Title: Notes on the VDM type model (document 44)

Ref: DBC/RBJ/069 *Issue:* 1.1 *Date:* 27th Aug 1987

Status: Draft *Type:*

Author:

Name	*Location*	*Signature*	*Date*
Roger Bishop Jones			

0 Document control

0.1 Contents list

0 Document control **1**
0.1 Contents list . . . 1
0.2 Document cross references . . . 2

1 Introduction **2**

2 A Language for Type Definition **2**

3 Type Universe and their Axiomatic Properties **2**
3.1 Basic Types . . . 2
3.2 Operators . . . 3
3.3 Closure conditions . . . 3

4 Extended Type Universes **3**
4.1 The subset closure operator . . . 3
4.2 Flat domain embedding operator . . . 3
4.3 Scott Universes . . . 3

5 CONSTRUCTIONS **3**
5.1 Base Types . . . 3
5.2 The Tag Operator . . . 4
5.2.1 Tags and the *mk* notation . . . 4
5.3 Building U . . . 4
5.4 Building the Scott Universe . . . 4

6 Type Universes and VDM Types **4**

7 Conclusions and Summary **4**

8 Supplimentary Discussion **5**
8.1 Continuity 5
8.2 Solutions to Recursive Type Equations 5

0.2 Document cross references

[1] *A Type Model for VDM*, B.Q.Monahan, Document no.44. Published in LNCS 252, ISBN 0-387-17654-3

1 Introduction

These are personal notes made partly to aid my own comprehension and partly to provide feedback for Brian. I have picked nits, including typos. The structure of this document follows that of Brian's paper exactly even to the extent of dummy sections to get the section numbers right.

2 A Language for Type Definition

3 Type Universe and their Axiomatic Properties

3.1 Basic Types

Are we to assume that all atoms introduced by name (subsequently) are distinct if their names are distinct?

Of which basic types is NIL VALUE a member? (none I would guess)

Proposition (7) should read:

$$\forall \mathbf{S} \subseteq \mathbf{Atom}(\mathbf{S} \neq \{\} \wedge |\mathbf{S}| \in \omega) \Rightarrow \mathbf{S} \in \mathbf{Bty} \tag{7}$$

The text asserts that Atom is not a basic type, but the formal propositions do not, and (8) is formalised in a way which suggests it might be (do the connotations of formulae matter?).

If (71) were added:

$$\mathbf{Atom} \notin \mathbf{Bty} \tag{71}$$

then (8) might be expressed:

$$\forall \mathbf{S} \in \mathbf{Bty} |\mathbf{S}| \notin \omega \Rightarrow \mathbf{Atom} \cap \mathbf{S} = \varnothing \tag{8}$$

Why is Atom not a Basic Type?

Supersets of Atom are not excluded, should they have been?

Maybe the following (instead of (7) and (8)) better expresses the requirements:

$$\forall \mathbf{SS} \cap \mathbf{Atom} \neq \varnothing \Rightarrow (\mathbf{S} \in \mathbf{Bty} \Leftrightarrow (|\mathbf{S}| \in \omega \wedge \mathbf{S} \subset \mathbf{Atom})) \tag{9}$$

In words, the only basic types containing any atoms are the finite (non-empty) subsets of Atom (which are all basic types). This prevents Atom being a member of $Sub(Bty)$.

Observations in subsequent sections seem to depend upon Bty being a countable set of countable sets, but this doesn't seem to follow from anything said here.

Perhaps we should also have:

$$|Byt| \in \omega \cup \{\omega\} \tag{10}$$

$$\forall S \in Bty.|S| \in \omega \cup \{\omega\} \tag{11}$$

The correctness of these propositions depends a property of the definitions of ω and $|s|$, which is not mentioned either here or in the appendix on set theory. Viz. $\forall a, b \in \omega.a < b \Leftrightarrow a \in b$

3.2 Operators

What is meant by a *family* of sets? Is this just a set of sets, or will any collection or class do?

Note that field selection corresponds to the application of the *mapping representing the record* to the *field name*.

3.3 Closure conditions

Fortuneatly should read *Fortunately*

4 Extended Type Universes

4.1 The subset closure operator

'Hence if..' is this because all the operators are monotonic?

4.2 Flat domain embedding operator

$\boxed{\perp}$ surely cannot be arbitrary, it must presumably not be a member of $\cup F$.

4.3 Scott Universes

How do they fail to be CCC's?

5 CONSTRUCTIONS

5.1 Base Types

I like the use of countable sets, but the constraints specified here on A are difficult to understand.

Floating point numbers are a proper subset of Q (I note in passing).

Surely the options are:

i Something realistic like floating point (not a field).

ii Compromise with a field (e.g. Q).

iii Go the whole hog with a complete field (countable models available if that matters).

'A' seems to be a vague compromise falling somewhere between [ii] and [iii].

5.2 The Tag Operator

Chose a countable *set* Tag...

Its not clear how we know that $\overline{U}$ is a countable set of countable sets. Perhaps this follows from a similar unstated constraint on Bty?

5.2.1 Tags and the *mk* notation

Why are record types ordered?

5.3 Building U

Your reference to Cohn doesn't give any satisfaction in the 1981 paperback edition.

5.4 Building the Scott Universe

Last sentence first paragraph, for *delt* read *dealt*.

Page 14, footnote, *fortuneatly* should read *fortunately*.

6 Type Universes and VDM Types

7 Conclusions and Summary

By *finitely generated* do you just mean countable or finite, or do you mean more than that, if so, what?

is either infinite *or not values..* doesn't make grammatical sense.

The examples concerning type equivalence are not very illuminating. Surely no notion of type equivalence, whether extensional or not, could fail to have the properties mentioned? The first

property is a consequence of equality being an equivalence relation,the second an instance of the requirement that type constructors must respect type equality.

That types are extensional requires that whenever two types have exactly the same members they are equal, which is not derivable from the properties shown.

8 Supplimentary Discussion

8.1 Continuity

The restriction to continuous function spaces seems to me to be a problem. This is O.K. for a programming language, but inadequate for a specification language. We want to be able to use quantification in implicit specifications, quantification is not continuous, so post-conditions will not in general be continuous functions, similarly for pre-conditions, invariants etc. In some applications the quantification in implicit specifications occurs nested several layers deep in subsidiary function definitions, and so functions in general may be discontinuous.

Of course computable functions will always be continuous, but surely we don't want to restrict implicit specifications to continuous functions?

Brian seems to suggest in section 2 that some separate language is used in type invariants, but this is not reflected elsewhere in the language definition where invariants are boolean expressions, and all expressions may have boolean expressions embedded in them.

It would be open to us to make a systematic distinction between continuous and discontinuous functions, or between computable and non-computable functions, but the distinction is not there at the moment, and continuous functions do not by themselves suffice.

8.2 Solutions to Recursive Type Equations

The closure conditions on the Universe of Types are specified, but this does not give us any obvious characterisation of which systems of type equations have non-trivial solutions.

Though interesting examples are given of cases where recursive equations have non-trivial solutions, there are also cases where the only solutions are trivial or where there are no solutions. We need some account of when solutions will be obtained.