \vee B)$ or $\mathrm{t}^*(A \vee B)$. For every such formula β, its *components* β_1 and β_2 are defined as in the Table 5.3. So our tableau rules can be 'packed' into the following two rules:

$$\text{Rule A} \quad \dfrac{\alpha}{\begin{array}{c}\alpha_1\\ \alpha_2\end{array}} \qquad\qquad \text{Rule B} \quad \dfrac{\beta}{\beta_1 \mid \beta_2}$$

We say that a branch ϕ of a tableau is *closed* if it contains both an s-formula and its conjugate. Otherwise we say that ϕ is *open*. A tableau is *closed* if all its branches are closed. Otherwise it is *open*. A formula A is *provable* from the set of formulae Γ if and only if there is a closed tableau

β	β_1	β_2
$\mathrm{f}(A \wedge B)$	$\mathrm{f}(A)$	$\mathrm{f}(B)$
$\mathrm{f}^*(A \wedge B)$	$\mathrm{f}^*(A)$	$\mathrm{f}^*(B)$
$\mathrm{t}(A \vee B)$	$\mathrm{t}(A)$	$\mathrm{t}(B)$
$\mathrm{t}^*(A \vee B)$	$\mathrm{t}^*(A)$	$\mathrm{t}^*(B)$

Table 5.3: Formulae of type β.

for $\{\mathrm{t}(B) | B \in \Gamma\} \cup \{\mathrm{f}(A)\}$.

An example of a closed tableau representing a proof of $((A \vee B) \wedge \neg A) \rightarrow (B \vee (A \wedge \neg A))$ is given in Fig. 5.1. A failed attempt at proving the disjunctive syllogism is shown in Fig. 5.2.

Soundness.

Definition 5.4.1 Let us say that a **4**-valuation v *realizes* $\mathrm{s}(A)$ if

1. $\mathrm{s}(A) = \mathrm{t}(A)$ and A is at least true under v.
2. $\mathrm{s}(A) = \mathrm{f}(A)$ and A is non-true under v.
3. $\mathrm{s}(A) = \mathrm{t}^*(A)$ and A is non-false under v
4. $\mathrm{s}(A) = \mathrm{f}^*(A)$ and A is at least false under v.

A set U of s-formulae is said to be *realizable* if there is a **4**-valuation v which realizes every element of U.

In order to prove soundness we need the following lemma:

Lemma 5.4.1 *If $\mathcal{T}$ is a closed tableau for U, where U is a set of signed formulae, then the tableau $\mathcal{T}'$ obtained from $\mathcal{T}$ by replacing every (occurrence of an) s-formula with (an occurrence of) its converse is a closed tableau for the set U^* of the* converses *of the s-formulae in U.*

The proof is a straightforward induction on the number of nodes in a closed tableau.

Now, it is easy to verify that our tableau rules are *correct* in the sense that every **4**-valuation which realizes the premise of the rule A realizes also both the conclusions of the rule, and every **4**-valuation which realizes the premise

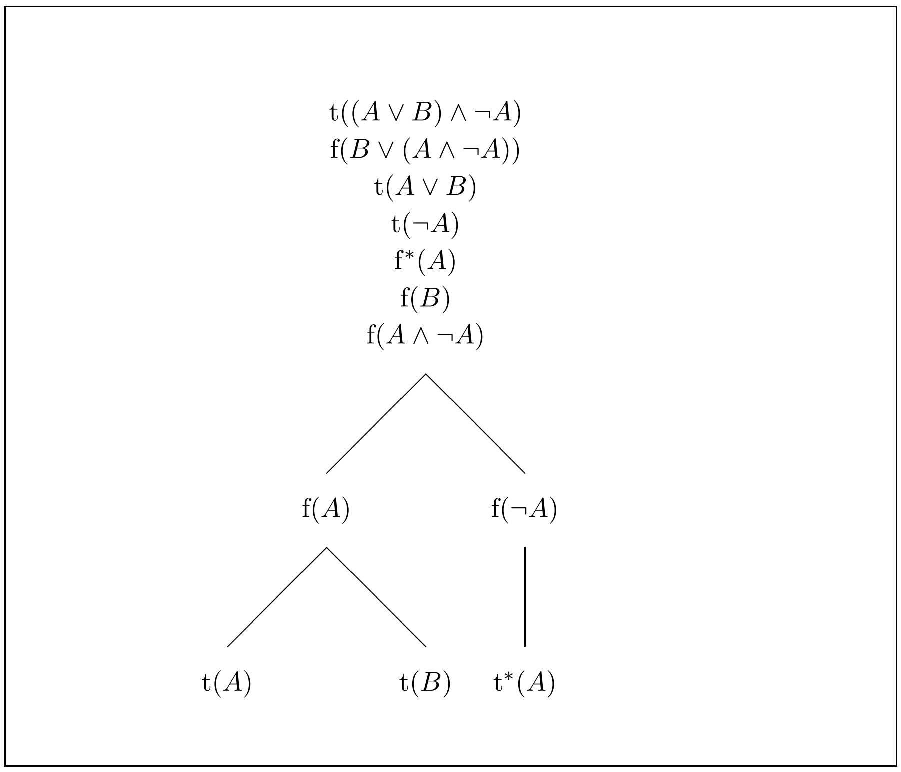


Figure 5.1: A proof of $((A \vee B) \wedge \neg A) \rightarrow (B \vee (A \wedge \neg A))$

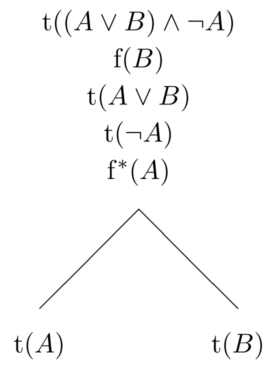


Figure 5.2: A failed attempt to prove the disjunctive syllogism

of the rule B realizes also at least one of the conclusions of the rule. Therefore it follows, by an elementary inductive argument, that if a **4**-valuation v realizes all the initial s-formulae of a tableau $\mathcal{T}$, then there is at least one branch ϕ of $\mathcal{T}$ such that v realizes all the s-formulae occurring in ϕ. But, of course, no **4**-valuation can realize two conjugate s-formulae simultaneously. Therefore if $\mathcal{T}$ is a a closed tableau, no **4**-valuation can realize all the initial s-formulae of $\mathcal{T}$. So, if $\mathcal{T}$ is a closed tableau for $\{\mathrm{t}(B) | B \in \Gamma\} \cup \{\mathrm{f}(A)\}$, it follows that for every **4**-valuation v, A is at least true in v whenever all formulae in Γ are. Moreover it follows from lemma 5.4.1 that no **4**-valuation can realize the *converses* of all the initial signed formulae of $\mathcal{T}$, i.e. no **4**-valuation can realize $\{\mathrm{t}^*(B) | B \in \Gamma\} \cup \{\mathrm{f}^*(A)\}$. Hence for every **4**-valuation v, A is non-false in v whenever all the formulae in Γ are. So, by def. 5.3.3, $\Gamma \vdash A$. This concludes the proof.

Completeness. Let us say that a branch ϕ of a tableau $\mathcal{T}$ is *complete* if (i) for every α in ϕ both α_1 and α_2 occur in ϕ and (ii) for every β in ϕ at least one of β_1, β_2 occurs in ϕ. Let us also say that a tableau $\mathcal{T}$ is *completed* when every branch of $\mathcal{T}$ is complete. We have the following theorem:

Theorem 5.4.1 *Every complete* open *branch of any tableau is realizable.*

We shall first define the analog of Hintikka sets within our framework. The theorem will then immediately follow from the analog of Hintikka's lemma.

Definition 5.4.2 Let us say that a set of *signed* formulae U is an *R-Hintikka set* if and only if it satisfies the following conditions:

H_0: No signed variable and its conjugate are both in U.

H_1: If $\alpha \in U$, then $\alpha_1 \in U$ and $\alpha_2 \in U$.

H_2: If $\beta \in U$, then $\beta_1 \in U$ or $\beta_2 \in U$.

It follows from our definitions that the set of s-formulae in a complete *open branch* of any tableau is an R-Hintikka set. Then the theorem is an immediate consequence of the following lemma:

Lemma 5.4.2 *Every R-Hintikka set is realizable.*

Proof. Let U be an R-Hintikkka set. Let us assign to each variable P which occurs in at least an element of U a value in **4** as follows:

(1) If $\mathrm{t}(P) \in U$, and $\mathrm{f}^*(P) \notin U$, give P the value **T**.
(2) If $\mathrm{t}(P) \in U$, and $\mathrm{f}^*(P) \in U$, give P the value **Both**.
(3) If $\mathrm{f}(P) \in U$, and $\mathrm{t}^*(P) \notin U$, give P the value **F**.
(4) If $\mathrm{f}(P) \in U$, and $\mathrm{t}^*(P) \in U$, give P the value **None**.
(5) if $\mathrm{t}^*(P) \in U$, and $\mathrm{f}(P) \notin U$, give P the value **T**.
(6) if $\mathrm{f}^*(P) \in U$, and $\mathrm{t}(P) \notin U$, give P the value **F**.

Now it is obvious that the **4**-valuation induced by this assignment realizes all the signed variables occurring in U. Then we only need to observe that

(i) If a **4**-valuation realizes both α_1 and α_2, then it realizes also α.

(ii) If a **4**-valuation realizes at least one of β_1,β_2 then it realizes also β.

The lemma then follows from an easy induction on the complexity of the s-formulae in U.

Theorem 5.4.1 implies

Theorem 5.4.2 (Completeness Theorem) *If* $\Gamma \vdash A$ *then there is a closed tableau for* $\{\mathrm{t}(B) | B \in \Gamma\} \cup \{\mathrm{f}(A)\}$.

5.4.2 Detecting inconsistencies

Although we do not want inconsistencies to have the catastrophic effect that they have in classical logic, it would be desirable to be able to check our databases for consistency. After all, we do not want to 'glorify' contradictions. On the contrary we usually want to remove them.

Fortunately, consistency checks are quite easy to carry out without modifying our framework. We shall show that, in fact, our tableaux 'contain' their classical version: to obtain classical tableaux we only need to modify the closure condition on a branch.

Definition 5.4.3 The *weak conjugate* of an s-formula is defined as in the following table:

Signed formula	Weak conjugate
$\mathrm{t}(A)$	$\mathrm{f}^*(A)$
$\mathrm{f}(A)$	$\mathrm{t}^*(A)$
$\mathrm{t}^*(A)$	$\mathrm{f}(A)$
$\mathrm{f}^*(A)$	$\mathrm{t}(A)$

We say that a branch ϕ of a tableau $\mathcal{T}$ is *weakly closed* if ϕ contains an s-formula and its *weak conjugate*. The tableau $\mathcal{T}$ is *weakly closed* if all its branches are weakly closed.

Let us use s as a variable ranging over *signs*, i.e. over the symbols $\{\mathrm{t}, \mathrm{f}, \mathrm{t}^*, \mathrm{f}^*\}$. By $\overline{s}$ we mean:

f	if	$s = \mathrm{t}$
t	if	$s = \mathrm{f}$
f^*	if	$s = \mathrm{t}^*$
t^*	if	$s = \mathrm{f}^*$

By s^* se mean:

t^*	if	$s = \mathrm{t}$
f^*	if	$s = \mathrm{f}$
t	if	$s = \mathrm{t}^*$
f	if	$s = \mathrm{f}^*$

We have the following lemma:

Lemma 5.4.3 *Let us say that a set of s-formulae S is* homogeneous *if all the s-formulae in it have the same sign s. Then if S is a homogeneous set of s-formulae, every tableau $\mathcal{T}$ for S contains only s-formulae with sign s or $\overline{s^*}$.*

Proof: By inspection of the rules and induction on the rank of the s-formulae occurring in $\mathcal{T}$.

The following theorem states the connection between classical and relevant derivability:

Theorem 5.4.3 Γ *is a (classically) unsatisfiable set of formulae if and only if there is a weakly closed tableau for* $\{\mathrm{t}(B)|B \in \Gamma\}$*. Therefore* A *is classically deducible from* Γ *if and only if there is a weakly closed tableau for* $\{\mathrm{t}(B)|B \in \Gamma\} \cup \{\mathrm{t}(\neg A)\}$*.*

Proof: If Γ is classically unsatisfiable, there is a closed classical tableau for $\{\mathrm{t}(B)|B \in \Gamma\}$. It is easy to see that the tableau obtained from it by replacing each sign f with f^* is a weakly closed tableau for $\{\mathrm{t}(B)|B \in \Gamma\}$.

Suppose there is a weakly-closed tableau $\mathcal{T}$ for $\{\mathrm{t}(B)|B \in \Gamma\}$. By Lemma 5.4.3 this tableau can contain only formulae signed with t or f^*. Therefore every weakly-closed branch contains $\mathrm{t}(A)$ and $\mathrm{f}^*(A)$ for some A. It follows that every **4**-valuation which realizes the initial formulae, must realize both $\mathrm{t}(A)$ and $\mathrm{f}^*(A)$ for some A, i.e. it must assign **Both** to A for some A. Hence there is no boolean valuation which satisfies Γ.

As we said before, the notion of weakly closed tableau is useful in that a weakly closed tableau shows an inconsistency in our database and, therefore, the need for revision. So our tableaux are sensitive to contradictions like classical tableaux; unlike classical tableaux, however, when they detect a contradiction they do not panic but keep on making 'sensible' deductions.

We observe that, in our approach, an initial formula of the form $\mathrm{f}(A)$ always represents a 'query', whereas data are represented by initial formulae of the form $\mathrm{t}(A)$. So, checking our data for consistency and answering questions on the base of our data are *formally* distinct tasks.

5.4.3 First-order tableaux

Another advantage of our single-tableau formulation is in the treatment of quantifiers. These are dealt with by means of rules obtained, as in the case of the binary connective rules, by taking over the standard classical rules (for s-formulae) and supplementing them with their 'starred' versions. By contrast, the suitable quantifier rules for the first-order version of the method of coupled-trees are not so straightforward.

We consider a standard first-order language with no functional symbols. We use the letters $x, y, z, \ldots$ (possibly with subscripts) as individual *variables* and the letters $a, b, c, \ldots$ (possibly with subscripts) as *parameters*. For any variable x and parameter a, $A(x/a)$ will be the result of substituting all the free occurrences of x in A with a. *Subformulae* are defined in the usual way, so that for every parameter a, $A(x/a)$ is a subformula of $\forall x A(x)$.

An obvious way of extending Belnap's semantics to formulae containing quantifiers is the following:

Let **U** be a non-empty universe. We assume a set U of *constants* naming the elements of **U** (of course neither **U** nor U need to be denumerable). By a U-formula we mean a formula built up from the logical operators, the relation symbols, the individual variables and the constants in U. (Thus a U-formula is like a formula with parameters but with elements of U in place of parameters.) Let F^U be the set of *closed* U-formulae. A *first-order valuation over* **U** is defined as a mapping v of all elements of F^U into **4** such that (i) v is a **4**-valuation (see above p. 86) and (ii) v satisfies the following additional conditions for quantifiers (where $\sqcap V$ and $\sqcup V$ denote, respectively, the g.l.b. and the l.u.b. of the set V in **L4**):

$$\begin{aligned} v(\forall x(A(x)) &= \sqcap\{v(A(k/x))|k \in U\} \\ v(\exists x(A(x)) &= \sqcup\{v(A(k/x))|k \in U\} \end{aligned}$$

The first-order consequence relation associated with this extended semantics is then:

Definition 5.4.4 $\Gamma \vdash A$ if and only if for every universe **U** and all first-order valuations v over **U**, (i) if all elements of Γ are at least true under v, then A is at least true under v *and* (ii) if all the elements of Γ are non-false under v, then A is non-false under v.

The quantifier rules are the expected ones:

$$\frac{\mathrm{t}(\forall xA(x))}{\mathrm{t}(A(a/x))} \quad \text{for } \textit{all } a \quad \frac{\mathrm{t}^*(\forall xA(x))}{\mathrm{t}^*(A(a/x))}$$

$$\frac{\mathrm{f}(\forall xA(x))}{\mathrm{f}(A(a/x))} \quad \text{with } a \textit{ new} \quad \frac{\mathrm{f}^*(\forall xA(x))}{\mathrm{f}^*(A(a/x))}$$

$$\frac{\mathrm{t}(\exists xA(x))}{\mathrm{t}(A(a/x))} \quad \text{with } a \textit{ new} \quad \frac{\mathrm{t}^*(\exists xA(x))}{\mathrm{t}^*(A(a/x))}$$

$$\frac{\mathrm{f}(\exists xA(x))}{\mathrm{f}(A(a/x))} \quad \text{for } \textit{all } a \quad \frac{\mathrm{f}^*(\exists xA(x))}{\mathrm{f}^*(A(a/x))}$$

It is convenient to use Smullyan's notation for quantified formulae. So γ will denote any formula of one of the four forms $\mathrm{t}(\forall xA(x))$, $\mathrm{t}^*(\forall xA(x))$, $\mathrm{f}(\exists xA(x))$, $\mathrm{f}^*(\exists xA(x))$ and by $\gamma(a)$ we shall mean, respectively, $\mathrm{t}(A(x/a))$, $\mathrm{t}^*(A(x/a))$, $\mathrm{f}(A(x/a))$, $\mathrm{f}^*(A(x/a))$. Similarly δ will denote any formula of one of the four forms $\mathrm{t}(\exists xA(x))$, $\mathrm{t}^*(\exists xA(x))$, $\mathrm{f}(\forall xA(x))$, $\mathrm{f}^*(\forall xA(x))$ and

and by $\delta(a)$ we shall mean, respectively, $\mathrm{t}(A(x/a))$, $\mathrm{t}^*(A(x/a))$, $\mathrm{f}(A(x/a))$, $\mathrm{f}^*(A(x/a))$.

Thus, our first-order rules can be succintly expressed by the following two rules:

Rule C $\dfrac{\gamma}{\gamma(a)}$ for *any* parameter a

Rule D $\dfrac{\delta}{\delta(a)}$ for a *new* parameter a

Soundness and completeness of this first-order system can be proved — given the corresponding proofs for the propositional fragment — by means of straightforward adaptations of the standard proofs for classical tableaux (as given in [Smu68a]). In fact, in our framework, quantifiers do not involve any non-standard notion which does not arise already at the propositional level. For instance, in order to prove completeness, we define the first-order analog of a complete tableau (where, of course, a complete branch may be infinite) and a procedure which generates a complete tableau for a set U of s-formulae. We can then define first-order R-Hintikka sets (over a universe $\mathbf{U}$) by adding to the clauses in definition 5.4.2 the following two clauses:

H_3**:** If $\gamma \in U$, then for *every* k in U, $\gamma(k) \in U$.

H_4**:** If $\delta \in U$, then for *some* k in U, $\delta(k) \in U$.

It is easy to see that the set of s-formulae occurring in every *open* branch of a complete tableau is an R-Hintikka set. Then, completeness follows from the lemma:

Lemma 5.4.4 *Every R-Hintikka set for a universe* $\mathbf{U}$ *is realizable.*

The previous discussion about the completeness of the first-order systems shows the *heuristic* advantage of this approach. The proofs of the analogs of most of the theorems which hold for the classical version can simply be *carried over* to the non-standard version with minor modifications. In fact this is true of most of the theorems included in [Smu68a]. We just mention here that the proofs of the compactness theorem and of the analog of Lowenheim-Skolem theorem require virtually no modification. Moreover, any implementation of a classical tableau-based theorem prover can be easily adapted to our framework, so providing a theorem prover for Belnap's logic *and* classical logic (via the notion of 'weak closure') simultaneously. A 'naive' implementation in Prolog, adapted from a program by Fitting, is given in [Gor90]

$\dfrac{\mathrm{t}(A \wedge B)}{\begin{array}{l}\mathrm{t}(A)\\ \mathrm{t}(B)\end{array}}$	$\dfrac{\mathrm{t}^*(A \wedge B)}{\begin{array}{l}\mathrm{t}^*(A)\\ \mathrm{t}^*(B)\end{array}}$	$\dfrac{\begin{array}{l}\mathrm{f}(A \wedge B)\\ \mathrm{t}(A)\end{array}}{\mathrm{f}(B)}$	$\dfrac{\begin{array}{l}\mathrm{f}(A \wedge B)\\ \mathrm{t}(B)\end{array}}{\mathrm{f}(A)}$
$\dfrac{\begin{array}{l}\mathrm{f}^*(A \wedge B)\\ \mathrm{t}^*(A)\end{array}}{\mathrm{f}^*(B)}$	$\dfrac{\begin{array}{l}\mathrm{f}^*(A \wedge B)\\ \mathrm{t}^*(B)\end{array}}{\mathrm{f}^*(A)}$	$\dfrac{\mathrm{f}(A \vee B)}{\begin{array}{l}\mathrm{f}(A)\\ \mathrm{f}(B)\end{array}}$	$\dfrac{\mathrm{f}^*(A \vee B)}{\begin{array}{l}\mathrm{f}^*(A)\\ \mathrm{f}^*(B)\end{array}}$
$\dfrac{\begin{array}{l}\mathrm{t}(A \vee B)\\ \mathrm{f}(A)\end{array}}{\mathrm{t}(B)}$	$\dfrac{\begin{array}{l}\mathrm{t}(A \vee B)\\ \mathrm{f}(B)\end{array}}{\mathrm{t}(A)}$	$\dfrac{\begin{array}{l}\mathrm{t}^*(A \vee B)\\ \mathrm{f}^*(A)\end{array}}{\mathrm{t}^*(A)}$	$\dfrac{\begin{array}{l}\mathrm{t}^*(A \vee B)\\ \mathrm{f}^*(B)\end{array}}{\mathrm{t}^*(A)}$
$\dfrac{\mathrm{t}(\neg A)}{\mathrm{f}^*(A)}$	$\dfrac{\mathrm{f}(\neg A)}{\mathrm{t}^*(A)}$	$\dfrac{\mathrm{t}^*(\neg A)}{\mathrm{f}(A)}$	$\dfrac{\mathrm{f}^*(\neg A)}{\mathrm{t}(A)}$
$\dfrac{}{\mathrm{t}(A) \mid \mathrm{f}(A)}$	$\dfrac{}{\mathrm{t}^*(A) \mid \mathrm{f}^*(A)}$		

Table 5.4: The rules of $\mathbf{RE}_{\mathrm{fde}}$.

5.5 An efficient alternative

Another simple tree method for first-degree entailment is obtained by adapting the rules of the system **KE** (see above, Chapter 3). The system so obtained will be baptised $\mathbf{RE}_{\mathrm{fde}}$. The propositional rules of $\mathbf{RE}_{\mathrm{fde}}$ are given in Table 5.4. The quantifier rules are the same as the rules for the tableau method given in the previous section.

Again, the logical rules can be expressed in a succint form by means of our extended use of Smullyan's notation (where β_i', $i = 1, 2$ denotes the *conjugate* of β_i):

$$\text{Rule A} \quad \dfrac{\alpha}{\begin{array}{c}\alpha_1\\ \alpha_2\end{array}}$$

$$\text{Rule B1} \quad \dfrac{\begin{array}{c}\beta\\ \beta_1'\end{array}}{\beta_2} \qquad\qquad \text{Rule B2} \quad \dfrac{\begin{array}{c}\beta\\ \beta_2'\end{array}}{\beta_1}$$

$$\text{Rule C} \quad \dfrac{\gamma}{\gamma(a)} \quad \text{for } \textit{any} \text{ parameter } a$$

$$\text{Rule D} \quad \dfrac{\delta}{\delta(a)} \quad \text{for a } \textit{new} \text{ parameter } a$$

In each application of the rules, the s-formulae α, β, γ and δ are called *major*

$$\begin{array}{c}
\mathrm{t}((A \vee B) \wedge \neg A) \\
\mathrm{f}(B \vee (A \wedge \neg A)) \\
\mathrm{t}(A \vee B) \\
\mathrm{t}(\neg A) \\
\mathrm{f}^*(A) \\
\mathrm{f}(B) \\
\mathrm{f}(A \wedge \neg A) \\
\mathrm{t}(A) \\
\mathrm{f}(\neg A) \\
\mathrm{t}^*(A)
\end{array}$$

Figure 5.3: An $\mathbf{RE}_{\mathrm{fde}}$-proof of $((A \vee B) \wedge \neg A) \rightarrow (B \vee (A \wedge \neg A))$

premises. In each application of rules B1 and B2 the s-formulae β_i', $i = 1, 2$ are called *minor premises* (rules A, C and D have no minor premises).

In the system $\mathbf{RE}_{\mathrm{fde}}$ all the logical rules have a linear format. However, they are not sufficient for completeness: our 'cut-rules', PB and PB*, are *not* eliminable.

Definition 5.5.1 An $\mathbf{RE}_{\mathrm{fde}}$-*tree for* U, where U is a set of s-formulae, is a tree of s-formulae constructed in accordance with the rules above starting from s-formulae in U. A branch of an $\mathbf{RE}_{\mathrm{fde}}$-tree is *closed* when it contains a signed formula and its conjugate. Otherwise it is *open*. The tree itself is said to be *closed* when all its branches are closed.

An s-formula occurring in a $\mathbf{RE}_{\mathrm{fde}}$-tree is said to be *analysed* if it has been used at least once as *major* premise of one of the rules.

In Fig. 5.3 we give an $\mathbf{RE}_{\mathrm{fde}}$-proof of the same example of which a tableau proof was given in Fig. 5.1. The reader can compare the structure of proofs in the two methods. Notice that the $\mathbf{RE}_{\mathrm{fde}}$-tree contains no branching. Indeed, the system $\mathbf{RE}_{\mathrm{fde}}$ is more efficient than the tableau method formulated in the previous section for much the same reason as the classical system **KE** is more efficient than the classical tableau method (see Chapters 2 and 4 above): the use of PB and PB* — our 'cut rules' — allow us to avoid many redundant branchings in the refutation tree.

A lemma analogous to Lemma 5.4.1 holds:

Lemma 5.5.1 *If $\mathcal{T}$ is an* $\mathbf{RE}_{\mathrm{fde}}$*-tree for U, where U is a set of signed formulae, then the tree $\mathcal{T}'$ obtained from $\mathcal{T}$ by replacing every (occurrence of an) s-formula with (an occurrence of) its converse is an* $\mathbf{RE}_{\mathrm{fde}}$*-tree for the set U^* of the* converses *of the s-formulae in U.*

Soundness. The proof is strictly analogous to the one given for the tableau method.

Completeness. We define a notion akin to the notion of R-Hintikka set :

Definition 5.5.2 Let us say that a set of *signed* formulae U is an *R-analytic set* if and only if it satisfies the following conditions:

A_0**:** No signed variable and its conjugate are both in U.

A_1**:** If $\alpha \in U$, then $\alpha_1 \in U$ and $\alpha_2 \in U$.

A_2**:** If $\beta \in U$ and $\beta_1' \in U$, then $\beta_2 \in U$.

A_3**:** If $\beta \in U$ and $\beta_2' \in U$, then $\beta_1 \in U$.

A_4**:** If $\gamma \in U$, then for *every* k in U, $\gamma(k) \in U$.

A_5**:** If $\delta \in U$, then for *some* k in U, $\delta(k) \in U$.

An R-analytic set differs from a R-Hintikka set in that it may be the case that for some β in the set neither β_1 nor β_2 are in the set.

Definition 5.5.3 We say that an R-analytic set is *β-complete* if for every β in U either of the following two conditions is satisfied:

1. either $\beta_1 \in U$ or $\beta_1' \in U$;
2. either $\beta_2 \in U$ or $\beta_2' \in U$;

It is then easy to verify that:

Fact 5.5.1 *If U is an R-analytic set and U is β-complete, then U is an R-Hintikka set.*

It is not difficult to define a procedure which, given a set of s-formulae U, generates either a closed $\mathbf{RE}_{\mathrm{fde}}$-tree or an open $\mathbf{RE}_{\mathrm{fde}}$-tree such that for every (possibly infinite) open branch ϕ the set of all the s-formulae occurring in ϕ is a R-analytic set which is also β-complete (the only tricky part of such a procedure concerns condition A_4 and can be dealt with as in [Smu68a, pp.58–60]). Thus, completeness follows from fact 5.5.1 and lemma 5.4.4.

We can define the notions of *weakly closed branch* and *weakly closed tree* in exactly the same way as we did for the tableau method discussed in the previous section. Again the relation between classical and relevant deducibility is the same *mutatis mutandis.*

Our proof of the completeness of $\mathbf{RE}_{\text{fde}}$ yields the subformula principle as a corollary:

Corollary 5.5.1 (Analytic Cut Property) *If there is a closed* $\mathbf{RE}_{\text{fde}}$*-tree* $\mathcal{T}$ *for* U*, then there is a closed* $\mathbf{RE}_{\text{fde}}$*-tree* $\mathcal{T}'$ *for* U *such that the rules PB and PB* are applied only to subformulae of s-formulae in* U*.*

In fact, the proof shows that, when applying PB or PB*, we need to consider only the immediate signed subformulae of signed formulae of type β occurring above in the same branch and which have not been already 'analysed'. Since all the logical rules preserve the subformula property, we have:

Corollary 5.5.2 (Subformula Principle) *If there is a closed* $\mathbf{RE}_{\text{fde}}$*-tree* $\mathcal{T}$ *for* U*, then there is a closed* $\mathbf{RE}_{\text{fde}}$*-tree* $\mathcal{T}'$ *for* U *such that every s-formula occurring in* $\mathcal{T}'$ *is a signed subformula of s-formulae in* U

A constructive proof of the subformula principle, which yields a procedure for transforming any $\mathbf{RE}_{\text{fde}}$-proof in an equivalent $\mathbf{RE}_{\text{fde}}$-proof which enjoys the subformula property, can be obtained by adapting the proof given by Mondadori for the system **KE** [Mon88b].

The completeness of the (propositional fragment of the) system $\mathbf{RE}_{\text{fde}}$ can also be proved by showing that all the axioms of the Hilbert-style formulation (given on p. 86) are theorems of $\mathbf{RE}_{\text{fde}}$ and $\mathbf{RE}_{\text{fde}}$-derivability is closed under the rules (8)-(11). All these facts are shown below.

In what follows $\vdash$ stands for $\vdash_{\mathbf{RE}_{\text{fde}}}$.

Fact 5.5.2 $A \wedge B \vdash A$

(1)	t$(A \wedge B)$	Assumption
(2)	f(A)	Assumption
(3)	t(A)	Et$\wedge$ (1)
(4)	t(B)	Et$\wedge$ (1)

Fact 5.5.3 $A \wedge B \vdash B$

Proof as above.

Fact 5.5.4 $A \vdash A \vee B$

(1)	t(A)	Assumption
(2)	f($A \vee B$)	Assumption
(3)	f(A)	Ef$\vee$ (2)
(4)	f(B)	Ef$\vee$ (2)

Fact 5.5.5 $B \vdash A \vee B$

Proof as above.

Fact 5.5.6 $A \wedge (B \vee C) \vdash (A \wedge B) \vee C$

(1)	t($A \wedge (B \vee C)$)	Assumption
(2)	f($(A \wedge B) \vee C$)	Assumption
(3)	t(A)	Et$\wedge$ (1)
(4)	t($B \vee C$)	Et$\wedge$ (1)
(5)	f($A \wedge B$)	Ef$\vee$ (2)
(6)	f(C)	Ef$\vee$ (2)
(7)	t(B)	Et $\vee$ 2 (4,6)
(8)	f(B)	Ef $\wedge$ 1 (5,3)

Fact 5.5.7 $A \vdash \neg\neg A$

(1)	t(A)	Assumption
(2)	f($\neg\neg A$)	Assumption
(3)	t*($\neg A$)	Ef$\neg$ (2)
(4)	f(A)	Et$^*\neg$ (3)

Fact 5.5.8 $\neg\neg A \vdash A$

(1)	t($\neg\neg A$)	Assumption
(2)	f(A)	Assumption
(3)	f*($\neg A$)	Ef$\neg$ (1)
(4)	t(A)	Ef$^*\neg$ (3)

Fact 5.5.9 *If $A \vdash B$ and $B \vdash C$, then $A \vdash C$*

It follows from the hypothesis that there are closed trees $\mathcal{T}_1$ and $\mathcal{T}_2$ for $\{\mathrm{t}(A), \mathrm{f}(B)\}$ and $\{\mathrm{t}(B), \mathrm{f}(C)\}$ respectively. Therefore the following is a closed tree for $\{\mathrm{t}(A), \mathrm{f}(C)\}$:

$$\begin{array}{ccc} & \mathrm{t}(A) & \\ & \mathrm{f}(C) & \\ \hline \mathrm{t}(B) & | & \mathrm{f}(B) \\ \mathcal{T}_2 & & \mathcal{T}_1 \end{array}$$

Fact 5.5.10 *If $A \vdash B$ and $A \vdash C$, then $A \vdash B \wedge C$*

It follows from the hypothesis that there are closed trees $\mathcal{T}_1$ and $\mathcal{T}_2$ for $\{\mathrm{t}(A), \mathrm{f}(B)\}$ and $\{\mathrm{t}(A), \mathrm{f}(C)\}$ respectively. Therefore the following is a closed tree for $\{\mathrm{t}(A), \mathrm{f}(B \wedge C)\}$:

$$\begin{array}{lcr} \multicolumn{3}{c}{\mathrm{t}(A)} \\ \multicolumn{3}{c}{\mathrm{f}(B \wedge C)} \\ \hline \mathrm{t}(B) & | & \mathrm{f}(B) \\ \mathrm{f}(C) & & \mathcal{T}_1 \\ \mathcal{T}_2 & & \end{array}$$

Fact 5.5.11 *If $A \vdash C$ and $B \vdash C$, then $A \vee B \vdash C$*

It follows from the hypothesis that there are closed trees $\mathcal{T}_1$ and $\mathcal{T}_2$ for $\{\mathrm{t}(A), \mathrm{f}(C)\}$ and $\{\mathrm{t}(B), \mathrm{f}(C)\}$ respectively. Therefore the following is a closed tree for $\{\mathrm{t}(A \vee B), \mathrm{f}(C)\}$:

$$\begin{array}{lcr} \multicolumn{3}{c}{\mathrm{t}(A \vee B)} \\ \multicolumn{3}{c}{\mathrm{f}(C)} \\ \hline \mathrm{t}(A) & | & \mathrm{f}(A) \\ \mathcal{T}_1 & & \mathrm{t}(B) \\ & & \mathcal{T}_2 \end{array}$$

Fact 5.5.12 *If $A \vdash B$, then $\neg B \vdash \neg A$.*

By hypothesis there is a closed trees $\mathcal{T}$ for $\{\mathrm{t}(A), \mathrm{f}(B)\}$. It follows from Lemma 5.5.1 that there is a closed tree $\mathcal{T}'$ for$\{\mathrm{t}^*(A), \mathrm{f}^*(B)\}$. Therefore the following is a closed tree for $\{\mathrm{t}(\neg B), \mathrm{f}(\neg A)\}$:

$$\begin{array}{c} \mathrm{t}(\neg B) \\ \mathrm{f}(\neg A) \\ \mathrm{f}^*(B) \\ \mathrm{t}^*(A) \\ \mathcal{T}' \end{array}$$

We conclude by mentioning that a 'naive' implementation in Prolog of the system $\mathbf{RE}_{\mathrm{fde}}$ has been developed by Rajev Gore [Gor90].