The Connection between Two Ways of Reasoning about Partial Functions

John S. Fitzgerald, Cliff B. Jones

School of Computing Science, Newcastle University, UK

Abstract

Undefined terms involving the application of partial functions and operators are common in program specifications and in discharging proof obligations that arise in design. One way of reasoning about partial functions with classical First-order Predicate Calculus (*FoPC*) is to use a non-strict equality notion so as to insulate logical operators from undefined operands. An alternative approach is to work only with strict (weak) equality but use an alternative Logic of Partial Functions (*LPF*) — a logic in which the "Law of the Excluded Middle" does not hold. This paper explores the relationships between the theorems that can be proved in the two approaches. The main result is that theorems in *LPF* using weak equality can be straightforwardly translated into ones that are true in *FoPC*; translation in the other direction results, in general, in more complicated expressions but in many cases these can be readily simplified. Such results are important if the laudable move towards interworking of formal methods tools is to be sound.

Key words: Formal Methods, Specification Languages, Logic, Partial Functions, *LPF*, Equality

1 Introduction

Partial functions and operators are common in specifications and designs; this is clear from the literature on formal languages such as VDM [Jon90,FL98], Z [Hay93] and B [Abr96].

Applying partial functions gives rise to potentially undefined terms (e.g. the **hd** operator in VDM extracts the initial element of non-empty sequences but the term **hd** [ ] is undefined). It is sometimes possible to "protect" applications of partial operators by guards. For example, in the following expression, the conditional avoids the evaluation of a potentially non-denoting term providing

conditionals are *non-strict* in their second and third operands:

$$\textbf{if } s \neq [\,] \textbf{ then } \textbf{hd}\, s \textbf{ else } 7$$

Undefined terms that are arguments to strict predicates give rise to undefined logical values but the propositional operators and quantifiers of First-order Predicate Calculus (*FoPC*) have no meaning for undefined logical values — the logical operators are *strict* (i.e. undefined if either operand is undefined). Writing $a = b$ for weak (strict) equality, the following expression involves a potentially undefined mapping application:

$$d \in \textbf{dom}\, m \wedge m(d) = 3$$

FoPC does not define a result for the case where ($d \notin \textbf{dom}\, m$, and) the application is undefined: although one might want to read the conjunction left-to-right, there is no formal sense in which the first operand being **false** overcomes the undefinedness of the second operand. As well as the specific example of weak equality, any strict predicate would exhibit similar problems but, rather than deal with arbitrary predicates, the discussion here focuses on undefinedness arising from equalities. The motivation for this is that equality is common as tests in recursive function definitions and equalities such as in

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

must be "computational" — i.e. strict.

Such recursive functions offer a particular challenge for reasoning because their domains are not necessarily obvious. The *subp* example from [CJ91] is used commonly because it is just difficult enough (e.g. no simple domain set over which to quantify) to illustrate plausible logical assertions whose status requires thought. The function is *deliberately* partial over pairs of integers but returns $i - j$ when $i \geq j$. The following formula appears plausible and seems to capture knowledge about the actual domain of *subp*:

$$\forall i, j \in \mathbb{Z} \cdot i \geq j \;\Rightarrow\; subp(i,j) = i - j \tag{1}$$

However, the quantified expression depends on $0 \geq 1 \;\Rightarrow\; subp(0,1) = 0 - 1$ which, since $subp(0,1)$ does not –in the least fixed point– denote an integer, comes down to **false** $\Rightarrow\; \bot_{\mathbb{Z}} = -1$ which, if $=$ is weak equality, reduces to **false** $\Rightarrow\; \bot_{\mathbb{B}}$ which makes no sense in *FoPC*.

Here again it is tempting (but unjustified by the normal axioms of *FoPC*) to read the antecedent in Formula 1 as a "guard" but a standard property of

FoPC is the equivalence of an implication to its contrapositive and

$$\forall i, j \in \mathbb{Z} \cdot subp(i, j) \neq i - j \;\Rightarrow\; i < j \tag{2}$$

does not offer a natural "guard" reading. A more problematic formula is

$$\forall i, j \in \mathbb{Z} \cdot subp(i, j) = i - j \vee subp(j, i) = j - i \tag{3}$$

where there is no guarding clause.

To avoid these problems, VDM uses The Logic of Partial Functions (*LPF*) in which, for example, the value of a conjunction in which *either* operand is **false** is defined to be **false**. There is a significant body of work on the *LPF* [Che86], its proof theory [BCJ84], its typed form [JM94], its mechanisation in a proof support environment [JJLM91] and its use for reasoning about models and refinements in VDM [BFL$^+$94,Jon06,Fit07]. A full description is not attempted here but the cited papers and references therein provide a history and link *LPF* to mainstream writings on logic.

Formulae 1 to 3 above are true in *LPF* with weak equality and their proofs are straightforward (see Section 2). *LPF* is, however, non-classical in the sense that the "law of the excluded middle" does not hold. Thus:

$$\forall i, j \in \mathbb{Z} \cdot subp(0, 1) = 42 \vee \neg\,(subp(0, 1) = 42) \tag{4}$$

is not a theorem in *LPF*.

There are approaches other than *LPF* that attempt to tame undefined terms; a categorisation is given in [CJ91] on the basis of where undefined values are "caught"; advantages and disadvantages of the approaches are also listed. In several approaches, Formula 1 may only be proved with side conditions. In even more approaches, Formula 3 would have to be ruled inadmissible. There is an intellectual interest in understanding the relationships between approaches but there is now a pressing practical issue because serious consideration is being given to moving conjectures between theorem proving tools, e.g. in the discussions around the "verification grand challenge" [JOW06].

1.1 Notions of Equality

As well as the axiomatisation of the logics, the question of the form of equality that best fits a particular logic is important: so we first explore these. In particular, although we assume a standard axiomatisation of *FOPC* (e.g. [Kle67]),

translations of Formulae 1–3 can be made provable by using different notions of equality.

The "=" used above is a *weak* relational operator. It is undefined if either operand is undefined. One way of insulating logical operators from undefined values is to provide versions of relational operators that absorb undefined terms. For example, existential equality ($=_\exists$) is a non-strict equality that yields false if either operand is undefined ("existence" is required for truth). In the *subp* example, a translation of Formula 1 with existential equality is

$$\forall i, j \in \mathbb{Z} \cdot i \geq j \;\Rightarrow\; subp(i, j) =_\exists i - j \tag{5}$$

This avoids the undefinedness problem for *FoPC* because, when the antecedent $i \geq j$ is false, the $subp(i, j)$ term is undefined and so the consequent is false because of the existential equality. (Note that, in this example, the weak relational operator $\geq$ is retained in the antecedent since it is always defined over $\mathbb{Z}$ and i and j are bound to $\mathbb{Z}$.) Similarly, the weak equality in the if-condition within the definition of *subp* has to be retained.

An alternative non-strict relation is strong equality ($==$) which differs from its existential cousin only in that $\bot_\mathbb{Z} == \bot_\mathbb{Z}$ is true. The following translated version of Formula 1, with strong equality is therefore also valid.

$$\forall i, j \in \mathbb{Z} \cdot i \geq j \;\Rightarrow\; subp(i, j) == i - j \tag{6}$$

Neither existential nor strong equality are computable because they are non-strict. The form of equality that arises in computations is "weak" or "strict" being undefined where either operand is undefined. One of the disadvantages of reasoning with non-strict operators is that one inevitably has to have different versions of the operators in the same proof because the functions, definitions etc. must use computable operators. Remember as well that equality is being used here to focus the discussion but the same issue of strict vs. non-strict versions has to be tackled for all predicates whose arguments can be (potentially undefined) terms.

1.2 Translations between Approaches

As indicated above, Formula 1 is true in *FoPC* if the equality operator is changed to existential equality as in Formula 5; and even the following translation of Formula 3:

$$\forall i, j \in \mathbb{Z} \cdot subp(i, j) =_\exists i - j \vee subp(j, i) =_\exists j - i \tag{7}$$

is true in $FoPC_E$.

The aim of this paper is to establish precise "translations" between approaches. Our chosen comparison is between *LPF* with weak equality (referred to below as LPF_w) and *FoPC* with existential equality (referred to below as $FoPC_E$). We also show below that the way definitions of partial functions are "imported" into proofs influences how theorems are proved.

2 The Typed Logic of Partial Functions

LPF admits undefined logical terms. The key omission from the logic is the law of the excluded middle. Classically, natural deduction [Pra65] provides exactly one introduction and elimination rule for each connective; in *LPF* it is necessary to have, for example, introduction rules for both $\wedge$ and $\neg\wedge$. *LPF* offers the strongest monotonic extension of *FoPC* with respect to the following ordering on truth values: [1]

$$\bot_{\mathbb{B}} \preceq \textbf{true}, \bot_{\mathbb{B}} \preceq \textbf{false}$$

The operators in *LPF* have the (monotonicity) property that any undefined operands being "completed" to either **true** or **false** will not cause any result to change between **true** and **false** (of course, a $\bot_{\mathbb{B}}$ might itself complete).

An important aspect of LPF_w proofs is the way that function definitions are handled by translation to inference rules [JM94]; for example, the definition of *subp* gives rise to the following rules (using only weak equality): [2]

$$\boxed{subp\text{-}b_w}\ \frac{}{subp(i,i) = 0}\ \text{Ax}$$

$$\boxed{subp\text{-}i_w}\ \frac{i \neq j; subp(i, j+1) = k}{subp(i,j) = k+1}\ \text{Ax}$$

These rules are used in the proof of Formula 1 shown in Fig. 1.

[1] Although his logic also admits undefinedness, Blamey [Bla80,Bla86] uses the term "gaps" in preference to explicit references to an undefined Boolean value $\bot_{\mathbb{B}}$.
[2] Names of rules are given in boxes to the left. Strictly, the *subp*-b_w axiom should have a hypothesis asserting that 0 is defined.

3 The Relationship between LPF_w and $FoPC_E$

This section presents the correspondence between theorems of LPF_w and those of $FoPC_E$; both directions are discussed. Since the operators and quantifiers of the two logics are the same, the mapping actually concerns what happens with the relational operators (for brevity, we consider only equality).

3.1 Translated Theorems of LPF_w hold in $FoPC_E$

Consider first taking a theorem of LPF_w and rewriting each weak equality to an existential equality: one would then have a formula of $FoPC_E$. There are indications above that this mapped formula should be true in $FoPC$: Formulae 1 and 3 are examples (cf. Formulae 5 and 7).

The intuition for the generality of theoremhood of the mapped formulae is the monotonicity property mentioned in Section 2: replacing weak equality by existential equality effectively "completes" the values of undefined terms and doing so cannot cause the value of a formula to change from **true** to **false**. Thus a proposition that is a theorem of LPF_w (where the weak equalities give $\bot_{\mathbb{B}}$) will not become untrue if the existential equalities yield **false**.

The formal justification of the truth of mapped formulae is pleasingly simple (once one has seen it). Essentially, proofs in LPF_w can be replayed as proofs in $FoPC_E$ because the former logic is strictly weaker than the latter. The simple translation above (does not change the logical operators, but) maps each (weak) relational operator to its existential counterpart. [3] Since all inference rules of LPF are valid in $FoPC$, the steps of the proofs that rely on the axiomatisation of the logic do not need to change. The only issue that this leaves open is the way that LPF_w proofs import the definitions of (recursive) functions. Although the obvious way to reason about a recursive definition is by using the definition as though the definition symbol is a strong equality, it is true that the following rules are sound:

$$\boxed{\textit{subp-}b_E}\ \frac{}{subp(i,i) =_\exists 0}\ \text{Ax}$$

$$\boxed{\textit{subp-}i_E}\ \frac{i \neq j; subp(i,j+1) =_\exists k}{subp(i,j) =_\exists k+1}\ \text{Ax}$$

[3] In fact, there are many cases where it is obvious that the strict, weak, operators yield the same result so there is no need to translate all instances.

Fig. 1 presents a proof of Formula 1 in LPF_w, much as it was done in [CJ91]. The conjecture obtained by converting Formula 1 to $FoPC_E$ is in Formula 6. The steps of its (translated) proof differ from Fig. 1 only in that the classical deduction theorem does not need Step 7 as a hypothesis.

from $i, j: \mathbb{Z}$		
1	$i - 0 = i$	h, $\mathbb{Z}$
2	$subp(i, i) = 0$	h, $subp$-b_w
3	$subp(i, i - 0) = 0$	= -$subs$(1,2)
4	**from** $n: \mathbb{N}; subp(i, i - n) = n$	
4.1	$i - (n + 1) \in \mathbb{Z}$	h, h4, $\mathbb{Z}$
4.2	$i \neq i - (n + 1)$	h, h4, $\mathbb{Z}$
	infer $subp(i, i - (n + 1)) = n + 1$	h, 4.1, 4.2, h4, $subp$-i_w
5	$\forall n: \mathbb{N} \cdot subp(i, i - n) = n$	$\forall$-I($\mathbb{N}$-ind(3, 4))
6	**from** $i \geq j$	
6.1	$(i - j): \mathbb{N}$	$\mathbb{Z}$, h6
	infer $subp(i, j) = i - j$	$\forall$-E(5, 6.1), $\mathbb{Z}$
7	$\delta(i \geq j)$	h, $\mathbb{Z}$
infer	$i \geq j \ \Rightarrow \ subp(i, j) = i - j$	$\Rightarrow$ -I(6,7)

Fig. 1. Proof of Property 1 in LPF_w

3.2 Translating Results in $FoPC_E$ to LPF_w

Formula 4 suggests that the translation from $FoPC_E$ to LPF_w is more complicated in that not all theorems of $FoPC_E$ hold in LPF_w. We do not need the law of the excluded middle to illustrate the difficulty; the discussion can be made more compact by considering:

$$\neg\,(subp(0, 1) =_\exists 42)$$

Since $subp(0, 1)$ fails to denote and existential equality gives false when either operand is undefined, this formula is a truth of $FoPC_E$. Trivial replacement by weak equality leads to a formula of LPF_w that collapses to $\bot_{\mathbb{B}}$.[4]

If one maps terms of the form

$$t_1 =_\exists t_2$$

[4] This observation led us to consider a mapping that excluded "negative occurrences" of terms. However, the following approach is more general.

into ($\delta(t)$ gives **true** for defined terms and **false** for $\perp_{\mathbb{B}}$)

$$t_1 = t_2 \wedge \delta(t_1 = t_2)$$

this converts $FoPC_E$ to LPF_w. It is clear that this preserves truth because the semantics is identical. Although this mapping appears cumbersome [5], it often simplifies nicely. For example Formulae 1 and 3 are both effectively mapped bi-directionally: the first because the definedness clause is the same as the left of the implication; in the second, the two definedness criteria reduce to $i \leq j \vee i > j$.

One other observation is worth making. Whenever one encounters a formula with a negative occurrence of a weak operator such as equality, one should consider the inverse operator. It is important to remember that $\neg\,(a =_\exists b)$ does *not* have the same meaning as $a \neq_\exists b$.

4 Discussion

This paper has an interesting history: Jones' talk at AVoCS [Jon06] in 2005 included the examples in Formulae 1 and 3; in the ensuing discussion, Michael Goldsmith of Oxford asked whether there was an exact match between the theorems of *LPF* and those of *FoPC* with existential equality. Jones' first reaction was that there would be counterexamples. However, subsequent efforts found useful examples that supported Goldsmith's conjecture and –as so often– only the attempt to prove the connection clarified the exact conditions.

The straightforward translation from LPF_w to $FoPC_E$ is pleasing; the fact that the translation in the other direction is often straightforward is also encouraging. As indicated above, such studies are important if the move to interworking of formal methods tools is to be conducted soundly.

It is a mistake to get over-enthusiastic about our trick of "replaying proofs" just because other approaches also employ the axioms of *FoPC*. Unfortunately, the technique does not appear to generalise because of the restrictions that, for example, Z-logic has to apply to expressions that contain non-denoting terms.

Why not just use $FoPC_E$ for all proofs? As argued in [CJ91], there is the danger of confusion arising from the need to manage two or more notions of equality; furthermore, there is a serious problem with needing non-strict

[5] The mapping for strong equality is even worse because of the need to recognise the case where both terms yield $\perp$!

versions of all relational operators that "come between" potentially undefined terms and the logical operators.

One of the anonymous referees used by the journal drew attention to the works of Farmer and Guttman. Their way to handle undefinedness (cf. [FG00, Theorem 2.1]) is to work with predicate symbols that denote graphs of functions (e.g. $((i, j), k) \in subp$). This makes it possible to obtain a defined result for a potentially undefined application by translating to an expression with an existential quantifier. This approach is discussed in the earlier [CJ91]. Such a mapping would yield expressions that were less closely connected to the original expression than with our mapping; there must also be a danger that the existential quantifiers would be inconvenient for mechanical theorem provers.

Acknowledgments We are grateful to Michael Goldsmith for the question that kindled exploration in this area and to Jim Woodcock for stimulating discussions on the topic. We hope that the comments from the journal's anonymous referees have helped clarify our explanation. Our work was supported by the EPSRC Platform project on Trustworthy Ambient Systems (TrAmS), the EU FP7 DEPLOY and ReSIST Projects.

References

[Abr96] J.-R. Abrial. *The B-Book: Assigning programs to meanings.* Cambridge University Press, 1996.

[BCJ84] H. Barringer, J. H. Cheng, and C. B. Jones. A logic covering undefinedness in program proofs. *Acta Informatica*, 21:251–269, 1984.

[BFL+94] Juan Bicarregui, John Fitzgerald, Peter Lindsay, Richard Moore, and Brian Ritchie. *Proof in VDM: A Practitioner's Guide.* FACIT. Springer-Verlag, 1994. ISBN 3-540-19813-X.

[Bla80] S. R. Blamey. *Partial Valued Logic.* PhD thesis, Oxford University, 1980.

[Bla86] S. Blamey. Partial logic. In D. Gabbay and F. Guenthuer, editors, *Handbook of Philosophical Logic, Volume III*, chapter 1. Reidel, 1986.

[Che86] J. H. Cheng. *A Logic for Partial Functions.* PhD thesis, University of Manchester, 1986.

[CJ91] J. H. Cheng and C. B. Jones. On the usability of logics which handle partial functions. In C. Morgan and J. C. P. Woodcock, editors, *3rd Refinement Workshop*, pages 51–69. Springer-Verlag, 1991.

[FG00] W. M. Farmer and J. D. Guttman. A Set Theory with Support for Partial Functions. *Studia Logica*, 66(1):59–78, 2000.

[Fit07] J. S. Fitzgerald. The Typed Logic of Partial Functions and the Vienna Development Method. In D. Bjørner and M. C. Henson, editors, *Logics of Specification Languages*, EATCS Texts in Theoretical Computer Science, pages 427–461. Springer, 2007. To appear.

[FL98] John Fitzgerald and Peter Gorm Larsen. *Modelling systems: practical tools and techniques in software development.* Cambridge University Press, 1998.

[Hay93] Ian Hayes, editor. *Specification Case Studies.* Prentice Hall International, second edition, 1993.

[JJLM91] C. B. Jones, K. D. Jones, P. A. Lindsay, and R. Moore. *mural: A Formal Development Support System.* Springer-Verlag, 1991. ISBN 3-540-19651-X.

[JM94] C. B. Jones and C. A. Middelburg. A typed logic of partial functions reconstructed classically. *Acta Informatica*, 31(5):399–430, 1994.

[Jon90] C. B. Jones. *Systematic Software Development using VDM.* Prentice Hall International, second edition, 1990. ISBN 0-13-880733-7.

[Jon06] Cliff B. Jones. Reasoning About Partial Functions in the Formal Development of Programs. *Electronic Notes in Theoretical Computer Science*, 145:3–25, January 2006.

[JOW06] Cliff Jones, Peter O'Hearn, and Jim Woodcock. Verified software: a grand challenge. *IEEE Computer*, 39(4):93–95, 2006.

[Kle67] S. C. Kleene. *Mathematical Logic.* Wiley, 1967.

[Pra65] Dag Prawitz. *Natural Deduction: a Proof-Theoretical Study.* Dover publications, 1965.