VDM: Axiomatising its Propositional Logic

P. F. GIBBINS
Faculty of Mathematics, The Open University, Milton Keynes MK7 6AA, England

The 3-valued truth-tables of LPF, which is both the logic of partial functions and also the logic of VDM, do not determine the logic unless the appropriate notion of logical consequence is specified.

Three alternative notions of logical consequence are considered and LPF is axiomatised as a sequent calculus for each such notion. The three axiomatisations employ the same introduction rules for the connectives and differ only in their axiom sequents.

The axiomatisations lead naturally to automatic theorem-provers for LPF.

Received April 1987, revised May 1987

1. INTRODUCTION

Users of VDM may well be dismayed to find that the current version of that formal specification system[1] employs a three-valued logic LPF – the **logic of partial functions** – as the vehicle for its proofs that a direct definition of a function or operation satisfies its implicit specification.

Proofs of correctness are central to the development method that VDM sponsors. But LPF, the logic of VDM, is non-standard. LPF is a restriction of classical logic: *every valid argument of LPF is a valid classical argument, but not conversely.*

The software engineer who uses LPF in his proof obligations therefore has a weaker logical apparatus at his disposal than his less fastidious colleague who avails himself of the full might of classical logic. Worse, LPF is more unwieldy than classical logic. Proving results in LPF is inherently more difficult.

How should be axiomatise LPF? And what can be done to unburden the software engineer? The second question suggests a large research programme which should result in a collection of software tools making VDM more easily usable. Among these tools should be a theorem-prover for LPF. In this paper we first consider the preliminary problem of **axiomatising** LPF.

In fact we consider several different **versions** of propositional LPF. Each version of LPF that we consider adopts the usual LPF three-valued truth-tables but each embodies its own particulȧr notion of **logical consequence.** We axiomatise each of these versions of LPF as a sequent calculus. Strikingly, the different versions of LPF have a **fixed** set of introduction rules for the connectives and differ only in their axiom sequents. Re-working the various versions of LPF as sequent calculi has the following benefits.

An anonymous referee has pointed out that two of these three logics, LPF-1 and LPF-3, have been treated in the publications by Cheng[2] and Kolestos.[3] A logic similar to LPF-3, the third version of LPF, has been studied by Blamey.[4]

First, proofs of the soundness and completeness of each version of LPF become almost trivial.

Secondly, it is very simple to obtain algorithms for LPF versions of Wang's algorithm.

Thirdly, sequent calculus axiomatisations of LPF are readily transformed into resolution theorem provers. In resolution theorem proving, one can think of the resolution rule of inference as handling the purely propositional aspects of classical first-order logic and the unification algorithm as handling the quantificational aspects. One can take a similar view of quantificational LPF.

Most of what is characteristically non-standard about LPF shows up at the purely propositional level. Therefore, in this paper we concentrate on propositional LPF and leave the connection with resolution theorem proving in propositional LPF and its extension to quantificational LPF to later papers.

2. THE TRUTH-TABLES OF LPF

In LPF the interpretations of the three logical connectives 'and' ($\wedge$), 'or' ($\vee$) and 'not' ($\neg$) are given by the following truth-tables in which **T** stands for *true*, **F** stands for *false* and $\perp$ stands for *undefined.* They are, incidentally, the **strong** truth-tables that Kleene gives for his three-valued **logic of partial functions.**[5] In Barringer, Cheng and Jones[6] the tables also appear, as does a proposed natural deduction axiomatisation of LPF which, in that paper, lacks soundness and completeness proofs.

For $\wedge$ we have

$\wedge$	T	$\perp$	F
T	T	$\perp$	F
$\perp$	$\perp$	$\perp$	F
F	F	F	F

for $\vee$

$\vee$	T	$\perp$	F
T	T	T	T
$\perp$	T	$\perp$	$\perp$
F	T	$\perp$	F

and for $\neg$

$\neg$	
T	F
$\perp$	$\perp$
F	T

We use Bool$_+$ and Bool to denote the following sets.

$$\text{Bool}_+ = \{\mathbf{T}, \perp, \mathbf{F}\}$$

and

$$\text{Bool} = \{\mathbf{T}, \mathbf{F}\}$$

As Jones remarks (see ref. 1, p. 77) these truth-tables are the **strongest monotonic extensions of the classical two-valued truth-tables** for the corresponding connectives, given a partial ordering on Bool_+ in which only $\perp$ is properly less than **T** and **F**. Just why this makes for an appropriate choice for the thirty-tables is a matter of philosophy as much as software engineering and we leave the question to a later paper.

It is important to recognise that fixing the logical connectives via these true-tables does not fix LPF. A logic describes a class of **valid arguments**, and we must define what we mean by **logical consequence** in LPF, a binary relation which holds between propositions, or between sets or lists of propositions. Note that a logic is not concerned merely with logical truth, or, in a propositional logic, 'tautologyhood'. In LPF there are no tautologies in the ordinary sense, but there are nevertheless valid arguments.

3. NOTIONS OF LOGICAL CONSEQUENCE

In this paper, we consider logical consequence to be a relation between sequences of well-formed formulae (wffs). We naturally demand that the sequences of wffs be finite. A **sequent** is a pair, each element of which is a finite sequence of wffs, so we have objects of the following types

$$\text{Wfflist} = \textbf{Seq of } \text{Wff}$$

$$\text{Sequent} = \text{Wfflist} \times \text{Wfflist}$$

A sequent is a pair $\langle \Gamma, \Delta \rangle$, but we employ the more usual, and more graphic notation

$$\Gamma \to \Delta$$

We write the concatenation of two wfflists Γ and Δ as

$$\Gamma, \Delta$$

and, given that 'A' is a wff, we write the concatenation of Γ and $[A]$ and Δ as

$$\Gamma, A, \Delta$$

We also abbreviate $[\,] \to \Delta$ and $\Gamma \to [\,]$ as

$$\to \Delta$$

and

$$\Gamma \to$$

respectively.

A wfflist Δ is said to be a logical consequence of a wfflist Γ if and only if the sequent

$$\Gamma \to \Delta$$

is valid. So what do we mean by the validity of a sequent? (Notice that a sequent

$$\Gamma \to \Delta$$

is valid only with respect to a definition of validity.) We consider the following three definitions of the **validity of a sequent**. The sequent

$$\Gamma \to \Delta$$

[*Valid-1*] iff whenever all the wffs in Γ are true, then at least on of the wffs in Δ is true,

[*Valid-2*] iff whenever all the wffs in Δ are false, at least one of the wffs in Γ is false,

[*Valid-3*] iff whenever all the wffs in Γ are true at least one of the wffs in Δ is true, **and** whenever all the wffs in Δ are false, at least one of the wffs in Γ is false.

There are several notions here that need to be made more precise, nevertheless it is clear that [Valid-1], [Valid-2] and [Valid-3] yield different logics. (We say that they yield different **versions** of LPF, namely LPF-1, LPF-2 and LPF-3). To see this, let P and Q be atomic wffs. (Note that all atoms are logically independent.) Then, given the LPF truth-tables, the sequent

$$P, \neg P \to Q$$

is valid according to [Valid-1] since 'P', '$\neg P$' can never both take the truth-value true. But it is invalid according to [Valid-2], since 'Q' may be false while both 'P' and '$\neg P$' are undefined. Similarly, the sequent

$$P \to Q, \neg Q$$

is valid according to [Valid-2] since 'Q' and '$\neg Q$' are never both false. But it is invalid according to [Valid-1] since 'P' may be true while both 'Q' and '$\neg Q$' are undefined. Clearly, neither of these sequents is valid according to [Valid-3]. It is also clear that [Valid-1], [Valid-2] and [Valid-3] are equivalent in classical two-valued logic.

It is worth noting that LPF-2 has the strange feature that some sequents of the form

$$\to A$$

are valid, even though 'A' is not an LPF tautology. Actually there are no LPF-tautologies, if by 'tautology' one means a wff which always takes the truth-value **T** for all assignments of truth-values to its component atoms. Thus in LPF-2 we have

$$\to P \vee \neg P$$

so that one cannot say that a tautology of LPF-2 is a special case of a valid sequent whose premises are null (Cp. ref. 1, p. 8). There is naturally a dual feature in LPF-1 in which there are no inconsistencies in the classical sense – no wffs which always take the truth-value **F** for all assignments of truth-values to its component atoms – even though there are valid sequents of the form

$$A \to$$

In none of the LPF's do we have a wff which always take the truth-value **T**, or which always takes the truth-value **F**, or indeed which always takes the truth-value $\perp$.

In this respect the LPF's are **expressiveness-incomplete,** a point which I owe to an anonymous referee. Those truth-functions which always take a fixed truth-value cannot be expressed. The situation can easily be remedied by extending the syntax of the LPF's with propositional constants which always take one of the three truth-values for all assignments of truth-values to the atoms of the language of the LPF's.

4. RULES OF INFERENCE AND Wffs

A rule of inference is a set of pairs. The first element of each pair is a sequent, the **conclusion** of an instance of the rule, and the second element is a list of sequents, the **premises** of the instance of the rule. In our versions of LPF the premises of an instance of a rule will be either a one- or a two-list. We graphically represent a rule of inference by listing the form of the premises above the form of the conclusion with a line (representing 'inference') separating premises from conclusion.

If a sequent is valid according to a definition of validity, we say that it is **valid under** that definition of validity. A rule of inference is then **valid (under a definition of validity)** iff for all its instances, whenever the premises of an instance of the rule are valid, the conclusion of that instance of the rule is valid.

Some classically valid rules of inference fail to be valid in some versions of LPF.

Thus consider

$$\frac{\Gamma, A \rightarrow \Delta}{\Gamma \rightarrow \Delta, \neg A}(\neg\text{: right})$$

This is a version of the standard classical '$\neg$-introduction on the right' rule. It could only be invalid if for one of its instances, the premise '$\Gamma, A \rightarrow \Delta$' were a valid sequent and the conclusion '$\Gamma \rightarrow \Delta, \neg A$' were an invalid sequent.

Classically, the conclusion would be an invalid sequent iff it is possible for all the wffs in Γ to be true and all the wffs in $\Delta, \neg A$ to be false. But in that case 'A' would be true, as would all the wffs in Γ, A. Therefore '$\Gamma, A \rightarrow \Delta$' would be invalid.

However, given the definition [Valid-1], the rule ($\neg$: right) is invalid. For consider the instance

$$\frac{A \rightarrow A}{\rightarrow A, \neg A}$$

The premise is clearly valid under [Valid-1], but the conclusion is not since both 'A' and '$\neg A$' may be undefined.

($\neg$: right) is a valid rule of inference under [Valid-2]. But the classical rule ($\neg$: left)

$$\frac{\Gamma \rightarrow A, \Delta}{\Gamma, \neg A \rightarrow \Delta}(\neg\text{: left})$$

is not, as may be seen by an argument similar to that above. ($\neg$: left) is valid under [Valid-1], but neither ($\neg$: right) nor ($\neg$: left) is valid under [Valid-3]. Therefore, if the introduction rules are to be common to LPF-1, LPF-2 and LPF-3, they must look somewhat different from the standard classical rules.

In a sequent calculus, there is usually a rule of inference dealing with the introduction of a formula dominated by each connective on the 'left' and the 'right'. However, the introduction rules common to all the LPF's take a slightly different form. We have the usual classical rules for $\vee$ and $\wedge$. But the rules involving $\neg$ are different, as the examples above might suggest.

We now make clear what we mean by **wff**, and also **literal sequent, rule of inference, deduction tree**, and **proof** of a sequent.

Wffs

The abstract syntax of wffs is given by

$$\text{Wff} ::= \textbf{ATOM} \mid \neg\text{Wff} \mid \text{Wff Bin\_Op Wff}$$

$$\text{Bin\_Op} ::= \wedge \mid \vee$$

But it is worth giving a concrete syntax which reflects the precedence and associativities of the binary operators.

$$\langle\text{Wff}\rangle ::= \langle\text{Conj\_Wff}\rangle \mid \langle\text{Conj\_Wff}\rangle \wedge \langle\text{Wff}\rangle$$

$$\langle\text{Conj\_Wff}\rangle ::= \langle\text{Neg\_Wff}\rangle \mid \langle\text{Neg\_Wff}\rangle \vee \langle\text{Conj\_Wff}\rangle$$

$$\langle\text{Neg\_Wff}\rangle ::= \langle\text{Factor}\rangle \mid \neg\langle\text{Neg\_wff}\rangle$$

$$\langle\text{Factor}\rangle ::= \langle\text{Atom}\rangle \mid (\langle\text{Wff}\rangle)$$

$$\langle\text{Atom}\rangle ::= A \mid \ldots \mid Z$$

from which the precedence of $\wedge$ over $\vee$ follows, as does the right associativity of both. More importantly, it follows by structural induction that any wff which contains a connective, has a **dominant connective** and there is an unique occurrence of that connective which is the dominant occurrence. This fact is important in the soundness and completeness proofs that follow.

A **literal** is a wff which is either an atom or the negation of an atom. So we have

$$\text{Literal} = \text{Atom} \mid \neg\text{Atom}$$

A sequent whose wffs are all literals is a **literal sequent** so that

$$\text{Literal\_Sequent} = (\textbf{Seq of}\ \text{Literal}) \times (\textbf{Seq of}\ \text{Literal})$$

We pick out some of the **literal** sequents and label them **axiom sequents**.

5. AXIOM SEQUENTS

A literal sequent S satisfies

[Axiom-1] if and only if for some **literal** L, S is of the form

$$\Gamma, L, \Delta \rightarrow \Theta, L, \Psi$$

that is, if and only if there is a literal common to both sequences of wffs in the sequent;

[Axiom-2-left] if and only if for some **atom** A, S is of the form

$$\Gamma, A, \Delta, \neg A, \Theta \rightarrow \Psi$$

that is, if and only if an atom and its negation appear in the 'left' sequence of wffs in S;

[Axiom-2-right] if and only if for some **atom** A, S is of the form

$$\Gamma \rightarrow \Delta, A, \Theta, \neg A, \Psi$$

that is, if and only if an atom and its negation appear in the 'right' sequence of wffs in S.

LPF-1, -2 and -3 differ not in their introduction rules, but only in their axiom sequents, which are as follows:

LPF-1

A literal sequent is an axiom sequent of LPF-1 if and only if it satisfies **either** (1) Axiom-1 **or** (2) Axiom-2-**left**.

LPF-2

A literal sequent is an axiom sequent of LPF-2 if and only if it satisfies **either** (1) Axiom-1 **or** (2) Axiom-2-**right**.

LPF-3

A literal sequent is an axiom sequent of LPF-3 if and only if it satisfies **either** (1) Axiom-1 **or** (2) **both** Axiom-2-left **and** Axiom-2-right.

A rule has the type

Rule = (**Seq of** Sequent) × Sequent

so that a particular rule of inference is an object of type Rule. It will be a set of pairs, the first element being a sequence of sequents – the premises of the rule – and the second element being a single sequent – the conclusion of the rule. The best way of describing the rules of a sequent calculus is as a collection of such items which follow a pattern. Each of the patterns is given a name and is associated with a particular rule. So we have the introduction rules of LPF, so-called because the conclusion 'introduces' an occurrence of one (or more) of the connectives. In effect, we have a predicate true of the rules of LPF, namely what we call ($\wedge$: left), ..., ($\neg\neg$: right), of type

Is_Introduction_Rule: Rule → Bool

which we define in the conventional way as follows.

6. THE INTRODUCTION RULES

$$\frac{\Gamma, \Delta, A, B \to \Theta}{\Gamma, A \wedge B, \Delta \to \Theta}(\wedge : \text{left})$$

$$\frac{\Theta \to \Gamma, \Delta, A; \Theta \to \Gamma\ \Delta, B}{\Theta \to \Gamma, A \wedge B, \Delta}(\wedge : \text{right})$$

$$\frac{\Gamma, \Delta, A \to \Theta; \Gamma, \Delta, B \to \Theta}{\Gamma, A \vee B, \Delta \to \Theta}(\vee : \text{left})$$

$$\frac{\Theta \to \Gamma, \Delta, A, B}{\Theta \to \Gamma, A \vee B, \Delta}(\vee : \text{right})$$

$$\frac{\Gamma, \Delta, \neg A \to \Theta; \Gamma, \Delta, \neg B \to \Theta}{\Gamma, \neg(A \wedge B), \Delta \to \Theta}(\neg \wedge : \text{left})$$

$$\frac{\Theta \to \Gamma, \Delta, \neg A, \neg B}{\Theta \to \Gamma, \neg(A \wedge B), \Delta}(\neg \wedge : \text{right})$$

$$\frac{\Gamma, \Delta, \neg A, \neg B \to \Theta}{\Gamma, \neg(A \vee B), \Delta \to \Theta}(\neg \vee : \text{left})$$

$$\frac{\Theta \to \Gamma, \Delta, \neg A; \Theta \to \Gamma, \Delta, \neg B}{\Theta \to \Gamma, \neg(A \vee B), \Delta}(\neg \vee : \text{right})$$

$$\frac{\Gamma, \Delta, A \to \Theta}{\Gamma, \neg\neg A, \Delta \to \Theta}(\neg\neg : \text{left})$$

$$\frac{\Theta \to \Gamma, \Delta, A}{\Theta \to \Gamma, \neg\neg A, \Delta}(\neg\neg : \text{right})$$

7. PROOFS IN LPF

Our discussion of the meta-theory of LPF is informal. Notice that though LPF is a non-standard logic in which there are three truth-values, its meta-theory is formulable in a standard two-valued meta-language. For example, a sequent is either probable or not provable in LPF, its provability is never 'undefined'. Therefore we can reason informally in the meta-language and **sketch** the proofs of our main results. (It would be possible to obtain the following results using LPF in the metalanguage since each of our lemmas and theorems employs only inductive and constructive reasoning. The meta-theory of LPF is describable using LPF and so LPF is **auto-descriptive**.[7]

We begin with the important notion of a deduction tree. A **deduction tree** is a tree of sequents. A literal sequent is a deduction tree. So is a tree whose root is a sequent and whose subtrees are deduction trees, provided that the root of the tree together with the roots of its immediate subtrees are an instance of one of the introduction rules. In the style of VDM, we have

```
Deduction_Tree = Literal_Sequent|
  Compound_Deduction_Tree

Compound_Deduction_Tree :: seq:
  Seq of Deduction_Tree
    s: Sequent
```

where

```
inv-Compound_Deduction_Tree(mk-Compound_
Deduction_Tree (seq, s))
Δ Is_Introduction_Rule (Roots (seq), s)
```

and

```
Roots: Seq of Deduction_Tree → Seq of Sequent
Roots (seq) Δ
  cases seq of
  [] → []
  cons (mk-Literal_Sequents (s), tl) → cons (s, Roots
    (tl))
  cons (mk-Compound_Deduction_tree (seq, s), tl) →
    cons (s, Roots (tl))
  end
```

A proof is a special case of a deduction tree, in which all the leaves of the tree are not merely literal sequents, but **axiom sequents**.

```
Is_Proof_Tree: Deduction_Tree → Bool
Is_Proof_Tree (t) Δ
  cases t of
  mk-Literal_Sequent(s) → Is_Axiom_Sequent(s)
  mk-Compound_Deduction_Tree (seq, s) →
    Is_Seq_Proof_Trees (seq)
  end

Is_Seq_Proof_Trees: Seq of Deduction_Tree → Bool
Is_Seq_Proof_Trees (seq) Δ
  cases seq of
  [] → []
  mk-Seq_Deduction_Trees (cons (hd, tl))
    → Is_Proof_Tree (hd) ∧ Is_Seq_Proof_Trees (tl)
  end
```

This leads to

Lemma 1

To every sequent there corresponds a deduction tree.

Proof. By structural induction on deduction trees.

If a deduction tree is a literal sequent then it is its own deduction tree.

If a sequent is not a literal sequent then the wffs in its premises and conclusion may be dismantled using the introduction rules applied in reverse, that is, from conclusion to premises. The process can be iterated until termination, at which point the leaves of the tree are literal sequents.

If the wff processed at a given state is an **atom**, then it needs no dismantling. If the wff is either a **conjunction** or a **disjunction** then it generates either one or two subtrees each of whose root sequent(s) has one less occurrence of a logical connective. If finally, the wff is a **negation**, then if it is the negation of an atom, its dismantling terminates in a literal, else if it is a double negation, both negations may be removed by the $\neg\neg$ rules, else it is the negation of a conjunction or a disjunction in which case one of the 'mixed' ($\neg\wedge$, etc.) rules may be used. In each case, the number of occurrences of logical connectives in the root sequent(s) are reduced. Hence the process terminates in a deduction tree.

8. SOUNDNESS AND COMPLETENESS

In what follows we employ the method set out in Cleaves treatment of the logic of indefiniteness.[8]

We need definitions of a valuation of WFFS and a definition of validity in LPF.

We define a class of atomic valuations, each of which is of the type

$$\text{Atomic\_Valuation} = \text{ATOMS} \rightarrow \text{Bool}_+$$

Each admissible atomic valuation is a total function. Admissible valuations have a larger domain than atomic valuations, each being of type

$$\text{Valuations} = \text{WFFS} \rightarrow \mathit{Bool}_+$$

A sequent is valid, according to the definitions LPF-1, -2, -3 iff it is true for all the appropriate **admissible valuations.** The important point is that the admissible valuations and the atomic valuations are in 1-1 correspondence. Every atomic valuation generates, and may be **extended** to, an admissible valuation. Equally, every admissible valuation may be restricted to an atomic valuation. This follows (by structural induction) from the facts that in the BNF grammar for well-formed formulae, the class of wffs is freely generated, and the fact that LPF is truth-functional. Therefore a sequent is valid iff it is valid for all atomic valuations.

Lemma 2

A literal sequent L is valid $\Leftrightarrow$ L is an axiom sequent.

Proof

LPF-1

($\Leftarrow$). If a literal sequent $\Gamma \rightarrow \Delta$ is an axiom sequent then **either** there is a literal common to both Γ and Δ, in which case whenever all the wffs (literals) in Γ are true so is at least one in Δ, **or** there is a pair of complementary literals in Γ, in which case at least one wff (literal) in Δ is true whenever all the wffs (literals) in Γ *are true, since the wffs in* Γ never are all true.

($\Rightarrow$) We show that if a literal sequent $\Gamma \rightarrow \Delta$ is **not** an LPF-1-axiom sequent then $\Gamma \rightarrow \Delta$ is **not** LPF-1-valid.

Let $\Gamma \rightarrow \Delta$ be a literal sequent which is not an LPF-1-axiom sequent.

Then

(1) there is no literal L such that $L \in \Gamma$ and $L \in \Delta$, and

(2) there is no atom P such that $P \in \Gamma$ and $\neg P \in \Gamma$.

We show that there is a total function v: ATOMS $\rightarrow$ Bool$_+$ as defined below which, since it is a function may be extended to a valuation **val** which refutes $\Gamma \rightarrow \Delta$.

```
Let $v(\alpha)$ = if $\alpha \in \Gamma$ then T
                  else
                  if $\neg\alpha \in \Gamma$ then F
                  else $\perp$
```

Note that by the final 'else' all atoms which fail to occur in both Γ and Δ take the value $\perp$. Note also that by (2) Γ contains no complementary literals, so all the literals in Γ may be assigned **T**.

Note that by (1) there is no literal common to both Γ and Δ, so none of the literals in Δ need be assigned **T**. However, if a literal L in Δ is the negation of an atom in Γ, or if it is an atom whose negation is in Γ, then L must be assigned **F**. In all other cases it may be assigned $\perp$.

Clearly v is a function. Since v is a function it may be extended to an admissible valuation **val** on WFFS. Under **val** all the literals in Γ are true while none of the literals in Δ is true. Therefore val refutes $\Gamma \rightarrow \Delta$ which is therefore not valid.

LPF-2

($\Leftarrow$). If a literal sequent $\Gamma \rightarrow \Delta$ is an axiom sequent then **either** there is a literal common to both Γ and Δ, in which case whenever all the wffs (literals) in Δ are false so is at least one in Δ, **or** there is a pair of complementary literals in Δ, in which case at least one wff (literal) in Γ is false whenever all the wffs (literals) in Δ are false, since the wffs in Δ never are all false.

($\Rightarrow$). We show that if a literal sequent $\Gamma \rightarrow \Delta$ is **not** an LPF-2-axiom sequent then $\Gamma \rightarrow \Delta$ is **not** LPF-2-valid.

As before, let $\Gamma \rightarrow \Delta$ be a literal sequent which is not an LPF-2-axiom sequent. Then

(1) there is no literal L such that $L \in \Gamma$ and $L \in \Delta$, and

(2) there is no atom P such that P such that $P \in \Delta$ and $\neg P \in \Delta$.

Again we show that there is a total function v: ATOMS $\rightarrow$ Bool$_+$ as defined below which, since it is a function, may be extended to an admissible valuation **val** which refutes $\Gamma \rightarrow \Delta$.

```
Let $v(\alpha)$ = if $\alpha \in \Delta$ then F
                  else
                  if $\neg\alpha \in \Delta$ then T
                    else $\perp$
```

An argument similar to that for the LPF-1 case shows that v is a function. As before, since v is a function it may be extended to an admissible valuation **val** on WFFS. Under **val** all the literals in Δ are false while none of the literals in Γ is false. Therefore $\Gamma \rightarrow \Delta$ is not valid.

LPF-3

($\Leftarrow$). If a literal sequent $\Gamma \rightarrow \Delta$ is an axiom sequent then **either** there is a literal common to both Γ and Δ, in which case whenever all the wffs (literals) in Δ are false so is at least one in Δ, **or** there is a pair of complementary literals in Γ and a pair of complementary literals in Δ. In the latter case at least one wff (literal) in Δ is true whenever all the wffs (literals) in Γ are true, and one wff (literal) in Γ is false whenever all the wffs (literals) in Δ are false, since the wffs in Γ never are all true and the wffs in Δ are never all false.

($\Rightarrow$). We show that if a literal sequent $\Gamma \rightarrow \Delta$ is **not** an LPF-3-axiom sequent then $\Gamma \rightarrow \Delta$ is **not** LPF-3-valid.

As before, let $\Gamma \rightarrow \Delta$ be a literal sequent which is not an LPF-2-axiom sequent.

Then

(1) there is no literal L such that $L \in \Gamma$ and $L \in \Delta$, **and**

(2) it is not the case that there are atoms P, Q such that $P \in \Gamma$ and $\neg P \in \Gamma$, and $Q \in \Delta$ and $\neg Q \in \Delta$.

Again we show that there is a total function v: ATOMS $\rightarrow$ Bool$_+$ as defined below which, since it is a function, may be extended to an admissible valuation **val** which refutes $\Gamma \rightarrow \Delta$.

```
Let $v(\alpha)$ = if $\alpha \in \Gamma$ then if $\neg\alpha \in \Gamma$ then $\perp$
                                   else T
              else
              if $\neg\alpha \in \Gamma$ then F
              else
              if $\alpha \in \Delta$ then if $\neg\alpha \in \Delta$ then $\perp$
                                   else F
              else
              if $\neg\alpha \in \Delta$ then T
              else $\perp$
```

The function v is, incidentally, elegantly displayed as a matrix in ref. 8, pp. 320–321. According to v, complementary literals both of which appear in either Γ or Δ (but not of course in both) are assigned $\perp$. v may be extended as before to a function **val** which refutes $\Gamma \rightarrow \Delta$.

Lemma 3

The LPF introduction rules are validity-preserving in **both directions**.

Proof. Consider the following **linear ordering** $\leqslant$ on Bool$_+$:

$$\begin{tikzcd} \mathbf{T} \\ \perp \arrow[u] \\ \mathbf{F} \arrow[u] \end{tikzcd}$$

where the (reflexive, transitive closure of the) arrow '$\rightarrow$' represents $\leqslant$. This is of course **not** the partial ordering of Bool$_+$ considered by Jones in his remarks on the monotonicity of the LPF connectives (ref. 1, pp. 76–77).

Define the functions

$$\min : \text{Bool}_+ \times \text{Bool}_+ \rightarrow \text{Bool}_+$$
$$\max : \text{Bool}_+ \times \text{Bool}_+ \rightarrow \text{Bool}_+$$

to conform to the truth-tables, so that

$$\min(b_1, b_2) = b_1 \wedge b_2$$
$$\max(b_1, b_2) = b_1 \vee b_2$$

where $b_1, b_2 \in \text{Bool}_+$. These definitions do conform to our linear ordering, in that max (**T**, **F**) = **T** and max (**F**, $\perp$) = $\perp$ etc., etc.

By analogy, we can define the functions $\min_{SEQ}$ and $\max_{SEQ}$, each of which takes a Wfflist and a Valuation and yields an element of Bool$_+$.

```
$\min_{SEQ}$: Wfflist × Valuations → Bool$_+$
$\min_{SEQ}(l, v) \underline{\Delta}$
cases $l$ of
  [ ] → T
  cons($hd$, $tl$) → min ($v(hd)$, $\min_{SEQ}(tl)$)
end
```

and similarly

```
$\max_{SEQ}$: Wfflist × Valuations → Bool$_+$
$\max_{SEQ}(l, v) \underline{\Delta}$
cases $l$ of
  [ ] → F
  cons($hd$, $tl$) → max($v(hd)$, $\max_{SEQ}(tl)$)
end
```

It is easy to see that we can re-write the conditions on the validity of an LPF sequent $\Gamma \rightarrow \Delta$ in the following ways:

[*Valid-1*] $(\forall \boldsymbol{v} \in \text{Valuations}).(\min_{SEQ}(\Gamma, \boldsymbol{v}) = \mathbf{T} \Rightarrow \max_{SEQ}(\Delta, \boldsymbol{v}) = \mathbf{T})$

[*Valid-2*] $(\forall \boldsymbol{v} \in \text{Valuations}).(\max_{SEQ}(\Delta, \boldsymbol{v}) = \mathbf{F} \Rightarrow \min_{SEQ}(\Gamma, \boldsymbol{v}) = \mathbf{F})$

[*Valid-3*) $(\forall \boldsymbol{v} \in \text{Valuations}).(\min_{SEQ}(\Gamma, \boldsymbol{v}) \leqslant \max_{SEQ}(\Delta, \boldsymbol{v}))$

Now we can check that each of the introduction rules preserves validity in both directions for each of the three definitions of sequent validity.

This is quite straightforward for rules like ($\wedge$: left) whose premises consist of a single sequent. In this particular case the result is obvious because

$(\forall \boldsymbol{v} \in \text{Valuations}).(\min_{SEQ}(\Gamma, A, B, \Delta, \boldsymbol{v}) = \min_{SEQ}(\Gamma, A \wedge B, \Delta, \boldsymbol{v}))$

For the rules which have two sequents in their premises we take ($\vee$: left) as typical. Here we have

$(\forall \boldsymbol{v} \in \text{Valuations}).(\min_{SEQ}(\Gamma, A \vee B, \Delta, \boldsymbol{v}) = \max(\min_{SEQ}(\Gamma, A, \Delta, \boldsymbol{v}), \min_{SEQ}(\Gamma, B, \Delta, \boldsymbol{v})))$

so that if

$$\Gamma, A \vee B, \Delta \rightarrow \Theta$$

is valid, so are

$$\Gamma, \Delta, A \rightarrow \Theta \quad \text{and} \quad \Gamma, \Delta, B \rightarrow \Theta$$

and vice versa.

A similar calculation handles the remaining rules of inference.

Theorem (Soundness and Completeness)

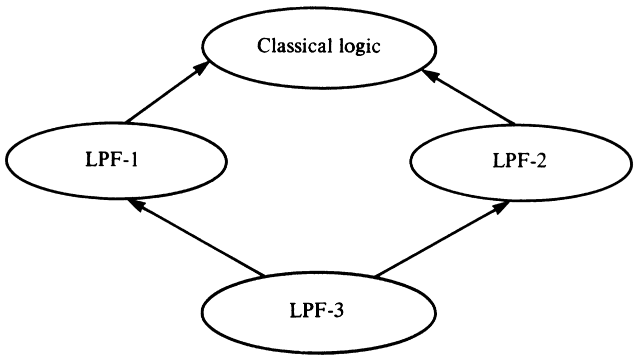
A sequent S is the root of some proof tree $\Leftrightarrow S$ is valid.

Proof. From Lemma 2 and 3.

A sequent generates a deduction tree. A sequent is valid iff the deduction tree it generates has leaves which are all and only axiom sequents.

Finally, notice that the relationship between LPF-1, -2 and -3 and classical logic is as illustrated in the following diagram. The class of valid sequents of LPF-3 is clearly the **intersection** of the classes of valid sequents of LPF-1 and LPF-2.

Of these logics, the current version of VDM chooses LPF-1. Just why is a matter of the philosophy of software engineering, a subject we leave to another paper.



REFERENCES

1. C. B. Jones, *Systematic Software Development Using VDM*. Prentice-Hall International (1986).
2. J. H. Cheng, A logic for partial functions, Ph.D. Thesis, University of Manchester (1986).
3. G. Kolestos, Sequent calculus and partial logic, M.Sc. Thesis, University of Manchester (1976).
4. S. R. Blamey, Partial valued logic. D.Phil. Thesis, Oxford University.
5. S. C. Kleene, *Introduction to Metamathematics*, p. 334. North-Holland Publishing Co. (1952).
6. H. Barringer, J. H. Cheng and C. B. Jones, A logic covering infiniteness in program proofs. *Acta Informatica* **21** 251–269 (1984).
7. N. Rescher, *Many-Valued Logics*, pp. 84 ff. McGraw-Hill (1969).
8. J. P. Cleave, The notion of logical consequence in the logic of inexact predicates, *Zeitshr. f. math. Logik d. Math. Bd.* **20**, S.307–324 (1974).

Announcement

3–6 July 1989

Seventh International Conference on **Software Engineering for Telecommunication Switching Systems,** Bournemouth.

Aims

To provide a forum for the exchange of ideas and the discussion of problems concerning **software for telecommunication systems.** The conference seeks to examine future trends and needs based on current practical experience. It also seeks to encourage contact with practitioners in related fields of application.

Based on the successful format of the six previous conferences, the programme will be limited to a maximum of about 35 papers. Emphasis will be on achievements of technical interest and advanced ideas rather than on reports of a routine nature. In addition there will be an opportunity to demonstrate and discuss **software engineering tools and support environments** outside the formal sessions. Facilities for the ever-popular **informal discussions** will also be available.

Scope

Since the last conference, there have been many changes in the world of telecommunications. The Organising Committee would welcome practical and theoretical submissions that address some of the issues raised by the following, deliberately provocative, discussions of those changes.

Liberalisation of the marketplace and rapid technological progress have changed the nature of the product and have placed new stresses on its deployment and exploitation. Software is playing an increasingly important part but remains among the least understood of the disciplines involved.

The duration of the **product life cycle** is decreasing. New services must be conceived, developed and deployed in **shorter timescales,** and have correspondingly shorter sales lifes. There are also moves towards '**unbundling**' software and offering, both to the customer and to third party suppliers, the opportunity to **add new services** to existing products. These, in turn, raise new issues in **requirements analysis, service specification, security, integrity, validation, cost estimation, metrication,** and the **integration of the support software** for all of these which cannot be resolved solely by the processes of international **standardisation.**

The concept of the '**layered**' **architecture,** popularised through the **OSI framework,** is being invoked throughout the system, but it lacks analytical power. **Synthesis,** of correct implementation from requirement, is eagerly sought, through the use of **declarative (logical and applicative) languages, object-oriented and process-based architectures and computer-assisted transformation** techniques. However, the conceptual and **theoretical foundations** involved are insufficiently understood and not mature enough to be used 'methodically' at the appropriate scale.

The introduction of large-scale **and information system management,** international 'integrated' **(ISDN, IBCS)** systems poses new problems in **strategic planning** and **decision analysis** which will require the widespread use of sophisticated computational and heuristic systems.

Working Language

The working language of the conference is English, which will be used for all printed material, presentations and discussions. Simultaneous translation will not be provided.

Venue and Date

The Conference will be held in the Bournemouth International Centre (BIC) from 3 to 6 July 1989. Hotel details will be sent with the provisional programme details.

Registration

Registration forms and further programme details will be made available a few months before the event.

For further information contact:
Conference Services, The Institution of Electrical Engineers, Savoy Place, London WC2R 0BL, United Kingdom.