The Connection between Two Ways of Reasoning about Partial Functions

J. S. Fitzgerald, C. B. Jones.

TECHNICAL REPORT SERIES

No. CS-TR-1044 August, 2007

TECHNICAL REPORT SERIES

No. CS-TR-1044 **August, 2007**

The Connection between Two Ways of Reasoning about Partial Functions

John S. Fitzgerald, Cliff B. Jones.

Abstract

This paper addresses the relationship between the theorems derived in two logics that provide alternative ways of reasoning about partial functions. Theorems in the Logic of Partial Functions using weak equality can be directly translated into First Order Predicate Calculus using existential equality. Translation in the other direction is, in general, more complicated but simplifies pleasingly in many cases. Such results are important if formal methods tool integration is to proceed safely.

Bibliographical details

FITZGERALD, J. S., JONES, C. B.

The Connection between Two Ways of Reasoning about Partial Functions
[By] J. S. Fitzgerald, C. B. Jones.

Newcastle upon Tyne: University of Newcastle upon Tyne: Computing Science, 2007.

(University of Newcastle upon Tyne, Computing Science, Technical Report Series, No. CS-TR-1044)

Added entries

UNIVERSITY OF NEWCASTLE UPON TYNE
Computing Science. Technical Report Series. CS-TR-1044

Abstract

This paper addresses the relationship between the theorems derived in two logics that provide alternative ways of reasoning about partial functions. Theorems in the Logic of Partial Functions using weak equality can be directly translated into First Order Predicate Calculus using existential equality. Translation in the other direction is, in general, more complicated but simplifies pleasingly in many cases. Such results are important if formal methods tool integration is to proceed safely.

About the author

John Fitzgerald is Reader in Computing Science at Newcastle University. His research addresses the use of formal methods in early stages of development for complex systems. He has pioneered the use of tool-supported formal modelling techniques in industrial practice through close collaborations, especially with the aerospace industry. He returned to academia following a period developing dynamic binary translation technology with Transitive Ltd and his work now focuses on user-guided validation of designs and the design of dynamic virtual organisations. He leads work on resilience-explicit computing in the ReSIST Network on Resilience in IST and, along with Cliff Jones and Alexander Romanovsky, leads the EPSRC platform project on Trustworthy Ambient Systems. John is Chairman of Formal Methods Europe.

Cliff Jones is currently Professor of Computing Science at Newcastle. He has spent more of his career in industry than academia. Fifteen years in IBM saw, among other things, the creation with colleagues of the Vienna Development Method. He went on to build the Formal Methods Group at Manchester University, which among other projects created the "mural" theorem proving assistant. A Senior Fellowship focused on formal (compositional) development methods for concurrent systems. In 1996 he moved to Harlequin directing some 50 developers on Information Management projects and finally became overall Technical Director before leaving to re-join academia in 1999. Cliff's interests in formal methods have now broadened to reflect wider issues of dependability. Cliff is a Fellow of the Royal Academy of Engineering, the ACM, BCS and IEE.

Suggested keywords

FORMAL METHODS,
SPECIFICATION LANGUAGES,
LOGIC,
PARTIAL FUNCTIONS,
LPF,
EQUALITY

The Connection between Two Ways of Reasoning about Partial Functions

John S. Fitzgerald, Cliff B. Jones
School of Computing Science, Newcastle University, UK

Abstract

This paper addresses the relationship between the theorems derived in two logics that provide alternative ways of reasoning about partial functions. Theorems in the Logic of Partial Functions using weak equality can be directly translated into First Order Predicate Calculus using existential equality. Translation in the other direction is, in general, more complicated but simplifies pleasingly in many cases. Such results are important if formal methods tool integration is to proceed safely.

1 Introduction

1.1 Partial Operators in Formal Specifications

Partial operators and functions are common in specifications and designs in formal languages such as Z [Hay93], B [Abr96] and VDM [Jon90, FL98]. Their application gives rise to potentially undefined terms. For example, the **hd** operator in VDM extracts the initial element of non-empty sequences, so the term $\mathbf{hd}\,[\,]$ is undefined. It is sometimes possible to "protect" applications of partial operators by guards, as in the following expression in which the non-strict conditional avoids the evaluation of a potentially non-denoting term:

$$\mathbf{if}\ s \neq [\,]\ \mathbf{then}\ \mathbf{hd}\, s\ \mathbf{else}\ \mathbf{nil}$$

However, plausible logical expressions can be written for which classical First-order Predicate Calculus (FoPC) does not define a value. For example, the following expression involves a potentially undefined mapping application:

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

FoPC does not define a result for the case where the lookup is undefined ($\mathbf{false} \wedge \bot_{\mathbb{B}}$ is undefined). Instead of using FoPC, VDM uses the Logic of Partial Functions [BCJ84] in which the value of $\mathbf{false} \wedge \bot_{\mathbb{B}}$ (and the commuted expression) is defined to be **false**.

Recursive functions offer a particular challenge for reasoning because their domains are not necessarily obvious. The following example from [CJ91] is used commonly because it is just difficult enough (e.g. no simple set over which to quantify) to illustrate plausible logical assertions whose status requires thought. The function *subp* is *deliberately* partial but returns $i - j$ provided $i \geq j$:

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

The following claim (in FoPC) appears plausible and seems to capture knowledge about *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 $0 \geq 1 \;\Rightarrow\; subp(0, 1) = 0 - j$ which, since $subp(0, 1)$ does not –in the least fixed point– denote an integer, comes down to **false** $\Rightarrow\; \perp_{\mathbb{Z}} = -1$.

For Claim 1, it is tempting to read the left side as a sort of "guard" but a standard property of propositional calculus 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.

An even more problematic claim is that

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

where there is no guarding clause. Both Claims 1 and 3 are true in LPF 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 many approaches other than LPF that attempt to tame undefined terms. A categorization of approaches is given in [CJ91] by analysing where undefined values are "caught"; advantages and disadvantages of the approaches are also listed. In several of these approaches, Claim 1 may only be proved with side conditions. In even more approaches, Claim 3 would be ruled out. There is obviously an intellectual interest in understanding the relationships between approaches. Furthermore, a pressing practical issue arises now that serious consideration is being given to moving conjectures between theorem proving tools.

As well as the axiomatisation of the predicate calculus itself, the question of the form of equality that best fits a particular logic is important: so we first explore these.

1.2 Notions of Equality

One way of avoiding handling undefinedness in the logic is to provide versions of relational operators such as equality that absorb undefined terms. For example, existential equality ($=_\exists$) is a non-strict equality that yields false if *any* operand is undefined. In the *subp* example, a version of Claim 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 poses no problems in FoPC because, if $i < j$, the antecedent is false, the $subp(i,j)$ term is undefined and so the consequent is false. 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* can be retained.

Another non-strict relation is strong equality ($==$) which differs from its existential cousin only in that $\perp_{\mathbb{Z}} == \perp_{\mathbb{Z}}$ is true. The following transliterated version of Claim 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.)

1.3 A Challenge

As indicated above, Claim 1 is true in FoPC if the equality operator is changed to existential equality as in Claim 5; and even the following:

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

is true in $FoPC_E$[1].

The main outcome 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 $FoPC_E$). Other choices are possible but these pairings of logics with equality notions appear to work well together. We also show below that how theorems are proved also depends on how the definitions of partial functions are "imported" into proofs.

One might ask why not just use $FoPC_E$. Apart from the danger of confusion arising from the need to manage two or more notions of equality, there is a serious problem with needing non-strict versions of all operators that "come between" undefined terms and the logical operators. We return to this point in Section 3.

[1]Following the second author's talk at AVoCS in Warwick on September 12th 2005 [Jon06], these examples led Michael Goldsmith of Oxford to ask whether there was an exact match between theorems of LPF with weak equality and theorems of FoPC with existential equality. Jones' first reaction was that there was not a match. 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.

2 The Typed Logic of Partial Functions

There is a body of work on the Logic of Partial Functions [Che86] and its typed form [JM94], its mechanization in a proof support environment [JJLM91] and its use as a foundation for reasoning about models and refinements in VDM [BFL$^+$94, Jon06, Fit07]. A complete description is not provided here (the references provide a fuller history linking LPF into mainstream writings on logic).

LPF admits undefined logical terms. The key omission from the logic is the law of the Excluded Middle. Classically, natural deduction provides exactly one introduction and elimination rule for each connective [Pra65]; in LPF it is necessary to have, for example, introduction rules for both $\wedge$ and $\neg\wedge$. Although the logic admits undefinedness, Blamey uses the term "gaps" in the logical values in preference to explicit references to $\bot_{\mathbb{B}}$ [Bla80, Bla86].

LPF offers the strongest monotonic extension of *FoPC* with respect to the following ordering on truth values:

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

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

An important facet of LPF_w proofs is the way that function definitions are handled by translation to inference rules; for example, the definition of *subp* gives rise to the following rules:[2]

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

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

These rules are used in the proof of Claim 1 in Figure 1.

3 The Relationship between LPF_w and $FoPC_E$

This section considers the correspondence the theorems of LPF_w with those of $FoPC_E$; both directions are discussed.

3.1 Theorems of LPF_w hold in $FoPC_E$

There are indications above that the truths of LPF_w map to $FoPC_E$: Claims 1 and 3 are examples (cf. Formulae 5 and 7; Claim 2 could also be mapped). The intuition for the generality of this link is the monotonicity property mentioned in

[2]Names of rules are given in boxes to the left. An "Ax" to the right indicates an axiom. Strictly, the *subp*-b_w axiom should have a hypothesis asserting that 0 is defined.

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 which is a theorem of LPF_w will never become untrue if the existential equalities yield **false** where the weak equalities they replace gave $\bot_{\mathbb{B}}$.

The formal justification of the mapping 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 for the statements of the theorems maps each (weak) Boolean operator to its existential counterpart. (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 such operators between defined terms.) 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 open is the way that LPF_w proofs import the definition of (recursive) functions. Although the obvious way to use the recursive definition is actually by using the definition as though the definition symbol is a strong equality, it is true that the following rules are sound:

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

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

For the *subp* example, Fig. 1 presents a proof of Claim 1 in LPF_w, much as it was done in [CJ91]. The conjecture obtained by converting Claim 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 the Step 7 as a hypothesis.

3.2 Translating Results in $FoPC_E$ to LPF_w

Formula 4 provides a warning example that not all theorems of $FoPC_E$ hold in LPF_w but we do not actually need the law of the excluded middle to illustrate the difficulty. The discussion can be made more compact by considering:

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

Since $\textit{subp}(0,1)$ fails to denote, and existential equality gives false with either operand undefined, this is a truth of $FoPC_E$ that clearly cannot be trivially translated into a weak equality (which itself collapses to $\bot_{\mathbb{B}}$).[3]

A correct mapping of $FoPC_E$ is to translate

$t_1 =_\exists t_2$

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

	from $i, j{:}\,\mathbb{Z}$	
1	$i - 0 = i$	h, $\mathbb{Z}$
2	$subp(i, i) = 0$	h, $subp$-b
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
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)

Figure 1: Proof of Property 1 in LPF_w

into

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

In fact, this rather gruesome[4] mapping often simplifies out nicely. For example our Claims 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 (say equality), one should consider the inverse operator. It is important to remember that $\neg\,(a =_\exists b)$ is not the same as $a \neq_\exists b$.

4 Conclusions

The simple link between the theorems of LPF_w and $FoPC_E$ 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 integration of formal methods tools is to be conducted soundly.

It is tempting to get over enthusiastic about our trick of "replaying proofs" since other approaches also employ the axioms of $FoPC$. Unfortunately, the technique does not appear to generalize because of the restrictions that say Z-logic has to apply to expressions that contain non-denoting terms.

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

Acknowledgments

The authors are grateful to Michael Goldsmith for the question that led us to the conjecture that sparked explorations in this area and to Jim Woodcock for stimulating discussions on the topic. The work was supported by the EPSRC Platform project on Trustworthy Ambient Systems and EU FP6 RODIN project and ReSIST Network of Excellence.

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.

[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.

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