THE FOUNDATIONS OF SPECIFICATION I

Raymond Turner
The University of Essex

February 13, 2004

Abstract

We develop a *Core Specification Theory* (**CST**) as a basis for the mathematical investigation of specification and specification languages.

1 Specification Theories

Specification languages such as **Z**, **VDM**, **B**, **PVS** ([28], [13], [21], [1], [7], [24], [25]) are based upon the Predicate Calculus (**PC**) in the sense that the language in which the specifications are expressed is some version or dialect of **PC**. However, they differ from the simple *single sorted* versions of **PC** in that they distinguish between different *types* of data where the *types* of these theories are usually presented in an inductive fashion: there are basic types (e.g. *numbers, characters*) together with a battery of type constructors such as *products*, *sets and recursive types*. Furthermore, presentations of specification languages ([20], [12], [15], [16], [9], [10], [22], [23], [27], [4], [15], [19], [3], [5], [8], [12], [18], [24]) sometimes include groups of axioms for the various types and their associated relation and function symbols. Typically these stipulate the membership conditions for the type, the criteria for two elements of the type to be equal and lay out the properties of any special relations and functions for the type. As such these axiomatic systems constitute *theories* of the underlying concepts of the language that we shall refer to as *Specification Theories*.

However, it is not an easy task to provide such axiomatisations for existing languages ([4], [15], [9], [10]). This is largely due to the way in which they have evolved: their form has been largely determined only by practical considerations. In practice, standard logical systems such as **HOL** or **ZF** have been lifted wholesale and employed as the host for a massive infrastructure of *syntactic sugar* dictated by practical needs. Unfortunately, this *sugaring* has taken over and obscured the logical core. Subsequently, we have languages for which it virtually impossible to provide mathematically tractable and usable axiomatisations. In particular, the axiom systems are often very large: a stab is made at providing axioms for every construct that might prove useful. Even so, the systems are often incomplete in the sense that some of the constructs

are not axiomatised. Indeed, some of the more rigorous attempts at developing axiomatisations for existing languages ([9], [10]) have been forced to part company with the target language for technical and conceptual reasons. More generally, many problems with current languages often only come to the fore when axiomatisations are attempted.

Indeed, current specification languages are rarely precisely and completely formulated as axiomatic theories and so are inadequate for metamathematical purposes; it is very hard to treat the theory as an object of study in its own right. As a result, our mathematical grasp of specification and specification languages is quite meagre.

This is the first of a series of papers which seek to address this foundational gap. Our objective in this first paper is to formulate and study a core *Specification Theory* and use it to explore the specification process. This theory is a sub-theory of the implicit theories of all the major specification languages; it is buried inside them even if it is not evidently so. Nevertheless, it is expressive enough to illustrate the different styles of specification employed by these languages and to explore the logical foundations of the actual process of specification.

2 A Core Specification Theory

We present a core specification theory (**CST**) which is a fragment of most, if not all, the major specification languages. We shall do so in several stages. Initially, we present the language, and compared with actual specification languages it is very small. We then develop the logic of the system: a version of a typed predicate logic. Finally, we present the rules and axioms for the various types.

2.1 The Syntax of CST

The language has three syntactic categories: *wff*, *types* and *terms*. We deal first with the syntax for the types since these drive the form of the language. The atomic type terms consist of type variables and the type constant N, the *natural number* type. There are two type constructors which permit the formation of *sets* (*Set)* and *Cartesian products* ($\otimes$). More formally, the syntax of type terms is given as follows.

$$T ::= X \mid N \mid T \otimes T \mid Set(T)$$

These types are taken as basic in both Z and **VDM**. Generally, we shall employ upper case Roman letters A, B, C, D,... for type terms with U, V, W, X, Y, Z reserved for type variables.

With these go the following individual term constants and function symbols. Apart from the individual variables we admit, for the natural numbers, the constant zero (0) and the numerical successor function ($^+$) and for Cartesian products, we include the pairing function () and the selection functions (π_i). Finally, sets are supported by a constant for the empty set ($\emptyset$) and a binary

insertion function ($\circledast$) for adding an element to a set. This leads to the following syntax for terms.

$$\begin{array}{rl} t ::= & x \mid 0 \mid t^+ \\ & (t, t) \mid \pi_1(t) \mid \pi_2(t) \\ & \emptyset \mid t \circledast t \end{array}$$

where generally we employ lower case Roman letters $a, b, c, ...$ for individual terms with u, v, w, x, y, z reserved for term variables. The basic operators of the theory are *polymorphic.* In particular, the set insertion function and the pairing operation operate globally over all types.

Finally, we introduce the well-formed formula (wff). We employ lower case Greek letters for these. The atomic wff include absurdity (Ω); equality, set membership ($\in$) and the ordering relation on the natural numbers ($<$). General wff are generated from these by the propositional connectives and the quantifiers.

$$\begin{array}{rl} \phi ::= & \Omega \mid t = t \mid t \in t \mid t < t \\ & \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \phi \rightarrow \phi \\ & \forall x < t \cdot \phi \mid \exists x < t \cdot \phi \\ & \forall x \in t \cdot \phi \mid \exists x \in t \cdot \phi \\ & \forall x : T \cdot \phi \mid \exists x : T \cdot \phi \\ & \forall X \cdot \phi \mid \exists X \cdot \phi \end{array}$$

Apart from the numerical and set quantifiers, we have quantification with respect to a given type and quantification over types. The last four are the main logical quantifiers of the theory and will be governed by standard introduction and elimination rules. One might think that the bounded quantifiers should be defined in terms of the others but, for theoretical and practical purposes, it is convenient to take them all as primitive. However, their properties will be stated in terms of the main quantifiers.

Where e is a term or wff, we shall write $FV(e)$ for the collection of free individual variables of e and $FTV(e)$ for the free type variables. For the purposes of substitution, we shall write $e[x_1, .., x_n]$ to mark free individual variables. This notation is not to be taken to imply all of the variables $x_1, .., x_n$ occur free in e nor that they exhaust all the free variables of e. We shall write $e[t_1, .., t_n/x_1, .., x_n]$ for the substitution of a terms t_i for the variables x_i. Similarly, we shall write $e[X_1, .., X_n]$ to mark type variables and $e[T_1, .., T_n/X_1, .., X_n]$ for type substitution. Finally, note that individual terms do not contain type variables and type terms do not contain individual variables.

We next present a few preliminary definitions. Propositional equivalence ($\leftrightarrow$) is defined in terms of implication in the standard way. We define *type membership* and of some other useful forms of quantification, as follows.

$$\begin{array}{rcl} t : X & \triangleq & \exists x : X \cdot t = x \\ \exists! x : X \cdot \phi[x] & \triangleq & \exists x : X \cdot \phi[x] \wedge \forall y : X \cdot \phi[y] \rightarrow x = y \\ \exists^{\leq 1} x : X \cdot \phi[x] & \triangleq & \forall x : X \cdot \forall y : X \cdot \phi[x] \wedge \phi[y] \rightarrow x = y \end{array}$$

This completes all the syntactic preliminaries.

2.2 The Logic

The logic is presented in a sequent-style natural deduction format. The rules are given relative to a *context* Γ which is a (possibly empty) finite set of wff. Sequents thus take the form:

$$\Gamma \vdash_{\mathbf{CST}} \phi$$

which is to be understood as asserting that in the theory **CST**, ϕ follows from Γ. We shall usually drop the suffix. Furthermore, we shall only include the contexts of a rule where they are modified in passing from the premises to the conclusion. We shall also write rules with no premises in the standard way.

There are two *structural* rules: an *assumption* axiom and a *weakening* rule.

$$\mathbf{Ax} \quad \phi \vdash \phi \qquad\qquad \mathbf{W} \quad \frac{\Gamma \vdash \psi}{\Gamma, \phi \vdash \psi}$$

There are the two standard equality axioms - adapted to a typed setting.

$$\mathbf{E_1} \quad \forall x : X \cdot x = x$$

$$\mathbf{E_2} \quad \forall x : X \cdot \forall y : X \cdot x = y \to (\phi[x] \to \phi[y])$$

The *logical* rules are the normal classical ones. We begin with the propositional connectives.

$$\mathbf{L_1} \ \frac{\neg\phi \qquad \phi}{\Omega} \qquad \mathbf{L_2} \ \frac{\Omega}{\phi}$$

$$\mathbf{L_3} \ \frac{\Gamma, \phi \vdash \Omega}{\Gamma \vdash \neg\phi} \qquad \mathbf{L_4} \ \frac{\Gamma, \neg\phi \vdash \Omega}{\Gamma \vdash \phi}$$

$$\mathbf{L_5} \ \frac{\phi \qquad \psi}{\phi \wedge \psi} \qquad \mathbf{L_6} \ \frac{\phi \wedge \psi}{\phi} \qquad \mathbf{L_7} \ \frac{\phi \wedge \psi}{\psi}$$

$$\mathbf{L_8} \ \frac{\phi}{\phi \vee \psi} \qquad \mathbf{L_9} \ \frac{\psi}{\phi \vee \psi} \qquad \mathbf{L_{10}} \ \frac{\Gamma \vdash \phi \vee \psi \qquad \Gamma, \phi \vdash \eta \qquad \Gamma, \psi \vdash \eta}{\Gamma \vdash \eta}$$

$$\mathbf{L_{11}} \ \frac{\Gamma, \phi \vdash \psi}{\Gamma \vdash \phi \to \psi} \qquad \mathbf{L_{12}} \ \frac{\phi \to \psi \qquad \phi}{\psi}$$

The main quantifier rules are for the typed and type quantifiers. These are standard and subject to the normal side conditions about dependency.

$$\mathbf{L_{13}}\ \frac{\Gamma, x : X \vdash \phi}{\Gamma \vdash \forall x : X \cdot \phi} \qquad \mathbf{L_{14}}\ \frac{\forall x : X \cdot \phi \qquad t : X}{\phi[t/x]}$$

$$\mathbf{L_{15}}\ \frac{\phi[t/x] \qquad t : X}{\exists x : X \cdot \phi} \qquad \mathbf{L_{16}}\ \frac{\Gamma \vdash \exists x : X \cdot \phi \qquad \Gamma, x : X, \phi \vdash \eta}{\Gamma \vdash \eta}$$

$$\mathbf{L_{17}}\ \frac{\phi}{\forall X \cdot \phi} \qquad \mathbf{L_{18}}\ \frac{\forall X \cdot \phi}{\phi[T/X]}$$

$$\mathbf{L_{19}}\ \frac{\phi[T/X]}{\exists X \cdot \phi} \qquad \mathbf{L_{20}}\ \frac{\Gamma \vdash \exists X \cdot \phi \qquad \Gamma, \phi \vdash \eta}{\Gamma \vdash \eta}$$

We shall deal with the bounded quantifiers in connection with their types. This completes the basic logic of the theory. We can now deal with the types themselves: for each we shall provide introduction, elimination and special equality rules together with any rules for their special relations and functions.

2.3 Natural Numbers

The first group of axioms for the numbers are those of *Peano* Arithmetic but with explicit quantifiers to restrict them to numbers. The first four inform us about the successor relation and the fifth is the standard scheme of induction.

$$\mathbf{N_1} \qquad 0 : N$$

$$\mathbf{N_2} \qquad \forall x : N \cdot x^+ : N$$

$$\mathbf{N_3} \qquad \forall x : N \cdot x^+ \neq 0$$

$$\mathbf{N_4} \qquad \forall x : N \cdot \forall y : N \cdot x^+ = y^+ \to x = y$$

$$\mathbf{N_5} \qquad (\phi[0] \land \forall x : N \cdot \phi[x] \to \phi[x^+]) \to \forall x : N \cdot \phi[x]$$

The next group provide the axioms for the ordering relation. Again, they are the standard axioms adapted to fit the present typed framework.

$$\mathbf{N_6} \qquad \forall x : N \cdot \neg(x < 0)$$

$$\mathbf{N_7} \qquad \forall y : N \cdot \forall x : N \cdot x < y^+ \leftrightarrow (x < y \lor x = y)$$

Finally, we deal with the numerical quantifiers. These are governed by the following axioms.

$$\mathbf{N_8} \qquad \forall y : N \cdot (\forall x < y \cdot \phi) \leftrightarrow (\forall x : N \cdot x < y \to \phi)$$

$$\mathbf{N_9} \qquad \forall y : N \cdot (\exists x < y \cdot \phi) \leftrightarrow (\exists x : N \cdot x < y \land \phi)$$

They insist that, in the context where the bound is a number, they can be unpacked in terms of quantification with respect to the natural number type. This style of axiom, where a construct is only provided a meaning in a given context, will form the basis for our general scheme of relation and function specifications. This completes the numerical axioms and rules.

2.4 Cartesian Products

Cartesian products are present in most specification languages and the axioms are the usual ones. The first three are the normal axioms for pairs and selections functions.

$$\begin{aligned}
&\mathbf{P_1} \quad \forall x : X \cdot \forall y : Y \cdot (x, y) : X \otimes Y \\
&\mathbf{P_2} \quad \forall z : X \otimes Y \cdot \pi_1(z) : X \\
&\mathbf{P}_3 \quad \forall z : X \otimes Y \cdot \pi_2(z) : Y
\end{aligned}$$

Finally, the special equality axioms demand that the selection functions behave appropriately on pairs and support surjective pairing.

$$\begin{aligned}
&\mathbf{P_4} \quad \forall x : X \cdot \forall y : Y \cdot \pi_1(x, y) = x \wedge \pi_2(x, y) = y \\
&\mathbf{P_5} \quad \forall z : X \otimes Y \cdot z = (\pi_1(z), \pi_2(z))
\end{aligned}$$

The construction can be iterated to the product of more than 2 types via

$$X_1 \otimes (X_2 \otimes ..... \otimes X_{n+1})$$

In particular, we shall write X^n for $X \otimes X \otimes ..... \otimes X$ i.e. n-copies of A. We shall often write $\pi_i(x)$ as x_i.

2.5 Sets

While it is more central in some than in others, this type constructor is present in some form in most specification languages and certainly in all the major logic-based languages. We present the axioms/rules in several waves.

The first group parallel the Peano axioms. The first pair state the closure conditions for the type: the empty set is a member of every type of sets and the sets of a given type are closed under element insertion. The next two ban the multiplicity of elements in sets and guarantee order independence. The final axiom in the group is the induction principle for sets.

$$\begin{aligned}
&\mathbf{S_1} \quad \emptyset : Set(X) \\
&\mathbf{S_2} \quad \forall x : X \cdot \forall y : Set(X) \cdot x \circledast y : Set(X) \\
&\mathbf{S_3} \quad \forall x : X \cdot \forall y : Set(X) \cdot x \circledast (x \circledast y) = x \circledast y \\
&\mathbf{S}_4 \quad \forall x : X \cdot \forall y : X \cdot \forall z : Set(X) \cdot x \circledast (y \circledast z) = y \circledast (x \circledast z) \\
&\mathbf{S_5} \quad (\phi[\emptyset] \wedge \forall x : X \cdot \forall y : Set(X) \cdot \phi[y] \rightarrow \phi[x \circledast y]) \rightarrow \forall y : Set(X) \cdot \phi[y]
\end{aligned}$$

The next pair govern the special relation symbol for this type, namely *set membership*. The first insists that the empty set has no elements and the second demands that the insertion function adds a single element to an existing set.

$$\begin{aligned}
&\mathbf{S_6} \quad \forall x : X \cdot x \notin \emptyset \\
&\mathbf{S_7} \quad \forall y : Set(X) \cdot \forall x : X \cdot \forall z : X \cdot z \in x \circledast y \leftrightarrow (z = x \vee z \in y)
\end{aligned}$$

Finally, we provide the *set quantifier* axioms. They mirror the numerical ones and insist that where the bound is a set they can be unpacked in terms of the main type quantifier.

$$\mathbf{S_8} \quad \forall y : Set(X) \cdot (\forall x \in y \cdot \phi) \leftrightarrow (\forall x : X \cdot x \in y \rightarrow \phi)$$
$$\mathbf{S_9} \quad \forall y : Set(X) \cdot (\exists x \in y \cdot \phi) \leftrightarrow (\exists x : X \cdot x \in y \wedge \phi)$$

This completes the statement of the theory **CST**. It is a very minimal theory of *numbers, sets and products* with very little meat on it. On the other hand, it is a highly expressive theory which supports a large portion of everyday specification. In the next paper we shall put it to work in exploring the foundations of the specification process. In the rest of this paper we shall explore the theory.

2.6 First Steps

We establish a few preliminary properties of the theory. The first couple present some elementary properties of the numbers.

Proposition 1 *The following are provable.*

1. $\forall y : N \cdot \forall x < y \cdot x : N$
2. $\forall y : N \cdot y = 0 \vee \exists u : N \cdot y = u^+$

Proof. The first follows from N_8 and the second by numerical induction with the induction wff

$$\phi[y] = y = 0 \vee \exists u : N \cdot y = u^+$$

■

We now do much the same for sets but here there are a few more obvious things to say.

Proposition 2 *The following are provable*

1. $\forall y : Set(X) \cdot \forall x \in y \cdot x : X$
2. $\forall y : Set(X) \cdot y = \emptyset \vee \exists u : X \cdot \exists v : Set(X) \cdot y = u \circledast v$
3. $\forall y : Set(X) \cdot \forall x : X \cdot x \circledast y \neq \emptyset$
4. $\forall y : Set(X) \cdot \forall x \in y \cdot x \circledast y = y$
5. $\forall z : Set(X) \cdot \forall x \in z \cdot \exists y : Set(X) \cdot x \notin y \wedge z = x \circledast y$
6. $\forall x \in Set(X) \cdot (\forall y \in x \cdot y : Y) \rightarrow x : Set(Y)$

Proof. The first follows from S_8. The rest employ the obvious set inductions. For example, for (2), we use set induction with the induction wff:

$$\phi[y] = y = \emptyset \vee \exists u : X \cdot \exists v : Set(X) \cdot y = u \circledast v$$

■

We can now establish the most important property of the *sets* of the theory namely their *extensional* nature. We first define *Extensional Equality* for sets as follows.

$$x \equiv y \triangleq \forall u \in x \cdot u \in y \wedge \forall v \in y \cdot v \in x$$

Proposition 3 *(Extensionality)*

$$\forall x : Set(X) \cdot \forall y : Set(X) \cdot x \equiv y \rightarrow x = y$$

Proof. Let $x : Set(X)$. We employ induction on y with the induction wff:

$$\phi[y] = x \equiv y \rightarrow x = y$$

Assume $y = \emptyset$. If $x = \emptyset$ we are finished by E_1. But if $x \neq \emptyset$ then $x = u \circledast v$ for some u, v which is impossible. This completes the base case. So assume that $y = u \circledast v$. We have to show that

$$x \equiv u \circledast v \rightarrow x = u \circledast v$$

If $u \in v$ we are finished by induction. If $u \notin v$, since $u \in x$, by the last proposition (part 5), $x = u \circledast v'$ for some set v', $u \notin v'$. Since $u \circledast v \equiv u \circledast v'$ and $u \notin v$ and $u \notin v'$, it follows that $v \equiv v'$. By induction, $v = v'$. So $u \circledast v = u \circledast v'$.■

This completes our basic introduction to the theory **CST**. It should be clear that this theory is a sub-theory of both the implicit theories of **VDM** and Z - and indeed all the major languages. One observation worthy of note is that the types of the theory are genuine *data types* in the sense that they could be implemented as types in a programming language. This should be seen in contrast to the types of the major specification languages.

3 Relation Specification

Implicit in all *logical* specification languages is the notion that specifications involve the introduction of new *relation* and *function* symbols. Furthermore, most languages allow a style of specification in which new *polymorphic* or *generic* relations and functions can be introduced. However, different specification languages present specifications in different syntactic forms. In particular, some (e.g. Z) are predominately relational in their style of specification and others (e.g. **VDM**) are more functionally inclined. We shall consider both styles. Indeed, we shall provide, within the formal framework provided by **CST**, a uniform treatment of, and a logical foundations for, both Z and VDM specification styles.

3.1 Schema

Our style of relation specification is based upon the Z *schema* notation: it introduces new *Polymorphic* relation symbols into the language via the following specification format.

Definition 4 *Let ϕ be any wff and $A_1, ..., A_k$ any type terms where $x_1, ..., x_k$ $(n, k \geq 0)$ are all the free individual variables of ϕ and where $X_1, ..., X_n$ include and exhaust all the type variables of $\phi, A_1, .., A_k$. We may then introduce a new relation symbol into the language via*

$$R[X_1, .., X_n] \triangleq [x_1 : A_1, .., x_k : A_k \mid \phi] \qquad (\mathbf{S})$$

*where each free individual variable is assigned exactly one type. We shall call these **Schema Specifications**. The type prefix*

$$x_1 : A_1, ..., x_k : A_k$$

*we call the **Declaration** of the schema and the wff ϕ its **Predicate**.*

How are these specifications to be unpacked *logically?* Here we shall be guided by the form of the axioms of our theory **CST**: **S** is to be understood as the introduction of a new relation symbol R that satisfies the following axiomatic condition.

$$\forall X_1 \cdot .. \cdot \forall X_n \cdot \forall x_1 : A_1 \cdot .. \cdot \forall x_k : A_k \cdot R[X_1, .., X_n](x_1, .., x_k) \leftrightarrow \phi \qquad (\mathbf{Rel})$$

This implicitly extends the syntax of wff to include new atomic wff of the form $R[T_1, .., T_n](t_1, .., t_k)$.

For the rest of this section we shall employ this notion of specification to enrich the theory. Indeed, the development of the theory will furnish us with material to illustrate the whole specification process.

Example 5 *A schema specification of the **Subset** relation is given as follows*

$$\subseteq_X \triangleq [x : Set(X), y : Set(X) \mid \forall z \in x \cdot z \in y]$$

which is written in its standard infix notation with the type variable as a subscript. Under the government of Rel, this is interpreted as the introduction of a new relation which satisfies

$$\forall X \cdot \forall x : Set(X) \cdot \forall y : Set(X) \cdot x \subseteq_X y \leftrightarrow \forall z \in x \cdot z \in y$$

The following are instances.

$$\subseteq_N \triangleq [x : Set(N), y : Set(N) \mid \forall z \in x \cdot z \in y]$$
$$\subseteq_{N^2} \triangleq \left[x : Set(N^2), y : Set(N^2) \mid \forall z \in x \cdot z \in y\right]$$

Example 6 *With subset in place we can specify a generic version of* ***Extensional Equivalence*** *for sets as follows*

$$\equiv_X \triangleq [x : Set(X), y : Set(X) \mid x \subseteq_X y \wedge y \subseteq_X x]$$

Example 7 *The following provides the specification of the* ***Pairing*** *relation on sets.*

$$Pair_X \triangleq [x : X, y : X, z : Set(X) \mid \forall u : X \cdot u \in z \leftrightarrow u = x \vee u = y]$$

Of course, just positing a relation is not the end of the story. We might for instance wish to show that the relation is not vacuous. In most cases this will be obvious, but we shall often investigate matters more thoroughly.

Following Z, (e.g. [28]) we shall also write schema in the more graphic form:

$R[X_1, ..., X_n]$

$x_1 : T_1, ..., x_k : T_k$

ϕ

In this presentation we shall often mark conjunctions with a new line. The following examples illustrate this.

Example 8 *The following is a schema specification of simple set theoretic* ***Union****.*

$Union[X]$

$u : Set(X), v : Set(X), w : Set(X)$

$\forall x : X \cdot x \in w \leftrightarrow x \in u \vee x \in v$

This is clearly non-vacuous since choosing all three sets to be the empty set provides an instance.

Example 9 *The following provides the definition of* ***Generalised Union***

$Genunion[X]$

$u : Set(Set(X)), v : Set(X)$

$\forall x : X \cdot x \in v \leftrightarrow \exists z \in u \cdot x \in z$

Example 10 *Given the specification of subset, the* ***Power set*** *relation may be specified as follows.*

$$\begin{array}{|l} Pow[X] \\ \hline u : Set(X), v : Set(Set(X)) \\ \hline \forall x : Set(X) \cdot x \in v \leftrightarrow x \subseteq_X u \\ \hline \end{array}$$

Observe that there is an instance: chose both to be the empty set.

We now come to the way of forming sets given by a scheme of *separation*. Notice that this operation is schematic (in the standard sense) with respect to a wff i.e. we introduce a new relation symbol for each wff.

Example 11 *Let ψ contain at most z free. We then specify*

$$\begin{array}{|l} Sep_\psi[X] \\ \hline u : Set(X), v : Set(X) \\ \hline \forall z : X \cdot z \in v \leftrightarrow z \in u \wedge \psi[z] \\ \hline \end{array}$$

Example 12 *The following provide specifications of the* ***Domain*** *and* ***Range*** *of set-theoretic relations.*

$$\begin{array}{|l} Dom[X, Y] \\ \hline u : Set(X \otimes Y), v : Set(X) \\ \hline \forall x : X \cdot x \in v \leftrightarrow \exists y : Y \cdot (x, y) \in u \\ \hline \end{array}$$

$$\begin{array}{|l} Ran[X, Y] \\ \hline u : Set(X \otimes Y), v : Set(X) \\ \hline \forall y : Y \cdot y \in v \leftrightarrow \exists x : X \cdot (x, y) \in u \\ \hline \end{array}$$

Although there is no type of *Maps* in the present theory, we can specify the relation of being a *Map* from one type to a second i.e. a *set theoretic relation* (an element of $Set(X \otimes Y)$) which is *single-valued*.

Example 13

$$Map[X, Y] \triangleq [z : Set(X \otimes Y) \mid \forall x \in z \cdot \forall y \in z \cdot x_1 = y_1 \rightarrow x_2 = y_2]$$

The next example is slightly different in that it is an example of how one can build new specifications uniformly from old ones. We shall have more to say about this in a later publication where we introduce our interpretation of the *Schema Calculus*.

Definition 14 *Let*

$$R[X_1, .., X_n] \triangleq [x : I, y : O \mid \psi]$$

Define the ***Domain*** *and* ***Range*** *of R as follows.*

$$\begin{aligned} Dom_R[X_1, .., X_n] &= [x : I \mid \exists y : O \cdot \psi] \\ Ran_R[X_1, .., X_n] &= [y : O \mid \exists x : I \cdot \psi] \end{aligned}$$

We shall study these and related examples in some detail since they will be employed to illustrate the whole process of specification. Moreover, many of them will provide some of the central infrastructure for the development of the theory as a more realistic specification language.

Operations express relationships between named *inputs* and *outputs*. However, the specification of an *operation* will have the same form as a schema definition but some of the variables will be interpreted as *inputs* and some as *outputs*. However, apart from the binary case of relations, where the convention is that the first argument is the input and the second the output, we have no convention to determine which is which. We shall often use Cartesian product types to reduce matters to the binary case and then employ this convention. However, in practical applications of the theory, this often proves inconvenient and so we need some more general notational devices to distinguish inputs from outputs. We shall adopt several styles of convention. We shall often just stick the inputs on the first line of the declaration and the outputs on the second. More general Z style conventions employ decorations, e.g. mark inputs with ? and outputs with !. Where there is no danger of ambiguity we shall usually drop the decoration in the predicate.

Example 15 *We may introduce* ***Generalised Intersection*** *as follows.*

$\cap[X]$

$$u? : Set(Set(X)), v! : Set(X)$$

$$\forall x : X \cdot x \in v \leftrightarrow \forall w \in u \cdot x \in w$$

This is our *interpretation* of the Z *schema notation*. However, we are not claiming that it is the official one; we are merely borrowing the schema notation as a convenient way of expressing our style of polymorphic definition. Indeed, there are some obvious differences. Firstly, we have explicitly interpreted schema to herald the axiomatic introduction of new relation symbols. This perspective is not explicitly adopted in the accounts of Z. Secondly, in the more formal accounts of Z given in the various standardization documents ([4], [20]) and the various logics of Z ([27], [9], [10]) it appears that a schema is taken to imply the type of its arguments. In the present setting this amounts to the introduction of a new relation via the following axiomatic stipulation.

$$\forall X_1 \cdot .. \cdot \forall X_n \cdot R[X_1, .., X_n](x_1, .., x_k) \leftrightarrow x_1 : A_1 \wedge .. \wedge x_k : A_k \wedge \phi$$

However, within the present theory, this is not a significant difference since the declaration will always form part of the proof context within which reasoning about the specification will be carried out. This perspective is the one adopted in the statement of the theory.

3.2 Conservative Extensions

Every time we introduce a new relational symbol via S we enrich the language and the theory. Moreover, we do so by introducing a new *axiom* into the theory. However, we require that such relational enrichment does not substantially change the theory. This would be unfortunate since then we would have no guarantee that the properties of the old theory, which we may rely on during specification, are maintained. Indeed, we shall use the word *Legitimate* to describe such conservative additions. This will mean different things within different styles of specification.

Suppose that we have extended the language of **CST** with a new relation R. Let $\mathbf{CST^R}$ be the theory in which all the rules of **CST** are extended to this new language together with the axiom *Rel*. The following informs us that such additions are conservative.

Theorem 16 *Let ψ be any wff of **CST**. Then*

$$\textit{if } \Gamma \vdash_{\mathbf{CST^R}} \psi \textit{ then } \Gamma \vdash_{\mathbf{CST}} \psi$$

The theorem follows as a direct result of the following lemma which shows how to compile any wff of the extended theory to one of the original.

Lemma 17 *There is a translation * from the language of $\mathbf{CST^R}$ to the language of **CST** such that*

1. If $\Gamma \vdash_{\mathbf{CST^R}} \phi$ then $\Gamma^* \vdash_{\mathbf{CST}} \phi^*$
2. If ϕ is a wff of **CST** then $\phi^* = \phi$

where Γ^* is the translated context

Proof. We shall spell out the details of the translation since it will be employed as a basis for several modifications. The translation is defined by recursion on the extended language. The major impact is obviously on the new relation symbol. We illustrate with the case of one type variable and one individual argument.

$$R[X] \triangleq [x : A \mid \phi]$$

This is transformed as follows.

$$R[T](t)^* = \phi[T/X, t/x]$$

All the other atomic wff translated to themselves

$$\alpha^* = \alpha$$

The translation passes through the propositional connectives and the quantifiers e.g.

$$\begin{aligned}
(\delta \wedge \eta)^* &= \delta^* \wedge \eta^* \\
(\forall x : T \cdot \eta)^* &= \forall x : T^* \cdot \eta^* \\
(\forall x \in b \cdot \eta)^* &= \forall x \in b \cdot \eta^* \\
(\forall X \cdot \eta)^* &= \forall X \cdot \eta^*
\end{aligned}$$

All types are translated to themselves. This completes the translation. Part (2) is immediate given the translation. Part (1) follows by induction on the derivations in $\mathbf{CST^R}$. All the rules of **CST** are automatic. This leaves us to check the new axiom which is immediate.■

We shall employ the whole translation again in connection with function specifications. For this reason we have provided the explicit details of the translation.

3.3 Comparing Schema

For theoretical purposes, we shall need to compare schema specifications. There are two important relationships that we shall employ in the sequel. The first is the obvious one.

Definition 18 *Let*

$$\begin{aligned}
R[X_1, .., X_n] &\triangleq [x_1 : A_1, .., x_k : A_k \mid \eta] \\
S[X_1, .., X_n] &\triangleq [x_1 : A_1, .., x_k : A_k \mid \delta]
\end{aligned}$$

be two schema specifications. We shall say they are ***Equivalent*** *(written $R \cong S$) iff*

$$\forall X_1 \cdot .. \cdot \forall X_n \cdot \forall x_1 : A_1 \cdot .. \cdot \forall x_k : A_k \cdot \\ R[X_1, .., X_n](x_1, .., x_k) \leftrightarrow S[X_1, .., X_n](x_1, .., x_k)$$

Example 19 *The schema specification of union is equivalent to the schema with the predicate*

$$(\forall x \in w \cdot x \in u \vee x \in v) \wedge (\forall x \in u \cdot x \in w) \wedge (\forall x \in v \cdot x \in w)$$

Although this notion will do a great deal of work for us in comparing schema it will sometimes be too constraining and so we introduce another standard notion that generalizes it.

Definition 20 *Let*

$$\begin{array}{rcl} R[X_1, .., X_n] & \triangleq & [x : I, y : O \mid \psi] \\ S[X_1, .., X_n] & \triangleq & [x : I, y : O \mid \eta] \end{array}$$

We shall say that S is a ***Refinement*** *of R, written as*

$$R \sqsubseteq S$$

iff

1. $\forall X_1 \cdot .. \cdot \forall X_n \cdot \forall x : I \cdot Dom_R[X_1, .., X_n](x) \rightarrow Dom_S[X_1, .., X_n](x)$
2. $\forall X_1 \cdot .. \cdot \forall X_n \cdot \forall x : I \cdot Dom_R[X_1, .., X_n](x) \rightarrow \forall y : O \cdot \psi[x, y] \leftrightarrow \eta[x, y]$

We shall say that the two Schema are ***Weakly Equivalent*** *iff they refine each other i.e. we write*

$$R \approx S \triangleq R \sqsubseteq S \wedge S \sqsubseteq R$$

Clearly if two schema of equivalent they are weakly equivalent i.e.

$$R \cong S \rightarrow R \approx S$$

We shall unpack the connection in the other direction shortly. These notions will aid us in our investigation of many aspects of specification, including the following.

3.4 Total Operations

There are some important special cases of schema which are theoretically and practically significant. We begin with *Totality.*

Definition 21 *Let*

$$R[X_1, .., X_n] \triangleq [x : I, y : O \mid \psi]$$

We shall say that R defines a ***Total*** *relation if*

$$\forall X_1 \cdot .. \cdot \forall X_n \cdot \forall x : I \cdot Dom_R[X_1, .., X_n](x) \qquad \textbf{(TOT)}$$

We shall say that a many place operation is ***Total*** *iff*

$$R[X_1, .., X_n] \triangleq [x : I_1 \otimes .. \otimes I_k, y : O_1 \otimes .. \otimes O_m \mid \psi[x_1, .., x_k, y_1, .., y_m]]$$

is.

Many of our operators are total.

Proposition 22 *$Pair, Union, Genunion, Separation, Dom$ and Ran are total.*

Proof. We shall not prove all of these but rather illustrate the technique with simple union. For union, fix $u : Set(X)$. We use induction with the wff

$$\psi[v] = \exists w : Set(X) \cdot \forall x : X \cdot x \in w \leftrightarrow x \in u \vee x \in v$$

If v is the empty set then we put $w = u$. Suppose that v has the form $x' \circledast y$. Assume inductively $\psi[y]$. Let w' be the guaranteed set. Then we put the required set for v to be $x' \circledast w'$. ■

Observe that for R and S total we have that weak equivalence implies equivalence i.e.

$$R \approx S \rightarrow R \cong S$$

By way of further unpacking these notions, also notice that:

Proposition 23 *If $R \sqsubseteq S$ and R is total then so is S*

Consequently, if one of the relations is total and they are weakly equivalent then they are equivalent. But obviously not all relations are total. However, given a relation we can always define one which is total.

Definition 24 *Let*

$$R[X_1, .., X_n] \triangleq [x : I, y : O \mid \psi]$$

Define the ***Totalisation of R*** *as*

$$R^T[X_1, .., X_n] \triangleq [x : I, y : O \mid Dom_R[X_1, .., X_n] \rightarrow \psi]$$

The reader might wish to compare this idea with that of [28]. The two notions are related but it will take us too far afield to say exactly how.

Example 25 *Consider the following specification of the predecessor relation.*

$$Pred \triangleq \left[x : N, y : N \mid x = y^+\right]$$

The domain of this relation is given as

$$Dom_{Pred} \triangleq \left[x : N \mid \exists y : N \cdot x = y^+\right]$$

Since,

$$\forall x : N \cdot x > 0 \leftrightarrow \exists y : N \cdot x = y^+$$

the totalisation is equivalent to the schema

$$Pred^T \triangleq \left[x : N, y : N \mid x > 0 \rightarrow x = y^+\right]$$

Observe that R and R^T are equivalent just in case R is total. Moreover, any relation can be refined to a total one.

Proposition 26 *For any R, R^T is total and $R \sqsubseteq R^T$*

Proof. Clearly $R^T$is total. Moreover its domain extends that of R but on the domain of R it agrees with R.■

3.5 Type Independence

This brings us to the second general property of schema. The alert reader will have noticed that there is a difference between the basic relations of the theory (i.e. $\in$) and the present style of relation specification where new relation symbols take type arguments. This leads to the following idea.

Definition 27 *Let*

$$R[X_1, .., X_n] \triangleq [x_1 : A_1, .., x_k : A_k \mid \psi]$$

*be any schema specification. We shall say that R is **Type Independent** just in case R is equivalent to a schema of the form*

$$S[X_1, .., X_n] \triangleq [x_1 : A_1, .., x_k : A_k \mid \eta]$$

where η contains no free type variables.

We then have:

Theorem 28 *For any type independent specification we may conservatively add a new relation symbol R^I which satisfies*

$$\forall x_1 : A_1 \cdot ... \cdot \forall x_k : A_k \cdot R^I(x_1, .., x_k) \leftrightarrow R[X_1, .., X_n](x_1, .., x_k)$$

Proof. We illustrate with the simple case.

$$\forall X \cdot \forall x : A[X] \cdot \forall y : B[X] \cdot R[X](x, y) \leftrightarrow \psi[X, x, y]$$

Suppose that

$$\forall X \cdot \forall x : A[X] \cdot \forall y : B[X] \cdot \psi[X, x, y] \leftrightarrow \eta[x, y]$$

We then we compile away in a similar manner to the explicit case but where the implicit relation is now interpreted as

$$R^I(a, b)^* = \eta(a, b)$$

This clearly satisfies the condition.■

Proposition 29 *Subset, Extensional Equivalence for sets, Union, Generalised Union, Separation, Power and Map are type independent.*

Proof. The first two we have met before and are immediate. We illustrate the rest with union and power set. For union, we know that the predicate is equivalent to

$$u \subseteq w \wedge v \subseteq w \wedge \forall x \in w \cdot x \in u \vee x \in v$$

To show that the power set is type independent, we show that the predicate is equivalent to

$$\emptyset \in y \wedge (\forall z \in x \cdot \forall u \in y \cdot z \circledast u \in y) \wedge \forall z \in y \cdot z \subseteq x \qquad (*)$$

Assume $x : Set(X)$, $y : Set(Set(X))$. First assume *. Then

$$\forall w : Set(X) \cdot w \in y \rightarrow w \subseteq x$$

is automatic. So let $w : Set(X) \wedge w \subseteq x$. We argue by induction with the wff

$$\psi[w] \triangleq w \subseteq x \rightarrow w \in y$$

If w is the empty set then we are done by $*$. If w has the form $x' \circledast y'$ then, by induction, we may assume $\psi[y']$ and we are done by *. Conversely, assume

$$\forall w : Set(X) \cdot w \in y \leftrightarrow w \subseteq x$$

We have to show that $\emptyset \in y \wedge \forall z \in x \cdot \forall u \in y \cdot z \circledast u \in y$. The first conjunct is immediate. Assume that $z \in x \wedge u \in y$. So $z \in x \wedge u \subseteq x$. Hence, $z \circledast u \subseteq w$ and so $z \circledast u \in y$. ■

Where we can establish type independence we shall use the same name for the relation in its implicit manifestation. In particular, this enables us to circumvent the irritating build-up of type information in the predicates of specifications where the type information is already in the declaration. For example, we may now specify power sets as

$$\begin{array}{|l} Pow[X] \\ \hline u : Set(X), v : Set(Set(X)) \\ \hline \forall x : Set(X) \cdot x \in v \leftrightarrow x \subseteq u \\ \hline \end{array}$$

It might be useful to develop some simple criteria for type independence. Many of these arise in connection with *schema calculus*. But this will take us too far afield. We content ourselves with the following idea.

Definition 30 *Let*

$$R[X_1, .., X_n] \triangleq [x : I, y : O \mid \psi]$$

*be any schema specification. We shall say that R is **Closed** if*

$$\forall X_1 \cdot .. \cdot \forall X_n \cdot \forall x : I \cdot \psi[x, y] \rightarrow y : O \qquad (\mathbf{CLO})$$

This is a natural condition on operations and simply demands that the type of the input determines the type of the output.

Proposition 31 *If R is type independent and satisfies closure then Dom_R and Ran_R are type independent.*

Proof. Let R be equivalent to

$$S[X] \triangleq [x : I, y : O \mid \eta]$$

where η contains no free type variables. Then Dom_R is equivalent to

$$[x : I \mid \exists y : O \cdot \eta]$$

Given closure, this is equivalent to

$$[x : I \mid \exists Y \cdot \exists y : O[Y/X] \cdot \eta]$$

■

This completes our introduction to relation specification. There is much more to say and more important examples to study but we have done enough to move on to *function specifications*.

4 Function Specification

This is substantially different from the introduction of new relations: whereas the latter enrich the class of atomic wff, new function symbols enrich the class of individual terms and, in particular, new functions return values that can themselves be passed as arguments to other functions and relations. This will bring us closer to the specification style of **VDM** ([6], [13]). Mathematically, this is a more subtle extension and *legitimacy* will be a more delicate matter. However, the addition of new functions is mathematically essential for the development of a useful theory of *Numbers, Sets* and *Cartesian Products*.

4.1 Function Application

For simplicity of notation, we shall employ the binary case to illustrate matters. However, as we shall see, given Cartesian products, one can easily extrapolate. To begin with we require the following notion.

Definition 32 *Let*

$$R[X_1, ..., X_n] \triangleq [x : I, y : O \mid \psi]$$

*then we shall say that R is **Functional** iff*

$$\forall X_1 \cdot ... \cdot \forall X_n \cdot \forall x : I \cdot \exists^{\leq 1} y : O \cdot \psi \qquad (\mathbf{PF})$$

The following are all easy consequences of extensionality.

Proposition 33 *Pair, Union, Genunion, Separation, Cartesian Product are all functional.*

However, relation specifications which happen to be functional are still relational in the sense that they are introduced as new relational symbols in the theory. They have not been introduced as *genuine* function symbols which can be applied to arguments in the standard way. This is the content of the following.

Definition 34 *Let*

$$R[X_1, ..., X_n] \triangleq [x : I, y : O \mid \psi]$$

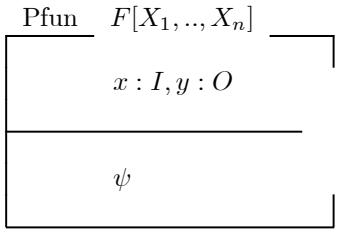
be functional. We may then introduce a new function symbol into the language via the following ***Function Schema***

$$F[X_1, .., X_n] \triangleq_{Pfun} [x : I, y : O \mid \psi] \qquad (\mathbf{FS})$$

The specification is intended to introduce a new (partial) function symbol to the language and in particular, for any $A_1, .., A_n$ and t, $F[A_1, .., A_n](t)$ is a new term. In the special case where R is total we shall write

$$F[X_1, .., X_n] \triangleq_{Fun} [x : I, y : O \mid \psi]$$

The new function symbol does not occur in ψ, so at this point no recursion is intended. We shall deal with this in a later publication. We shall also write this specification in more graphic notation as



More generally, the specification of many place functions takes the form

$$F[X_1, .., X_n] \triangleq_{Pfun} [x_1? : I_1, ..., x_k? : I_k, y! : O \mid \psi]$$

which is to be unpacked in terms of the product as the specification

$$F[X_1, .., X_n] \triangleq_{Pfun} [x? : I_1 \otimes ... \otimes I_k, y! : O \mid \ \psi[x?_1/x_1?, .., x?_n/x_n?, y/y!]]$$

FS is intended to go beyond S. More specifically, given PF, FS is to be logically interpreted as the introduction of a new function symbol which satisfies the following axioms.

$$\forall X_1 \cdot . \cdot \forall X_n \cdot \forall x : I \cdot Dom_R[X_1, .., X_n](x) \rightarrow F[X_1, .., X_n](x) : O \qquad (\mathbf{F}_1)$$

$$\forall X_1 \cdot . \cdot \forall X_n \cdot \forall x : I \cdot Dom_R[X_1, .., X_n](x) \rightarrow \psi[x, F[X_1, .., X_n](x)] \qquad (\mathbf{F}_2)$$

Example 35 *The following is a specification of the predecessor function*

$$Pred \triangleq_{Pfun} [x : N, y : N \mid x = y^+]$$

This introduces predecessor as a new function symbol which satisfies

$$\forall x : N \cdot x > 0 \rightarrow Pred(x) : N$$

$$\forall x : N \cdot x > 0 \rightarrow pred(x)^+ = x$$

VDM uses the term *Implementable* for the PF requirement (or rather the total version of it) but this seems inappropriate since it obviously does not guarantee that the function is in any sense *computable*. Instead, we continue to use the word *Legitimate* but now to describe a function specification for which PF has been established.

Another way of looking at matters is instructive. Let

$$R[X_1, ..., X_n] \triangleq [x : I, y : O \mid \psi]$$

be functional. Then we may introduce *application* for R via the axioms

$$\forall x : I \cdot Dom_R[X_1, .., X_n](x) \rightarrow App_R[X_1, .., X_n](x) : O$$

$$\forall x : I \cdot Dom_R[X_1, .., X_n](x) \rightarrow \psi[x, App_R[X_1, .., X_n](x)]$$

Of course, the two routes are formally identical and the only difference is the explicit declaration that a function is being defined. Our uniform use of schema notation for both, highlights the relationship and difference.

Example 36 *The following is a functional specification of the Cartesian product operator on sets. It is total but it is non-trivial to show it.*

Fun $\otimes[X]$

$$x? : Set(X), y? : Set(Y), z! : Set(X \otimes Y)$$

$$\forall w : X \otimes Y \cdot w \in z \leftrightarrow w_1 \in x \wedge w_2 \in y$$

Example 37 *The following is a specification of* ***Application*** *for maps*

Pfun $Mapp[X, Y]$

$$z? : Set(X \otimes Y), u? \in X, w! : Y$$

$$Map[X, Y](z) \wedge (u, w) \in z$$

The following will prove useful shortly.

Proposition 38 *The following is a total function. It is also type independent*

Fun $In[X]$

$$u? : X, v? : Set(Set(X)), z! : Set(Set(X))$$

$$\forall x \in v \cdot u \circledast x \in z$$
$$\forall y \in z \cdot \exists x \in v \cdot y = u \circledast x$$

Proof. We first prove totality by induction on v with the wff

$$\exists z : Set(Set(X)) \cdot \forall x \in v \cdot u \circledast x \in z$$

The case where v is empty we put $z = \{\{u\}\}$. So assume that v has the form $y \circledast w$. By induction,

$$\exists z' : Set(Set(X)) \cdot \forall x \in w \cdot u \circledast x \in z'$$

The required set for $y \circledast w$ is then $(u \circledast y) \circledast z'$. Hence

$$\exists z : Set(Set(X)) \cdot \forall x \in v \cdot u \circledast x \in z$$

Now given this set, the set required for the induction step is given by separation as

$$\{y \in z \cdot \exists x \in v \cdot y = u \circledast x\}$$

Functionality follows from the extensional nature of sets. Independence is immediate.■

We can now return to proof that the power-set constructor defines a total function.

Proposition 39 *Power is total, functional and type independent*

Proof. Type independence has already been established. For the other two, totality is the non-trivial part. We prove the result by induction where the induction wff is

$$\phi[x] = \exists y : Set(Set(X)) \cdot \forall z : Set(X) \cdot z \in y \leftrightarrow z \subseteq x$$

If $x = \emptyset$ then the required set is $\emptyset$. If $x = u \circledast v$ then there are two cases. If $v = \emptyset$ then the required set is $\{\emptyset, \{u\}\}$. Otherwise, let v' be guaranteed by induction i.e. the power set of v. Now put the power set of $u \circledast v$ to be $In(u, v') \cup v'$.■

The following provides a slightly different perspective on the introduction of new functions and has a more logical origin.

Lemma 40 *Given PF, in the language with a new function symbol added, F_1 and F_2 are equivalent to the following*

$$\forall X_1 \cdot .. \cdot \forall X_n \cdot \forall x : I \cdot \forall y : O \cdot Dom_R(x) \rightarrow (F[X_1, .., X_n](x) = y \leftrightarrow \psi) \quad (\mathbf{F})$$

Proof. We illustrate with the case where $n = 1$. Assume F_1 and F_2. Assume $x : I[X]$ and $y : O[X]$. Assume $Dom_R(x)$. Assume first $F[X](x) = y$. Then by F_2, $\psi[x, F[X](x)]$. By F_1, $F[X](x) : O[X]$. By E_2, $\psi[x, y]$. On the other hand, given $\psi[x, y]$, and, by F_2, $\psi[x, F[X](x)]$, functionality yields $y = F[X](x)$. Conversely, assume F. F_1 is immediate by definition of type membership in **CST** and the equivalence given by F. Moreover, given F_1, F_2 is immediate: put $y = F[X](x)$ in F.■

We can now easily see why PF is necessary. If we introduce a new function symbol without it, the theory may be rendered inconsistent. To see this suppose that $x : I[X]$, $y : O[X]$, $y' : O[X]$ and

$$\psi[x, y] \wedge \psi[x, y'] \wedge y \neq y'$$

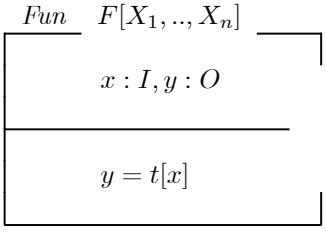
i.e. it is not *single-valued*. Then by F,

$$F[X](x) = y \wedge F[X](x) = y' \wedge y \neq y'$$

So we cannot just specify a function via PS and stop; we must establish PF to be sure that the theory remains consistent. Finally note that ours is a very different approach to that adopted by **VDM** ([12]) which employs a 3-valued logic to deal with partial functions.

We shall refer to the above style of function definition as *Indirect* since functions are being characterized logically rather than by indicating how to compute them. **VDM** uses the term *Implicit* but we have already adopted this description for polymorphism. In fact, **VDM** does not support the definition of *indirect polymorphic* functions in this very general form; it only allows such functions to be *Directly* specified. Although this is actually a special case of the Indirect style, it is important enough to consider separately.

Definition 41 *Let I, O be any type terms and t any term, which may now contain free type variables. We assume that $X_1, ..., X_n$ include and exhaust all the type variables of I, O, t. We may then introduce a new function symbol* ***Directly*** *by the simple style of function specification*



Notice that this will be a total function just in case

$$\forall x : I \cdot t[x] : O$$

Example 42 *Pairing provides a simple example where there are no type variables in the predicate.*

$$Pair[X] \triangleq_{Fun} [x? : X, y? : X, z! : Set(X) \mid \ z = x \circledast y \circledast \emptyset]$$

Direct function definitions are common in mathematics and computer science. Indeed, much of the infrastructure of the theory will be constructed by a judicious mix of both indirect and direct ones. Typically, one presents an indirect definition followed by a sequence of direct ones supported by it.

4.2 Conservative Extensions

We now turn to showing that such additions are conservative. Suppose that we can prove PF in **CST**. Let $\mathbf{CST^F}$ be the theory in which all the rules of **CST** are extended to this new language together with the axioms F_1, F_2. Then we have:

Theorem 43 $\boldsymbol{CST^F}$ *is a conservative extension of* $\boldsymbol{CST}$

This follows from the following.

Lemma 44 *There is a translation* * *from the language of* $\boldsymbol{CST^F}$ *to the language of* $\boldsymbol{CST}$ *such that*

1. If $\Gamma \vdash_{\mathbf{CST^F}} \phi$ then $\Gamma^* \vdash_{\mathbf{CST}} \phi^*$
2. If ϕ is a wff of **CST** then $\phi^* = \phi$

Proof. We illustrate with the simple case where there is only one type variable and the relation is total. We proceed as follows. First, using DeMorgan's laws we push all the negation's through to atomic assertions. The translation then proceeds as in the relational case for all the connectives and quantifiers. Atomic assertions and their negation's which do not contain F are compiled as before. This leaves us to deal with the atomic assertions and their negation's which do contain F. These are transformed using the following.

$$\alpha[F[A](x)/y]^* = x \in I[A] \wedge \exists u : O[A] \cdot \psi[A, x, u] \wedge \alpha[u/y]$$
$$(\neg\alpha[F[A](x)(x)/y])^* = x \in I[A] \wedge \exists u : O[A] \cdot \psi[A, x, u] \wedge \neg\alpha[u/y]$$

Part (2) is immediate from the definition of the translation. Part (1) follows by induction on the derivations. Almost all the axioms and rules are automatic as they are in the relational case. This leaves us to check F_1 and F_2. The former unpacks to

$$\forall x : I[A] \cdot \exists u : O[A] \cdot \exists v : O[A] \cdot \psi[A, x, u] \wedge u = v$$

which is true. For the latter we establish

$$\forall x \in I[X] \cdot (\exists u : O[X] \cdot \psi[X, x, u]) \rightarrow \psi[X, x, F[X](x)]$$

since, given totality, this immediately yields the result. We achieve this by induction on ψ. We may assume that ψ is in normal form. Suppose that ψ is an atomic wff. Then F_2 unpacks to the true:

$$\forall x : I[X] \cdot \exists u : O[X] \cdot \psi[X, x, u] \rightarrow \exists u : O[X] \cdot \psi[X, x, u]$$

The negative case is similar. Given that ψ is in normal form, all the induction cases are easy to check. For example, by induction

$$\forall x \in I[X] \cdot (\exists u : O[X] \cdot \eta[X, x, u]) \rightarrow \eta[X, x, F[X](x)]$$
$$\forall x \in I[X] \cdot (\exists u : O[X] \cdot \delta[X, x, u]) \rightarrow \delta[X, x, F[X](x)]$$

Hence,

$$\forall x \in I[X] \cdot (\exists u : O[X] \cdot \eta[X, x, u] \vee \delta[X, x, u]) \rightarrow \eta[X, x, F[X](x)] \vee \delta[X, x, F[X](x)]$$

This concludes part(2).■

Once again this is a crucial result for the process of specification. The conservativeness of specifications has not been sufficiently discussed in the literature on specification languages.

4.3 Functions With Pre-Conditions

VDM allows the specification of functions with *pre-conditions*. In this section we develop our style of functional specification to permit them. We first establish the following.

Lemma 45 *Let ϕ, ψ be any wff and I, O any type terms. We assume that $X_1, ..., X_n$ include and exhaust all the type variables of I, O, ϕ, ψ. Furthermore, we assume that x, y are the free individual variables of ψ and x is the only free individual variable of ϕ. Suppose that*

$$\forall X_1 \cdot ... \cdot \forall X_n \cdot \forall x : I \cdot \phi[x] \rightarrow \exists! y : O \cdot \psi[x, y] \qquad \text{(PPF)}$$

We may then legitimately introduce a new function symbol into the language that satisfies

$$\forall X_1 \cdot . \cdot \forall X_n \cdot \forall x : I \cdot \phi[x] \rightarrow F[X_1, .., X_n](x) : O \qquad (\mathbf{PREF}_1)$$

$$\forall X_1 \cdot . \cdot \forall X_n \cdot \forall x : I \cdot \phi[x] \rightarrow \psi[x, F[X_1, .., X_n](x)] \qquad (\mathbf{PREF}_2)$$

Proof. Given PPF, the following schema satisfies the above

$$F[X_1, ..., X_n] \triangleq_{Pfun} [x : I, y : O \mid \psi]$$

■

Definition 46 *We shall write*

$$F[X_1, .., X_n] \triangleq_{Fun} [x : I, y : O \mid \phi; \psi]$$

for a specification that is to be logically unpacked as the introduction of a new function symbol which satisfies $PREF_1$, $PREF_2$

The above shows that, where PPF is provable, such specifications are legitimate.

Example 47 *The following provides a specification, with pre-conditions, of Map application.*

$Fun \quad Mapp[X, Y]$

$$z? : Set(X \otimes Y), u? \in X, w! : Y$$

$$Map(z) \wedge u \in Dom[X, Y](z);$$
$$(u, w) \in z$$

This provides us with a very general mechanism for the legitimate introduction of new function symbols.

4.4 Type Independence

In parallel with relation specifications we may also drop the type variables in function specifications but now we need not only functionality but also type independence.

Theorem 48 *If R is functional and type independent then we conservatively introduce a new function symbol F given axiomatically by*

$$\forall X_1 \cdot . \cdot \forall X_n \cdot \forall x : I \cdot Dom_R[X_1, .., X_n](x) \rightarrow F(x) : O \qquad (\mathbf{IF}_1)$$

$$\forall X_1 \cdot . \cdot \forall X_n \cdot \forall x : I \cdot Dom_R[X_1, .., X_n](x) \rightarrow \psi[x, F(x)] \qquad (\mathbf{IF}_2)$$

Proof. We mimic the style of proof for explicit polymorphism. We illustrate with the following total case. We shall assume that ψ contains no free type variables-given type independence we can always reduce matters to such wff. We proceed as in the explicit case but translate

$$\alpha[F(x)/y]^* = \exists X \cdot x \in I[X] \wedge \exists u : O[X] \cdot \psi[x, u] \wedge \alpha[u/y]$$
$$(\neg\alpha[F(x)(x)/y])^* = \exists X \cdot x \in I[X] \wedge \exists u : O[X] \cdot \psi[x, u] \wedge \neg\alpha[u/y]$$

It is easy to check that IF_1 is satisfied. For IF_2, we establish that the following is sound under the translation.

$$\forall x \in I[X] \cdot (\exists u : O[X] \cdot \psi[x, u] \rightarrow \psi[x, F(x)])$$

Given totality, this immediately yields the result. We achieve this by induction on ψ. The proof then parallels the original. ■

Notice that as an upshot of this, we could take many of our relations as new implicitly polymorphic functions. For example, Genunion would take the form of a new function symbols which satisfies

$$\forall x : Set(Set(X)) \cdot \cup(x) : Set(X)$$
$$\forall x : Set(Set(X)) \cdot \forall y : X \cdot y \in \cup(x) \leftrightarrow \exists z \in x \cdot y \in z$$

A theory with Pair, Genunion, Powerset and separation is a typed version of Zermelo set theory but with numbers forming a type and not a set. It is also a sub-theory of both Z and **VDM**.

5 Further Work

There are many topics left to explore. The formulation and exploration of *type inference* systems for specification will form the topic of the next paper. A paper on set theoretic models of the theory will follow that. Other topics will be considered in future publications.

6 Acknowledgments

I would like to thank Martin Henson, Norbert Völker and Amnon Eden for detailed comments and discussion on this paper and the next.

References

[1] Abrial, J.R. The B-Book. Cambridge University Press. Cambridge 1996.

[2] Barwise, J. Admissible Sets and Structures. Springer Verlag. Berlin. 1975.

[3] Blikle, A. Three-valued predicates for Software Specification and Validation. VDM '88 VDM – The Way Ahead, pp. 243-266, Springer-Verlag, September 1988.

[4] Brien, S.M. and Nicholls, J.E. Z Based Standard Version 1.0, Oxford University Computing Laboratory. PRG 107, 1992.

[5] Cheng, J.H. and Jones, C.P.. On the usability of logics which handle partial functions. Proceedings of the Third Refinement Workshop, Springer-Verlag, 1990.

[6] Dawes, J. The VDM-SL Reference Guide. Pitman. 1991.

[7] Diller, A. Z: An Introduction to Formal Methods. John Wiley& Sons, 1990.

[8] Gibbins, P.F. VDM: Axiomatising its propositional logic. BCS Computer Journal, Vol. 31, pp. 510-516, 1988.

[9] Henson, M. and Reeves, S. Revising Z -I Semantics and Logic. Journal of Formal Aspects of Computing, vol 11,no 4, pp 359-380, 1999.

[10] Henson, M. and Reeves, S. Revising Z -II Logical Development. Journal of Formal Aspects of Computing, vol 11,no 4, pp 381-401, 1999.

[11] Hodges, W. The Semantics of Specification Languages. Queen Mary College, London. 1990

[12] Jones, C,B. and Middleburg. C.A. A typed Logic of Partial Functions reconstructed classically, Acta Informatica, 31(5); 399-430, 1994.

[13] Jones, C.B. Systematic Software Development Using VDM. Prentice Hall, Hemel Hemstead, 1986.

[14] Kooij, M. Interface Specification with Temporal Logic. Proc. 5th International Workshop on Software Specification and Design, pp. 104-110, May 1989.

[15] Larson, P,G. and Pawlowski, W. The Formal Semantics of ISO VDM-SL. The Journal of Computer Standards and Interfaces. 1995.

[16] Milne, R., The proof theory of the Raise specification language. Technical report. Raise/STC/REM/12/V3. England. 1990.

[17] Maibaum, T.S.E. and Turski, M. The Specification of Computer Programs. Addison Wesley. 1987.

[18] Middelburg, K. The Logical Semantics of Flat VVSL. Technical Report, Neher Laboratories, Number 954 RNL/89, December 1989.

[19] Moore, R. and Ritchie, Proof in VDM-A practitioners Guide, FACIT, Springer Verlag, ISBN 3-540-19813-X, 1994.

[20] Nicholls, J.E. (ed). Z-Notation version 1.2, 1995.

[21] Owre, S. Rushby, John. Natarajan, S, and Von Henke, F. Formal Verification of Fault Tolerant Architecture Prolegomena to the design of PVS, IEEE Transactions of Software Engineering, Vol. 21, no 2, pp 107-125, Feb. 1995.

[22] Owre, Sam and Shankar Natarajan: The Formal Semantics of PVS. NASA/CR-1999-209321, May 1999.

[23] Potter, B., Sinclair, S. and Till, D. : An introduction to Formal Specification and $\mathbb{Z}$. Prentice Hall, 1991.

[24] Spivey, J.M. : Understanding Z. Cambridge University Press, 1988.

[25] Spivey, J.M.: The Z Notation: A Reference Manual. Prentice Hall, 1992.

[26] Tarlecki, A., Konikowska, B. and Blikle, A. A Three-valued logic for Software Specification and Validation. VDM '88 VDM – The Way Ahead, pp. 218-242, Springer-Verlag, September 1988.

[27] Woodcock, J. and Brien, S.M. : W: a logic for Z, in J.E. Nicholls (ed), Z Users Workshop, York, 1991, Proceedings of sixth Annual $\mathbb{Z}$ User Meeting. Springer Verlag, 1992.

[28] Woodcock, J. and Davies, J. Using Z- Specifications, Refinement and Proof, Prentice Hall, 1996.