Six theories of operation refinement for partial relation semantics

Moshe Deutsch[1], Martin C. Henson[1], Steve Reeves[2]

[1]Department of Computer Science, University of Essex, UK.
[2]Department of Computer Science, University of Waikato, New Zealand.
mdeuts@essex.ac.uk, hensm@essex.ac.uk, stever@cs.waikato.ac.nz

Abstract. In this paper we analyse total correctness operation refinement on a partial relation semantics for specification. In particular we show that three theories: a relational completion approach, a proof-theoretic approach and a functional models approach, are all equivalent. This result holds whether or not preconditions are taken to be minimal or fixed conditions for establishing the postcondition.

Keyword: Specification Language; Specification Logic; Refinement;

1 Introduction

In this paper we provide a mathematical analysis of total correctness operation refinement for partial relation semantics. An important example of such a semantics is the specification language Z. We will examine refinement when preconditions are interpreted as minimal conditions for establishing the postconditions (they may be weakened) or fixed conditions (they are firing or trigger conditions). The former approach is covered in sections 3 and 4, and the latter in sections 5 and 6.
Our aim is to understand better the various techniques which have been proposed and to link them carefully to what appear to be *prima facie* alternative approaches. In particular we will look at the standard relational completion approaches (see for example [10] and [2]) and relate them to a variety of proof theoretic approaches and to frameworks in which specifications are interpreted to be sets of implementations (rather in the spirit of the uses to which Matin-Löf's theory has been put [6], though our investigation takes place in classical logic).
Such an investigation becomes possible in virtue of the logic for Z reported in, for example, [4] and a novel and simple technique of rendering all the theories of refinement in an unusual proof-theoretic form: sets of introduction and elimination rules. This leads to a uniform and simple method for proving the various equivalence results. As such it contrasts with the more semantic based techniques employed in [1].
Our paper concludes with a review of what has been established and an agenda for further investigation.

2 Preliminaries

In this first section we will revise a little Z logic, settling some notational conventions in the process. Additional detail can be found in appendix A and in [4].
In [4] Z schemas, and operation schemas in particular, were formalised as sets of bindings. This captures the informal account to be found in the literature (*e.g.* [3], [10]). In this paper we will use the meta-variable U (with decorations) to range over operation schema. As an example, consider the operation schema:

$$\begin{array}{|l}
\multicolumn{1}{l}{\mathit{Ex}_0} \\ \hline
\mathtt{x}, \mathtt{x}' : \mathbb{N} \\ \hline
\mathtt{x} = 0 \\
\mathtt{x}' < 10 \\ \hline
\end{array}$$

Ex_0 has the the type $\mathbb{P}[\mathtt{x} : \mathbb{N}, \mathtt{x}' : \mathbb{N}]$, and is understood to be a set of bindings of schema type $[\mathtt{x} : \mathbb{N}, \mathtt{x}' : \mathbb{N}]$. Recall that unprimed labels (such as $\mathtt{x}$) are understood to be observations of the state before the operation, whereas primed labels (such as $\mathtt{x}'$) are observations of the state afterwards. Each operation schema U will have a type of the form $\mathbb{P}\ T$ where T is a schema type. The type T can, additionally, always be partitioned as the (compatible) union of its input (or before) type T^{in}, and its output (or after) type $T^{out'}$. That is, $T = T^{in} \curlyvee T^{out'}$. For the schema Ex_0 we have $T^{in} = [\mathtt{x} : \mathbb{N}]$ and $T^{out'} = [\mathtt{x}' : \mathbb{N}]$. In this paper, since we are only dealing with operation refinement, we can assume that all operation schemas have the type $\mathbb{P}\ T$ where $T = T^{in} \curlyvee T^{out'}$. With this in place we can omit the type superscripts in most places in the sequel.

Definition 1 (Semantics for atomic schemas).

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

Note that this definition, in which the operation schema has been written horizontally, draws bindings from the *natural carrier* of the type T. As a consequence, writing $t(\bot)$ for any term of the appropriate type which contains an instance of a constant $\bot$, we have:

Lemma 1.

$$\frac{t(\bot) \in U}{\mathit{false}}$$

□

The bindings $\langle\!|\ x \Rrightarrow 0, x' \Rrightarrow n\ |\!\rangle$, where $n < 10$, are all elements of Ex_0. In fact there are no other elements in this case. We can formalise the idea of the *preconditions* of the schema (domain of the relation the schema denotes) to express the partiality involved.

Definition 2. *Let $T^{in} \preceq V$.*

$$Pre\ \ U\ \ x^V =_{df} \exists\, z \in U \bullet x =_{T^{in}} z$$

Proposition 1. *The following introduction and elimination rules are immediately derivable for preconditions:*

$$\frac{t_0 \in U \quad t_0 =_{T^{in}} t_1}{Pre\ \ U\ \ t_1} \qquad \frac{Pre\ \ U\ \ t \quad y \in U, y =_{T^{in}} t \vdash P}{P}$$

where y is fresh. □

Clearly, the precondition of Ex_0 is not (and for operation schemas in general, will not be) the whole of $[x : \mathbb{N}]$ (in general, T^{in}). In this sense operation schemas denote partial relations.
Naturally, an immediate question arises: what does it mean for one operation schema to refine another? More generally, we are asking: what does it mean for one partial relation to refine another?

3 Refinement with preconditions considered minimal

We begin by introducing three distinct notions of refinement based on three distinct answers to the questions above and then we go on to compare them. This serves to illuminate them all, particularly the notion based on the lifted-totalisation (see below) which is the *de facto* standard for Z.

3.1 S-refinement

In this section we introduce a purely proof theoretic characterisation of refinement, which is closely connected to refinement as introduced by Spivey in, for example, [8] and as discussed in [5] and [7].
This notion is based on two basic observations regarding the properties one expects in a refinement: firstly, that a refinement may involve the reduction of non-determinism; secondly that, if preconditions are minimal, a refinement may also involve the expansion of the domain of definition. Put another way, we have a refinement providing that *postconditions do not weaken* (we do not permit an increase in non-determinism in a refinement) and that *preconditions do not strengthen* (we do not permit requirements in the domain of definition to disappear in a refinement).
This notion can be captured by forcing the refinement relation to hold *exactly* when these conditions apply. S-refinement is written $U_0 \sqsupseteq_s U_1$ and is given by the definition that leads directly to the following rules:

Proposition 2. *Let z, z_0, z_1 be fresh variables.*

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

$$\frac{U_0 \sqsupseteq_s U_1 \quad Pre\ \ U_1\ \ t}{Pre\ \ U_0\ \ t}\ (\sqsupseteq_s{}_0^-)$$

$$\frac{U_0 \sqsupseteq_s U_1 \quad Pre\ \ U_1\ \ t_0 \quad t_0 \star t_1' \in U_0}{t_0 \star t_1' \in U_1}\ (\sqsupseteq_s{}_1^-)$$

□

This theory does not depend on, and makes no reference to, the $\perp$ value. It can be formalised in the core theory $\mathcal{Z_C}$.

3.2 The chaotic relational completion

In this section we review $W_\bullet$*-refinement* (written $U_0 \sqsupseteq_{w_\bullet} U_1$): this notion, adapted from, for example, [10], is based on a relational completion operator. For notational convenience we will write $T^\star$ for the set $T_\perp^{in} \star T_\perp^{out'}$.
The *lifted totalisation* of a set of bindings can be defined as follows:

Definition 3.

$$\overset{\bullet}{U} =_{df} \{z \in T^\star \mid Pre\ \ U\ \ z \Rightarrow z \in U\}$$

Proposition 3. *The following introduction and elimination rules are derivable for lifted totalised sets:*

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

and:

$$\frac{t \in \overset{\bullet}{U} \quad Pre\ \ U\ \ t}{t \in U}\ (\bullet_0^-) \qquad \frac{t \in \overset{\bullet}{U}}{t \in T^\star}\ (\bullet_1^-)$$

□

Lemma 2. *The following are derivable:*

$$\frac{}{U \subseteq \overset{\bullet}{U}}\ (i) \qquad \frac{}{\perp \in \overset{\bullet}{U}}\ (ii) \qquad \frac{\neg Pre\ \ U\ \ t \quad t \in T^\star}{t \in \overset{\bullet}{U}}\ (iii)$$

□

Lemmas 2(i), (ii) and (iii) demonstrate that definition 3 is consistent with the intentions described in [10] chapter 16: the underlying partial relation is contained in the completion; the undefined element is present in the relation, and more generally, each value outside the precondition maps to every value in the range of the relation.
$W_\bullet$*-refinement* is defined as follows:

Definition 4.

$$U_0 \sqsupseteq_{w_\bullet} U_1 =_{df} \overset{\bullet}{U_0} \subseteq \overset{\bullet}{U_1}$$

Obvious introduction and elimination rules follow from this.

3.3 F-refinement

To a logician, a specification resembles a theory; so a natural question is: what are the models of the theory? A computer scientist may ask a closely related question: when is a program an implementation of the specification? We will, in this section, consider deterministic programs and model them as (total) functions.
From the logical perspective, we are interested in all the models of a theory, so given a putative model g and a theory U, we would be inclined to write:

$$g \models U$$

to represent the statement that g is a model of U. Within our application area in computer science we might prefer to read this as a relation of *implementation*. To signal this interpretation we shall in fact write this judgement as:

$$g \Subset U$$

to be pronounced "g implements (is an implementation of) U".
Our third approach to refinement is to consider specifications as sets of implementations and then to define refinement as containment of implementations.

Definition 5.

$$g \Subset_f U =_{df} (\forall z \in T_{\perp}^{in} \bullet \mathit{Pre}\ U\ z \Rightarrow z \star (g\ z)' \in U) \land g \in T_{\perp}^{in} \to T_{\perp}^{out'}$$

Then we can prove the following.

Proposition 4. *The following introduction and elimination rules are derivable:*

$$\frac{z \in T_{\perp}^{in}, \mathit{Pre}\ U\ z \vdash z \star (g\ z)' \in U \quad g \in T_{\perp}^{in} \to T_{\perp}^{out'}}{g \Subset_f U}\ (\Subset_f^+)$$

where z is a fresh variable.

$$\frac{g \Subset_f U \quad \mathit{Pre}\ U\ t \quad t \in T_{\perp}^{in}}{t \star (g\ t)' \in U}\ (\Subset_{f_0}^-) \qquad \frac{g \Subset_f U}{g \in T_{\perp}^{in} \to T_{\perp}^{out'}}\ (\Subset_{f_1}^-)$$

□

This is sufficient technical development to allow us to explore refinement. We can answer the question: when is U_0 a refinement of U_1? A reasonable answer is: when any implementation of U_0 is also an implementation of U_1. After all, we wish to be able to replace any specification U_1 by its refinement U_0, and if all potential implementations of the latter are implementations of this former we are quite safe. Thus we are led to:

Definition 6.

$$\widehat{U} =_{df} \{z \mid z \Subset_f U\}$$

Then we have F-refinement ("F" for function).

Definition 7.

$$U_0 \sqsupseteq_f U_1 =_{df} \widehat{U_0} \subseteq \widehat{U_1}$$

Obvious introduction and elimination rules for F-refinement follow from this definition.

4 Three equivalent theories

In this section we demonstrate that our three theories of refinement are all equivalent. In doing this we will see clearly the critical role that the $\perp$ value plays.

We shall be showing that all judgements of refinement in one theory are contained among the refinements sanctioned by another. Such results will always be established proof-theoretically. Specifically we will show that the refinement relation of a theory T_0 satisfies the elimination rule (or rules) for refinement of another theory T_1. Since the elimination rules and introduction rules of a theory enjoy the usual symmetry property, this is sufficient to show that all T_0-refinements are also T_1 refinements.

4.1 W$_\bullet$-refinement and S-refinement are equivalent

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

Proposition 5. *The following rule is derivable:*

$$\frac{U_0 \sqsupseteq_{w_\bullet} U_1 \quad Pre\ U_1\ t}{Pre\ U_0\ t}$$

Proof

Consider the following derivation:

$$\dfrac{\dfrac{\dfrac{U_0 \sqsupseteq_{w_\bullet} U_1 \qquad \dfrac{\dfrac{}{\neg Pre\ U_0\ t}\,(1) \qquad \dfrac{\dfrac{Pre\ U_1\ t}{t_0 \in T^{in}_{\perp}} \qquad \dfrac{}{\perp' \in T^{out'}_{\perp}}}{t_0 \star \perp' \in T^{\star}}}{t_0 \star \perp' \in \overset{\bullet}{U_0}}\,(2(iii))}{t_0 \star \perp' \in \overset{\bullet}{U_1}} \qquad Pre\ U_1\ t}{\dfrac{\dfrac{t_0 \star \perp' \in U_1}{false}\,(1)}{Pre\ U_0\ t}\,(1)}$$

where $t_0 =_{df} t \restriction T^{in}$. □

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

Proposition 6. *The following rule is derivable:*

$$\frac{U_0 \sqsupseteq_{w_\bullet} U_1 \quad \mathit{Pre}\ U_1\ t_0 \quad t_0 \star t_1' \in U_0}{t_0 \star t_1' \in U_1}$$

Proof

$$\cfrac{\cfrac{U_0 \sqsupseteq_{w_\bullet} U_1 \quad \cfrac{t_0 \star t_1' \in U_0}{t_0 \star t_1' \in \overset{\bullet}{U_0}}\ (2(i))}{t_0 \star t_1' \in \overset{\bullet}{U_1}} \qquad \mathit{Pre}\ U_1\ t_0}{t_0 \star t_1' \in U_1}$$

□

Theorem 1.

$$\frac{U_0 \sqsupseteq_{w_\bullet} U_1}{U_0 \sqsupseteq_s U_1}$$

□

Proof

This follows immediately, by $(\sqsupseteq_s^+)$, from propositions 5 and 6 .[1]
□

We now show that S-refinement satisfies the $\mathrm{W}_\bullet$-elimination rule.

Proposition 7.

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

Proof

$$\dfrac{\dfrac{t \in \overset{\bullet}{U}_0}{t \in T^{\star}} \qquad \dfrac{U_0 \sqsupseteq_s U_1 \qquad \overline{Pre\ U_1\ t}^{\,(1)} \qquad \dfrac{t \in \overset{\bullet}{U}_0 \qquad \dfrac{U_0 \sqsupseteq_s U_1 \qquad \overline{Pre\ U_1\ t}^{\,(1)}}{Pre\ U_0\ t}}{t \in U_0}}{t \in U_1}}{t \in \overset{\bullet}{U}_1}\,(1)$$

□

Theorem 2.

$$\dfrac{U_0 \sqsupseteq_s U_1}{U_0 \sqsupseteq_{w_\bullet} U_1}$$

□

Theorems 1 and 2 together establish that the theories of S-refinement and W$_\bullet$-refinement are equivalent.

4.2 F-refinement and W$_\bullet$-refinement are equivalent (in $\mathcal{Z}_{\mathcal{C}}^{\perp}$ + AC)

R-refinement We begin this analysis by defining, by way of an intermediate stage, the set of total functions compatible with an operation schema. This forms a bridge between F-refinement and W$_\bullet$-refinement.

Definition 8.

$$\overline{U} =_{df} \{z \in T_{\perp}^{in} \rightarrow T_{\perp}^{out'} \mid z \subseteq \overset{\bullet}{U}\,\}$$

Then we have:

Definition 9.

$$g \Subset_r U =_{df} g \in \overline{U}$$

And then R-refinement is simply: $U_0 \sqsupseteq_r U_1 =_{df} \overline{U_0} \subseteq \overline{U_1}$ with the obvious introduction and elimination rules.

R-refinement and W$_\bullet$-refinement are equivalent We begin by showing that R-refinement satisfies the W$_\bullet$-refinement elimination rule.

Proposition 8. *The following rule is derivable:*

$$\dfrac{U_0 \sqsupseteq_r U_1 \qquad t \in \overset{\bullet}{U}_0}{t \in \overset{\bullet}{U}_1}$$

Proof

The proof requires the axiom of choice (see the step labelled (AC) below).

$$\dfrac{\dfrac{t \in \overset{\bullet}{U}_0}{\exists\, g \in T_\perp^{in} \to T_\perp^{out'} \bullet t \in g \wedge g \subseteq \overset{\bullet}{U}_0}\,(AC) \qquad \begin{matrix}\delta \\ \vdots \\ t \in \overset{\bullet}{U}_1\end{matrix}}{t \in \overset{\bullet}{U}_1}\,(1)$$

where δ is:

$$\dfrac{\dfrac{\dfrac{\dfrac{U_0 \sqsupseteq_r U_1 \qquad \dfrac{\dfrac{}{y \in T_\perp^{in} \to T_\perp^{out'}}\,(1) \qquad \dfrac{}{y \subseteq \overset{\bullet}{U}_0}\,(1)}{y \in \overline{U_0}}}{y \in \overline{U_1}}}{y \subseteq \overset{\bullet}{U}_1} \qquad \dfrac{}{t \in y}\,(1)}{t \in \overset{\bullet}{U}_1}}{}$$

$\square$

From this we have:

Theorem 3.

$$\frac{U_0 \sqsupseteq_r U_1}{U_0 \sqsupseteq_{w_\bullet} U_1}$$

We now show that W$_\bullet$-refinement satisfies the R-refinement elimination rule.

Proposition 9.

$$\frac{U_0 \sqsupseteq_{w_\bullet} U_1 \quad g \in \overline{U_0}}{g \in \overline{U_1}}$$

Proof

$$\begin{prooftree}
\AxiomC{$g \in \overline{U_0}$}
\UnaryInfC{$g \in T_\perp^{in} \to T_\perp^{out'}$}
\AxiomC{$U_0 \sqsupseteq_{w_\bullet} U_1$}
\AxiomC{$g \in \overline{U_0}$}
\UnaryInfC{$g \subseteq \overset{\bullet}{U_0}$}
\AxiomC{$\overline{t \in g}\ (1)$}
\BinaryInfC{$t \in \overset{\bullet}{U_0}$}
\BinaryInfC{$t \in \overset{\bullet}{U_1}$}
\RightLabel{(1)}
\UnaryInfC{$g \subseteq \overset{\bullet}{U_1}$}
\BinaryInfC{$g \in \overline{U_1}$}
\end{prooftree}$$

□

Theorem 4.

$$\frac{U_0 \sqsupseteq_{w_\bullet} U_1}{U_0 \sqsupseteq_r U_1}$$

□

Theorems 3 and 4 together demonstrate that W$_\bullet$-refinement and R-refinement are equivalent.

R-refinement and F-refinement are equivalent In this case we show that the notions of *implementation* (rather than refinement) are equivalent by the same strategy involving elimination rules. We first establish that F-implementation implies R-implementation:

Proposition 10. *The following rules are derivable:*

$$\frac{g \Subset_f U}{g \subseteq \overset{\bullet}{U}} \qquad \frac{g \Subset_f U}{g \in T_\perp^{in} \to T_\perp^{out'}}$$

Proof

$$\dfrac{\dfrac{\dfrac{\dfrac{g \Subset_f U}{g \in T_{\perp}^{in} \to T_{\perp}^{out'}} \qquad \dfrac{}{z_0 \star z_1' \in g}\,(1)}{z_0 \star z_1' \in T^*} \qquad \dfrac{\dfrac{g \Subset_f U \quad \dfrac{}{Pre\;\; U\;\; z_0}\,(2) \quad \dfrac{\dfrac{}{Pre\;\; U\;\; z_0}\,(2)}{z_0 \in T_{\perp}^{in}}}{z_0 \star (g\, z_0)' \in U} \qquad \begin{matrix}\delta\\ \vdots\\ z_1' = (g\;\; z_0)'\end{matrix}}{z_0 \star z_1' \in U}}{z_0 \star z_1' \in \overset{\bullet}{U}}\,(2)}{g \subseteq \overset{\bullet}{U}}\,(1)$$

where δ is:
$$\dfrac{\dfrac{g \Subset_f U}{g \in T_{\perp}^{in} \to T_{\perp}^{out'}} \qquad \dfrac{}{z_0 \star z_1' \in g}\,(1)}{z_1' = (g\;\; z_0)'}$$

The second rule is immediate. □

Theorem 5.
$$\dfrac{g \Subset_f U}{g \Subset_r U}$$

□

Now we show that R-implementation implies F-implementation.

Proposition 11.
$$\dfrac{g \Subset_r U \quad Pre\;\; U\;\; t \quad t \in T_{\perp}^{in}}{t \star (g\;\; t)' \in U} \qquad \dfrac{g \Subset_r U}{g \in T_{\perp}^{in} \to T_{\perp}^{out'}}$$

Proof

$$\dfrac{\dfrac{\dfrac{g \Subset_r U}{g \subseteq \overset{\bullet}{U}} \quad \dfrac{\dfrac{g \Subset_r U}{g \in T_{\perp}^{in} \to T_{\perp}^{out'}} \quad t \in T_{\perp}^{in}}{t \star (g\;\; t)' \in g}}{t \star (g\;\; t)' \in \overset{\bullet}{U}} \qquad Pre\;\; U\;\; t}{t \star (g\;\; t)' \in U}$$

The second rule is immediate. □

Theorem 6.

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

□

Then, from 5 and 6, we see that the two notions of implementation are equivalent. Hence, so are the two notions of refinement.
Despite their superficial dissimilarity, all three theories are, then, equivalent. We will examine in section 7 some consequences of these results.

5 Refinement with preconditions considered fixed

We now introduce three further notions of refinement; in this case where non-determinism may be reduced but where the preconditions are considered fixed.

5.1 SP-refinement

This is an alternative proof theoretic characterisation of refinement, which is closely connected to refinement in the *behavioural approach*, as discussed, for example, in [2] and [9].
This special case of S-Refinement may involve reduction of non determinism but insists on the *stability of the preconditions.* SP-refinement is written $U_0 \sqsupseteq_{sp} U_1$ and is given by the definition that leads directly to the following rules:

Proposition 12. *Let z, z_0, z_1 be fresh variables.*

$$\frac{\mathit{Pre}\ U_1\ z \vdash \mathit{Pre}\ U_0\ z \quad z_0 \star z_1' \in U_0 \vdash z_0 \star z_1' \in U_1}{U_0 \sqsupseteq_{sp} U_1}\ (\sqsupseteq^+_{sp})$$

$$\frac{U_0 \sqsupseteq_{sp} U_1 \quad \mathit{Pre}\ U_1\ t}{\mathit{Pre}\ U_0\ t}\ (\sqsupseteq^-_{sp_0}) \qquad \frac{U_0 \sqsupseteq_{sp} U_1 \quad t_0 \star t_1' \in U_0}{t_0 \star t_1' \in U_1}\ (\sqsupseteq^-_{sp_1})$$

5.2 The abortive relational completion

In this section we review *$W_\Box$-refinement* (written $U_0 \sqsupseteq_{w_\Box} U_1$). This notion is based on a relational completion operator, but this time takes an *abortive approach* with respect to values outside the precondition. The *abortive lifted totalisation* of a set of bindings is defined as follows:

Definition 10.

$$\overset{\Box}{U} =_{df} \{z_0 \star z_1' \in T^\star \mid z_0 \star z_1' \in U \lor (\neg\ \mathit{Pre}\ U\ z_0 \land z_1' = \perp')\}$$

Proposition 13. *The following introduction and elimination rules are derivable:*

$$\frac{t_0 \star t_1' \in U}{t_0 \star t_1' \in \overset{\scriptscriptstyle\square}{U}}\ (\square_0^+) \qquad \frac{t_0 \star t_1' \in T^\star \quad \neg\, Pre\ U\ t_0 \quad t_1' =\perp'}{t_0 \star t_1' \in \overset{\scriptscriptstyle\square}{U}}\ (\square_1^+)$$

$$\frac{t_0 \star t_1' \in \overset{\scriptscriptstyle\square}{U} \quad t_0 \star t_1' \in U \vdash P \quad \neg\, Pre\ U\ t_0, t_1' =\perp' \vdash P}{P}\ (\square_0^-)$$

$$\frac{t_0 \star t_1' \in \overset{\scriptscriptstyle\square}{U}}{t_0 \star t_1' \in T^\star}\ (\square_1^-)$$

□

Note that it is, sometimes, useful to use the following version of $(\square^+)$ rule (e.g. in the proof of proposition 21), which is based upon implication introduction, rather than disjunction introduction.

Proposition 14.

$$\frac{t_0 \star t_1' \in T^\star \quad Pre\ U\ t_0 \vee t_1' \neq\perp' \vdash t_0 \star t_1' \in U}{t_0 \star t_1' \in \overset{\scriptscriptstyle\square}{U}}\ (\square^+)$$

□

Lemma 3. *The following are derivable:*

$$\frac{}{\overset{\scriptscriptstyle\square}{U} \subseteq \overset{\bullet}{U}}\ (i) \qquad \frac{}{\perp \in \overset{\scriptscriptstyle\square}{U}}\ (ii) \qquad \frac{\neg\, Pre\ U\ t \quad t \in T_\perp^{in}}{t \star \perp' \in \overset{\scriptscriptstyle\square}{U}}\ (iii)$$

$$\frac{t_0 \star t_1' \in \overset{\scriptscriptstyle\square}{U} \quad t_1' \neq\perp'}{t_0 \star t_1' \in U}\ (iv) \qquad \frac{t_0 \star t_1' \in \overset{\scriptscriptstyle\square}{U} \quad t_0 =\perp}{t_1' =\perp'}\ (v)$$

□

$W_\square$-refinement is then defined as follows:

Definition 11.

$$U_0 \sqsupseteq_{w_\square} U_1 =_{df} \overset{\scriptscriptstyle\square}{U_0} \subseteq \overset{\scriptscriptstyle\square}{U_1}$$

Obvious introduction and elimination rules follow from this.

5.3 FP-refinement

Like F-refinement, *FP-refinement* considers specifications to be sets of implementations, and then we define refinement as containment of implementations. Unlike F-refinement, implementations abort outside the domain of definition, rather than behave chaotically.

Definition 12.

$$g \mathrel{\in\!\!\!\!\mid}_{fp} U =_{df} (\forall\, z \in T_{\perp}^{in} \bullet \mathit{Pre}\ U\ z \vee (g\ z)' \neq \perp' \Rightarrow z \star (g\ z)' \in U) \wedge g \in T_{\perp}^{in} \rightarrow T_{\perp}^{out'}$$

Then we can prove the following.

Proposition 15. *The following introduction and elimination rules are derivable:*

$$\frac{z \in T_{\perp}^{in}, \mathit{Pre}\ U\ z \vee (g\ z)' \neq \perp' \vdash z \star (g\ z)' \in U \qquad g \in T_{\perp}^{in} \rightarrow T_{\perp}^{out'}}{g \mathrel{\in\!\!\!\!\mid}_{fp} U}\ (\mathrel{\in\!\!\!\!\mid}^{+}_{fp})$$

where z is a fresh variable.

$$\frac{g \mathrel{\in\!\!\!\!\mid}_{fp} U \qquad t \in T_{\perp}^{in} \qquad \mathit{Pre}\ U\ t}{t \star (g\ t)' \in U}\ (\mathrel{\in\!\!\!\!\mid}^{-}_{fp_0}) \qquad \frac{g \mathrel{\in\!\!\!\!\mid}_{fp} U \qquad t \in T_{\perp}^{in} \qquad (g\ t)' \neq \perp'}{t \star (g\ t)' \in U}\ (\mathrel{\in\!\!\!\!\mid}^{-}_{fp_1})$$

$$\frac{g \mathrel{\in\!\!\!\!\mid}_{fp} U}{g \in T_{\perp}^{in} \rightarrow T_{\perp}^{out'}}\ (\mathrel{\in\!\!\!\!\mid}^{-}_{fp_2})$$

□

Definition 13.

$$\overset{\boxdot}{U} =_{df} \{z \mid z \mathrel{\in\!\!\!\!\mid}_{fp} U\}$$

Then we have FP-refinement.

Definition 14.

$$U_0 \sqsupseteq_{fp} U_1 =_{df} \overset{\boxdot}{U_0} \subseteq \overset{\boxdot}{U_1}$$

Obvious introduction and elimination rules for FP-refinement follow from this definition.

6 Three equivalent theories

In this section we demonstrate that this second set of three theories of refinement are all equivalent.

6.1 W$_\Box$-refinement and SP-refinement are equivalent

We begin by showing that W$_\Box$-refinement satisfies the two SP-refinement elimination rules. Firstly the rule for preconditions.

Proposition 16. *The following rule is derivable:*

$$\frac{U_0 \sqsupseteq_{w_\Box} U_1 \quad \mathit{Pre}\ U_1\ t}{\mathit{Pre}\ U_0\ t}$$

Proof

Consider the following derivation:

$$\dfrac{\dfrac{U_0 \sqsupseteq_{w_\Box} U_1 \qquad \dfrac{\dfrac{}{\neg\ \mathit{Pre}\ U_0\ t}\,(1) \quad \dfrac{\mathit{Pre}\ U_1\ t}{t \in T^{in}_{\perp}}}{t\star \perp' \in \overset{\Box}{U_0}}\,(3(iii))}{t\star \perp' \in \overset{\Box}{U_1}} \qquad \dfrac{\dfrac{}{t\star \perp' \in U_1}\,(2)}{\mathit{false}} \qquad \dfrac{\mathit{Pre}\ U_1\ t \quad \dfrac{}{\neg\ \mathit{Pre}\ U_1\ t}\,(2)}{\mathit{false}}}{\dfrac{\mathit{false}}{\mathit{Pre}\ U_0\ t}\,(1)}\,(2)$$

□

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

Proposition 17. *The following rule is derivable:*

$$\frac{U_0 \sqsupseteq_{w_\Box} U_1 \quad t_0 \star t_1' \in U_0}{t_0 \star t_1' \in U_1}$$

Proof

$$\dfrac{\dfrac{U_0 \sqsupseteq_{w_\Box} U_1 \quad \dfrac{t_0 \star t_1' \in U_0}{t_0 \star t_1' \in \overset{\Box}{U_0}}}{t_0 \star t_1' \in \overset{\Box}{U_1}} \qquad \dfrac{}{t_0 \star t_1' \in U_1}\,(1) \qquad \dfrac{\dfrac{t_0 \star t_1' \in U_0 \quad \dfrac{}{t_1' = \perp'}\,(1)}{\mathit{false}}}{t_0 \star t_1' \in U_1}}{t_0 \star t_1' \in U_1}\,(1)$$

□

Theorem 7.
$$\frac{U_0 \sqsupseteq_{w_\Box} U_1}{U_0 \sqsupseteq_{sp} U_1}$$

$\Box$

We now show that SP-refinement satisfies the W$_\Box$-elimination rule.

Proposition 18.
$$\frac{U_0 \sqsupseteq_{sp} U_1 \quad t_0 \star t_1' \in \overset{\Box}{U_0}}{t_0 \star t_1' \in \overset{\Box}{U_1}}$$

Proof

$$\dfrac{t_0 \star t_1' \in \overset{\Box}{U_0} \qquad \dfrac{\dfrac{U_0 \sqsupseteq_{sp} U_1 \quad \overline{t_0 \star t_1' \in U_0}\,^{(1)}}{t_0 \star t_1' \in U_1}}{t_0 \star t_1' \in \overset{\Box}{U_1}} \qquad \begin{array}{c}\beta\\ \vdots\\ t_0 \star t_1' \in \overset{\Box}{U_1}\end{array}}{t_0 \star t_1' \in \overset{\Box}{U_1}}\,(1)$$

where β stands for the following branch:

$$\dfrac{\dfrac{t_0 \star t_1' \in \overset{\Box}{U_0}}{t_0 \star t_1' \in T^\star} \qquad \dfrac{U_0 \sqsupseteq_{sp} U_1 \quad \overline{\neg\ Pre\ U_0\ t_0}\,^{(1)}}{\neg\ Pre\ U_1\ t_0} \qquad \overline{t_1' = \perp'}\,^{(1)}}{t_0 \star t_1' \in \overset{\Box}{U_1}}$$

$\Box$

Theorem 8.
$$\frac{U_0 \sqsupseteq_{sp} U_1}{U_0 \sqsupseteq_{w_\Box} U_1}$$

$\Box$

Theorems 7 and 8 together establish that the theories of SP-refinement and W$_\Box$-refinement are equivalent.

6.2 FP-refinement and W$_\Box$-refinement are equivalent (in $\mathcal{Z}_{\mathcal{C}}^{\perp}$ + AC)

RP-refinement As in section 4.2, we begin the analysis by defining the set of total functions compatible with an operation schema, which forms a bridge between FP-refinement and W$_\Box$-refinement.

Definition 15.

$$\widetilde{U} =_{df} \{z \in T_{\perp}^{in} \rightarrow T_{\perp}^{out'} \mid z \subseteq \overset{\Box}{U}\}$$

Then we have:

Definition 16.

$$g \in_{rp} U =_{df} g \in \widetilde{U}$$

And then RP-refinement is simply: $U_0 \sqsupseteq_{rp} U_1 =_{df} \widetilde{U_0} \subseteq \widetilde{U_1}$ with the usual introduction and elimination rules.

RP-refinement and W$_{\Box}$-refinement are equivalent We begin by showing that RP-refinement satisfies the W$_{\Box}$-refinement elimination rule.

Proposition 19. *The following rule is derivable:*

$$\frac{U_0 \sqsupseteq_{rp} U_1 \quad t \in \overset{\Box}{U_0}}{t \in \overset{\Box}{U_1}}$$

Proof

The proof is identical to the one of proposition 8, where every $\overset{\bullet}{U_i}$ is substituted by $\overset{\Box}{U_i}$ and every $\overline{U_i}$ is substituted by $\widetilde{U_i}$ $(i \in 2)$. The proof requires the axiom of choice. □

From this we have:

Theorem 9.

$$\frac{U_0 \sqsupseteq_{rp} U_1}{U_0 \sqsupseteq_{w_{\Box}} U_1}$$

We now show that W$_{\Box}$-refinement satisfies the RP-refinement elimination rule.

Proposition 20.

$$\frac{U_0 \sqsupseteq_{w_{\Box}} U_1 \quad g \in \widetilde{U_0}}{g \in \widetilde{U_1}}$$

Proof

The proof is identical to the one of proposition 9, where every $\overset{\bullet}{U_i}$ is substituted by $\overset{\Box}{U_i}$ and every $\overline{U_i}$ is substituted by $\widetilde{U_i}$ $(i \in 2)$.
□

Theorem 10.

$$\frac{U_0 \sqsupseteq_{w_\Box} U_1}{U_0 \sqsupseteq_{rp} U_1}$$

$\Box$

Theorems 9 and 10 together demonstrate that $\mathrm{W}_\Box$-refinement and RP-refinement are equivalent.

RP-refinement and FP-refinement are equivalent We now show that the notions of implementation are equivalent by the same strategy involving elimination rules. We first establish that FP-implementation implies RP-implementation:

Proposition 21. *The following rules are derivable:*

$$\frac{g \Subset_{fp} U}{g \subseteq \overset{\Box}{U}} \qquad \frac{g \Subset_{fp} U}{g \in T_\perp^{in} \to T_\perp^{out'}}$$

Proof

$$\dfrac{\dfrac{\dfrac{\dfrac{g \Subset_{fp} U}{g \in T_{\perp}^{in} \to T_{\perp}^{out'}} \quad \dfrac{}{z_0 \star z_1' \in g}\,(1)}{z_0 \star z_1' \in T^{*}} \qquad \dfrac{\begin{array}{c}\delta\\ \vdots\\ z_0 \star (g\ z_0)' \in U\end{array} \quad \dfrac{\dfrac{g \Subset_{fp} U}{g \in T_{\perp}^{in} \to T_{\perp}^{out'}} \quad \dfrac{}{z_0 \star z_1' \in g}\,(1)}{z_1' = (g\ z_0)'}}{z_0 \star z_1' \in U}}{z_0 \star z_1' \in \overset{\square}{U}}\,(2)}{g \subseteq \overset{\square}{U}}\,(1)$$

where δ stands for the following branch:

$$\dfrac{\dfrac{}{\mathit{Pre}\ U\ z_0 \vee z_1' \neq \perp'}\,(2) \qquad \begin{array}{c}\beta_0\\ \vdots\\ z_0 \star (g\ z_0)' \in U\end{array} \quad \begin{array}{c}\beta_1\\ \vdots\\ z_0 \star (g\ z_0)' \in U\end{array}}{z_0 \star (g\ z_0)' \in U}\,(3)$$

where β_0 is:

$$\dfrac{g \Subset_{fp} U \quad \begin{array}{c}\alpha\\ \vdots\\ z_0 \in T_{\perp}^{in}\end{array} \quad \dfrac{}{\mathit{Pre}\ U\ z_0}\,(3)}{z_0 \star (g\ z_0)' \in U}$$

and β_1 is:

$$\dfrac{g \Subset_{fp} U \quad \begin{array}{c}\alpha\\ \vdots\\ z_0 \in T_{\perp}^{in}\end{array} \qquad \dfrac{\dfrac{}{z_1' \neq \perp'}\,(3) \quad \dfrac{\dfrac{g \Subset_{fp} U}{g \in T_{\perp}^{in} \to T_{\perp}^{out'}} \quad \dfrac{}{z_0 \star z_1' \in g}\,(1)}{z_1' = (g\ z_0)'}}{(g\ z_0)' \neq \perp'}}{z_0 \star (g\ z_0)' \in U}$$

where α stands for the following branch:

$$\dfrac{\dfrac{\dfrac{g \Subset_{fp} U}{g \in T_{\perp}^{in} \to T_{\perp}^{out'}} \quad \dfrac{}{z_0 \star z_1' \in g}\,(1)}{z_0 \star z_1' \in T^{\star}}}{z_0 \in T_{\perp}^{in}}$$

The second rule is immediate. □

Theorem 11.

$$\frac{g \Subset_{fp} U}{g \Subset_{rp} U}$$

□

Now we show that RP-implementation implies FP-implementation.

Proposition 22. *The following rules are derivable:*

$$\frac{g \Subset_{rp} U \quad t \in T_{\perp}^{in} \quad Pre\ \ U\ \ t}{t \star (g\ \ t)' \in U} \qquad \frac{g \Subset_{rp} U \quad t \in T_{\perp}^{in} \quad (g\ t)' \neq \perp'}{t \star (g\ \ t)' \in U}$$

$$\frac{g \Subset_{rp} U}{g \in T_{\perp}^{in} \to T_{\perp}^{out'}}$$

Proof

The first rule:

$$\frac{\dfrac{\begin{array}{c}\delta \\ \vdots \\ t \star (g\ \ t)' \in \overset{\Box}{U}\end{array}}{t \star (g\ \ t)' \in \overset{\bullet}{U}}\ (3(i)) \qquad Pre\ \ U\ \ t}{t \star (g\ \ t)' \in U}$$

The second rule:

$$\frac{\begin{array}{c}\delta \\ \vdots \\ t \star (g\ \ t)' \in \overset{\Box}{U}\end{array} \quad (g\ t)' \neq \perp'}{t \star (g\ \ t)' \in U}\ (3(iv))$$

where δ is:

$$\frac{\dfrac{g \Subset_{rp} U}{g \subseteq \overset{\Box}{U}} \quad \dfrac{\dfrac{g \Subset_{rp} U}{g \in T_{\perp}^{in} \to T_{\perp}^{out'}} \quad t \in T_{\perp}^{in}}{t \star (g\ \ t)' \in g}}{t \star (g\ \ t)' \in \overset{\Box}{U}}$$

The third rule is immediate. □

Theorem 12.

$$\frac{g \Subset_{rp} U}{g \Subset_{fp} U}$$

□

Then, from 11 and 12, we see that the two notions of implementation are equivalent. Hence, so are the two notions of refinement.

7 Conclusions and future work

The model of schemas introduced in $W_\bullet$-refinement not only totalises the schema as a set of bindings, it also introduces the $\perp$ 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?
Our analysis provides a very clear mathematical explanation for lifting: with non-lifted totalisation it is not possible to prove proposition 5. Note that the proof of that result made explicit use of $\perp$ value. Indeed, we can do better: the following is an explicit counterexample:

Definition 17.

$$(i)\, \overset{\diamond}{U} =_{df} \{z \in T \mid Pre\ U\ z \Rightarrow z \in U\} \ \ (ii)\, True =_{df} [T \mid true]$$

Proposition 23.

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

□

It is an immediate consequence that the more permissive notion of refinement does not, for example, insist that preconditions do not strengthen.
Much the same observation can be made for the other family of refinement theories: again the lifting is critical in preventing the preconditions from strengthening.
Our refinement theories S-refinement and SP-refinement are entirely proof-theoretic, characterising refinement directly in terms of the behaviour of the predicates involved. These are quite closely related to conditions proposed originally by Spivey (these roughly correspond to the premises of our introduction rule for S-refinement). By reformulating this approach as a theory, rather than sufficient conditions, we establish an equivalent framework in which the model extension, with the lifting and completions involved, are unnecessary. Although we have not shown it here, there are very simple connections between S-refinement and an equivalent theory of refinement based on weakest preconditions: this will be reported in future work.
The approach to specification based on sets of implementations is a well established but somewhat different tradition, and is most usually investigated in a

constructive setting. We have demonstrated that what look like radically different models of specification and refinement are in fact intimately related.
What we have not reported here is an extension to data refinement in which data simulation relations play a significant role. There is much to say on this topic, but that requires the present work as a necessary precursor. We will in future work show that it is possible to formulate S-like theories which are equivalent to generalisations of the W-frameworks (the obvious generalisations are only equivalent for restricted forms of simulation). Moreover there are interesting results in weakest precondition data refinement to be developed and explored.
Finally, we have not mentioned the implications for the schema calculus. In considering Z as a prime example of a specification language that fits the technical development explored in this paper, one will want to know how the schema operations interact with refinement: in particular a treatment of monotonicity properties. It is quite well-known that the Z schema calculus has poor monotonicity properties in the relational model. Our results demonstrate that this is not a special feature of the relational model (because all the alternative approaches are equivalent). Indeed these poor properties are it seems intimately linked with the underlying partial relation semantics. One interesting set of approaches which need to be fully investigated are the consequences of restricting any one of the refinement theories we have outlines here to *atomic* schemas only; and then to redefine the semantics of the schema operators over the new semantics (rather than over partial relations). In this way refinement would reduce to the subset relation on the semantics and would be fully monotonic. Naturally the *nature* of the schema algebra would change, but those changes would be very interesting to explore.

8 Acknowledgements

We would like to thank the New Zealand Foundation for Research, Science and Technology (grant reference: UOWX0011) for financially supporting this research. Moshe Deutsch is supported by the British Council through an ORS award. Special thanks for particularly important discussions and comments go to Lindsay Groves, Greg Reeve, Ray Turner and Jim Woodcock.

References

1. W. P. DeRoever and K. Engelhardt. *Data refinement: model-oriented proof methods and their comparison.* Prentice Hall International, 1998.
2. John Derrick and Eerke Boiten. *Refinement in Z and Object-Z: Foundations and Advanced Applications.* Formal Approaches to Computing and Information Technology. Springer, May 2001.
3. A. Diller. *Z: An introduction to formal methods (2^{nd} ed.).* J. Wiley and Sons, 1994.
4. M. C. Henson and S. Reeves. Investigating Z. *Journal of Logic and Computation*, 10(1):1–30, 2000.

5. 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.
6. P. Martin-Löf. Constructive mathematics and computer programming. In *Logic, Methodology and Philosophy of Science VI*, pages 153–175. North Holland, 1982.
7. B. Potter, J. Sinclair, and D. Till. *An introduction to formal specification and Z.* Prentice Hall, 2nd. edition, 1996.
8. J. M. Spivey. *The Z notation: A reference manual, 2^{nd} ed.* Prentice Hall, 1992.
9. B. Strulo. How firing conditions help inheritance. In *Proceedings* ZUM '95*, LNCS Vol. 967*, pages 264–275. Springer Verlag, 1995.
10. J. Woodcock and J. Davies. *Using Z: Specification, Refinement and Proof.* Prentice Hall, 1996.

A Appendix

Our mathematical account takes place in a simple conservative extension $\mathcal{Z}_{\mathcal{C}}^{\bot}$ of $\mathcal{Z}_{\mathcal{C}}$ the core Z-logic of [4]. The only modification we need to make is to include the new undefined terms which are explicitly needed in the approach taken in [10]. 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.

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

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 $\mathcal{Z}_{\mathcal{C}}^{\bot}$ which introduces schemas as sets of bindings and the various operators of the schema calculus is undertaken as usual (see [4]) 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 18. *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.[2]

As a result the schema calculus is *hereditarily* $\bot$*-free.*
We indicated in the first section that we can always write the type of 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 also permit *binding concatenation*, written $t_0 \star t_1$ when the alphabets of t_0 and t_1 are disjoint. This is, in fact, exclusively used for partitioning bindings in operation schemas into before and after components, so the terms involved are necessarily disjoint. We lift this operation to sets (of appropriate type):

$$C_0 \star C_1 =_{df} \{z_0 \star z_1 \mid z_0 \in C_0 \wedge z_1 \in C_1\}$$

[2] 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 same restriction obviously applies here: the types of the sets involved must be disjoint.
We will need total functions over types. These are easily introduced.

Definition 19.

$$T_0 \to T_1 =_{df} \{g \in \mathbb{P}(T_0 \star T_1) \mid \mathit{unicity}(g) \wedge \mathit{total}(g)\}$$

Note that functions are modelled as subsets of $T_0 \star T_1$ rather than $T_0 \times T_1$, and that for notational convenience, we let g (*etc.*) range over terms of type $\mathbb{P}(T_0 \star T_1)$. In fact we only do this when such a term is a function.
When g is known to be an element of $T_0 \to T_1$ and $z_0 \in T_0$, we will write $g\ z$ (as usual) for the unique element $z_1 \in T_1$ such that $z_0 \star z_1 \in g$.