Formal Aspects of Computing (2020) 5: 208–223

Partial Logics Reconsidered: A Conservative Approach

Olaf Owe

Department of Informatics, University of Oslo, Norway

Keywords: Partial logic; Three-valued logic; Partial functions; Recursive functions

Abstract. Partial functions play an important role in computer science. In order to reason about partial functions one may extend classical logic to a logic supporting partial functions, a so-called partial logic. Usually such an extension necessitates side-conditions on classical proof rules in order to ensure consistency, and introduces non-classical proof rules in order to maintain completeness. These complications depend on the choice of consequence relation and non-monotonic operators. In computer science applications such complications are undesirable, because they affect (semi-) mechanical reasoning methods, and make manual reasoning difficult for computer scientists who are not logicians.

By carefully choosing the consequence relation and non-monotonic operators, a simple calculus for partial functions arises. The resulting logic is "healthy" in the sense that "meaningless" formulas (such as $top(emptystack) > 1$) cannot be concluded, except from contradictory or false assumptions, and a meaningless assumption provides no information. This requires all axioms to be healthy; and as a consequence the "excluded middle" $(a \vee \neg a)$ must be weakened (to meaningful a's). All the well-known classical rules preserve healthiness and are therefore sound in the logic, provided substitutions are restricted to meaningful terms. This means that program reasoning methods based on classical logic usually can be adapted to the presented partial logic.

1. Introduction

By a partial logic we mean a logic with which one may reason about partial functions, errors and well-definedness. The need for a partial logic in computer

Correspondence and offprint requests to: Olaf Owe, Department of Informatics, University of Oslo, PB 1080 Blindern, 0316 Oslo, Norway

science applications is well documented [BCJ84, Bla86, Bli88, Che86, ChJ90, Gog78, Hoo87, Kle52, KTB88]. Examples of partial functions are the indexing operation on an array, division, suc- and pred-operations on a bounded interval. And a recursively defined function may be partial, being well-defined when the recursion terminates without causing an error.

In some partial logics many proof rules of classical first order logic are not sound, and must be modified by well-definedness conditions. This makes such a partial logic difficult to learn and use in practice, especially for non-logicians. It would be desirable to have a partial logic which is *conservative* (with respect to classical logic) in the sense that the classical proof rules are sound, and that few additional proof rules and axioms are needed in order to make the logic (relatively) complete — so that the logic can be presented with few non-classical additions.

In this paper we will compare different partial logics with respect to conservation of critical classical proof rules. We consider only partial logics in which the classical logical connectives may be interpreted with the extension of Kleene [Kle52], which preserves essential propositional laws (see section 2). The logic called WS, with weak interpretation of assumptions and strong interpretation of theorems, is found to be the most conservative (see section 3).

The last part of the paper gives an introduction to WS logic, presenting a sound and relatively complete set of proof rules (see section 4). The system is easily extended with the well-definedness operator (Δ) and strong equality ($==$) as non-monotonic logical operators, preserving soundness and relative completeness. We define the well-definedness operator constructively. This simplifies the presentation of the logic, and has the advantage that all reasoning can be done within the monotonic part of the logic. Monotonicity makes it easier to relate abstract proofs to program execution, for instance to prove properties of recursively defined functions (see section 5).

2. Semantic aspects

Let us consider three truth values, corresponding to true, false, and error, denoted T, F, and $\bot$, respectively. In order to obtain protection against errors we want

$$\mathrm{F} \Rightarrow a \;==\; \mathrm{T} \qquad\qquad \mathrm{F} \wedge a \;==\; \mathrm{F}$$

even when a is error, where $==$ denotes two-valued equality with respect to the three truth values, for instance $\bot == \mathrm{T}$ is F and $\bot == \bot$ is T. Then formulas such as

$$x \neq 0 \Rightarrow x/x = 1$$
$$0 < i < N \Rightarrow tab[i] < tab[s(i)]$$
$$\neg isempty(q) \wedge isempty(pop(q))$$

are meaningful. For reasoning purposes it is desirable that the and- and or-operators are commutative (thus, $a \wedge \mathrm{F} == \mathrm{F}$), and that $a \Rightarrow b == \neg a \vee b$. This leads to the following truth tables for $\Rightarrow$, $\wedge$, $\vee$, and negation:

$\Rightarrow$	T	F	$\bot$
T	T	F	$\bot$
F	T	T	T
$\bot$	T	$\bot$	$\bot$

$\wedge$	T	F	$\bot$
T	T	F	$\bot$
F	F	F	F
$\bot$	$\bot$	F	$\bot$

$\vee$	T	F	$\bot$
T	T	T	T
F	T	F	$\bot$
$\bot$	T	$\bot$	$\bot$

$\neg$	
T	F
F	T
$\bot$	$\bot$

It follows that all operators are monotonic. Both $\wedge$ and $\vee$ are associative, and they satisfy the classical distribution laws. Furthermore, the classical equivalences between $\wedge$, $\vee$, $\Rightarrow$, and De Morgan's laws are satisfied:

$$\begin{array}{lcl@{\qquad}lcl} a \vee b & == & b \vee a & (a \vee b) \wedge c & == & a \wedge c \vee b \wedge c \\ a \wedge b & == & b \wedge a & (a \wedge b) \vee c & == & (a \vee c) \wedge (b \vee c) \end{array}$$

$$\begin{array}{lcl@{\qquad}lcl} a \Rightarrow b & == & \neg a \vee b & \neg(a \Rightarrow b) & == & a \wedge \neg b \\ a \vee b & == & \neg(\neg a \wedge \neg b) & \neg(a \vee b) & == & \neg a \wedge \neg b \\ a \wedge b & == & \neg(\neg a \vee \neg b) & \neg(a \wedge b) & == & \neg a \vee \neg b \end{array}$$

where a and b denote formulas. With these laws it is possible to write a formula in conjunctive, or disjunctive, normal form. The following simplification laws hold:

$$\begin{array}{lcl@{\qquad}lcl} a \wedge a & == & a & (a \vee b) \wedge a & == & a \\ a \vee a & == & a & (a \wedge b) \vee a & == & a \end{array}$$

$$\begin{array}{lcl@{\qquad}lcl} \neg\neg a & == & a & a \wedge \mathrm{T} & == & a \\ \neg \mathrm{T} & == & \mathrm{F} & a \wedge \mathrm{F} & == & \mathrm{F} \\ \neg \mathrm{F} & == & \mathrm{T} & a \vee \mathrm{T} & == & \mathrm{T} \\ \neg \bot & == & \bot & a \vee \mathrm{F} & == & a \end{array}$$

The following laws do *not* hold:

$$\begin{array}{lcl@{\qquad}l} a \wedge \neg a & == & \mathrm{F} & (\text{only } \neg((a \wedge \neg a) == \mathrm{T})) \\ a \vee \neg a & == & \mathrm{T} & (\text{only } \neg((a \vee \neg a) == \mathrm{F})) \end{array}$$

which reflect the fact that there are three truth values. It follows that $a \Rightarrow a == \mathrm{T}$ does not hold!

2.1. Quantifiers

As Kleene, we interpret the universal quantifier as a generalized and-operator, and the existential quantifier as a generalized or-operator. This ensures that the quantifiers are monotonic and that the classical prenex transformations rules are sound.

Let x denote a variable (of any type). The universal quantification $\forall x.\ a$ is true if a is true for all possible values of x, false if a is false for (at least) one value of x, otherwise undefined. Thus the quantification is undefined when a is undefined for at least one value of x and never false.

The existential quantification $\exists x.\ a$ is true if a is true for at least one value of x, false if a is false for all possible values of x, otherwise undefined. Thus the existence is undefined when a is undefined for at least one value of x and never true. It follows that

$$\neg\forall x.a == \exists x.\neg a \qquad\qquad \neg\exists x.a == \forall x.\neg a$$

2.2. Definedness

The well-definedness of a formula, or a term, is defined constructively, by defining the well-definedness operator Δ by structural induction:

$$
\begin{array}{lcl}
\Delta(\bot) & == & \text{F} \\
\Delta(\text{T}) & == & \text{T} \\
\Delta(\text{F}) & == & \text{T} \\
\Delta(x) & == & \text{T} \\
\Delta(\neg a) & == & \Delta a \\
\Delta(a \wedge b) & == & \Delta(a,b) \vee (\Delta a \wedge \neg a) \vee (\Delta b \wedge \neg b) \\
\Delta(a \vee b) & == & \Delta(a,b) \vee (\Delta a \wedge a) \vee (\Delta b \wedge b) \\
\Delta(a \Rightarrow b) & == & \Delta(a,b) \vee (\Delta a \wedge \neg a) \vee (\Delta b \wedge b) \\
\Delta(\forall x.a) & == & (\forall x.\Delta a) \vee (\exists x.\Delta a \wedge \neg a) \\
\Delta(\exists x.a) & == & (\forall x.\Delta a) \vee (\exists x.\Delta a \wedge a)
\end{array}
$$

where $\Delta(a,b)$ denotes $(\Delta a \wedge \Delta b)$. A formula a is said to be *meaningful* when Δa holds, and *meaningless* when $\neg\Delta a$ holds. It follows by structural induction that the formula denoted by Δa is meaningful. Notice that $\Delta a \wedge a$ and $\Delta a \Rightarrow a$ are meaningful, and that $\Delta(a \Rightarrow b) \wedge (a \Rightarrow b) == (\Delta a \Rightarrow a) \Rightarrow (\Delta b \wedge b)$, $\Delta(\forall x.a) \wedge (\forall x.a) == (\forall x.\Delta a \wedge a)$, and $\Delta(\forall x.a) \Rightarrow (\forall x.a) == (\forall x.\Delta a \Rightarrow a)$ (and similarly for $\exists$).

The well-definedness of non-logical predicates and functions can be defined as in [Owe84], by introducing a total and strict "ghost-predicate" d_f for each partial predicate or function f, with the same domain as f. For instance, for the non-negative numbers, $d_{pred}(x)$ is $x > 0$. The well-definedness of a non-logical function (or predicate) f is defined by

$$\Delta(f(e)) == \Delta e \wedge d_f(e)$$

when f is partial, and by $\Delta(f(e)) == \Delta e$ when f is total, where e denotes a list of terms, (consisting of variables and n-ary functions, n$\geq$ 0). This gives a strict semantics for all non-logical functions and predicates. Any application of the non-monotonic well-definedness operator is now constructively defined in terms of monotonic operators only (since d_f is monotonic). Non-logical axioms about d_f can be given directly, or indirectly through a definition of f, as explained in section 5.

We may introduce weak equality (=) as a strict and total predicate, thus $\Delta(e1 = e2) == \Delta(e1, e2)$. Weak equality corresponds to the executable equality used in programming. One may translate the strong equality $e1 == e2$ to $\neg\Delta e1 \wedge \neg\Delta e2 \vee \Delta(e1, e2) \wedge e1 = e2$, thereby avoiding the non-monotonic == operator. Thus, applications of the well-definedness operator as well as strong equality may be reduced to monotonic formulas.

Notice that variables (of any type) are considered meaningful. If variables were to range over error-values, there would be few interesting meaningful formulas with free or bound variables. Let $a[x := e]$ denote a with all free occurrences of x replaced by e (renaming bound variables in a when needed). $a[x := e]$ is meaningful whenever both a and e are meaningful.

3. Comparison of different partial logics

3.1. Classical proof rules

In this section, we present proof rules in the style of Gentzen sequent calculus [Gal86] where a sequent has the form $A \vdash c$ where c, the *conclusion*, is a formula and A is a (possibly empty) list of formulas, called *assumptions* — to

be understood as follows: from the set of formulas in A we may infer c. Commas in A may be interpreted as and's. For instance, reflexivity of the proof symbol may be expressed as the axiom

$$c \vdash c \qquad \text{(trivial sequent)}$$

General structural rules

The classical structural rules for sequent calculus are formulated below, letting x denote variables (of any type), $e,\ t$ denote terms (of the same type as x), $a,\ b,\ c$ denote formulas and letting $A,\ B,\ C$ denote lists of formulas.

We may enrich the assumption set

$$\frac{A \vdash c}{B \vdash c} \qquad \text{(enrichment rule)}$$

provided the set of formulas in A is a subset of the set of formulas in B. Observe that the rule also may be used to remove duplicated assumptions (contraction).

Free variables (of any type) in a sequent may be replaced by any term (of the same type):

$$\frac{A \vdash c}{A[x := t] \vdash c[x := t]} \qquad \text{(substitution rule)}$$

In the following, we use the convention that a rule (with n premises, $n > 0$) of the form $\dfrac{\overset{\cdots}{A_i \vdash c_i}\ (i \in 1..n)}{\underset{\cdots}{A \vdash c}}$ is an abbreviation for the rule $\dfrac{\overset{\cdots}{B_i, A_i \vdash c_i}}{\underset{\cdots}{B, A \vdash c}}$ where the set of formulas in B is the union of the sets of formulas in B_i $(i \in 1..n)$, i.e. it is understood that the conclusion inherits all assumptions not explicitly mentioned. (This convention does not apply to the substitution rule.)

Transitivity of the proof symbol may now be formulated by the rule

$$\frac{\begin{array}{c}\vdash a \\ a \vdash c\end{array}}{\vdash c} \qquad \text{(cut)}$$

Critical Logical Rules

We will here consider well-known proof rules of classical sequent calculus and natural deduction [Gal86, Pra65] which are sensitive to the choice of validity. The most sensitive proof rules are those involving non-trivial use of assumptions or elimination of formulas in the conclusion. The quantifier rules do not raise any additional soundness issues (as explained later) and are omitted.

$$\frac{a \vdash c}{\vdash a \Rightarrow c} \qquad \text{(deduction rule)}$$

$$\frac{\begin{array}{c}\vdash a \\ \vdash a \Rightarrow c\end{array}}{\vdash c} \qquad \text{(Modus Ponens)}$$

$$\frac{a \vdash c \quad \neg a \vdash c}{\vdash c} \qquad \text{(proof by cases)}$$

$$\frac{\vdash c \quad \vdash \neg c}{\vdash \mathrm{F}} \qquad \text{(proof of contradiction)}$$

$$\frac{\neg a \vdash \mathrm{F}}{\vdash a} \qquad \text{(proof by contradiction)}$$

$$\frac{\vdash \mathrm{F}}{\vdash a} \qquad \text{(false-elimination)}$$

$$\frac{\vdash a}{\neg a \vdash \mathrm{F}} \qquad \text{(false-introduction)}$$

$$\frac{a \vdash b}{\neg b \vdash \neg a} \qquad \text{(swap)}$$

$$\frac{\vdash a \vee b \quad \vdash \neg a \vee c}{\vdash b \vee c} \qquad \text{(resolution)}$$

$$\frac{\neg a \vdash b}{\vdash a \vee b} \qquad \text{(or-introduction)}$$

$$\frac{\vdash a \vee b \quad a \vdash c \quad b \vdash c}{\vdash c} \qquad \text{(or-elimination)}$$

In addition to the above classical rules we consider the following rules involving well-definedness:

$$\frac{\vdash c}{\vdash \Delta c} \qquad \text{(well-definedness rule)}$$

expressing that only meaningful formulas may be proved, and

$$\frac{\vdash \Delta t \quad A \vdash c}{A[x := t] \vdash c[x := t]} \qquad \text{(weak substitution rule)}$$

(no implicit assumption with x free) expressing that substitutions should be meaningful, which is reasonable in three-valued logic.

3.2. Comparison

As usual we let a sequent with free variables be valid iff it is valid for all meaningful, variable-free substitutions. We may then characterize different partial logics by presenting tables for the validity of $a \vdash b$ when a and b are variable-free:

SW	T	F	⊥
T	v	i	v
F	v	v	v
⊥	v	v	v

WW	T	F	⊥
T	v	i	v
F	v	v	v
⊥	v	i	v

SS	T	F	⊥
T	v	i	i
F	v	v	v
⊥	v	v	v

WS	T	F	⊥
T	v	i	i
F	v	v	v
⊥	v	i	i

DS	T	F	⊥
T	v	i	i
F	v	v	v
⊥	i	i	i

DD	T	F	⊥
T	v	i	i
F	v	v	i
⊥	i	i	i

DD+	T	F	⊥
T	v	i	i
F	v	v	i
⊥	i	i	v

WS+	T	F	⊥
T	v	i	i
F	v	v	v
⊥	v	i	v

(v for valid.) The four first logics are obtained by interpreting the assumption part (a) and the conclusion part (b) independently; the letter W indicates weak interpretation (weak interpretation of a is $\Delta a \Rightarrow a$) and the letter S indicates strong interpretation (strong interpretation of a is $\Delta a \wedge a$). Of these, WS is clearly the most restrictive. The DS-logic is even more restrictive, insisting that the assumption part is meaningful, in the sense that nothing (not even T) can be proved from undefined assumptions. DD is the most restrictive, insisting that both parts are meaningful independently. WS+, DD+, and DS+ are like WS, DD, and DS, respectively, except that $\bot \vdash \bot$ is valid. WS+ is more restrictive than WW and SS, but less than WS. Other extensions of classical logic than these are possible, but they seem quite unreasonable.

A table of sound/unsound rules for the different partial logics follows:

	SW	WW	SS	WS	DS	DD	WS+	DS+	DD+
trivial sequent	s	s	s	u	u	u	s	s	s
cut	u	s	s	s	s	s	s	s	s
enrichment	s	s	s	s	u	u	s	u	u
deduction rule	s	s	u	s	u	u	u	u	u
Modus Ponens	u	u	s	s	s	s	u	u	u
proof by contradiction	s	s	u	s	u	u	u	u	u
false-elimination	s	s	s	s	s	u	s	s	u
false-introduction	s	u	s	s	s	s	u	u	u
swap	s	u	u	s	u	u	u	u	u
proof by cases	u	s	u	s	s	s	u	u	u
well-definedness rule	u	u	s	s	s	s	u	u	u

(s for sound). The classical substitution rule is not sound in any partial logic, since it allows $\Delta\bot$ to be concluded from Δx , but it is sound in SW-logic when restricted to monotonic formulas. However, the weak substitution rule is sound in all logics. (The same discussion applies to classical $\forall$-elimination.)

The soundness of resolution and proof *of* contradiction is like that of Modus Ponens. The soundness of or-introduction is like the deduction rule; and the soundness of or-elimination is like cut. The classical and-introduction and elimination rules are sound in all logics.

For instance, the deduction rule is not sound in DS (and DD) because $\bot, \mathrm{F} \vdash c$ is valid, but $\bot \vdash \mathrm{F} \Rightarrow c$ is not; and it is not sound in SS because $\bot \vdash \bot$ is valid, but $\vdash \bot \Rightarrow \bot$ is not. Modus Ponens is sound in SS because from $\Delta a \wedge a$ and $\Delta(a \Rightarrow c) \wedge (a \Rightarrow c)$ follows $\Delta c \wedge c$. The same proof goes for DS and WS. It is not sound in WS+ since $\bot \vdash \bot$ and $\bot \vdash \bot \Rightarrow \mathrm{F}$ are both valid but not $\bot \vdash \mathrm{F}$.

The strongest objections against SW are that the proof symbol is not transitive (which implies that standard induction rules are unsound) and that Modus Ponens is not sound. The strongest objections against WW are that Modus Ponens and false-introduction are not sound. The strongest objections against SS are that the deduction rule and proof by contradiction are not sound. (In addition, proof by cases, or-introduction and swap are not sound.) WS, DS, and DD are semantically similar, but DS and DD have more restrictions on the use of assumptions, and therefore less rules are sound. The + logics do not seem very attractive.

WS is the only logic which satisfies both Modus Ponens and the deduction rule, and is the only logic which satisfies all the rules in the table. Furthermore, all the classical introduction and elimination rules of sequent calculus (including quantifier rules) are sound in WS, provided substitutions are meaningful.

The strongest objection against WS is that the trivial sequent $a \vdash a$ is not sound. This may be explained from the fact that there are three truth values (T, F, and $\bot$); thus $a \vee \neg a \vee \neg\Delta a$ holds, which is the same as $\Delta a \wedge a \Rightarrow a$ which (in WS) is equivalent to $\Delta a\ ,\ a \vdash a$, called the WS-axiom. Replacing $a \vdash a$ by the WS-axiom has the benefit that one never can prove meaningless formulas, not even from meaningless assumptions (only from false assumptions). Since meaningless assumptions may seem meaningful (for instance when the assumption is an induction hypothesis), the WS-axiom may even be considered advantageous.

It seems therefore that WS provides reasoning most close to traditional logic. The only major difference concerns the trivial sequents. By accepting that trivial sequents are dangerous in three valued logic, WS becomes the most natural logic.

Example 1. Consider suc- and pred-operators restricted to the bounded interval $0..N$. The formula

$$x < N \Rightarrow pred(suc(x)) = suc(pred(x))$$

is meaningless when x is zero. Even in this simple example it is not obvious to see that the condition $0 < x$ is missing. In SW and SS one could derive false when this formula, universally quantified, is taken as an assumption (and one could derive $0 < x$ when the assumption is not quantified). In DS and DD one cannot prove anything from the assumption, not even true. In DS+, DD+ and WS+ one could prove meaningless theorems (for instance $pred(0) = 0$) from this assumption — in WW even without the assumption. In WS nothing can be proved about $pred(0)$; the assumption provides no information when x is 0. Not even the sequent $pred(0) < 0 \vdash pred(0) < 0$ is valid. From the assumption $\forall x.pred(x) < x$ one may in WS conclude $pred(1) < 1$, but not $pred(0) < 0$.

Since an assumption a in WS is interpreted as $\Delta a \Rightarrow a$, one may omit well-definedness conditions in assumptions. This gives a practical way of providing consistent information about partial functions, in particular when their well-definedness is not fully known. And when their definedness is known, one need not repeat the appropriate well-definedness conditions for each assumption.

WS logic provides the possibility of defining "strong axioms", of the form

$\vdash\ a$ as well as "weak axioms" in the form of (closed) assumptions. For instance, the sequent $a,\ a \Rightarrow c\ \vdash\ c$ is not valid in WS, even though Modus Ponens is sound in WS. (In [GaL89], where only the former formulation was considered, it was claimed that Modus Ponens is not sound in WS, and further studies of WS was omitted.) In a paper by Konikowska et al [KTB88], WS logic is excluded from their considerations since it is claimed to "lead to theories where an axiom does not need to be a theorem". Clearly, this applies to weak axioms only.

The logics SS, WW, and to some extent SW, are discussed in [KTB88]. SW logic is investigated in [Hoo77, Hoo87] and [GaL89]. SS logic (called LPF) is investigated in [BCJ84, Che86, ChJ90]. WW logic (but with non-monotonic quantifiers) is investigated in [Owe84, Hol91].[1]

4. The WS logic

4.1. Sequents without assumption part

In WS, the sequent $A\ \vdash\ c$ is equivalent to $\vdash\ c\ ,\ \overline{A}$, where commas behind the proof symbol are interpreted as or's, and where $\overline{A}$ is A with every formula negated. It is therefore possible to present WS logic with sequents of the latter form. The enrichment and substitution rules are then expressed as follows:

$$\frac{\vdash\ A}{\vdash\ B} \qquad \text{(enrichment)}$$

provided the set of formulas in A is a subset of the set of formulas in B.

$$\frac{\vdash\ A \qquad \vdash\ \Delta t}{\vdash\ A[x := t]} \qquad \text{(weak substitution)}$$

In the following we use the convention that a rule of the form $\dfrac{\overset{\ldots}{\vdash\ A_i}}{\underset{\ldots}{\vdash\ A}}$ $(i \in 1..n)$ abbreviates the rule $\dfrac{\overset{\ldots}{\vdash\ A_i, B_i}}{\underset{\ldots}{\vdash\ A, B}}$ where the set of formulas in B is (as before) the union of all those in B_i $(i \in 1..n)$. A double line as in $\dfrac{\overset{\ldots}{\vdash\ A_i}}{\underset{\ldots}{\vdash\ A}}$ $(i \in 1..n)$ means that the rule may be used both ways; in particular, for each i it implies $\dfrac{\vdash\ A}{\vdash\ A_i}$.

[1] In WW, the sequent $\vdash\ \exists x.a$ has a very weak meaning; it is valid for instance when a is meaningless for one or more values of x, and false for the others. However, with the non-monotonic versions of the quantifiers used in [Owe84], this sequent has the same meaning as in WS.

Logical rules

$$\frac{\vdash\ a, b}{\vdash\ a \vee b} \qquad (\vee)$$

$$\frac{\vdash\ a \quad \vdash\ b}{\vdash\ a \wedge b} \qquad (\wedge)$$

$$\frac{\vdash\ \neg a, b}{\vdash\ a \Rightarrow b} \qquad (\Rightarrow)$$

$$\frac{\vdash\ a \quad \vdash\ \neg a}{\vdash\ \mathrm{F}} \qquad (\neg)$$

$$\frac{\vdash\ \mathrm{F}, a}{\vdash\ a} \qquad (\mathrm{F})$$

$$\frac{\vdash\ a, \neg a}{\vdash\ \Delta a} \qquad (\Delta)$$

$$\frac{\vdash\ a}{\vdash\ \forall x.a} \qquad \text{no implicit clause with } x \text{ free when used forwards}$$

$$\frac{\vdash\ \exists x.a \quad \vdash\ \neg a}{\vdash\ \mathrm{F}} \qquad \text{no implicit clause with } x \text{ free}$$

$$\frac{\vdash\ a[x := t]}{\vdash\ \exists x.a} \qquad (\exists\text{-introduction})$$

Logical axioms

$$\vdash\ a, \neg a, \neg \Delta a$$

$$\vdash\ \Delta \Delta t$$

Lemma 1. The proof rules and axioms above form a sound and relatively complete system when Δ and negation (when outside a logical operator/constant) are taken as meta-operators (as defined in section 2).

If we consider meaningful formulas only, the system reduces to classical sequent calculus. Since the formula denoted by Δa is meaningful, our system is complete for such formulas. One may prove that completeness is preserved when

atomic non-meaningful formulas, possibly negated, are included. The rest of the completeness proof can be done as in the classical case. A more formal proof of soundness and completeness will appear in [ElG91].

When there are no non-logical axioms, the Δ-rule can be proved by structural induction. Thus no non-classical rules are needed to ensure completeness in this case.

Example 2. In order to prove $\vdash \mathrm{T}$, we may start with $\vdash \mathrm{T}, \neg\mathrm{T}, \neg\Delta\mathrm{T}$ (by the first axiom), which reduces to $\vdash \mathrm{T}, \neg\mathrm{T}, \neg\mathrm{T}$ (evaluating Δ), and then to $\vdash \mathrm{T}, \mathrm{F}, \mathrm{F}$ (evaluating $\neg$). By enrichment we obtain $\vdash \mathrm{T}, \mathrm{F}$ and by the F-rule we obtain $\vdash \mathrm{T}$.

We may derive the following rules and theorems concerning well-definedness aspects:

$$\frac{\vdash a}{\vdash \Delta a \wedge a} \qquad \frac{\vdash a \wedge b}{\vdash (\Delta a \wedge a) \wedge (\Delta b \wedge b)}$$

$$\frac{\vdash \forall x.a}{\vdash \forall x.(\Delta a \wedge a)} \qquad \frac{\vdash a \vee b}{\vdash (\Delta a \wedge a) \vee (\Delta b \wedge b)}$$

$$\frac{\vdash \exists x.a}{\vdash \exists x.(\Delta a \wedge a)} \qquad \frac{\vdash a \Rightarrow b}{\vdash (\Delta a \Rightarrow a) \Rightarrow (\Delta b \wedge b)}$$

$$\vdash \Delta(\Delta a \wedge a) \qquad \vdash \Delta(\Delta a \Rightarrow a) \qquad \vdash \Delta t, \neg\Delta t$$

Notice that when Δ is taken as a meta-operator, all formulas are monotonic; and the following improved substitution rule follows by structural induction:

$$\frac{\vdash A \quad \vdash \Delta A[x := t]}{\vdash A[x := t]} \qquad A \text{ monotonic, no implicit clause}$$

4.2. Equality

When the monotonic language of lemma 1 is extended with weak equality, soundness and completeness are reestablished by adding the following axiom and rules:

$$\vdash x = x \qquad \frac{\vdash e1 = e2 \quad \vdash a[x := e1]}{\vdash a[x := e2]} \qquad \frac{\vdash a \Rightarrow b \quad \vdash b \Rightarrow a}{\vdash a = b}$$

By allowing strong equality as a logical symbol, we may include the laws of section 2 as axioms (including $(e1 == e2) == \neg\Delta e1 \wedge \neg\Delta e2 \vee \Delta(e1, e2) \wedge e1 = e2$), and the following rule formalizes the computation of negation and well-definedness mentioned in lemma 1:

$$\frac{\vdash e1 == e2 \quad \vdash a[x := e1]}{\vdash a[x := e2]}$$

It follows that == is a congruence relation and that all error-values are equal.

Lemma 2. The system of lemma 1 extended with weak and strong equality, negation and Δ as logical symbols, the equality-rules and -axiom above, and the strong equations of section 2 as axioms, forms a sound and relatively complete system, provided the $\exists$-introduction rule is restricted to meaningful terms t.

The last restriction is also needed in the other partial logics, when not restricted to monotonic formulas.

Non-logical axioms

A non-logical axiom of the form $\vdash a$ leads to an inconsistent theory if a is meaningless, or has meaningless instances (for meaningful substitutions). By using strong equality one may avoid such problems: one could weaken the axiom to $\vdash \neg(a == \mathrm{F})$. Furthermore, axioms in the form of strong equations are useful when describing abstract data types. For instance, the sequence-axioms

$$\begin{array}{lll} \vdash \mathit{empty}[i] & == & \bot \\ \vdash (x+q)[i] & == & \textbf{if } i = 1 \textbf{ then } x \textbf{ else } q[\mathit{pred}(i)] \end{array}$$

(where the function *empty* gives an empty sequence and where + adds an element to the left of a sequence) are meaningful, even though indexing and *pred* are partial functions.

By introducing a non-monotonic approximation relation $\rightarrow$, defining $e \rightarrow t$ as $\Delta e \Rightarrow (e == t)$, one may conveniently express rewrite rules such as

$$\begin{array}{lll} \vdash \mathit{pop}(\mathit{push}(s,x)) & \rightarrow & s \\ \vdash \mathit{top}(\mathit{push}(s,x)) & \rightarrow & x \\ \vdash \mathit{pred}(x) < x & \rightarrow & \mathrm{T} \\ \vdash \mathit{pred}(0) & \rightarrow & \bot \end{array}$$

With the approximation relation one often avoids conditions (but may loose the Church-Rosser property, for instance, consider the two rules $x/0 \rightarrow \bot$ and $x/x \rightarrow 1$).

4.3. Sequents with assumption part

One may obtain a complete and sound system for sequents with assumption part, allowing lists of formulas both before and after the proof symbol (interpreted as and's and or's, respectively), by extending the system above with the "swap"-rules

$$\frac{A \vdash a, B}{A, \neg a \vdash B} \qquad\qquad \frac{A \vdash \neg a, B}{A, a \vdash B}$$

We may then derive all the logical rules listed in section 3, as well as the following rules (with implicit lists on both sides of $\vdash$):

$$\frac{\vdash \Delta a}{a \vdash a} \qquad\qquad \frac{a \vdash b}{\vdash a \Rightarrow b} \qquad\qquad \frac{\vdash a \qquad \Delta a, a \vdash b}{\vdash b}$$

The first rule shows how to introduce and eliminate "trivial" sequents, the second

expresses the direct correspondence in WS between $\vdash$ and $\Rightarrow$, and the last is an improved cut-rule (which follows from classical cut and the well-definedness rule).

Notice that there is no logical difference between WS and SS when restricted to sequents without assumptions (and with commas behind $\vdash$ interpreted as or's). However, whereas all of WS follows trivially from this restricted logic (by the swap-rules), this is not the case for SS. As a result WS may be presented much more simply and elegantly than SS.

5. Recursive function definitions

Consider a function definition of the form

$$f(y) \triangleq e$$

where y is a list of variables and e is a monotonic expression. This definition gives the following two non-logical axioms:

$$\vdash\ f(y) == e \qquad (f\text{-axiom})$$

$$\vdash\ d_f(y) == \Delta e \qquad (d_f\text{-axiom})$$

where d_f is total (see [DLO86]). These axioms may not fully define f and d_f if the definition is recursive. In order to obtain least fix-point semantics (with $\rightarrow$ as the ordering), one may add a rule corresponding to "computational induction":

$$\frac{\vdash\ a[x := \bot] \qquad ..,a[x := f(y)][y := e_i],..\ \vdash\ a[x := e]}{\vdash\ a[x := f(y)]}$$

where the second premise has one assumption for each recursive call $f(e_i)$, and where a has no occurrence of f and is "admissible" with respect to x [Man74]. (The rule may be improved by replacing the first premise by $\vdash\ a[x := e']$ where e' is e with each recursive call replaced by $\bot$.) This rule is neither sound in SS (without requirements to the well-definedness of e_i) nor SW, but it is sound in WW, which is well suited for partial correctness-like reasoning. It is also sound in WS, but not as strong as in WW; however, the following rule is as strong:

$$\frac{..,d_f(e_i) \Rightarrow a[x := f(y)][y := e_i],..\ \vdash\ \Delta e \Rightarrow a[x := e]}{d_f(y)\ \vdash\ a[x := f(y)]} \qquad (d_f\text{-rule})$$

(a restricted as above). This rule is sound in WS, and also in WW (see [DLO86]), but not in SS or SW. The fact that all of WS can be expressed within the monotonic part, helps when verifying the requirement that a is admissible: In particular, the monotonic constructs introduced so far (including non-logical functions and ghost-predicates from the examples) give rise to admissible formulas only.

With this rule one may prove results about when f is not well-defined, and about the function value of f when well-defined. Results about when f is well-defined, may be proved by induction over a well founded set, such as the non-negative numbers, or finitely generated abstract data types. The classical induction rule on non-negative numbers is sound in WS logic — as well as in SS

and WW, but not in SW. Notice that the d_f-predicate makes reasoning about well-definedness of f more direct, by the above rule and axiom about d_f.

Example 3.

$$f(i,j) \triangleq \textbf{if } i = 0 \textbf{ then } 0 \textbf{ else } f(i-j,j) + 1$$

where i, j are non-negative numbers, and where the minus-operation is partial, restricted to non-negative results ($d_-(i,j)$ is $j \leq i$). As usual the if-construct is strict in the first argument (the test):

$$\Delta\textbf{if } a \textbf{ then } e1 \textbf{ else } e2 == \Delta a \ \wedge \ \textbf{if } a \textbf{ then } \Delta e1 \textbf{ else } \Delta e2$$

The d_f-axiom gives

$$d_f(i,j) == (i \neq 0 \Rightarrow j \leq i \wedge d_f(i-j,j))$$

By taking a of the d_f-rule as $i = x \cdot j$, its premise reduces to

$$d_f(i-j,j) \Rightarrow i - j = f(i-j,j) \cdot j \ \vdash \\ \textbf{if } i = 0 \textbf{ then } i = 0 \cdot j \textbf{ else } (j \leq i \wedge d_f(i-j,j) \Rightarrow i = f(i-j,j) \cdot j + j)$$

which is provable in WS (given the relevant axioms about numbers), since the assumption is meaningful when $j \leq i$. We conclude

$$d_f(i,j) \ \vdash \ i = f(i,j) \cdot j$$

and by $\exists$-introduction we obtain

$$d_f(i,j) \ \vdash \ \exists x.\, i = x \cdot j$$

By induction on non-negative numbers, we may easily prove $\vdash \ i = x \cdot j \Rightarrow d_f(i,j)$ for x non-negative (using the d_f-axiom). It follows that

$$\vdash \ (d_f(i,j) = \exists x.\, i = x \cdot j) \wedge i = f(i,j) \cdot j$$

(i, j non-negative), which says everything about d_f and f, except that $f(0,0)$ is 0 (which follows directly from the definition). Adding this fact, we may conclude

$$\vdash \ f(i,j) == \textbf{if } \exists x.\, i = x \cdot j \textbf{ then if } j = 0 \textbf{ then } 0 \textbf{ else } i/j \textbf{ else } \bot$$

which defines f non-recursively (and also d_f, since the outer then-branch clearly is meaningful).

6. Conclusions

By comparing classical proof rules which are sensitive to the choice of partial logic, we have found that all rules are sound in WS logic (provided substitutions are meaningful). In particular, introduction and elimination of the logical symbols may be done as in natural deduction. For relative completeness in the presence of non-logical axioms, the only additional rule required in WS is the well-definedness rule, saying that only meaningful formulas are provable. We conclude that WS is the partial logic which provides reasoning most close to traditional classical logic. In particular, the fact that $\vdash \ a \Rightarrow b$ is equivalent to $a \ \vdash \ b$ in WS makes reasoning quite natural. (No other partial logic satisfies this equivalence and the well-definedness rule.)

The only major difference concerns trivial sequents. In WS the trivial sequent

$a \vdash a$ is valid only when a is meaningful. Clearly this results in non-classical proof obligations. For instance, the sequent $pred(x) < x \vdash pred(x) < x$ requires that $x > 0$ is proved or assumed. However, the advantage is that it is not possible to prove meaningless formulas, not even from meaningless assumptions (only from false or contradictory assumptions).

One may say that reasoning from and about errors is quite restricted in WS, since meaningless (instances of) assumptions are ignored in WS, and meaningless (instances of) theorems are considered contradictory (false). In practical applications this seems to be healthy; in particular, reasoning about recursively defined functions is straightforward, and reasoning about their definedness is more direct than usual (due to the d_f-axiom and d_f-rule).

The fact that the "swap" rule may be used both ways in WS (to move formulas back and forth over $\vdash$, while negating), makes the logical presentation of WS simple, because introduction and elimination rules on assumptions are not needed.

All in all it seems that WS is well suited for teaching purposes as well as for automated reasoning. Its application to other areas in computer science, such as Hoare logic and abstract data type theory, is demonstrated in [Dah92].

Acknowledgements

This research has benefited from close interaction with Ole-Johan Dahl. I also thank Neelam Soundararajan and Morten Elvang for their interest and comments.

References

[BCJ84] Barringer, H., Cheng, J.H. and Jones, C.B.: "A Logic Covering Undefinedness in Program Proofs." *Acta Informatica*, **21**, 251-269, (1984).

[Bla86] Blamey, S.: "Partial Logic." In *Handbook of Philosophical Logic, Volume III,* D. Gabbay and F. Guenthner (eds.), D. Reidel Publishing Company, 1986.

[Bli88] Blikle, A.: "Three-Valued Predicates for Software Specification and Validation." In *VDM — The Way Ahead,* Lecture Notes in Computer Science 328, Springer Verlag, 1988.

[Che86] Cheng, J.H.: "A Logic for Partial Functions." Ph.D. Thesis, Report UMCS-86-7-1, Dept. of CS, University of Manchester, 1986.

[ChJ90] Cheng, J.H. and Jones, C.B.: "On the Usability of Logics which Handle Partial Functions." In *Proceedings of the Third Refinement Workshop*, C. Morgan and J. Woodcock (eds.), Springer Verlag, 1990.

[DLO86] Dahl, O.-J., Langmyhr, D.F. and Owe, O.: "Preliminary Report on the Specification and Programming Language ABEL." Research Report 106, Dept. of Informatics, University of Oslo, Norway, 1986.

[Dah92] Dahl, O.-J.: *Verifiable Programming.* The Hoare Series, Prentice Hall, 1992.

[ElG91] Elvang-Gøransson, M.: "Some properties of WSL." Preprint, Dept. of Informatics, University of Oslo, 1991.

[Gal86] Gallier, J.H.: *Logic for Computer Science.* Harper & Row, 1986.

[GaL89] Gavilanes-Franco, A.G. and Lucio-Carrasco, F.: "A First Order Logic for Partial Functions." Report DIA/89/1, Dept. of Informatics, Universidad Complutense, Madrid, Spain, 1989.

[Gog78] Goguen, J.A.: "Abstract Errors for Abstract Data Types." In *Formal Description of Programming Concepts,* North Holland, 1978.

[Hol91] Holden, M.: "Weak Logic Theory." *Theoretical Computer Science,* **79**, 295-321, (1991).

[Hoo77] Hoogewijs, A.: "A Calculus of Partially Defined Predicates." Technical Report, Rijksuniversiteit, Gent, 1977.
[Hoo87] Hoogewijs, A.: "Partial Predicate Logic in Computer Science." *Acta Informatica,* **24**, 381-393, (1987).
[Kle52] Kleene, S.C.: *Introduction to Metamathematics.* North Holland, 1952.
[KTB88] Konikowska, B., Tarlecki, A. and Blikle, A.: "A Three-Valued Logic for Software Specification and Validation." In *VDM — The Way Ahead,* Lecture Notes in Computer Science 328, Springer Verlag, 1988.
[Man74] Manna, Z.: *Mathematical Theory of Computation.* McGraw-Hill, 1974.
[Owe84] Owe, O.: "An Approach to Program Reasoning Based on a First Order Logic for Partial Functions." Report CS-081, Dept. of EECS, UCSD, USA, 1984.
[Pra65] Prawitz, D.: *Natural Deduction.* Almquist & Wiksell, Stockholm, 1965.

Received July 1991
Accepted in revised form May 1992 by A. Blikle