An analysis of operation refinement in Z

Moshe Deutsch (mdeuts@essex.ac.uk)
Martin C. Henson (hensm@essex.ac.uk)
Department of Computer Science, University of Essex, UK

Steve Reeves (stever@cs.waikato.ac.nz)
Department of Computer Science, University of Waikato, NZ

Abstract

In this paper we analyse and compare several notions of operation refinement for specifications in Z. In particular we show that three theories: relational completion, proof-theoretic and functional (models) are all equivalent.

1 Introduction

In [5] the notion of refinement underlying data refinement is based on the totalisation of lifted sets of bindings. In [4] a notion of refinement is suggested that is based upon preconditions and postconditions (this idea was also taken up in [2] and in [3]). The former is essentially a model-theoretic approach, involving the addition of an extra undefined element; the latter is a proof-theoretic approach, based upon the interaction of the various predicates involved. In this paper we demonstrate that the two approaches are equivalent. We then introduce a third model of refinement based on the sets of functions (that is, essentially, deterministic implementations) which can, in a natural way, be said to meet specifications.

Our mathematical account takes place in $\mathcal{Z}_\mathcal{C}$ the core Z-logic of [1]. The only modification we need to make is to include the new undefined terms which are explicitly needed in the approach taken in [5]. Specifically: the types of $\mathcal{Z}_\mathcal{C}$ are extended to include terms $\bot^T$ for every type T. There are, additionally, a number of axioms which ensure that all the new $\bot^T$ values interact properly, e.g.

$$\overline{\bot^{[l_0:T_0 \cdots l_n:T_n]} = \langle\!|\, l_0 \Rrightarrow \bot^{T_0} \cdots l_n \Rrightarrow \bot^{T_n} \,|\!\rangle}$$

In other words, $\bot^{[l_0:T_0 \cdots l_n:T_n]}.l_i = \bot^{T_i}$ $(0 \leq i \leq n)$. Note that this is the *only* axiom concerning undefined bindings, hence, binding construction is *non-strict* with respect to the $\bot^T$ values.

Finally, the extension of $\mathcal{Z}_\mathcal{C}$ which introduces schemas as sets of bindings and the various operations of the schema calculus is undertaken as usual (see [1]) but the carrier sets of the types must be adjusted to form what we call the *natural carrier sets* which are those sets of elements of types which *explicitly exclude* the $\bot^T$ values:

Definition 1.1 *The* natural carriers *for each type are defined by closing:*

$$\mathbb{N} =_{df} \{z^{\mathbb{N}} \mid z \neq \bot^{\mathbb{N}} \wedge z = z\}$$

under the operations of cartesian product, powerset and schema set.[1]

[1] The notational ambiguity does not introduce a problem, since only a set can appear in a term or proposition, and only a type can appear as a superscript.

As a result the schema calculus is *hereditarily $\perp$-free*: the undefined constant cannot appear in any term belonging to an operation schema. This is critical in the proof of proposition 2.3(ii) below.
This extended $\mathcal{Z}_C$ system is summarised in appendix A, but the reader may need to consult [1], particularly for the full details of the notational and meta-notational conventions we employ.
We finish this introduction with an overview of the Z notion of precondition, and how this is formalised in $\mathcal{Z}_C$. In section 2 we describe W-refinement, based on [5] and in section 3, S-refinement, based on [4]. Then in section 4 we show that these two approaches are in fact equivalent. We then go on to show that there is a fairly natural model of refinement, based on containment of implementation functions, which is also equivalent. The paper concludes with some final remarks.
Z is distinguished from many other formal methods by offering a notation for specifications employing only a *single* predicate. In view of this, preconditions are the weakest conditions which guarantee that the predicate overall is met. The notion is easily formalised in the $\mathcal{Z}_C$:

Definition 1.2

$$Pre\ U^{\mathbb{P}(T^{in} \curlyvee T^{out'})}\ x^{T^{in}} =_{df} \exists z \in U \bullet x =_{T^{in}} z$$

Proposition 1.3 *The following rules are derivable for preconditions:*

$$\frac{z \in U \quad z =_{T^{in}} x}{Pre\ U\ x} \qquad \frac{Pre\ U\ x \quad y \in U, y =_{T^{in}} x \vdash P}{P}$$

□

2 W-Refinement

In this section we review *W-refinement* (written $U_0 \sqsupseteq_w U_1$, named for Jim Woodcock): this notion, adapted from [5], is introduced by showing how it can be formalised in (the extended) $\mathcal{Z}_C$.
The *lifted* totalisation of a set of bindings can be defined as follows:

Definition 2.1

$$\overset{\bullet}{U}^{\mathbb{P}\,T} = \{z^T \mid Pre\ U\ z \Rightarrow z \in U\}$$

Then we can prove the following.

Proposition 2.2 *The following introduction and elimination rules are derivable for lifted totalised sets.*

$$\frac{Pre\ U\ t \vdash t \in U}{t \in \overset{\bullet}{U}}\ (\bullet^+)$$

and:

$$\frac{t \in \overset{\bullet}{U} \quad Pre\ U\ t}{t \in U}\ (\bullet^-)$$

□

Proposition 2.3

$$(i) \quad U \subseteq \overset{\bullet}{U}$$
$$(ii) \quad \perp^T \in \overset{\bullet}{U}^{\mathbb{P}\,T}$$
$$(iii) \quad \neg Pre\ U^{\mathbb{P}(T^{in} \curlyvee T^{out'})}\ t_0^{T^{in}} \Rightarrow t_0 \star t_1^{out'} \in \overset{\bullet}{U}$$

□

Propositions 2.3(i), (ii) and (iii) demonstrate that definition 2.1 is consistent with the intentions described in [5] chapter 16.

W-refinement is then defined so that $U_0 \sqsupseteq_w U_1 =_{df} \dot{U_0} \subseteq \dot{U_1}$. Obvious introduction and elimination rules, written ($\sqsupseteq_w{}^+$) and ($\sqsupseteq_w{}^-$) follow from this.

3 S-Refinement

This notion can be captured by forcing the refinement relation to hold *exactly* when preconditions are not strengthened and post-conditions are not weakened. S-refinement, named for Mike Spivey, is written $U_0 \sqsupseteq_s U_1$ and is adapted from [4] and [2]. The definition is that which directly leads to the following rules:

Proposition 3.1

$$\frac{Pre\ U_1\ z \vdash Pre\ U_0\ z \quad Pre\ U_1\ z_0, Post\ z_0 \star z_1 \in U_0 \vdash z_0 \star z_1 \in U_1}{U_0 \sqsupseteq_s U_1}\ (\sqsupseteq_s{}^+)$$

$$\frac{U_0 \sqsupseteq_s U_1 \quad Pre\ U_1\ z}{Pre\ U_0\ z}\ (\sqsupseteq_{s\ 0}{}^-)$$

$$\frac{U_0 \sqsupseteq_s U_1 \quad Pre\ U_1\ z_0 \quad z_0 \star z_1 \in U_0}{z_0 \star z_1 \in U_1}\ (\sqsupseteq_{s\ 1}{}^-)$$

□

4 W-Refinement and S-Refinement are equivalent

We begin by showing that W-refinement satisfies the two S-refinement elimination rules. Firstly the rule for preconditions.

Proposition 4.1 *The following rule is derivable.*

$$\frac{U_0 \sqsupseteq_w U_1 \quad Pre\ U_1\ t}{Pre\ U_0\ t}$$

Proof. Consider the following derivation:

$$\dfrac{\dfrac{Pre\ U_1\ t \qquad \dfrac{U_0 \sqsupseteq_w U_1 \quad t\star \perp \in \dot{U_0}}{t\star \perp \in \dot{U_1}}}{t\star \perp \in U_1}}{\dfrac{false}{Pre\ U_0\ t}\ 1} \qquad \dfrac{\dfrac{}{\neg Pre\ U_0\ t}\ 1}{2.3(iii)}$$

□

Turning now to the second elimination rule in S-refinement.

Proposition 4.2 *The following rule is derivable.*

$$\frac{U_0 \sqsupseteq_w U_1 \quad Pre\ U_1\ t_0 \quad Post\ U_0\ t_0\ t_1}{Post\ U_1\ t_0\ t_1}$$

Proof.

$$\dfrac{\dfrac{Pre\ U_1\ t_0 \qquad \dfrac{\dfrac{}{t_0 \star t_1 \in U_0}}{\dfrac{U_0 \sqsupseteq_w U_1 \quad t_0 \star t_1 \in \overset{\bullet}{U_0}}{t_0 \star t_1 \in \overset{\bullet}{U_1}}}}{t_0 \star t_1 \in U_1}}{}$$

□

We are now in a position to prove the following theorem.

Theorem 4.3

$$\frac{U_0 \sqsupseteq_w U_1}{U_0 \sqsupseteq_s U_1}$$

Proof. This follows immediately, by $(\sqsupseteq_s^+)$, from propositions 4.1 and 4.2. □

We now show that S-refinement satisfies the W-elimination rule.

Proposition 4.4

$$\frac{U_0 \sqsupseteq_s U_1 \quad t \in \overset{\bullet}{U_0}}{t \in \overset{\bullet}{U_1}}$$

Proof.

$$\dfrac{\dfrac{U_0 \sqsupseteq_s U_1 \quad \dfrac{}{Pre\ U_1\ t}\ 2 \qquad \dfrac{t \in \overset{\bullet}{U_0} \qquad \dfrac{U_0 \sqsupseteq_s U_1 \quad \dfrac{}{Pre\ U_1\ t}\ 2}{Pre\ U_0\ t}}{t \in U_0}}{t \in U_1}}{t \in \overset{\bullet}{U_1}}\ 2$$

□

Theorem 4.5

$$\frac{U_0 \sqsupseteq_s U_1}{U_0 \sqsupseteq_w U_1}$$

Proof. This follows immediately, by $(\sqsupseteq_w^+)$, from proposition 4.4.

Theorems 4.3 and 4.5 together establish that the theories of S-refinement and W-refinement are equivalent. The model of schemas introduced in W-refinement not only totalises the schema as a set of bindings, it also introduces the $\perp^T$ values and extends the domains and co-domains accordingly. The totalisation then stipulates chaotic behaviour outside the precondition and additionally for the $\perp$ values.
Why is it necessary to include the new values? What are the consequences of totalisation *without* lifting? It is not possible to prove versions of proposition 4.1 with non-lifted totalisation. Note that the proofs of those results made explicit use of $\perp^T$ values. Indeed the following is an explicit counterexample:

Proposition 4.6 *Let* $Chaos =_{df} [\ |\ false]$, $True =_{df} [\ |\ true]$ *and write* $\overset{\diamond}{U}$ *for the* non-lifted *totalisation of* U. *Then*

$$\overset{\diamond}{True} = \overset{\diamond}{Chaos}$$

□

It is an immediate consequence that this more permissive notion of refinement does not, for example, insist that preconditions do not strengthen.

5 R-refinement

A third approach to refinement is to consider specifications as sets of implementations and then to define refinement as containment of implementations.
We begin, by way of an intermediate stage, by defining the set of total functions compatible with an operation schema.

Definition 5.1

$$\hat{U} =_{df} \{C \mid C \subseteq \overset{\bullet}{U} \wedge unicity(C) \wedge total(C)\}$$

Then we have:

Definition 5.2

$$f \in_r U =_{df} f \in \hat{U}$$

And then R-refinement is simply $U_0 \sqsupseteq_r U_1 =_{df} \hat{U_0} \subseteq \hat{U_1}$.

6 R-refinement and W-refinement are equivalent

We begin by showing that R-refinement satisfies the W-refinement elimination rule.

Proposition 6.1 *The following rule is derivable.*

$$\frac{U_0 \sqsupseteq_r U_1 \quad t \in \overset{\bullet}{U_0}}{t \in \overset{\bullet}{U_1}}$$

Proof.

$$\dfrac{\dfrac{z_0 \star z_1 \in \overset{\bullet}{U_0}}{\exists f \bullet z_0 \star z_1 \in f \wedge f \subseteq \overset{\bullet}{U_0} \wedge unicity(f) \wedge total(f)}\ AC \qquad \begin{matrix}\overset{\delta}{\vdots} \\ z_0 \star z_1 \in \overset{\bullet}{U_1}\end{matrix}}{z_0 \star z_1 \in \overset{\bullet}{U_1}}\ 1$$

Where δ stands for the following branch:

$$\dfrac{\dfrac{U_0 \sqsupseteq_r U_1 \qquad \dfrac{\dfrac{}{y \subseteq \overset{\bullet}{U_0}}\,1 \quad \dfrac{}{unicity(y)}\,1 \quad \dfrac{}{total(y)}\,1}{y \in \hat{U}_0}}{y \in \hat{U}_1}}{\dfrac{y \subseteq \overset{\bullet}{U_1} \qquad \dfrac{}{z_0 \star z_1 \in y}\,1}{z_0 \star z_1 \in \overset{\bullet}{U_1}}}$$

□

Theorem 6.2

$$\dfrac{U_0 \sqsupseteq_r U_1}{U_0 \sqsupseteq_w U_1}$$

Proof. This follows immediately, by $(\sqsupseteq_r^+)$, from proposition 6.1. □

We now show that W-refinement satisfies the R-refinement elimination rule.

Proposition 6.3

$$\dfrac{U_0 \sqsupseteq_w U_1 \quad f \in \hat{U}_0}{f \in \hat{U}_1}$$

Proof.

$$\dfrac{\dfrac{U_0 \sqsupseteq_w U_1 \qquad \dfrac{\dfrac{f \in \hat{U}_0}{f \subseteq \overset{\bullet}{U_0}} \quad \dfrac{}{z_0 \star z_1 \in f}\,1}{z_0 \star z_1 \in \overset{\bullet}{U_0}}}{z_0 \star z_1 \in \overset{\bullet}{U_1}}\,1}{\dfrac{f \subseteq \overset{\bullet}{U_1} \qquad \dfrac{f \in \hat{U}_0}{unicity(f)} \quad \dfrac{f \in \hat{U}_0}{total(f)}}{f \in \hat{U}_1}}$$

□

Then we have:

Theorem 6.4

$$\dfrac{U_0 \sqsupseteq_w U_1}{U_0 \sqsupseteq_r U_1}$$

Proof. This follows immediately, by $(\sqsupseteq_r^+)$, from proposition 6.3. □

Theorems 6.2 and 6.4 together demonstrate that W-refinement and R-refinement are equivalent.

7 F-refinement

We define the notion of F-implementation as follows:

Definition 7.1

$$f \Subset_f U =_{df} (\forall z \bullet Pre\ U\ z \Rightarrow z \star (f\ z) \in U) \wedge tfun(f)$$

Then we can prove the following.

Proposition 7.2 *The following introduction and elimination rules are derivable.*

$$\frac{Pre\ U\ z \vdash z \star (f\ z) \in U \quad tfun(f)}{f \Subset_f U}\ (\Subset_f^+)$$

$$\frac{f \Subset_f U \quad Pre\ U\ z}{z \star (f\ z) \in U}\ (\Subset_{f_0}^-) \qquad \frac{f \Subset_f U}{tfun(f)}\ (\Subset_{f_1}^-)$$

□

Proposition 7.3

$$\begin{array}{ll} (i) & f \Subset_f U \Rightarrow f \subseteq \overset{\bullet}{U} \\ (ii) & f \Subset_f U \Rightarrow unicity(f) \\ (iii) & f \Subset_f U \Rightarrow total(f) \end{array}$$

□

Theorem 7.4

$$\frac{f \Subset_f U}{f \Subset_r U}$$

Proof. This follows immediately, by $(\Subset_r^+)$, from proposition 7.3 (i), (ii) and (iii). □

Proposition 7.5

$$\begin{array}{ll} (i) & f \Subset_r U \wedge Pre\ U\ z \Rightarrow z \star (f\ z) \in U \\ (ii) & f \Subset_r U \Rightarrow tfun(f) \end{array}$$

□

Theorem 7.6

$$\frac{f \Subset_r U}{f \Subset_f U}$$

Proof. follows immediately, by $(\Subset_f^+)$, from proposition 7.5 (i) and (ii). □

Then we have: $f \Subset_f U \Leftrightarrow f \Subset_r U$, followed immediately from 7.4 and 7.6.
We define $\sqsupseteq_f$ using $\Subset_f$ in the obvious way. So we get the following corollary:

Corollary 7.7

$$U_0 \sqsupseteq_w U_1 \Leftrightarrow U_0 \sqsupseteq_f U_1$$

□

8 Conclusions

The analysis we have provided explores three, *prima facie* different approaches to operation refinement: one based on the completion of a relation, one based directly weakening preconditions and strengthening postconditions and the third based on sets of reasonable deterministic implementations. Perhaps surprisingly these three approaches turn out to be equivalent. The proofs demonstrate clearly the nature of the relationships involved. Of special note, our analysis explains exactly why lifting, as well as totalisation is required in the relation completion approach.

In future work we will explore similar relationships for notions of refinement based on the alternative notion of preconditions: that they are firing conditions and may not be weakened. It is, furthermore, well-known that the relational approach to refinement does not provide good monotonicity properties for the schema operators of Z. This must be fully investigated in the current framework, and alternative models developed which have more useful properties.

This work was partially supported by the British Council and the EPSRC (grant GR/L57913).

References

[1] M. C. Henson and S. Reeves. Investigating Z. *Journal of Logic and Computation*, 10(1):1–30, 2000.

[2] S. King. Z and the Refinement Calculus. In D. Bjørner, C. A. R. Hoare, and H. Langmaack, editors, *VDM '90 VDM and Z—Formal Methods in Software Development*, volume 428 of *Lecture Notes in Computer Science*, pages 164–188. Springer-Verlag, April 1990.

[3] B. Potter, J. Sinclair, and D. Till. *An introduction to formal specification and Z.* Prentice Hall, 2nd. edition, 1996.

[4] J. M. Spivey. *The Z notation: A reference manual.* Prentice Hall, 1989.

[5] J. Woodcock and J. Davies. *Using Z: Specification, Refinement and Proof.* Prentice Hall, 1996.

A The specification logic $\mathcal{Z}_\mathcal{C}$

In this appendix we shall describe the simple specification logic $\mathcal{Z}_\mathcal{C}$ from [1]. This is included for convenience only and the reader may need to consult the earlier paper at least in order to fully understand our notational and meta-notational conventions.

$\mathcal{Z}_\mathcal{C}$is a typed theory in which the types of higher-order logic are extended with *schema types* which are unordered, label-indexed tuples. For example, if the T_i are types and the l_i are labels (constants) then:

$$[\cdots l_i : T_i \cdots]$$

is a (schema) type. The symbols $\preceq$, $\curlywedge$ and $\curlyvee$ denote the *schema subtype* relation, and the operations of *schema type intersection* and (compatible) *schema type union.* We let U (with diacriticals when necessary) range over operation schema expressions. These are sets of bindings linking, as usual before observations with after observations. We can always, then, write the type of such operation schemas as $\mathbb{P}(T^{in} \curlyvee T^{out'})$ where T^{in} is the type of the input sub-binding and $T^{out'}$ is the type of the output sub-binding. We will also write bindings belonging to operation schemas in partitioned form: $z_0^{T^{in}} \star z_1^{T^{out'}} \in U^{\mathbb{P}(T^{in} \curlyvee T^{out'})}$. In this way reasoning in Z becomes no more complex than reasoning with binary relations.

Amongst the usual terms such as bindings, sets, pairs and numbers *etc.* we have the terms $\bot^T$ at every type.

The judgements of the logic have the form $\Gamma \vdash_{\mathcal{Z}_\mathcal{C}} P$ where Γ is a set of formulæ.

The logic is presented as a natural deduction system *in sequent form.* Derivations in the logic, above, were presented in *pure* natural deduction form. We omit the rules which establish the underlying classical logic.

All data (entailment symbol, contexts, type *etc.*) which remains unchanged by a rule are omitted. In the rule $(\exists^-)$, the variable y may not occur in C, P_0, P_1 nor any other assumption. We begin with the usual rules for $\mathcal{Z}_\mathcal{C}$.

$$\frac{}{\langle\!|\, \cdots l_i \Rrightarrow t_i \cdots \,|\!\rangle.l_i = t_i}\ (\Rrightarrow^=_0) \qquad \frac{}{\langle\!|\, \cdots l_i \Rrightarrow t.l_i \cdots \,|\!\rangle = t^{[\cdots l_i : T_i \cdots]}}\ (\Rrightarrow^=_1)$$

$$\frac{}{(t, t').1 = t}\ ((\,)^=_0) \qquad \frac{}{(t, t').2 = t'}\ ((\,)^=_1) \qquad \frac{}{(t.1, t.2) = t}\ ((\,)^=_2)$$

$$\frac{P[z/t]}{t \in \{z \mid P\}}\ (\{\}^+) \qquad \frac{t \in \{z \mid P\}}{P[z/t]}\ (\{\}^-)$$

$$\frac{t_0 \equiv t_1}{t_0 = t_1}\ (ext) \qquad \frac{t^T.l_i = t_i}{(t \upharpoonright T').l_i = t_i}\ (\upharpoonright^=) \quad (l_i \in \alpha T';\ T' \preceq T)$$

where

$$t_0 \equiv t_1 =_{df} \forall z \in t_0 \bullet z \in t_1 \wedge \forall z \in t_1 \bullet z \in t_0$$

The usual side-conditions apply to rule $(\exists^-)$.
The symmetry and transitivity of equality and numerous *equality congruence* rules for the various term forming operations are all derivable in view of rule (*sub*).
In addition to these rules, the extended theory we need in this paper has the following axioms which ensure that the $\perp^T$ values interact appropriately:

$$\frac{}{\perp^{[l_0:T_0 \cdots l_n:T_n]} = \langle\!|\, l_0 \Rrightarrow \perp^{T_0} \cdots l_n \Rrightarrow \perp^{T_n} \,|\!\rangle} \qquad \frac{}{\perp^{T_0 \times T_1} = (\perp^{T_0}, \perp^{T_1})} \qquad \frac{}{\perp^{\mathbb{P}\, T} = \{z^T \mid z = \perp^T\}}$$

In [1] we showed how to extend $\mathcal{Z}_\mathcal{C}$ to the schema calculus. For example:

$$[S \mid P] =_{df} \{z^T \mid z \in S \wedge z.P\}$$

defines atomic schemas, and:

$$S_0^{\mathbb{P}\, T_0} \vee S_1^{\mathbb{P}\, T_1} =_{df} \{z^{T_0 \curlyvee T_1} \mid z \in S_0 \vee z \in S_1\}$$

defines schema disjunction.
In the extended theory we have described, it is crucially important that the $\perp^T$ values are excluded from the schemas. This is achieved by defining the *natural carrier sets* for each type:
The *natural carriers* for each type are defined by closing:

$$\mathbb{N} =_{df} \{z^{\mathbb{N}} \mid z \neq \perp^{\mathbb{N}} \wedge z = z\}$$

under the operations of cartesian product, powerset and schema set. The notational ambiguity does not introduce a problem, since only a set can appear in a term or proposition, and only a type can appear as a superscript.
The *full carriers* are defined for all types as follows:

$$T_\perp =_{df} \{z^T \mid z = z\}$$

Given this, we modify the defintions for the schema expressions accordingly, for example:

$$[S \mid P] =_{df} \{z \in T \mid z \in S \wedge z.P\}$$

defines atomic schemas, and:

$$S_0^{\mathbb{P}\, T_0} \vee S_1^{\mathbb{P}\, T_1} =_{df} \{z \in (T_0 \curlyvee T_1) \mid z \in S_0 \vee z \in S_1\}$$

defines schema disjunction.
As a consequence, all bindings in the schema calculus are *hereditarily $\perp$-free*. We used this in the proof of theorem 4.1, for example.
In many contexts we need to compare bindings over a common restricted type.

Definition A.1 *Let $T \preceq T_0$ and $T \preceq T_1$.*

$$t_0^{T_0} =_T t_1^{T_1} =_{df} t_0 \upharpoonright T = t_1 \upharpoonright T$$