



Complete Sets of Connectives and Complete
Sequent Calculus for Belnap's Logic

Paul RUET

LIENS - 96 - 28

Département de Mathématiques et Informatique

CNRS URA 1327

**Complete Sets of Connectives and
Complete Sequent Calculus for Belnap's
Logic**

Paul RUET

LIENS - 96 - 28

December 1996

Laboratoire d'Informatique de l'Ecole Normale Supérieure
45 rue d'Ulm 75230 PARIS Cedex 05

Tel : (33)(1) 44 32 00 00

Adresse électronique : ... @dmi.ens.fr

Complete sets of connectives and complete sequent calculus for Belnap's Logic

Paul Ruet
LIENS, Ecole Normale Supérieure
45 rue d'Ulm, 75005 Paris, France

Thomson - Laboratoire Central de Recherches
Domaine de Corbeville, 91404 Orsay Cedex, France

Email: ruet@dmi.ens.fr

Abstract

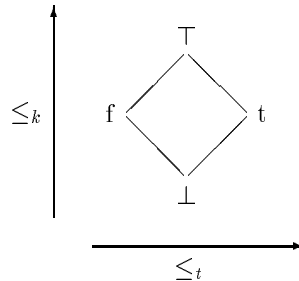
This paper presents a sequent calculus for Belnap's four-valued logic, and proves its soundness and completeness with respect to model theory. The proofs are adaptations of Gallier's proofs for classical logic.

It is also shown that the extensions of the classical connectives \neg and \wedge to the four-valued case do not form a complete set, and a new connective is introduced to obtain the completeness of the connectives.

1 INTRODUCTION

The interest of three-valued logic for computer science has been motivated by research on the semantics of negation in logic programming, from the model-theoretical [4, 11, 12] and the proof-theoretical viewpoints [14].

Moving from three-valued to four-valued logic provides a framework for logic programs dealing with missing or conflicting information, e.g information distributed over several sites [6]. It also sheds a new light on the stable model semantics for logic programming (see [3]), allowing to define a new stable model, the largest one in a certain ordering, and to recover at the same time the usual smallest stable model and alternating fixed points: the whole family of stable models is then enclosed between these 4 fixed points [8].



The four-valued logic extending Kleene's three-valued logic was introduced by Belnap in [2]. Let us briefly recall its main features. The 4 truth values (denoted f , t , \perp and \top) can be

thought of as subsets of $\{false, true\}$, where *false* and *true* are the ordinary truth values: in this interpretation, *t* is simply $\{true\}$, *f* is $\{false\}$, \perp is \emptyset representing a lack of information, and \top is $\{false, true\}$ indicating inconsistency. The truth values of Belnap's logic have two natural orderings \leq_t (the *truth ordering*) and \leq_k (the *knowledge ordering*):

- $f \leq_t \perp$, $f \leq_t \top$, $\perp \leq_t t$, $\top \leq_t t$,
- $\perp \leq_k f$, $\perp \leq_k t$, $f \leq_k \top$, $t \leq_k \top$ (subset ordering).

Another way of looking at these truth values is to consider them as members of $\{0, 1\} \times \{0, 1\}$, where *t* is $(1, 0)$, *f* is $(0, 1)$, \perp is $(0, 0)$ and \top is $(1, 1)$: the first projection represents then the degree of confidence we have in a sentence, and the second projection the degree of doubt.

The set *Four* of truth values has the structure of a bilattice, i.e it is a lattice for each of both orderings \leq_t and \leq_k . Meet and join under \leq_t are denoted \wedge and \vee and are natural generalisations of the usual conjunction and disjunction. Meet and join under \leq_k are denoted \otimes and \oplus : $x \otimes y$ is the most information x and y can agree on (*consensus operator*); $x \oplus y$ combines the knowledges represented by x and y , without checking for consistency (*gullibility operator*).

There is a natural notion of negation, denoted \neg , that flips the diagram from right to left, exchanging *f* and *t*, leaving \perp and \top unchanged; in the $\{0, 1\} \times \{0, 1\}$ interpretation of *Four*, \neg switches confidence and doubt. There is also an operator for vertical symmetry, called *conflation* and denoted $-$, that exchanges \perp and \top , leaving *f* and *t* unchanged.

	f	t	\perp	\top		f	t	\perp	\top
\neg	t	f	\perp	\top	$-$	f	t	\top	\perp

\wedge	f	t	\perp	\top
f	f	f	f	f
t	f	t	\perp	\top
\perp	f	\perp	\perp	f
\top	f	\top	f	\top

\vee	f	t	\perp	\top
f	f	t	\perp	\top
t	t	t	t	t
\perp	\perp	t	\perp	t
\top	\top	t	t	\top

\otimes	f	t	\perp	\top
f	f	\perp	\perp	f
t	\perp	t	\perp	t
\perp	\perp	\perp	\perp	\perp
\top	f	t	\perp	\top

\oplus	f	t	\perp	\top
f	f	\top	f	\top
t	\top	t	t	\top
\perp	f	t	\perp	\top
\top	\top	\top	\top	\top

Contrary to three-valued logic, four-valued logic has not been studied from the proof-theoretical viewpoint: however the symmetry induced by the bilattice structure and the intuitiveness of the connectives in the four-valued case suggest to take an interest in a proof-theoretical approach.

The rest of this paper is divided in two parts, corresponding to the propositional and predicate calculi. In each part, the main result is the completeness of the calculus. The methods used are adaptations of Gallier's for classical logic [9], which are suitable for sequent calculi: the adaptation to the four-valued case necessitates in particular to define the notion of satisfaction, first for sequents and then for formulae (Section 2), and to modify the definition of Hintikka sets (Section 3). In section 2, we also prove in passing that the extensions of the classical connectives \neg and \wedge to four-valued logic do not form a complete set, and a new connective \curlywedge is introduced, such that \neg , \wedge and \curlywedge form a complete set. We could have shown

that the extensions of \neg and \wedge to three-valued logic are not complete either: in this sense, the step from three-valued to four-valued logic is “free”, because in either case we have to add a new connective.

Further research interests include:

- Logic programming: deduce a resolution principle like Gallier for classical logic [9]; extend the work of [14] to the four-valued semantics.
- Computational interpretation: the notion of cut is not taken into account in this paper; a deduction system with a notion of cut that allows a global cut-elimination procedure (like natural deduction or free deduction) could give a framework for studying the algorithmic meaning of Belnap’s logic.
- Comparison with other sequent calculi for three-valued logic [10, 14].
- Extension to the case of truth spaces that are (general) bilattices [6, 7].

2 PROPOSITIONAL BELNAP’S LOGIC

2.1 THE SET OF TRUTH VALUES

The set of truth values is the complete bilattice $\mathcal{Four} = \{f, t, \perp, \top\}$ together with the two orderings \leq_t and \leq_k as presented in the Introduction.

Complete Sets of Connectives.

In classical logic, the connectives \neg , \wedge and \vee form a complete set, i.e for every integer n , all the mappings from $\{f, t\}^n$ to $\{f, t\}$ can be expressed by composition of some of them. In fact, \neg and \wedge suffice to form a complete set for classical logic.

When moving to Belnap’s logic, the connectives \neg and \wedge extending the corresponding classical ones do not form a complete set. Precisely: if K_n are sets of mappings from $\{f, t, \perp, \top\}^n$ to $\{f, t, \perp, \top\}$, and $K = \bigcup K_n$, let us call \overline{K} the intersection of all the sets S such that:

- (i) S contains K , the 0-ary f, t, \perp, \top , and the k^{th} n -ary projection π_n^k for every integers $n \geq k \geq 1$,
- (ii) if $f_1 \dots f_p$ are in S and all n -ary, and if κ is in K and p -ary, then $\kappa \circ \langle f_1 \dots f_p \rangle$ is in S and is n -ary.

We say that the set K is *complete for the bilattice \mathcal{Four}* if \overline{K} contains all the mappings from $\{f, t, \perp, \top\}^n$ to $\{f, t, \perp, \top\}$, for every integer n .

Proposition 1 *The connectives \neg and \wedge do not form a complete set for the bilattice \mathcal{Four} .*

Proof: let K be this set of connectives. We shall show that the unary function \curlyvee which maps f to \perp , \perp to t , t to \top and \top to f , does not belong to \overline{K} . (We will call this function *quarter turn* because it lets the bilattice \mathcal{Four} “do a quarter turn”)

In fact, to proceed by induction on the mappings in \overline{K} , we prove a stronger result, namely: for each unary function f in \overline{K} , one of the following situations holds:

$$P(f) \left\{ \begin{array}{l} \text{(i) } f \text{ is one of the unary constants } f, t, \perp \text{ or } \top \text{ (type 1),} \\ \text{(ii) } f(\top) = \top \text{ and } f(\perp) = \perp \text{ (type 2),} \\ \text{(iii) } f(\top) = \top, f(\perp) \neq \top, \perp \notin \text{Im}(f) \text{ and either } t \text{ or } f \notin \text{Im}(f) \text{ (type 3),} \\ \text{(iv) } f(\perp) = \perp, f(\top) \neq \perp, \top \notin \text{Im}(f) \text{ and either } t \text{ or } f \notin \text{Im}(f) \text{ (type 4).} \end{array} \right.$$

The identity π_1^1 and negation are trivially of type 2. One proves easily that $P(f)$ implies $\neg f$ is of the same type as f . Now we have to prove that $P(f)$ and $P(g) \implies P(f \wedge g)$, distinguishing between the respective types of f and g ; as examples, we shall examine two cases (among the 10 possible ones):

• f is of type 1 and g is of type 3: if $f = f$, then $f \wedge g = f$. If $f = t$, then $f \wedge g = g$. If $f = \top$, then either $g(\perp) = t$ and then $(f \wedge g)(\perp) = \top$, besides $g(f)$ and $g(t)$ are among t and \top , so $f \wedge g = \top$; or $g(\perp) = f$ and then $(f \wedge g)(\perp) = f$, besides $g(f)$ and $g(t)$ are among f and \top , so $f \wedge g$ is of type 3. If $f = \perp$, then either $g(\perp) = f$ and then $f \wedge g = f$; or $g(\perp) = t$ and then $(f \wedge g)(\perp) = \perp$ and $(f \wedge g)(\top) = f$, besides $g(f)$ and $g(t)$ are among t and \top , so $(f \wedge g)(f)$ and $(f \wedge g)(t)$ are among f and \perp , and $f \wedge g$ is of type 4.

• f is of type 3 and g is of type 4: if $f(\perp) = g(\top) = t$, then $f \wedge g$ is of type 2. If $f(\perp) = g(\top) = f$, then $f \wedge g = f$. If $f(\perp) = t$ and $g(\top) = f$, then $(f \wedge g)(\perp) = \perp$ and $(f \wedge g)(\top) = f$, besides $g(t)$ and $g(f)$ are among t and \perp , hence so are $(f \wedge g)(t)$ and $(f \wedge g)(f)$, and $f \wedge g$ is of type 4. Similarly, if $f(\perp) = f$ and $g(\top) = t$, then $f \wedge g$ is of type 3.

Hence $P(f)$ holds for each unary $f \in \overline{K}$. But $P(\curlyvee)$ does not hold, so $\curlyvee \notin \overline{K}$. \square

Hence, to get a complete set of connectives, we need add at least the quarter turn \curlyvee to \neg and \wedge . In fact it suffices:

Proposition 2 *The connectives \neg , \wedge and \curlyvee form a complete set for the bilattice Four.*

Proof: let K' be this set of connectives. We prove, by induction on n , that every function from $\{f, t, \perp, \top\}^n$ to $\{f, t, \perp, \top\}$ is in $\overline{K'}$.

• $n = 0$: because of condition (i), all truth values are in $\overline{K'}$.

• $n + 1$: let us first notice that for any truth values x and y , $x \vee y = \neg(\neg x \wedge \neg y)$; if we define unary connectives $/$ and \backslash by $/x = \neg\curlyvee x$ (exchanges f and \top , \perp and t) and $\backslash x = \neg/\neg x$ (exchanges f and \perp , t and \top), then $\neg x = / \neg / x$. Hence the connectives \vee , $/$, \backslash and \neg belong to $\overline{K'}$.

Now let f be a $(n + 1)$ -ary function. We define the n -ary functions g, h, k, l by:

$$\begin{cases} f(x_1 \dots x_n, f) = g(x_1 \dots x_n) \\ f(x_1 \dots x_n, t) = h(x_1 \dots x_n) \\ f(x_1 \dots x_n, \perp) = k(x_1 \dots x_n) \\ f(x_1 \dots x_n, \top) = l(x_1 \dots x_n) \end{cases}$$

If $(?t)$ is the unary function that lets t invariant and maps the other truth values to f , then we have: $f(x_1 \dots x_{n+1}) = [(?t)(\neg x_{n+1}) \wedge g(x_1 \dots x_n)] \vee [(?t)x_{n+1} \wedge h(x_1 \dots x_n)] \vee [(?t)(/x_{n+1}) \wedge k(x_1 \dots x_n)] \vee [(?t)(\backslash x_{n+1}) \wedge l(x_1 \dots x_n)]$.

But $(?t)x = x \wedge \neg x$. Hence f can be expressed in K' , and the result is proved for any integer n . \square

The connectives \neg , \vee , \wedge and \otimes , presented in our Introduction (as in [2, 5]) also form a complete set:

Proposition 3 *The connectives \neg , \vee , \wedge and \otimes form a complete set for the bilattice Four.*

Proof: let K'' be this set of connectives. Observe that $x \oplus y = \neg(\neg x \otimes \neg y)$. So $(?t)$, \vee and \oplus can be expressed in K'' . With the preceding notations we have: $f(x_1 \dots x_{n+1}) = [(?t)(\neg x_{n+1}) \wedge g(x_1 \dots x_n)] \vee [(?t)x_{n+1} \wedge h(x_1 \dots x_n)] \vee [(?t)((x_{n+1} \oplus \neg x_{n+1}) \vee \top) \wedge k(x_1 \dots x_n)] \vee [(?t)((x_{n+1} \otimes \neg x_{n+1}) \vee \perp) \wedge l(x_1 \dots x_n)]$. \square

However, from the next Section, we shall consider \neg , \wedge and \curlyvee because they are a bit more primitive and easier to deal with.

Algebraic Properties.

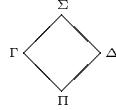
Negation and conflation are idempotent and commute, as well as $/$ and \backslash . Quarter turn \curlyvee is nilpotent with degree 4. Considering the binary connectives as meets and joins, one easily

proves that they are commutative and associative. All 12 distributive laws between \wedge , \vee , \otimes and \oplus hold. Furthermore \wedge and \vee are dual for negation \neg , \otimes and \oplus are dual for conflation \neg , \wedge and \oplus (resp. \vee and \otimes) are dual for $/$, \wedge and \otimes (resp. \vee and \oplus) are dual for \backslash (De Morgan laws).

2.2 ELEMENTARY MODEL THEORY

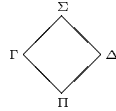
In the preceding section, connectives were seen as mappings between truth values; in this section, the same symbols will be used to represent syntactic symbols inside propositions. The link is made, as usual, when we interpret the valuation of a non-atomic proposition with the truth tables.

As in the classical setting, a *propositional language* is a set \mathcal{P} of propositional symbols A, B, \dots , also called *atomic propositions*. A *proposition* is built up, in the usual way, from propositional symbols and a set of *connectives*: we will consider negation \neg and quarter turn \curvearrowright of arity 1, and the binary connective \wedge . A *sequent* is a quadruplet $(\Gamma, \Delta, \Pi, \Sigma)$, where $\Gamma, \Delta, \Pi, \Sigma$ are finite sets of propositions. Sequents are represented by diamonds, recalling the bilattice structure of *Four*:




A *valuation* is a mapping from \mathcal{P} to *Four*. Valuations are extended to propositions according to the truth tables (for example, $v(A \wedge B) = v(A) \wedge v(B)$).

As explained above, we first define the notion of satisfaction for sequents, and then particularise it to propositions. We say that a valuation v *satisfies* the sequent:

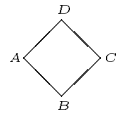


if for v , either:

- ⎧ one of the propositions in Γ has truth value f , or
- ⎧ one of the propositions in Δ has truth value t , or
- ⎧ one of the propositions in Π has truth value \perp , or
- ⎧ one of the propositions in Σ has truth value \top .

A valuation v satisfies a proposition A if it satisfies the sequent  A .

Remark: contrary to the classical setting, a valuation may satisfy the proposition $C \vee \neg A \vee /B \vee \backslash D$ but not the sequent:



A sequent (or a proposition) is said *valid* if every valuation satisfies it. A sequent (or a proposition) is said *invalid* if it is not valid.

Example: $/A \vee \backslash A$ is valid.

Remark: we use the term *invalid* instead of *falsifiable* because it can happen that, e.g a proposition is invalid, but there exists no valuation that provides it with truth value f.

“Equivalent” Connective: two propositions A and B are said *equivalent*, noted $A \approx B$, if for every valuation v , $v(A) = v(B)$. There exists a connective \equiv such that $A \equiv B$ is valid iff $A \approx B$, but it is slightly more sophisticated than in the classical case: because a disjunction $C \vee D$ can be t, but neither C nor D is, we are led to define $A \equiv B = (?t)[A \wedge B] \vee (?t)[\neg A \wedge \neg B] \vee (?f)[A \wedge \neg B] \vee (?f)[\neg A \wedge B]$. In fact, $A \equiv B$ says a bit more: for any valuation v , $v(A \equiv B) = t$ iff $v(A) = v(B)$, and f in the other cases.

2.3 SEQUENT CALCULUS

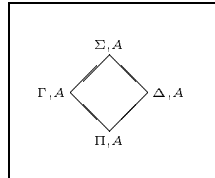
In this section, we present a deduction system LB'_0 for propositional Belnap’s logic, and we prove its soundness with respect to model theory. Completeness is proved in the next section.

Usual sequent calculi are characterized by two properties: logical rules are all *introductions* (of connectives), and each connective is introduced by some rules (maybe just one) in the left part of the sequent conclusion, and by some other rules in the right part (*symmetry*). Our deduction system is a sequent calculus in the sense that it has both same properties: all logical rules are introductions, and there is an introduction rule for each of the connectives \neg , \wedge and \vee in each part of the sequent conclusion (left, right, bottom and top).

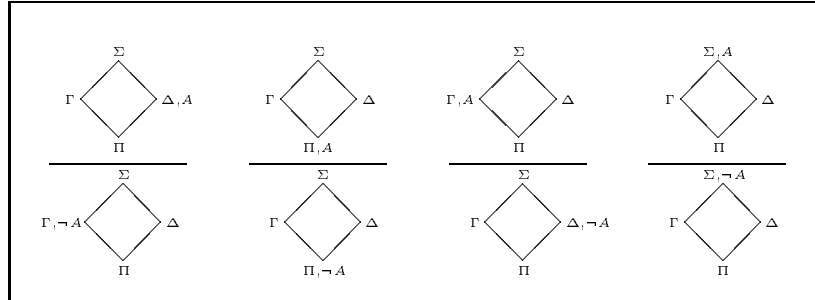
In order to simplify the proof of completeness, system LB'_0 has no structural rule, and only one logical rule for each connective and each sequent part, like Gallier’s system G' for classical logic [9]. Usual weakenings are contained in axioms. Exchanges and contractions are implicit, since we consider Γ , Π , Δ and Σ as sets of formulae.

Remark: the counterpart is an increase of the number of sequents premises: some logical rules have 3 or 4 premises. Of course, one can define sequent calculi whose rules have only 2 premises, thus maybe more readable, but not that manageable for the purpose of this paper.

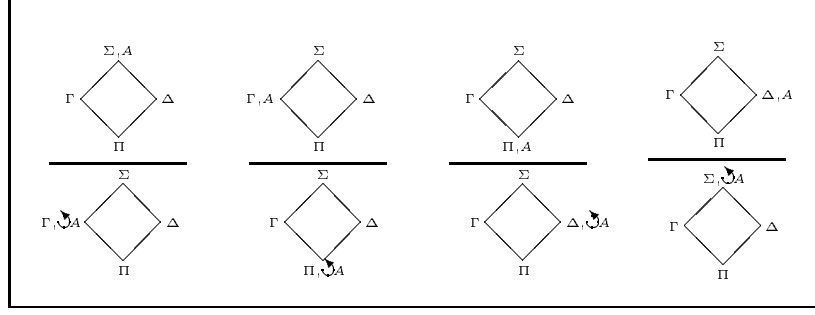
Axiom



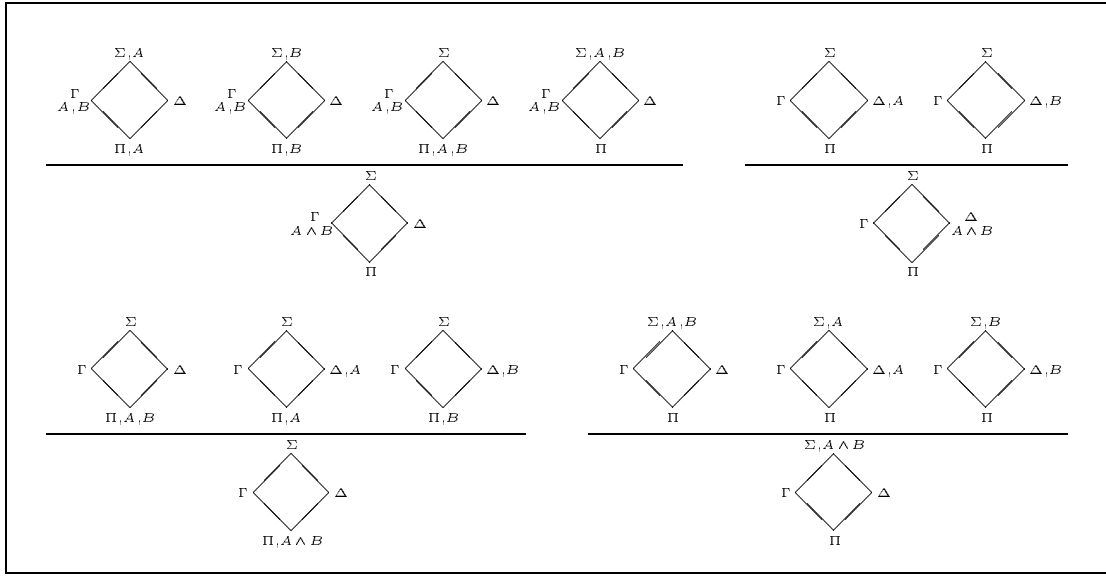
Negation



Quarter turn



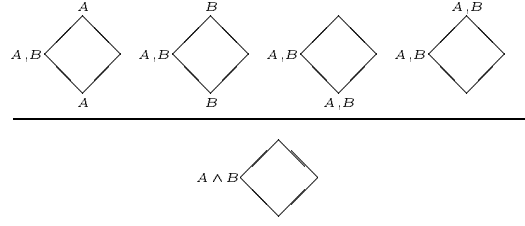
Conjunction



Theorem 1 For each logical rule of LB'_0 and each valuation v , v satisfies the sequent conclusion iff it satisfies all the sequents premises. As a consequence, for each logical rule of LB'_0 , the sequent conclusion is valid iff the sequents premises are.

Proof: this proof shows in passing how deduction system LB'_0 is obtained from truth tables.

- rules for negation \neg and quarter turn \curvearrowright : clear.
- rules for conjunction \wedge : let us consider the case of the introduction to the left, with 4 premises, and let v be a valuation. We first forget the context part of the rule, i.e we prove the result for the simpler rule:



It is easy to convince oneself that it suffices.

Then we look at the truth table for \wedge and ask when $A \wedge B$ is f for v . The answer is an equivalent sentence in “disjunctive normal form”: $(A \text{ is f}) \text{ or } (B \text{ is f}) \text{ or } (A \text{ is } \perp \text{ and } B \text{ is } \top) \text{ or } (A \text{ is } \top \text{ and } B \text{ is } \perp)$, for v . Finally, we change this sentence into one in “conjunctive normal form”. We obtain a conjunction of 4 disjunctions . . . which are exactly the semantical meanings of the 4 premises of the rule:

$(A \text{ is f or } B \text{ is f or } A \text{ is } \perp \text{ or } A \text{ is } \top)$ and $(A \text{ is f or } B \text{ is f or } B \text{ is } \perp \text{ or } B \text{ is } \top)$ and $(A \text{ is f or } B \text{ is f or } A \text{ is } \perp \text{ or } B \text{ is } \perp)$ and $(A \text{ is f or } B \text{ is f or } A \text{ is } \top \text{ or } B \text{ is } \top)$, for v .

Hence the sequent conclusion is satisfied by v iff the premises are.

The method is the same for the three other rules, except that some unnecessary parts in the final sentence can be discarded, thus eliminating unuseful premises. \square

A (*logical*) *axiom* is any sequent of the form $\Gamma, A \quad \Delta, A$.

The set of *derivation trees* (of LB'_0) is the set of finite trees, whose nodes are labeled by sequents, containing all one-node trees labeled with an axiom, and closed under the rules of LB'_0 , as usual. A *proof tree* is a derivation tree whose leaves are all labeled with axioms.

The label of the root of a derivation tree is also called its *conclusion*. A sequent is *provable* if there exists a proof tree of which it is the conclusion.

Corollary 1 (Soundness of LB'_0) *If a sequent is provable, then it is valid.*

Proof: the *height* of a derivation tree is the maximal length of its paths. The proof is easy by induction on the height of a proof tree for a given sequent. \square

Note that the sequent calculus with only *atomic axioms* (i.e sequents containing occurrences of the same atomic formula in each part) is equivalent to LB'_0 (a sequent is provable in one system iff it is provable in the other one).

2.4 COMPLETENESS

A sequent is said *finished* if it is an axiom or if it contains only atomic propositions. A derivation tree is said finished if all its leaf sequents are finished. Finished trees that are not proof trees are called *counter-example trees*.

Lemma 1 *A finished sequent is valid iff it is an axiom.*

Proof: let \mathcal{S} be a finished sequent. If it is an axiom, then it is clearly valid. If it is not an axiom, then no proposition occurring in \mathcal{S} occurs in each part of \mathcal{S} : each of them has some “forbidden” truth values (if e.g A does not occur in the left and bottom parts of \mathcal{S} , its forbidden truth values are f and \perp); since \mathcal{S} contains only propositional symbols (because it is finished), we can define a valuation v that associates to any propositional symbol one of its forbidden truth values; this valuation v does not satisfy \mathcal{S} . \square

Let ϕ be a mapping from a set X to X , and $x \in X$. We say that *algorithm ϕ terminates with input x* if there is an integer n such that $\phi^n(x) = \phi^{n+1}(x)$, and in this case, we also say that ϕ terminates with x on $\phi^n(x)$.

Theorem 2 *There is an algorithm Search0 , mapping derivation trees to derivation trees with same conclusion, and such that, for each sequent \mathcal{S} :*

- *if \mathcal{S} is valid, then Search0 terminates with (the one-node tree labeled by) \mathcal{S} on a proof tree;*
- *if \mathcal{S} is invalid, then Search0 terminates with (the one-node tree labeled by) \mathcal{S} on a counter-example tree T and all invalidating valuations can be deduced from T .*

Proof: the set of leaves of a derivation tree can be well-ordered (so that we can speak of the *first* of them), as well as the set of propositions occurring in a sequent.

Search0 is defined by:

- if U is a finished derivation tree, then $\text{Search0}(U) = U$;
- given an non-finished tree U , let \mathcal{S}' be one of its non-finished leaf sequents (say the first one), A an occurrence of a non-atomic proposition of \mathcal{S}' (say the first one), and R the tree (labeled by \mathcal{S}') which is the instance of the logical rule of LB'_0 that introduces A in \mathcal{S}' . (For example, if $A = B \wedge C$ is on the top part of \mathcal{S}' , then R is the instance of the introduction of \wedge to the top, with conclusion \mathcal{S}' .) Then $\text{Search0}(U)$ is obtained from U by replacing \mathcal{S}' by R . (In the following we shall say that $\text{Search0}(U)$ is obtained by expansion of U on the occurrence A)

Let T_0 be the one-node tree labeled by \mathcal{S} , and $T_{n+1} = \text{Search0}(T_n)$.

Define the *complexity* $c(A)$ of a proposition A as the number of connectives occurring in it. Finite sets of integers are ordered in the following way: if X and Y are finite sets of integers, we say that $X <_0 Y$ iff X is obtained from Y by replacing some $n \in Y$ by an arbitrary finite set of integers strictly smaller than n . The transitive closure of $<_0$, denoted $<$, is a strict order, and is well-founded. If U is a derivation tree, define $lf(U)$ as the (finite) set consisting of the complexities of the propositions occurring in its leaf sequents. Derivation trees are ordered in the following way: if U and V are derivation trees, we also note $U < V$ to mean $lf(U) < lf(V)$. The relation $<$ between derivation trees is a well-founded strict order as well, so we may induct on it.

For every derivation tree U , $\text{Search0}(U) < U$. Hence Search0 terminates on a finished derivation tree T . (This is easily proved by a top-down $<$ -induction on T_0 .)

Because of Theorem 1, a valuation v satisfies \mathcal{S} iff it satisfies all leaf sequents of T .

Hence if \mathcal{S} is valid, then all leaf sequents of T are valid, i.e they are axioms by Lemma 1: T is a proof tree.

If not, some valuation v does not satisfy \mathcal{S} , and some leaf sequent \mathcal{S}' of T can then be invalidated by v ; thus \mathcal{S}' is not an axiom (by Lemma 1) and T is a counter-example tree. Furthermore in this case, any valuation that invalidates \mathcal{S} also invalidates some leaf sequent of T , so that all the valuations that invalidate \mathcal{S} can be deduced from non-axiom leaves of T . \square

Corollary 2 (Completeness of LB'_0) *If a sequent is valid, then it is provable.*

3 FIRST-ORDER BELNAP'S LOGIC

3.1 ELEMENTARY MODEL THEORY

A *first-order language* \mathcal{L} consists of an infinite set \mathcal{V} of *variables* x, y, z, \dots , a set Σ_f of *function symbols* and a set Σ_r of *relation symbols*. As usual, one builds a *term* from variables and function symbols, an *atomic formula* from terms and relation symbols, and in general, *formulae* are built from atomic formulae, connectives and *quantifiers* \forall (universal quantifier), \exists (existential quantifier), \prod and \sum .

A term is said *closed* if it contains no variable. In the formulae $\forall xA$, $\exists xA$, $\prod xA$ and $\sum xA$, all the occurrences of the variable x are said *bound*. An occurrence of a variable that is not bound is said *free*. A formula is said *closed* if it has no free occurrences of variables. As a consequence, a closed atomic formula has no variable.

Sequents are quadruplets of finite sets of formulae, and represented by diamonds like in the propositional case.

A \mathcal{L} -*structure* \mathcal{M} consists of a non-empty set M (the *domain* of interpretation), and an *interpretation function* I such that:

- (i) for every n -ary function symbol f , $I(f) : M^n \rightarrow M$ is a n -ary function,
- (ii) for every n -ary relation symbol R , $I(R)$ is a mapping from M^n to the set $\mathcal{F}our$ of truth values (instead of a classical n -ary relation, ie a subset of M^n).

In order to give interpretations to formulae in a structure, one needs interpret a larger class of words: one defines *pseudo-terms* as either terms or members of M , and *pseudo-formulae* are built up from pseudo-terms, in the same way as formulae are built up from terms.

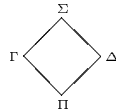
Furthermore, since a formula may contain free variables, its truth value will generally depend on the specific values in the domain assigned to variables: an *assignment* is a function from \mathcal{V} to M .

The interpretation of a pseudo-term for an assignment s is defined inductively: if $m \in M$, then $I(m)[s] = m$; if x is a variable, then $I(x)[s] = s(x)$; if $t_1 \dots t_n$ are pseudo-terms, f is a n -ary function symbol and $t = ft_1 \dots t_n$, then we define $I(t)[s] = I(f)(I(t_1)[s] \dots I(t_n)[s])$, which is also a member of M . In particular, the interpretation of a constant or a member of M does not depend on s .

An atomic pseudo-formula is then interpreted by a mapping from $\mathcal{V} \rightarrow M$ to truth values: for any assignment s , n -ary relation symbol R and pseudo-terms $t_1 \dots t_n$, $I(Rt_1 \dots t_n)[s] = I(R)(I(t_1)[s] \dots I(t_n)[s])$.

Finally I is extended to pseudo-formulae with the truth tables and rules for quantifiers (following [2], [6]): if $\bigwedge, \bigvee, \bigotimes, \bigoplus$ respectively stand for infinitary meets and joins under \leq_t and \leq_k , then we define $I(\forall xA(x))[s] = \bigwedge_{m \in M} I(A(m))[s]$, $I(\exists xA(x))[s] = \bigvee_{m \in M} I(A(m))[s]$, $I(\prod xA(x))[s] = \bigotimes_{m \in M} I(A(m))[s]$, $I(\sum xA(x))[s] = \bigoplus_{m \in M} I(A(m))[s]$. This has a meaning because we have interpreted pseudo-formulae and not only formulae.

We say that a \mathcal{L} -structure *satisfies* the sequent



for the assignment s , if for I and s , either

- one of the formulae in Γ has truth value f , or
- one of the formulae in Δ has truth value t , or
- one of the formulae in Π has truth value \perp , or
- one of the formulae in Σ has truth value \top .

A \mathcal{L} -structure *satisfies* a sequent, or is a *model* of it, if it satisfies it for every assignment s .

A \mathcal{L} -structure satisfies a formula A if it satisfies the sequent \Diamond^A .

A sequent (or a formula) is said *valid* if every \mathcal{L} -structure satisfies it. A sequent (or a formula) is said *invalid* if it is not valid.

Algebraic Properties.

All properties for propositional Belnap's logic still hold. \forall is commutative ($\forall x \forall y A \equiv \forall y \forall x A$); so are \exists , \prod and \sum . \forall and \exists are dual for \neg , \prod and \sum are dual for $-$, \forall and \sum (resp. \exists and \prod) are dual for $/$, and \forall and \prod (resp. \exists and \sum) are dual for \backslash .

Hence \exists , \prod and \sum can be expressed from \neg , \wedge , \bigcup and \forall . For the rest of this paper, we only consider the universal quantifier \forall .

“Equivalent” Connective: two formulae A and B are said *equivalent*, noted $A \approx B$, if for every \mathcal{L} -structure \mathcal{M} and every assignment s , $I(A)[s] = I(B)[s]$. One can prove that $I(A \equiv B)[s] = \text{t}$ iff $I(A)[s] = I(B)[s]$, and f otherwise.

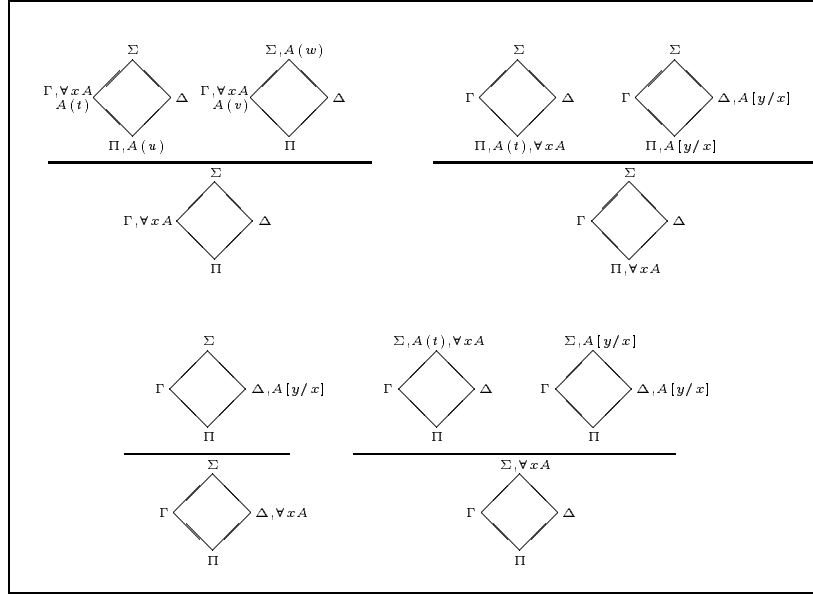
3.2 SEQUENT CALCULUS

System LB'_0 for propositional Belnap's logic is extended here to first-order Belnap's logic with rules for the universal quantifier \forall : this gives system LB'_1 .

LB'_1 consists of all rules of LB'_0 , plus 4 logical rules for \forall , where x denotes any variable, y denotes a variable not-free in Γ , Δ , Π , Σ , and t, u, v, w are any terms.

LB'_1 is a sequent calculus as well: there is an introduction for the universal quantifier \forall in each sequent part. Moreover, like Gallier's sequent calculus G [9], it has exactly one introduction rule for \forall in each sequent part.

Universal quantifier



Lemma 2 *If A is a formula, and y a variable not-free in A , then $\forall x A \approx \forall y(A[y/x])$.*

Proof: because y is not free in A , y has the same occurrences in $A[y/x]$ as x in A . Hence for each \mathcal{L} -structure \mathcal{M} and each assignment s , $I(\forall x A \equiv \forall y(A[y/x]))[s] = t$, which implies $\forall x A \approx \forall y(A[y/x])$. \square

This lemma says that bound variables are “mute”: they can be renamed outside the free variables of the formula.

Consider a \mathcal{L} -structure \mathcal{M} , an assignment s , a variable x and m a member of M . $[s, x := m]$ denotes the assignment s' such that: $s'(y) = s(y)$ if $y \neq x$, and $s'(x) = m$.

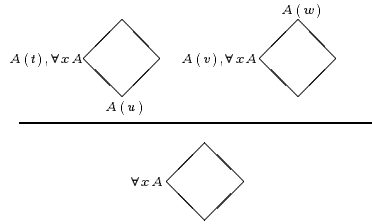
Lemma 3 *Let \mathcal{M} be a \mathcal{L} -structure, s an assignment, A a formula and x a variable not-free in A . For each $m \in M$, $I(A)[s, x := m] = I(A)[s]$.*

Proof: variable x is first (eventually) renamed in A (see Lemma 2), so that it does not occur any more in A . The result is then proved by induction on A . \square

Theorem 3 *For each logical rule of LB'_1 , each \mathcal{L} -structure \mathcal{M} and each assignment s , \mathcal{M} satisfies the sequent conclusion for s iff it satisfies all the sequents premises for s . As a consequence, for each logical rule of LB'_1 , the sequent conclusion is valid iff the sequents premises are.*

Proof: we only consider the rules for universal quantifier \forall .

• *introduction to the left*: as above, we prove the result for the simpler rule:



If a \mathcal{L} -structure \mathcal{M} satisfies the conclusion for an assignment s , then it satisfies both premises for s . Conversely, if \mathcal{M} satisfies both premises for s , then $\forall x A$ has truth value f for \mathcal{M} and s : suppose on the contrary that $I(\forall x A)[s] \neq f$; then we distinct between two cases: if $I(A(t))[s] = f$ or $I(A(v))[s] = f$, then $I(\forall x A)[s] = f$ (absurd); if not, then $I(A(u))[s] = \perp$ and $I(A(w))[s] = \top$, in this case $I(A(u) \wedge A(w)) = f$, and $I(\forall x A)[s] = f$ as well (absurd).

• *introduction to the right*: we need consider the full rule here, because satisfaction of a sequent depends globally on its free variables.

Let \mathcal{M} be a \mathcal{L} -structure that satisfies the premise, and s an assignment. We shall say that a formula F occurring in a given sequent \mathcal{S} is *given the convenient value by \mathcal{M} and s in \mathcal{S}* , when $I(F)[s] = f$ if F occurs on the left of \mathcal{S} , $I(F)[s] = t$ if F occurs on the right, $I(F)[s] = \perp$ if F occurs on the bottom and $I(F)[s] = \top$ if F occurs on the top. Now, either one of the formulae of $\Gamma, \Delta, \Pi, \Sigma$ is given the convenient value by \mathcal{M} and s in the sequent premise (and then \mathcal{M} satisfies the conclusion for s). Or, if not, none of these formulae is given the convenient value by \mathcal{M} and $[s, y := m]$ either, for any $m \in M$ (this is because of Lemma 3, y being not free in $\Gamma, \Delta, \Pi, \Sigma$); therefore in this case, for each $m \in M$, $I(A[y/x])[s, y := m] = t = I(A(m))[s]$, and thus $I(\forall x A)[s] = t$, so that \mathcal{M} still satisfies the conclusion for s .

Let \mathcal{M} be a \mathcal{L} -structure that satisfies the conclusion, and s an assignment. Either one of the formulae of $\Gamma, \Delta, \Pi, \Sigma$ is given the convenient value by \mathcal{M} and s in the sequent conclusion (and then \mathcal{M} satisfies the premise for s). Or $I(\forall x A)[s] = t$ and then, for each $m \in M$, $I(A(m))[s] = t$; in particular, $I(A(s(y)))[s] = t = I(A[y/x])[s]$, thus \mathcal{M} satisfies again the premise for s .

- *introduction to the bottom*: let \mathcal{M} be a \mathcal{L} -structure that satisfies the premises, and s an assignment. Either one of the formulae of $\Gamma, \Delta, \Pi, \Sigma$ is given the convenient value by \mathcal{M} and s in the sequent premises (and then \mathcal{M} satisfies the conclusion for s). Or, if not, none of these formulae is given the convenient value by \mathcal{M} and $[s, y := m]$ either, for any $m \in M$; if we suppose that $I(\forall x A)[s] \neq \perp$, the second premise implies that for each $m \in M$, $I(A[y/x])[s, y := m] = I(A(m))[s]$ is t or \perp ; besides, the first premise says that $I(A(t))[s] = \perp$; therefore $I(\forall x A)[s] = \perp$, which provides a contradiction. Hence \mathcal{M} satisfies the conclusion for s .

Let \mathcal{M} be a \mathcal{L} -structure that satisfies the conclusion, and s an assignment. Note that \mathcal{M} satisfies the first premise for s . Let us concentrate on the second one. As above, either one of the formulae of $\Gamma, \Delta, \Pi, \Sigma$ is given the convenient value by \mathcal{M} and s in the sequent conclusion (and then \mathcal{M} satisfies the premise for s). Or $I(\forall x A)[s] = \perp$, and then for every $m \in M$, $I(A(m))[s]$ is t or \perp ; in particular $I(A(s(y)))[s] = I(A[y/x])[s]$ is t or \perp : both premises are then satisfied by \mathcal{M} for s .

- *introduction to the top*: similar to preceding case. \square

Derivation trees and *proof trees* of LB'_1 are defined similarly to the propositional case. The label of the root of a derivation tree is also called its *conclusion*. A sequent is *provable* if there exists a proof tree of which it is the conclusion.

If X is any set of formulae, let us call $\text{Term}(X)$ the set of terms built from function symbols and free variables in formulae in X , and of height not greater than the maximum of the heights of terms occurring in formulae in X .

A derivation tree with conclusion \mathcal{S} is said *finished* if all its leaf sequents \mathcal{S}' have the following property (where S denotes the set of occurrences of formulae in the path between \mathcal{S} and \mathcal{S}'):

- either \mathcal{S}' is an axiom,
- or \mathcal{S}' contains only atomic formulae, or formulae of the form $\forall x A$ on the left, bottom or top, and such that: if $\forall x A$ occurs on the left in \mathcal{S}' , then $A(t)$ occurs on the left, bottom and top in S , for each term $t \in \text{Term}(\mathcal{S}')$; if $\forall x A$ occurs on the bottom (resp. top) in \mathcal{S}' , then $A(t)$ occurs on the bottom (resp. top) in S , for each term $t \in \text{Term}(\mathcal{S}')$.

Finished trees that are not proof trees are called *counter-example trees*.

Corollary 3 (Soundness of LB'_1) *If a sequent is provable, then it is valid.*

Note that the sequent calculus with only atomic axioms is still equivalent to LB'_1 .

3.3 COMPLETENESS

Preliminary Remark: like in classical logic, the main argument for proving completeness of the propositional calculus is the a priori majoration of the size of an eventual proof tree. This argument does not hold any more for the first-order calculus, what complicates the proof.

Signed Formulae and Four-Valued Hintikka Sets.

Following [13], we define the concepts of signed formulae and four-valued Hintikka sets (a adaptation of Hintikka sets to Belnap's logic). A *signed formula* is any expression of the form A_t or A_f or A_\perp or A_\top , where A is any formula. Signed formulae denote occurrences of formulae in four-valued sequents. Note the difference with usual signed formulae [9, 13] that denote “non-occurrences” of formulae.

If X is any set of formulae, or signed formulae, define \mathcal{L}_X as the subset of \mathcal{L} consisting of all function and relation symbols occurring in formulae in X . A set \mathcal{D} of terms is said

downward-closed over \mathcal{L}_X if for every n -ary function symbol f in \mathcal{L}_X and every terms $t_1 \dots t_n$, $ft_1 \dots t_n \in \mathcal{D} \implies t_1 \dots t_n$ are in \mathcal{D} .

A *four-valued Hintikka set with respect to \mathcal{D}* is any set S of signed formulae such that \mathcal{D} is a non-empty downward-closed set of terms over \mathcal{L}_S , and satisfying the following conditions:

(H0) For every atomic formula A , at least one of the signed formulae A_t , A_f , A_\perp , A_\top is not in S .

(H1) $A = B \wedge C$: if A_t is in S , then both B_t and C_t are in S ; if A_f is in S , then either B_f , C_f , B_\perp , B_\top are in S , or B_f , C_f , C_\perp , C_\top are in S , or B_f , C_f , B_\perp , C_\perp are in S , or B_f , C_f , B_\top , C_\top are in S ; if A_\perp is in S , then either B_\perp , C_\perp are in S , or B_\perp , C_t are in S , or C_\perp , B_t are in S ; finally if A_\top is in S , then either B_\top , C_\top are in S , or B_\top , C_t are in S , or C_\top , B_t are in S .

(H2) $A = \neg B$: if A_t is in S , then B_f is in S ; if A_f is in S , then B_t is in S ; if A_\perp is in S , then B_\perp is in S ; if A_\top is in S , then B_\top is in S .

(H3) $A = \bigvee B$: if A_t is in S , then B_\perp is in S ; if A_\perp is in S , then B_f is in S ; if A_f is in S , then B_\top is in S ; if A_\top is in S , then B_t is in S .

(H4) $A = \forall x B$: if A_t is in S , then $B(u)_t$ is in S for at least one term $u \in \mathcal{D}$; if A_f is in S , then either for every term $t \in \mathcal{D}$, $B(t)_f$, $B(t)_\perp$ are in S , or for every term $t \in \mathcal{D}$, $B(t)_f$, $B(t)_\top$ are in S ; if A_\perp is in S , then either $B(t)_\perp$ is in S for every term $t \in \mathcal{D}$, or $B(u)_\perp$, $B(u)_t$ are in S for at least one term $u \in \mathcal{D}$; if A_\top is in S , then either $B(t)_\top$ is in S for every term $t \in \mathcal{D}$, or $B(u)_\top$, $B(u)_t$ are in S for at least one term $u \in \mathcal{D}$.

Intuitively, conditions (H1) to (H3) specify that: if $A_\xi \in S$ ($\xi \in \{f, t, \perp, \top\}$), if \mathcal{S} is the sequent with no formula but A on the ξ -part, and if R is the tree labeled by \mathcal{S} which is the instance of the logical rule of \mathbf{LB}'_1 that introduces A in \mathcal{S} , then for each leaf \mathcal{S}' of R , S contains at least one signed formula that denotes the occurrence of a formula in \mathcal{S}' .

Given a \mathcal{L} -structure \mathcal{M} , an assignment s and a formula A , we shall say that \mathcal{M} *satisfies* A_t for s iff $I(A)[s] = t$, A_f iff $I(A)[s] = f$, A_\perp iff $I(A)[s] = \perp$ and A_\top iff $I(A)[s] = \top$.

Condition (H0) allows to prove the following crucial property of our four-valued Hintikka sets:

Lemma 4 *For every four-valued Hintikka set S with respect to \mathcal{D} , there is a \mathcal{L} -structure with domain \mathcal{D} that satisfies no formula of S .*

Proof: let S be a four-valued Hintikka set with respect to \mathcal{D} . Let s be a function from \mathcal{V} to \mathcal{D} that is identical on the variables in \mathcal{D} . Let \mathcal{H}_S be any \mathcal{L} -structure with (non-empty) domain \mathcal{D} and interpretation function $I_{\mathcal{H}_S}$ such that:

- for every n -ary function symbol f in \mathcal{L}_S , and terms $t_1 \dots t_n$ in \mathcal{D} such that $ft_1 \dots t_n \in \mathcal{D}$, $I_{\mathcal{H}_S}(f)(t_1 \dots t_n)[s] = ft_1 \dots t_n$;
- for every n -ary relation symbol R in \mathcal{L}_S , and terms $t_1 \dots t_n$ in \mathcal{D} ,

$$\begin{cases} I_{\mathcal{H}_S}(R)(t_1 \dots t_n)[s] = f & \text{if } (Rt_1 \dots t_n)_f \notin S, \\ I_{\mathcal{H}_S}(R)(t_1 \dots t_n)[s] = \perp & \text{if } (Rt_1 \dots t_n)_f \in S \text{ and } (Rt_1 \dots t_n)_\perp \notin S, \\ I_{\mathcal{H}_S}(R)(t_1 \dots t_n)[s] = t & \text{if } (Rt_1 \dots t_n)_f \in S, (Rt_1 \dots t_n)_\perp \in S \text{ and } (Rt_1 \dots t_n)_t \notin S, \\ I_{\mathcal{H}_S}(R)(t_1 \dots t_n)[s] = \top & \text{if } (Rt_1 \dots t_n)_f \in S, (Rt_1 \dots t_n)_\perp \in S, (Rt_1 \dots t_n)_t \in S \\ & \text{and } (Rt_1 \dots t_n)_\top \notin S. \end{cases}$$

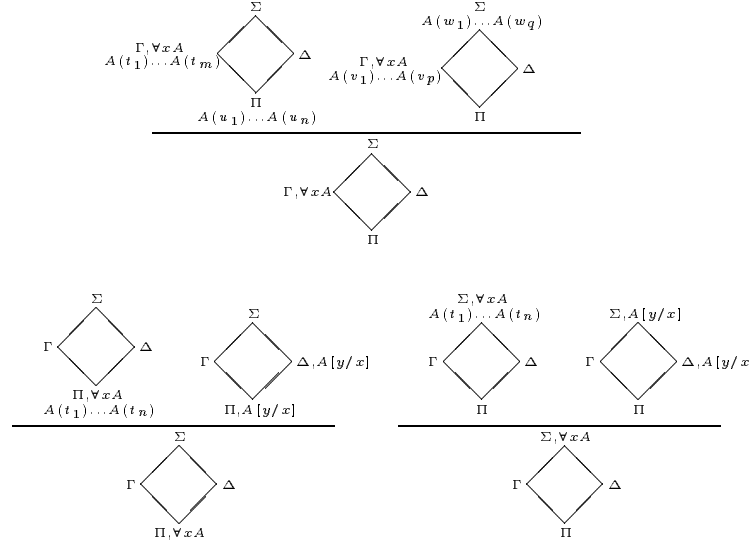
By condition (H0) this definition is proper. (Note that in general, the interpretation of a function symbol f on terms $t_1 \dots t_n$ has no importance if $ft_1 \dots t_n \notin \mathcal{D}$, and indeed this case is not taken into account, in general, by the conditions above. It is taken into account if S contains terms of arbitrary large height, because \mathcal{D} is then a term algebra.)

Using the fact that \mathcal{D} is downward-closed, one easily shows that for each $t \in \mathcal{D}$, $I_{\mathcal{H}_S}(t)[s] = t$, and thus for each relation symbol R in \mathcal{L}_S and terms $t_1 \dots t_n$ in \mathcal{D} , $I_{\mathcal{H}_S}(Rt_1 \dots t_n)[s] = I_{\mathcal{H}_S}(R)(t_1 \dots t_n)[s]$.

Let us prove that \mathcal{H}_S does not satisfy any signed formula of S for s . We proceed by induction on the signed formulae of S .

- It is true for atomic formulae: if $(Rt_1 \dots t_n)_f$ is an atomic formula occurring in S , then $t_1 \dots t_n$ are in \mathcal{D} and $R \in \mathcal{L}_S$, so $I_{\mathcal{H}_S}(Rt_1 \dots t_n)[s] = I_{\mathcal{H}_S}(R)(t_1 \dots t_n)[s] \neq f$; idem for $(Rt_1 \dots t_n)_t$, $(Rt_1 \dots t_n)_\perp$ and $(Rt_1 \dots t_n)_\top$.
- For non-atomic formulae, we use the definition of the interpretation of a universally quantified formula, and conditions (H1) to (H4). Consider for example the case where $A = \forall x B$ and $A_\perp \in S$, and suppose that \mathcal{H}_S satisfies A_\perp for s , i.e. $I_{\mathcal{H}_S}(A)[s] = \perp$; because of (H4), either $B(t)_\perp \in S$ for all $t \in \mathcal{D}$, and the existence of a term u in domain \mathcal{D} such that $I_{\mathcal{H}_S}(B(u))[s] = \perp$ contradicts the induction hypothesis, or there is a term $u \in \mathcal{D}$ such that $B(u)_\perp$ and $B(u)_t$ are in S , and then $I_{\mathcal{H}_S}(B(u))[s]$ is \perp or t , what contradicts the induction hypothesis too. The other cases are similar. \square

We slightly modify system LB'_1 , replacing the introductions of \forall to the left, bottom and top respectively by the following (where the t_i , u_i , v_i , w_i are any terms):



These rules are called the *modified* rules, and the system obtained is equivalent to the old one.

König's Lemma *If a tree T is infinite and finite-branching (i.e. each node has a finite number of successors), then there is some infinite path in T .*

Theorem 4 *There is a algorithm Search1, mapping derivation trees to derivations tree with same conclusion, and such that, for each sequent \mathcal{S} :*

- if \mathcal{S} is valid, then Search1 terminates with (the one-node tree labeled by) \mathcal{S} on a proof tree;
- if \mathcal{S} is invalid, then either Search1 terminates with (the one-node tree labeled by) \mathcal{S} on a counter-example tree that provides a finite invalidating \mathcal{L} -structure, or Search1 does not terminate with (the one-node tree labeled by) \mathcal{S} and \mathcal{S} can be invalidated in an infinite \mathcal{L} -structure.

Proof: informally, if U is a derivation tree, $\text{Search1}(U)$ is obtained by successively expanding each non-axiom leaf sequent of U on each non-atomic formula occurring in it (in the sense

given in the proof of Theorem 2). The terms involved in the expansions of \forall -formulae (with the modified rules) are just those that have a chance to let new axioms appear in the tree; in particular, if all such terms have already been used for expanding a given \forall -formula A occurring in a leaf \mathcal{S}' , and if \mathcal{S}' is not an axiom, then \mathcal{S}' shall not be expanded any more on A . Before defining *Search1*, we introduce two functions *Expand* and *Develop*:

If T and U are derivation trees and A is an occurrence of a non-atomic formula, define *Expand*(T, A, U) by:

- if $A = B \wedge C$, $A = \neg B$, or $A = \bigcup B$, then *Expand*(T, A, U) is the tree obtained by expansion of each leaf node of T on the occurrence A ;
- if $A = \forall xB$ and A is an occurrence on the right, then *Expand*(T, A, U) is obtained by expansion of each leaf node of T on A , with variable y not occurring anywhere in U (see the logical rules for \forall);
- if $A = \forall xB$ and A is an occurrence on the left, then *Expand*(T, A, U) is obtained by expansion of each leaf node of T on A (according to the modified introduction of \forall to the left), with $\{t_1 \dots t_m\} = \{u_1 \dots u_n\} = \{v_1 \dots v_p\} = \{w_1 \dots w_q\} = \text{Term}(U)$, if this expansion really “does something”, i.e if the leaf nodes produced are not all identical to the one consumed; else *Expand*(T, A, U) = T ;
- if $A = \forall xB$ and A is an occurrence on the bottom or on the top, then *Expand*(T, A, U) is obtained by expansion of each leaf node of T on A (according to the modified introductions of \forall to the bottom or to the top), with $\{t_1 \dots t_m\} = \text{Term}(U)$, if this expansion really “does something”; else *Expand*(T, A, U) = T .

If L is a one-node tree, U a derivation tree and $\{A_1 \dots A_n\}$ the set of non-atomic formulae occurring in L , define *Develop*(L, U) = *Expand*($\dots[\text{Expand}(L, A_1, U), A_2, U] \dots A_n, U$).

Finally let *Search1*(U) be the result of replacing in U each non-axiom leaf node L by *Develop*(L, U).

Let T_0 be the one-node tree labeled by \mathcal{S} , and $T_{n+1} = \text{Search1}(T_n)$.

Observe that if *Search1* terminates, then it terminates on a finished tree.

Now the T_n are all labeled by \mathcal{S} , so:

- (1) If *Search1* terminates on a proof tree, then \mathcal{S} is valid.
- (2) If *Search1* terminates on a tree T which is not a proof tree, then T is a counter-example tree (since it is finished), and there is a path from the root to a non-axiom leaf sequent. If *Search1* does not terminate, then the limit tree $\bigcup_{n \geq 0} T_n$ is finite-branching but infinite, so by König's Lemma it contains an infinite path.

In either case, let \mathcal{D} be the (non-empty) set of all terms occurring along that path. Let S_t be the set of all formulae occurring on the right in a sequent along the path, S_r the set of all formulae occurring on the left, S_\perp the set of all formulae occurring on the bottom, and S_\top the set of all formulae occurring on the top. Let $S = \{A_t \mid A \in S_t\} \cup \{A_r \mid A \in S_r\} \cup \{A_\perp \mid A \in S_\perp\} \cup \{A_\top \mid A \in S_\top\}$.

If \mathcal{D} is not downward-closed, there is no signed formula of the form $(\forall xA)_r$, $(\forall xA)_\perp$, or $(\forall xA)_\top$ in S , so the path has to be finished, and one can easily prove that the non-axiom leaf sequent contains only atomic formulae. It is then trivial to build a finite \mathcal{L} -structure that does not satisfy S .

If \mathcal{D} is downward-closed, then S is a four-valued Hintikka set with respect to \mathcal{D} : (H0) holds because none of the T_n is a proof tree; (H1) to (H3) hold trivially (they have been specially defined on purpose to be true); for (H4), the only interesting case is when $A_r = (\forall xB)_r \in S$ and A is expanded infinitely many times in the path: at each expansion step of A (on the left), the path goes along the right branch or the left one, but the same direction (say the

right) has to be taken infinitely many times, hence in this case, for every term $t \in \mathcal{D}$, $B(t)_f$ and $B(t)_\top$ are in S .

Hence by Lemma 4 there is a \mathcal{L} -structure \mathcal{H}_S with domain \mathcal{D} that satisfies no formula of S . Thus no sequent along the path is satisfied by \mathcal{H}_S , in particular the root sequent \mathcal{S} is invalid.

Therefore we have shown the following:

- if *Search1* terminates on a proof tree, then \mathcal{S} is valid;
- if *Search1* terminates on a counter-example tree, then \mathcal{S} can be invalidated in a \mathcal{L} -structure with domain \mathcal{D} , which is finite;
- if *Search1* does not terminate, then \mathcal{S} can be invalidated in a \mathcal{L} -structure with domain \mathcal{D} , which is infinite. \square

Corollary 4 (Completeness of LB'_1) *If a sequent is valid, then it is provable.*

References

- [1] ANDERSON A.R., N.D. BELNAP N.D. and DUNN J.M., Entailment: the Logic of Relevance and Necessity. *Princeton University Press, vol.2 p.506-541* (1992).
- [2] BELNAP Jr N.D., A Useful Four-Valued Logic. *In: J.M. Dunn and G. Epstein (eds.), Reidel: Modern Uses of Multiple-Valued Logic* (1977).
- [3] FAGES F., A New Fixpoint Semantics for General Logic Programs Compared with the Well-Founded and the Stable Model Semantics. *Proc. of the 7th Int. Conf. on Logic Programming* (1990).
- [4] FITTING M., A Kripke-Kleene Semantics for Logic Programs. *J. of Logic Programming* 2 (1985).
- [5] FITTING M., Bilattices and the Theory of Truth. *J. of Philosophical Logic* 18 (1989).
- [6] FITTING M., Bilattices and the Semantics of Logic Programming. *J. of Logic Programming* 11 (1991).
- [7] FITTING M., Kleene's Three-Valued Logics and their Children. *Fundamenta Informaticae* (1993) *forthcoming*.
- [8] FITTING M., The Family of Stable Models. *J. of Logic Programming* 17 (1993).
- [9] GALLIER J.H., Logic for Computer Science, Foundations of Automated Theorem Proving. *John Wiley & Sons* (1987).
- [10] GIRARD J.-Y., Three-Valued Logic and Cut-Elimination: the Actual Meaning of Takeuti's Conjecture. *Dissertationes Mathematicae* 136, *Warsaw* (1976).
- [11] KUNEN K., Negation in Logic Programming. *J. of Logic Programming* 4 (1987).
- [12] KUNEN K., Signed Data Dependences in Logic Programs. *J. of Logic Programming* 7 (1989).
- [13] SMULLYAN R.M., First-Order Logic. *Springer Verlag* (1968).
- [14] VAUZEILLES J. and STRAUSS A., Intuitionistic Three-Valued Logic and Logic Programming. (1988) *manuscript*.