Available online at www.sciencedirect.com
SCIENCE DIRECT®
Electronic Notes in Theoretical Computer Science
ELSEVIER Electronic Notes in Theoretical Computer Science 137 (2005) 67–92
www.elsevier.com/locate/entcs

An Analysis of Operation-Refinement in an Abortive Paradigm

Moshe Deutsch[1] Martin C. Henson[2]

Department of Computer Science
University of Essex
Wivenhoe Park, Colchester, Essex, CO4 3SQ, UK

Abstract

This paper begins a new strand of investigation which complements our previous investigation of refinement for specifications whose semantics is given by *partial* relations (using Z as a linguistic vehicle for this semantics). It revolves around extending our mathematical apparatus so as to continue our quest for examining mathematically the essence of the lifted-totalisation semantics (which underlies the *de facto* standard notion of refinement in Z) and the role of the semantic elements $\perp$ in model-theoretic refinement, but this time in the *abortive paradigm*. We consider the simpler framework of *operation-refinement* and, thus, (at least at this stage) abstract from the complications emerging when *data simulations* are involved: we examine the (*de facto*) standard account of operation-refinement in this regime by introducing a simpler, *normative* theory (SP-refinement) which captures the notion of *firing conditions* refinement directly in the language and in terms of the natural properties of preconditions and postconditions; we then summarise our observations and link them to the particular role each of the possible *extreme specifications* in Z plays in the abortive paradigm - this lays the foundations to a more intricate future investigation of *data-refinement* in this paradigm. We conclude by providing a detailed account of *future work* which generalises Miarka, Boiten and Derrick's work of *combining* the *abortive* and *chaotic* paradigms for refinement, in our mathematical framework of $\mathcal{Z}_C$ and $\mathcal{Z}_C^{\perp}$.

Keywords: Operation-Refinement, Specification Language, Specification Logic.

1 Introduction

The concept of stepwise-refinement constitutes a pragmatic interpretation of the *Transformational Software Process Model* widely-known in the Software

[1] Email: mdeuts@essex.ac.uk
[2] Email: hensm@essex.ac.uk

doi:10.1016/j.entcs.2005.04.025

Engineering literature. It embodies the most important strategy employed for managing the immense complexity arising during the development process of large-scale software systems: *separation* and *orthogonality* of software components so as to *structure* the development in a systematic manner that would be easier for *human consumption*. The idea is to *decompose* the *design* phase of the software life cycle into a number of *simpler* (and, thus, *manageable*) steps, each of which *transforms* a system description *of a certain level of abstraction* into a more *concrete* one; the final step transforms a concrete system description into a *computer program*. Each of these transformations is verified by refinement rules; this ensures that, firstly, the process is *gradual* (each step includes some more design decisions within the system under development) and, secondly, that *correctness* is preserved in the course of development (the final program is guaranteed to meet the initial abstract specification). *Operation-refinement* concerns the derivation of a more concrete operation from a given abstract one, without changing the representation of its underlying *state space*.[3] It is, effectively, the degenerate case of *data-refinement* in which *data simulations* are *identity functions*.

Z is a state-based formalism based on an underlying *partial relation semantics*, where a specification (of an operation) denotes a set of *bindings* which can be construed to be a partial relation between *input* sub-bindings and *output* sub-bindings (see, for example, [25] and [40,10,37] for accounts of Z logic and semantics along these lines). Unlike some other formalisms such as B [1] and VDM [30], the (standard) language definition of Z does not provide any account of refinement. Therefore, in light of *both* the popularity of Z and the increasing interest in incorporating refinement within development processes, it is very important to acquire a comprehensive understanding of *foundational* (as well as *pragmatic*) aspects of refinement in Z. Note that, in the world of state-based formalisms, there are two major paradigms for refinement of partial specifications; we refer to these as the *chaotic* and the *abortive* paradigms. The former represents a more *sequential* view where preconditions may be weakened in the course of refinement, whereas the latter represents a more *concurrent* view in which preconditions remain fixed during refinement.

Indeed, we have, in previous work (*e.g.* [16,14,13,15,12]), concentrated on a foundational investigation of *both* operation-refinement and data-refinement in the *chaotic paradigm*. The (*de facto*) standard approach for refinement in this paradigm is a *model-theoretic* one, where the specifications (partial relations) are both *completed* (made total) and *extended* (by means of an additional semantic value, often called *bottom* and written $\perp$); this semantics is often known as the *chaotic*-lifted-totalisation. We examined (in *ibid.*) the

[3] Hence, it is often known as *algorithm / algorithmic refinement* or *algorithm design*.

essence of this semantics and explained precisely the *mathematical* (as well as the *conceptual*) role of the $\perp$ values in model-theoretic refinement in this paradigm.

In this paper, we begin a new strand of investigation which complements our previous work: we will extend our mathematical apparatus so as to establish an investigation of operation-refinement in the abortive paradigm. We shall, thus, *begin* to shed some light on *firing conditions* refinement, in general, and on the (*de facto*) standard (model-theoretic) account of refinement in this paradigm, in particular. This account is based on a distinct lifted-totalisation semantics which manifests itself in the *strictness* it imposes, on *all* the initial values outside the precondition of the underlying operation, with respect to the distinguished value $\perp$. We will begin to expound, fundamentally, the crucialness of this setting, in obtaining an acceptable model-theoretic notion of *firing conditions* refinement, and the critical role that the $\perp$ values play within it. Indeed, we shall see that the mathematical role of $\perp$ in this paradigm is entirely different to that of $\perp$ in the chaotic paradigm.

We begin our pursuit by revising various concepts related to the partial relation semantics of Z and to refinement of specifications in view of this semantics (section 2). We then proceed with the definition of the two basic theories of operation-refinement, each of which captures a particular aspect of *firing conditions* refinement (section 3). These are: a theory capturing the properties expected in a refinement in an *apparent* mathematical manner (section 3.1) - this is a purely proof-theoretic notion that is used as our *benchmark* for determining the *validity* of any other notion of operation-refinement in the abortive paradigm; and a theory capturing the standard model-theoretic account, in this paradigm, which is based on the *abortive*-lifted-totalisation semantics (section 3.2). We then prove that these two theories of refinement are *equivalent* (section 4). In section 5, we summarise our observations from the comparison between the theories; in particular, we emphasise the critical role of the $\perp$ values, as well as the unique manner in which they interact with the completion, in substantiating the equivalence results in section 4. We then link these observations to the particular role that each of the possible *extreme specifications* in Z plays in the abortive paradigm; not only is this analysis very revealing, but it also paves the way for our future work in generalising this to a more intricate investigation of *data-refinement* in the abortive paradigm. Finally, in section 6, we provide a detailed account of additional interesting future work which would investigate some generalisations of Miarka's (*et al.*) framework [32] for *combining* the two abortive and chaotic paradigms for refinement. We revise this framework, emphasise the role of $\perp$ within it and discuss (in view of our current and previous analyses) whether a *supplemen-*

tary semantic element, that is *distinct* from $\perp$, would be crucial for capturing model-theoretic characterisations of *operation-refinement* and *data-refinement* within it.

Such an investigation becomes possible in virtue of $\mathcal{Z}_{\mathcal{C}}$, the logic for Z reported in *e.g.* [25], and a simple *conservative extension* $\mathcal{Z}_{\mathcal{C}}^{\perp}$, reported in *e.g.* [16], which incorporates $\perp$ terms into the types of $\mathcal{Z}_{\mathcal{C}}$. We summarise this, and additional notational conventions, in appendix A.[4] We employ a novel technique of rendering all the theories of refinement in a proof-theoretic form: as sets of *introduction* and *elimination* rules. This leads to a uniform and simple method for proving the various results in the sequel. As such, it contrasts with the more semantic-based techniques employed in [9].

2 The Partial Relation Semantics of Z

In this first section we will lay the basic mathematical and conceptual scenes which underlie our investigation. In the process, we will revise a little Z logic, settling some notational conventions; additional detail can be found in appendix A.

2.1 Schemas

The schema notation constitutes the most recognisable feature of Z (partly due to its semi-graphical form) and it, indeed, occupies a central place within the language as a means of structuring not only the mathematical text, that is used for describing rigorously properties of the system, but also the entire system itself.

In [25], 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.* [19], [40]), where an operation schema may be understood as a relation between states: a *transition relation* from an *unprimed state*, denoting the state "before" the operation, to a *primed state*, denoting the state "after" the operation. In this paper, we will use the meta-variable U (with decorations) to range over operation schemas. As an example, consider the operation schema (written horizontally) specifying the *predecessor operation*:

$$Pred \mathrel{\widehat=} \left[\, \mathtt{x}, \mathtt{x}' : \mathbb{N} \mid \mathtt{x} > 0 \land \mathtt{x}' = \mathtt{x} - 1 \,\right]$$

Pred has the type $\mathbb{P}[\mathtt{x} : \mathbb{N}, \mathtt{x}' : \mathbb{N}]$, and is understood to be a set of bindings of schema type $[\mathtt{x} : \mathbb{N}, \mathtt{x}' : \mathbb{N}]$. The bindings $\langle\!|\; x \Rrightarrow n, x' \Rrightarrow m \;|\!\rangle$, where $n > 0$,

[4] This is included for convenience only and the reader may wish to consult [25], [28] and [16] for further detail concerning our notational and meta-notational conventions.

are all elements of *Pred*. In fact, there are no other elements in this case. Recall that unprimed labels (such as $\mathtt{x}$) are understood to be observations of the state before the operation takes place, 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, $T =_{df} T^{in} \curlyvee T^{out'}$. For the schema *Pred*, we have $T^{in} =_{df} [\mathtt{x} : \mathbb{N}]$ and $T^{out'} =_{df} [\mathtt{x}' : \mathbb{N}]$. 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 $T =_{df} T^{in} \curlyvee T^{out'}$. With this in place, we can omit the type superscripts in most places in the sequel.

2.2 Preconditions

Z takes the logical (*i.e.* "postcondition only") approach to pre and postconditions [33,39,38,22]. That is, being a *single-predicate* framework, preconditions in Z are *implicit* and may be *calculated* by existential closure of the defining predicate with respect to all its *after* observations.

We can formalise the idea of the precondition of an operation schema (*domain* of the relation, between before and after states, the schema denotes) to express the *partiality* involved:

Definition 2.1 Let $T^{in} \preceq V$. $\quad Pre\ \ U\ x^V =_{df} \exists\, z \in U \bullet x \doteq z$

Notice that if V is precisely T^{in}, the definition above amounts to no more than:

$$\exists\, z \bullet x \star z' \in U$$

This facilitates the analysis significantly when reasoning about the *precondition-status* [5] of before-state variables (as opposed to variables ranging over a larger schema type).

The following introduction and elimination rules are immediately derivable for preconditions: [6]

Proposition 2.2

$$\frac{t_0 \in U \quad t_0 \doteq t_1}{Pre\ \ U\ t_1}\ (Pre^+) \qquad \frac{Pre\ \ U\ t \quad y \in U, y \doteq t \vdash P}{P}\ (Pre^-)$$

---

[5] Whether or not the variable is in the precondition of the specification in question.

[6] For later convenience, the notion of precondition is introduced as a predicate. In vernacular Z, the precondition constitutes a *state schema* comprised of all the *valid* before-states (bindings) of the operation in question. This is easily captured when necessary as: $\{z^{T^{in}} \mid Pre\ \ U\ z\}$.

The usual sideconditions apply to the eigenvariable y*.* □

Clearly, the precondition of *Pred* is not (and for operation schemas in general, will not be) the whole of $[\mathtt{x} : \mathbb{N}]$ (in general, T^{in}). In this sense, operation schemas denote *partial relations*. Indeed, *Pred* is a partial operation because it ranges over all natural numbers (its before-type) but is defined for only those natural numbers that are *greater than zero* (its domain); that is, it does not specify the behaviour:

$$\langle\!| \ \mathtt{x} \Rrightarrow 0, \mathtt{x}' \Rrightarrow m \ |\!\rangle$$

for any $m \in \mathbb{N}$. More precisely, it is *silent* with regards to the outcome of the operation when it is applied *outside its precondition*.

2.3 What Happens Outside the Preconditions?

The above raises an immediate question: what behaviour is permitted for a correct implementation of a (partial) specification (of an operation) outside its precondition? To answer this question we need a theory of *refinement*: a means of comparing such an implementation with such a specification. The general answer to this question is based on *total correctness* refinement. That is, refinement is based on a subsequent *total relation* semantics, known as the *lifted-totalisation*. This interpretation serves as the semantic basis for refinement in Z. It is modelled by, first, *extending* (*i.e. lifting*) the *source set* and *co-domain* of the operation in question with a distinguished (semantic) element (often referred to as "bottom") $\perp$, which represents some *unwelcome behaviour*, and then *totalising* the operation *in a certain way* with which the distinguished elements ***interact*** *in a certain way*.

In the world of state-based specifications, there are two well-known fundamentally different paradigms for refinement of partial specifications, where each of these paradigms induces a distinct lifted-totalisation semantics underlying refinement. This leads to two different concepts of refinement, each of which is based on a different answer to the question above. First is the *chaotic* paradigm, sometimes also known as the *contractual* approach [32] [10, ch.2-3]; this represents a more *sequential* view and, thus, underlies the standard interpretation of refinement in Z. In this paradigm, preconditions are considered as *minimal conditions* for establishing the postconditions (*i.e.* they may be *weakened* in a refinement process), therefore the answer to the above question is: *anything can happen* outside the precondition of the operation. That is, the operation behaves as specified when it is applied within its precondition and may establish any arbitrary outcome, *including unwelcome behaviour*, when applied outside its precondition; this is often referred to as "divergence"

[36,32]. In previous work (cited in section 1) we examined thoroughly models of operation-refinement and data-refinement in this paradigm, where our mathematical foundation enabled us to surgically scrutinise these models in a manner that would not be possible otherwise. Indeed, we resolved various difficulties many informal and semi-formal accounts ran into in an attempt to determine the essence of the *chaotic*-lifted-totalisation semantics, in general, and the role of the $\perp$ elements, in particular, in model-theoretic refinement.

The second paradigm for refinement is the *abortive* one, sometimes also known as the *behavioural* [10] or *blocking* [32] approach (in [21], this is, effectively, what Grundy denotes as the *partial model*); this represents a more *concurrent* view: it is reminiscent of the notion of "refusals" or "deadlock" in process algebras and, therefore, it is typically employed when state-based formalisms are combined with process algebras (*e.g.* [20], [4] and [10, ch.18-19]). In this paradigm, preconditions are considered as *guards* [32,10] or *firing conditions* [31,36] (*i.e.* they are trigger/fixed conditions - *not to be* ***weakened*** *in a refinement process*) [7], therefore the answer to the above question is: *nothing* can happen outside the precondition of the operation. More precisely, when applied within its precondition, the operation behaves as specified and it is *blocked* outside its precondition: it may *not* be applied outside its precondition and if it is, it will result solely in an *unwelcome behaviour*.

The bulk of this paper is devoted to the investigation of the basic notions of operation-refinement in the *abortive paradigm*. This rather simplified framework enables us to abstract from the complications arising when *data simulations* are involved and, thus, to, at least, *begin* to reason about the mathematical role of the $\perp$ values in this paradigm (and in comparison to their role in the chaotic paradigm).

3 A Basic Analysis of Refinement

Naturally, the partial relation semantics of operation schemas in Z raises an immediate question: what does it mean for one operation schema to refine another in the abortive paradigm? More generally, we are asking: what does it mean for one partial relation to refine another in this paradigm?

We begin our analysis by introducing two distinct notions of operation-refinement based on two distinct answers to the questions above. We, then, proceed with an analysis, of the relationships amongst these, which throws a new light on both of them but, in particular, on the standard (model-theoretic)

[7] Naturally, they are not to be strengthened either, in order not to violate the principles of refinement (see *e.g.* [9], [10], [16] and [13]); hence they, effectively, remain *fixed* in the process of refinement.

notion in this paradigm, based on the *abortive*-lifted-totalisation semantics.

3.1 SP-Refinement

Our first theory is SP-refinement, the *normative* theory of operation-refinement in the abortive paradigm. SP-refinement is a purely proof-theoretic characterisation which serves as a *benchmark* for determining the *validity* of any of the other notions of operation-refinement in this paradigm.

This notion is based on *three* basic properties one expects in a *firing conditions* refinement: firstly, that a refinement guarantees that *preconditions do not strengthen*; secondly, that a refinement guarantees that *postconditions do not weaken*; and, finally, that a refinement guarantees that *preconditions do not* ***weaken***. Notice that the first two properties are standard in a refinement, thus SP-refinement may involve the reduction of *nondeterminism* (and, hence, it is, indeed, a special case of S-refinement, the *normative* characterisation of operation-refinement in the *chaotic paradigm* - see *e.g.* section 3.2 of [16] for its definition); however, the first and the last properties impose *stability* of the domain of the definition (*i.e.* the *precondition*) throughout the refinement process (hence "SP"-refinement). SP-refinement can be captured by forcing the refinement relation to hold *exactly* when these conditions apply. It is written $U_0 \sqsupseteq_{sp} U_1$ and is given by the following $\mathcal{Z}_\mathcal{C}$ definition:

Definition 3.1

$$\begin{aligned} U_0 \sqsupseteq_{sp} U_1 =_{df} & (\forall\, z \bullet Pre\ \ U_1\ \ z \Rightarrow Pre\ \ U_0\ \ z) \wedge \\ & (\forall\, z_0, z_1 \bullet z_0 \star z_1' \in U_0 \Rightarrow z_0 \star z_1' \in U_1) \end{aligned}$$

The following introduction and elimination rules are derivable for SP-refinement:

Proposition 3.2 *Let z, z_0, z_1 be fresh variables.*

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

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

□

Notice that, in contrast to S-refinement, the conjunct $Pre\ U_1\ z_0$ is *absent* from the antecedent of the postcondition premise in the introduction rule for SP-refinement ($\sqsupseteq^+_{sp}$). This conjunct is precisely what distinguishes between

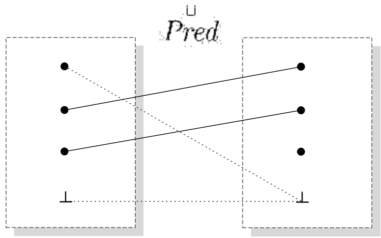

Fig. 1. An example: the abortive-lifted-totalisation of the predecessor operation.

S-refinement and SP-refinement: the two premises of $(\sqsupseteq^{+}_{sp})$ together guarantee that the precondition remains *fixed* in the course of refinement. As a result, the two elimination rules above establish *necessary conditions* for refinement which are distinct from the ones for S-refinement: akin to S-refinement, $(\sqsupseteq^{-}_{sp_0})$ guarantees that preconditions do not strengthen, yet, as opposed to S-refinement, $(\sqsupseteq^{-}_{sp_1})$ guarantees that *both* postconditions *and* preconditions do not weaken. This proof-theoretic representation enables us to take SP-refinement as the *normative* characterisation of operation-refinement in the abortive paradigm: this is our *prescription* for refinement, and another theory is acceptable providing it is at least *sound* with respect to SP-refinement since (as we shall see in section 4) soundness necessarily means that it must satisfy the two *necessary conditions* for SP-refinement above;[8] *completeness*, on the other hand, means that the other theory sanctions *at least* what SP-refinement does, namely *strengthening of postconditions*.

3.2 $W_{\Box}$-Refinement

In this section, we provide the formal technical development underlying the (*de facto*) standard (model-theoretic) notion of operation-refinement in the abortive paradigm. Akin to all our model-based theories, this takes place within our extended theory of $\mathcal{Z}_{\mathcal{C}}^{\perp}$.

We begin by expressing, in our mathematical framework, the intentions behind the *abortive-lifted-totalisation* semantics discussed in the literature (*e.g.* [4], [10, ch.3] and [3]). An example of applying this semantics to the operation *Pred* (written $\overset{\Box}{Pred}$ in our nomenclature) is depicted in Fig. 1. Recall (from *e.g.* [16]) that the chaotic-lifted-totalisation semantics makes no distinction between *arbitrary* and *unwelcome* behaviour resulting from applying an operation outside its precondition: anything may happen outside the precondition of the operation; this may include unwelcome behaviour as well as a possible

[8] This measurement of validity manifests itself more considerably in the generalisations to data-refinement (see *e.g.* [11]).

"divergence" of results. In the abortive paradigm, however, there is no such latitude: an operation is *blocked* outside its precondition; hence the "block" notation, used to denote the abortive-lifted-totalisation of a set of bindings, in the following definition:[9]

Definition 3.3 $\overset{\square}{U} =_{df} \{z_0 \star z_1' \in T^\star \mid z_0 \star z_1' \in U \vee (\neg\, \mathit{Pre}\ U\ z_0 \wedge z_1' = \perp')\}$

Notice the way this definition explicitly deploys the $\perp$ value in order to capture the *blocking* interpretation. This suggests (though, at this stage, fairly superficially) that the mathematical role of $\perp$ in this paradigm is different to that of $\perp$ in the chaotic paradigm; indeed, Boiten and de Roever [3] refer to $\perp$, *in this context*, as the element representing "deadlock". We will discuss this issue in further detail, following the rest of our analysis, in section 5.

The following introduction and elimination rules are derivable for abortive-lifted-totalised sets:

Proposition 3.4

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

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

□

The following additional rules are derivable for abortive-lifted-totalised sets:

Lemma 3.5

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

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

$$\frac{t_0 \star t_1' \in \overset{\square}{U} \quad t_0 = \perp}{t_1' = \perp'}\ (vi) \qquad \frac{t_0 \star t_1' \in T^\star \quad \mathit{Pre}\ U\ t_0 \vee t_1' \neq \perp' \vdash t_0 \star t_1' \in U}{t_0 \star t_1' \in \overset{\square}{U}}\ (vii)$$

□

[9] For notational convenience, we write $T^\star$ for the set $T_\perp^{in} \star T_\perp^{out'}$ (note the use of $\star$ for sets, as opposed to $\curlyvee$ used for types).

Lemmas 3.5(i) to (vi) show that definition 3.3 is consistent with the intentions described in the literature (embodied in Fig. 1): (i) to (iv) demonstrate that the abortive completion is contained in the chaotic completion (written $\overset{\bullet}{U}$ in our nomenclature - see *e.g.* section 3.3 of [16] for its definition), the distinguished value is present in the completion and everything outside the precondition is mapped onto it, where all the states in the underlying relation remain unchanged in the completion; (v) and (vi) together express the *strictness*, of *all* the *initial values* outside the precondition of the underlying relation, with respect to $\perp$ in the completion. Additionally, note that definition 3.3 may be expressed using *implication* (in the obvious way) instead of disjunction: lemma 3.5(vii) constitutes the introduction rule, for the abortive-lifted-totalisation, based on *implication introduction*.

With this in place, we can easily define the standard notion of refinement in the abortive paradigm. We name this $W_{\Box}$*-refinement*; it is written $U_0 \sqsupseteq_{w_{\Box}} U_1$ and is defined as follows:

Definition 3.6 $U_0 \sqsupseteq_{w_{\Box}} U_1 =_{df} \overset{\Box}{U_0} \subseteq \overset{\Box}{U_1}$

Obvious introduction and elimination rules follow from this definition.

4 Two Equivalent Theories

In this section, we shall demonstrate that our two theories of refinement are equivalent. This analysis will aid us to *begin* to shed some light on the mathematical and conceptual roles that the $\perp$ values play, in model-theoretic refinement, in the abortive paradigm.

Methodologically, we shall be showing that all judgements of refinement in one theory are contained among the refinements sanctioned by another. Such results can always be established proof-theoretically because we have expressed even our model-theoretic approach as a theory (set of introduction and elimination rules). Specifically, we will show that the refinement relation of a theory $\mathcal{T}_0$ satisfies the elimination rule (or rules) for refinement of another theory $\mathcal{T}_1$. Since the elimination rules and introduction rules of a theory enjoy the usual symmetry properties, this is sufficient to show that all $\mathcal{T}_0$-refinements are also $\mathcal{T}_1$-refinements. Equivalence can then be shown by interchanging the roles of $\mathcal{T}_0$ and $\mathcal{T}_1$ in the above.

We begin by showing that $W_{\Box}$-refinement satisfies the two SP-refinement elimination rules. Firstly, the rule which guarantees *non-augmentation* of undefinedness.

Proposition 4.1 *The following rule is derivable:*

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

Proof

$$\cfrac{\cfrac{\begin{matrix}\delta\\ \vdots \\ t\star\perp'\in \overset{\square}{U_1}\end{matrix} \quad \cfrac{\overline{t\star\perp'\in U_1}\ (2)}{false}\ (L.\ B.4) \quad \cfrac{Pre\ U_1\ t \quad \overline{\neg\ Pre\ U_1\ t}\ (2)}{false}}{false}\ (2)}{Pre\ U_0\ t}\ (1)$$

Where δ stands for the following branch:

$$\cfrac{U_0 \sqsupseteq_{w_\square} U_1 \qquad \cfrac{\overline{\neg\ Pre\ U_0\ t}\ (1) \qquad \cfrac{Pre\ U_1\ t \qquad \cfrac{\cfrac{\cfrac{\overline{t\star y'\in U_1}\ (3)}{t\star y'\in T}}{t\in T^{in}}}{t\in T^{in}_{\perp}}}{t\in T^{in}_{\perp}}\ (3)}{t\star\perp'\in \overset{\square}{U_0}}\ (L.\ 3.5(iii))}{t\star\perp'\in \overset{\square}{U_1}}$$

□

Turning now to the SP-elimination rule which guarantees *non-augmentation* of *both* definedness and nondeterminism.

Proposition 4.2 *The following rule is derivable:*

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

Proof

$$\cfrac{\cfrac{U_0 \sqsupseteq_{w_\square} U_1 \quad \cfrac{t_0\star t_1'\in U_0}{t_0\star t_1'\in \overset{\square}{U_0}}}{t_0\star t_1'\in \overset{\square}{U_1}} \qquad \overline{t_0\star t_1'\in U_1}\ (1) \qquad \cfrac{\cfrac{\cfrac{t_0\star t_1'\in U_0 \quad \overline{t_1'=\perp'}\ (1)}{t_0\star\perp'\in U_0}}{false}\ (L.\ B.4)}{t_0\star t_1'\in U_1}}{t_0\star t_1'\in U_1}\ (1)$$

□

The following theorem is then immediately derivable by propositions 4.1 and 4.2, in addition to the rule $(\sqsupseteq^{+}_{sp})$:[10]

[10] The proofs of such theorems are always automatic by the structural symmetry between introduction and elimination rules. We shall, therefore, not provide them explicitly.

Theorem 4.3

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

□

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

Proposition 4.4 *The following rule is derivable:*

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

Proof

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

Where δ stands for the following branch:

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

□

Then the following theorem immediately follows, by $(\sqsupseteq^+_{w_\Box})$, from proposition 4.4:

Theorem 4.5

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

□

Together, theorems 4.3 and 4.5 establish that the theories of SP-refinement and $W_\Box$-refinement are equivalent. Notice that, unlike the chaotic paradigm counterpart results (substantiating equivalence between $W_\bullet$-refinement [11] and S-refinement - see *e.g.* section 4.2 of [16]) where the explicit use of $\perp$ is crucial only for guaranteeing that preconditions do not strengthen, the explicit use of $\perp$ here is crucial for establishing *all* three results: the two results (propositions 4.1 and 4.2) underlying the *soundness* theorem 4.3 and the result (proposition

[11] $W_\bullet$-refinement, in our nomenclature, is the (*de facto*) standard (model-theoretic) notion of operation-refinement in the chaotic paradigm - see *e.g.* section 3.3 of [16] for its definition.

4.4) underlying the *completeness* theorem 4.5. We will further elaborate on this observation in the next section.

5 Discussion

In this paper, we have conducted a foremost foundational analysis of operation-refinement in the abortive paradigm. We have developed two theories of refinement, each of which constitutes a specialisation, of the corresponding theory in the chaotic paradigm (see *e.g.* section 3 of [16]), which adheres to the concept of *firing conditions* refinement. The standard account in this paradigm is a model-theoretic one; it is based on a particular notion of lifted-totalisation, in which *all* the initial values outside the precondition of the underlying operation are *strict* with respect to the distinguished value $\perp$ in the completion. SP-refinement belongs to our proof-theoretic family of refinement theories. Again, it serves as the *normative* characterisation of operation-refinement in the abortive paradigm because it captures the intentions behind *firing conditions* refinement in an apparent mathematical manner and *directly*, within the language, in terms of the predicates involved: it does not involve the introduction of an auxiliary semantics, nor the introduction of auxiliary elements. We have demonstrated that, by establishing this approach as a *theory* (rather than *sufficient conditions*), we can attain an equivalent framework in which the model extensions with auxiliary semantic elements are unnecessary for formalising the concept of *firing conditions* refinement. Once again, we have demonstrated that what look like different models of specification and refinement are, in fact, *intimately related*.

However, being confined to only operation-refinement, this analysis aids us to only *begin* to explain some of the mathematical reasons why the abortive-lifted-totalisation, underlying the standard characterisation of refinement in this paradigm, has been defined in just the way it has (*i.e.* insisting on *strictness* with respect to the distinguished value $\perp$), and what the mathematical role of the $\perp$ values, in the context of *firing conditions* refinement, is. Indeed, conceptually, akin to the chaotic paradigm, $\perp$ here represents some *unwelcome behaviour*, but its mathematical role is, evidently, different to that in the chaotic paradigm. This difference manifests itself precisely in the proofs of propositions 4.1, 4.2 and 4.4: it is clear that the accomplishment of *all* these proofs is *critically* contingent not only on the *explicit* use of the $\perp$ value, but also on the *strictness*, of all the initial values outside the precondition, with respect to it; recall that, in contrast to that, the explicit use of $\perp$ in the chaotic paradigm is crucial only for substantiating that the standard account of refinement *guarantees that preconditions do not strengthen* (see *e.g.* proposition 4.11

of [16]). This suggests that the distinguished values, in this paradigm, and *the way they interact with the completion (and vice versa)* are absolutely *vital* for capturing *correctly* the *guarded* interpretation underlying model-theoretic refinement, and there seems to be no other way to model *refusals* in relational completion (*i.e.* without utilising $\perp$ values).[12]

As usual, we can emphasise this point (or at least *begin* to emphasise it) by considering the idea of *extreme specifications* in the abortive paradigm.[13] Akin to the chaotic paradigm, the specification *True* denotes *explicit permission to behave*. The only sanctioned property in a *firing conditions* refinement is *reduction of nondeterminism*; hence, since *True* is the most nondeterministic specification, any specification which reduces this behaviour constitutes its refinement. Now recall that *True* is a *total* specification. A mathematical fact is that *all the relational completion models at our disposal underlie equivalent theories of refinement when the underlying specifications are **total***. The proof of this is very simple: we established, in section 6 of [16], that the *strict* and *non-strict-chaotic-lifted-totalisation* models underlie equivalent theories of refinement;[14] moreover, it is evident that the strict-chaotic-lifted-totalisation and the abortive-lifted-totalisation models are *equivalent* when the underlying specifications are *total*, in which case refinement *in both paradigms* amounts to a *firing conditions* refinement *so as to prevent augmentation of undefinedness*.

The specification *Chaos*, however, denotes something completely different in the abortive paradigm. Recall that, in the chaotic paradigm, *Chaos* denotes *implicit permission to behave* and, as a result, *any* specification refines it (see *e.g.* section 4.4 of [16]). This, of course, coincides with the fact that, in that paradigm, *anything can happen outside the precondition of any operation*. In the abortive paradigm, on the other hand, any operation *aborts* outside its precondition, *blocking* any possibility of *recovery* from this outcome in the context of refinement (as we have seen in section 4, this setting seems inevitable in order to obtain an acceptable model-theoretic notion of *firing conditions* refinement). Now since *everything* is outside the precondition of

[12] This is in contrast to the chaotic paradigm, in which we established a model-theoretic characterisation of operation-refinement based on a relational completion model that is *totalised*, but *not lifted*; we proved that this characterisation captures the (necessary and sanctioned) properties expected in a refinement in that paradigm. Notwithstanding, there is a price for this formulation, namely the necessity of relying on an *alternative interpretation of the concept of **preconditions*** (for further detail, see section 5 of [16]).

[13] See appendix A for the definitions of the two extreme specifications *True* and *Chaos* in our nomenclature (definition A.4).

[14] The *strict-chaotic-lifted-totalisation* semantics in *ibid.* captures (in our mathematical framework) the informal intentions described in [7]: *divergence* outside the precondition of the underlying relation and *strictness* for $\perp$ (*i.e.* $\perp$ in the *source set* maps only to its *co-domain* counterpart).

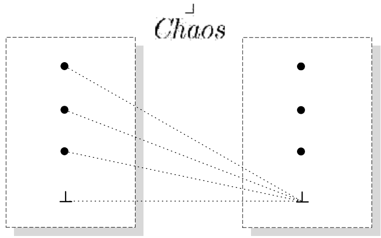

Fig. 2. An illustration: the specification *Chaos* represents *implicit deadlock* in the abortive paradigm.

Chaos, it *implicitly* denotes here a situation in which *everything is blocked*, namely *implicit deadlock*; this situation is defined by, for example, Roscoe [35, ch.0] as follows: "*a concurrent system is **deadlocked** if no component can make any progress...*". Indeed, in a *firing conditions* refinement *Chaos* cannot refine *anything* (which is, of course, natural in order to prevent preconditions from *strengthening*) but, actually, *nothing* can refine it either in this regime. We, therefore, refer to *abortive-lifted-totalised Chaos* (*i.e.* $\overset{\square}{Chaos}$) as *Deadlock*, from which no *recovery* is possible; this is illustrated in Fig. 2.

In [3], Boiten and de Roever refer to $\perp$ (in the context of the abortive paradigm) as the element representing "deadlock". We, for reasons discussed earlier, prefer to follow the lines of the concurrent formalisms' literature (*e.g.* [35]) and refer to the *situation* (*i.e.* the lifted-totalised specification) as *Deadlock* and to $\perp$, which evidently has a crucial role in preventing non-strict *recovery* from this situation, as the *abortive element.* In this way, we still manifest the intuitions discussed in [3], in this context. Indeed, we will, in future work - in the generalisations to *simulation-based data-refinement* - reinforce our conclusions (and terminology) regarding the role of $\perp$ in the abortive paradigm, as well as the significance of the strictness of both *completions* of the operations and the *lifting* of data simulations in obtaining a ***valid*** *forward simulation* and a ***useful*** *backward simulation* model-theoretic characterisations of *firing conditions* refinement. Akin to the analysis of data-refinement in the chaotic paradigm (*e.g.* [13,15,12]), we will uncover many issues, concerning data-refinement in the abortive paradigm, when *non-trivial* data simulations are permitted. A particularly interesting revelation is the fact that the *abortive-lifted-totalisation* semantics can underlie the standard notion of *forward simulation refinement* in *both* the chaotic and the abortive paradigms, where the actual paradigm depends solely on the *way* in which the *data simulations* are *lifted.* [15]

[15] The bulk of this analysis is reported in [11].

6 Future and Related Work: Generalisations of Miarka, Boiten and Derrick's Framework for Combining the Abortive and Chaotic Paradigms for Refinement

6.1 Intentions, Intuitions and Motivations

In [32], Miarka *et al.* develop a framework which *combines* the two *abortive* and *chaotic* paradigms for refinement. This framework allows the representation of *both* ***refusals*** and ***underspecification*** *in the same account*. The authors' motivation lies in the fact that the two paradigms are neither *exclusive* [16] nor *mutually exclusive*. They begin with a simple, yet very tangible example which illustrates this point. The example is given by means of a Z specification modelling a simple *money transaction* system: *Bank* is a state schema which specifies a repository of *bank accounts* by means of a partial function between (unique) *account numbers* and *integer numbers*, each of which represents the *balance* of the account it is devoted to. *Transfer* is an operation schema on *Bank* which effectively specifies an *increase* of the balance of a *given account* by a *given value*; the precondition of *Trasfer* comprises two predicates: one guarantees that the *given account* exists in the repository and the other guarantees that the *given value* (to be added to the balance) is *not negative*, so that no money can be *withdrawn* as a consequence of this operation. They then demonstrate that non of the chaotic and abortive characterisations provides an adequate solution of (operation-) refinement for this system. Refinement in the chaotic paradigm is *too permissive*: it enables a sensible approach of *extending* the repository with a new bank account (whose balance is the given value), in case the given account number does not exist in the repository; however, it also allows the dangerous case of *money withdrawal* (both of these cases constitute a natural consequence of weakening the precondition). Conversely, refinement in the abortive paradigm is *too restrictive*: it prevents the dangerous case of money withdrawal, but it also prevents the sensible case of extending the repository with a new account.

The reason why non of these paradigms provides a satisfactory solution for refining this operation is that, apparently, the two predicates in its precondition have *different roles*: the one insisting that the added value is not negative is more like a *guard*, whereas the one insisting that the given account already exists is more like a *precondition*. Indeed, the authors' solution is based on combining the two paradigms for refinement and it revolves around the idea of essentially separating the predicates which form the *guard*, of a

[16] Indeed, we presented in [18] another characterisation of refinement (SC-refinement) in which preconditions may weaken, but *postconditions* remain *fixed*.

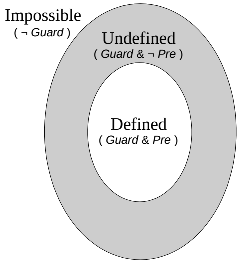

Fig. 3. The possible regions of operation behaviour in the combined paradigm for refinement.

certain operation schema, from those which form the *precondition*. In this way, they enable *both* guards and preconditions *in the same specification*. By and large, in this framework, an operation *outside its guard* behaves like in the *abortive paradigm*: it is *blocked*, regardless of whether or not its precondition holds; whereas *inside its guard*, the operation behaves like in the *chaotic paradigm*: the outcome depends on whether or not the precondition holds (as usual, anything can happen outside the precondition). The way in which they implement this idea is by viewing an operation schema as being comprised of a *brace* of schemas: the first is the "enabled" schema which denotes the guard, whereas the second is the "effect" schema which denotes the actual operation; this is an operation schema in the usual sense (*i.e.* its precondition may be calculated as usual). It is important to note that this is merely a cosmetic way of separating the guard explicitly from the predicate of the operation schema: naturally, the signature of the "enabled" schema is either identical to, or contained in, the signature of the "effect" schema; thus when required in a schema expression, the operation is taken as simply the *conjunction* of these two schemas. In this way, no changes are required for the schema calculus operations. Having said that, notwithstanding, an operation is given a non-standard interpretation, based on *three-valued* logic, which explicitly gives manifestation to *three regions* of operation behaviour delineated in Fig. 3: the operation is *defined* when *both* the guard and the precondition hold (this region is represented by *true*); the operation is *impossible* (*i.e. blocked*) when the guard does not hold (this region is represented by *false*); the operation is *undefined* when the guard holds, but the precondition does not hold (this is a "don't care" situation in which anything can happen, thus it is represented by the *third* logical value which precisely embodies this interpretation).

The authors then define a characterisation of (sufficient conditions for) operation-refinement, in accordance with these three regions, whereby *post-conditions may strengthen*, *preconditions may weaken* and *the guard may strengthen*. Having said that, it is important to note that "*the precondition is the* ***upper bound*** *for strengthening the guard and the guard is the* ***lower***

***bound** for weakening the precondition*" [32]. Put another way: going back to Fig. 3, the *defined* region may be enlarged, *but not beyond the impossible region's boundaries*; likewise, the *impossible* region may be enlarged, *but not beyond the defined region's boundaries*.

It would be very interesting to capture these ideas in our mathematical framework. Such an extension of the mathematical apparatus for refinement would emphasise three salient issues. We discuss these, in detail, in the remaining sub-sections.

6.2 Proof-Theoretic Operation-Refinement

Firstly, the three-valued logic interpretation of operations does not seem to be entirely crucial for capturing the intentions from [32] in the form of a proof-theoretic characterisation of refinement in our usual setting based on classical logic. The key issue here is, of course, capturing adequately the concept of separation between preconditions and guards. Aside from the approach suggested in [32], there are two alternative interesting ways of capturing this concept. One way is to have an operation schema defined with *two predicates*: one constituting the *guard* and the other constituting the (usual) Z *postcondition*; in a sense, this approach is reminiscent of the one taken in [32], only that this would require some changes to be made in the definitions of the schema calculus operations. Another way is to generalise the approach taken in [26,27], where the pre and postconditions are *syntactically separated*, by adding a third predicate for the guard. Either way, it would be very interesting to examine the ramifications of these approaches.

6.3 Model-Theoretic Operation-Refinement

The second issue (which was not covered in [32]) concerns model-theoretic refinement. Our conjecture (based on the experience we acquired through the *entire* project investigating foundational issues in refinement - pursued over the last four years) is that $\mathrm{W}_{\boxdot}$-refinement, that is a model-theoretic notion based on a lifted-totalisation semantics delineated in Fig. 4, would also capture the intentions from [32]. This *combined*-lifted-totalisation semantics is based on an *additional semantic element* "top" $\top$ (see *e.g.* [29, ch.2]) we require in the completion. Consider, for example, the combined-lifted-totalisation of the abstract specification U_1 (written $\overset{\boxdot}{U_1}$) in Fig. 4; this precisely adheres to the three possible regions of operation behaviour illustrated in Fig. 3: the before-state `x` is in *both* the *guard* and the *precondition* of the underlying operation and, thus, the two states forking from it belong to the *defined* region of the operation; `y` is *in the guard* but *outside the precondition*, thus all the

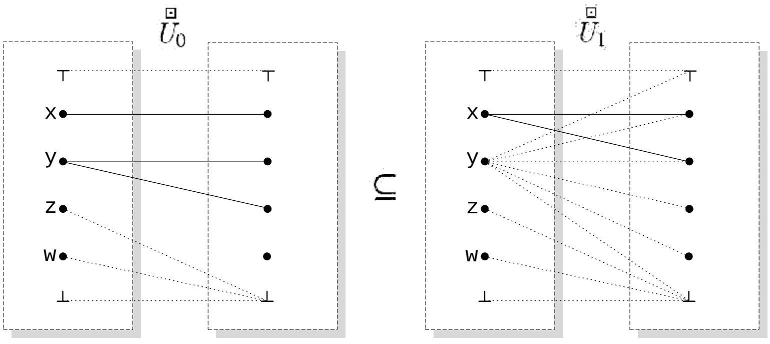

Fig. 4. A conjecture: $W_{\boxdot}$-refinement is based on a lifted-totalisation semantics that captures the intentions depicted in [32].

states forking from it belong to the *undefined* region - anything is possible including $\perp$ and $\top$; `z` and `w` are both *outside the guard* and, thus, belong to the *impossible* region - the operation is *blocked*, regardless of whether or not any of these before-states is in the precondition. As we can see in Fig. 4, it seems that the subset relation (in conjunction with this notion of lifted-totalisation) guarantees precisely the properties addressed in [32]: that postconditions can *only strengthen* (see `x` in $\overset{\boxdot}{U}_0$), that preconditions can *only weaken* (see `y` in $\overset{\boxdot}{U}_0$) and that guards *may not weaken* (see `z` and `w` in $\overset{\boxdot}{U}_0$). Moreover, the fact that anything within the guard but outside the precondition maps to *anything*, including $\perp$, enables *strengthening of the guard* within the *undefined region*: had `y` in $\overset{\boxdot}{U}_0$ been mapped onto only $\perp$, as a result of strengthening the guard, the subset relation would have still held. It would be very interesting to examine mathematically this conjecture, as well as the *monotonicity properties* of this notion.

6.4 Generalisations to Data-Refinement

Finally, the generalisations to *data-refinement*. *Prima facie*, one might argue that the top element $\top$ is not particularly crucial for establishing a combined-lifted-totalisation semantics, underlying $W_{\boxdot}$-refinement, which would capture the intentions depicted in Fig. 4. This is, indeed, the case for *operation-refinement*: one might use the same model as in Fig. 4, but with $\top$ (and any of its interactions with the completion) excluded, without loss of any generality discussed earlier. Nonetheless, our point of departure here is influenced by a broader perspective acquired from our investigation of data-refinement (some of which is reported in earlier work and some of which will be reported in future work). Recall (from *e.g.* [13] and [11]) that the way in which data simulations interact with the standard notion of refinement in each paradigm is by means of *lifting*.[17] In the chaotic paradigm, simulations are *non-strictly*-lifted, whereas in the abortive paradigm, they are *strictly*-lifted. The use of non-standard lifting of simulations has consequences which either *restrict* or *invalidate* the notion of refinement expected in each of the two paradigms. Now, for the sake of argument, consider the combined-lifted-totalisation without the top element $\top$. Let us examine the consequences of using a non-standard lifting of simulations in each paradigm; we can, thus, demonstrate that a single semantic element is (generally) insufficient for capturing *adequately* the intentions we discussed earlier, in the context of data-refinement:

- *Forward Simulation.* The use of ***strictly-lifted simulations*** in the ***chaotic paradigm*** *prevents weakening of preconditions* [13], whereas the use of ***non-strictly-lifted simulations*** in the ***abortive paradigm*** *permits weakening of preconditions* [11]; in which case, *weakening of guards would be permitted in our combined model for refinement.* Hence, non of these settings is adequate for forward simulation refinement in the combined paradigm;
- *Backward Simulation.* The use of ***strictly-lifted simulations*** in the ***chaotic paradigm*** *induces a theory of refinement that is* ***equivalent*** *to the standard one* [15,12], whereas the use of ***non-strictly-lifted simulations*** in the ***abortive paradigm*** *prevents strengthening of postconditions* [11]. Hence, the former setting might be adequate for backward simulation refinement in the combined paradigm, but the latter is not sufficiently general.

In conclusion, it is evident that, in the context of *forward simulation*, the only way to explicitly distinguish the *undefined* region of an operation from its *impossible* region, so as to obtain an adequate model-based theory of *forward simulation refinement* in the combined paradigm, is by means of employing an additional semantic element distinct from $\perp$. In which case, the notion of *lifting* of data simulations would have to apply for *both* semantic elements; the optimal setting in this case seems to be *strict-lifting with respect to* $\perp$ and *non-strict-lifting with respect to* $\top$. On the other hand, it seems that, in order to obtain an adequate model-based theory of *backward simulation refinement* in this paradigm, $\perp$ on its own is sufficient *providing that simulations are* ***strictly-lifted***.

Indeed, all the intuitions above are based on our existing mathematical analysis of data-refinement and, therefore, they would certainly provide a good

[17] Lifting signifies mapping $\perp$ of the *source set* of the relation onto all the states in its *co-domain*. In general, the notion of strictness discussed in this paper is with respect to $\perp$; therefore, strict-lifting denotes mapping $\perp$ onto only its *co-domain* counterpart.

start for future investigation. However, by all means, these would have to be carefully examined mathematically, in the context of the combined paradigm for refinement.

7 Acknowledgements

This work has been influenced in its development by too many people to name explicitly. However, special thanks for particularly important discussions and comments go to Steve Reeves, Ray Turner, Eerke Boiten, John Derrick, Steve Dunne, Neil Evans, Ralph Miarka, Greg Reeve, Steve Schneider, David Streader, Helen Treharne and Rob Arthan. Also, many thanks to Jonathan Bowen, John Derrick and Steve Schneider for supporting Moshe's research proposal to the Royal Academy of Engineering.

References

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

[2] D. Azada and P. Muenchaisri, editors. *APSEC 2003: 10th Asia-Pacific Software Engineering Conference, Chiangmai, Thailand, December 10-12, 2003, Proceedings*. IEEE Computer Society Press, December 2003.

[3] E. A. Boiten and W. P. de Roever. Getting to the Bottom of Relational Refinement: Relations and Correctness, Partial and Total. In R. Berghammer and B. Möller, editors, *RelMiCS 7: 7th International Seminar on Relational Methods in Computer Science, Malente, Germany, 12-17 May, 2003, Proceedings*, pages 82–88. University of Kiel, 2003.

[4] C. Bolton, J. Davies, and J. C. P. Woodcock. On the Refinement and Simulation of Data Types and Processes. In K. Araki, A. Galloway, and K. Taguchi, editors, *Integrated Formal Methods (IFM '99)*. Springer, 1999.

[5] J. P. Bowen, S. Dunne, A. Galloway, and S. King, editors. *ZB 2000: Formal Specification and Development in Z and B, First International Conference of B and Z Users, York, UK, August 29 - September 2, 2000, Proceedings*, volume 1878 of *Lecture Notes in Computer Science*. Springer, 2000.

[6] J. P. Bowen, A. Fett, and M. G. Hinchey, editors. *ZUM '98: The Z Formal Specification Notation, 11th International Conference of Z Users, Berlin, Germany, September 24-26, 1998, Proceedings*, volume 1493 of *Lecture Notes in Computer Science*. Springer, 1998.

[7] A. Cavalcanti and J. C. P. Woodcock. A Weakest Precondition Semantics for Z. *Technical Monograph PRG-TR-16-97. Oxford University Computing Laboratory*, 1997.

[8] D.Bert, J. P. Bowen, S. King, and M. Waldén, editors. *ZB 2003: Formal Specification and Development in Z and B, Third International Conference of B and Z Users, Turku, Finland, June 4-6, 2003, Proceedings*, volume 2651 of *Lecture Notes in Computer Science*. Springer, 2003.

[9] W. P. de Roever and K. Engelhardt. *Data Refinement: Model-Oriented Proof Methods and Their Comparison*. Prentice Hall International, 1998.

[10] J. Derrick and E. A. Boiten. *Refinement in Z and Object-Z: Foundations and Advanced Applications*. Formal Approaches to Computing and Information Technology – FACIT. Springer, May 2001.

[11] M. Deutsch. Firing Conditions. *University of Essex, technical report CSM-386*, April 2002.

[12] M. Deutsch and M. C. Henson. An Analysis of Backward Simulation Data-Refinement for Partial Relation Semantics. In *APSEC 2003 [2]*, pages 38–48, 2003.

[13] M. Deutsch and M. C. Henson. An Analysis of Forward Simulation Data Refinement. In *ZB 2003 [8]*, pages 148–167, 2003.

[14] M. Deutsch and M. C. Henson. An analysis of total correctness refinement models for partial relation semantics II. *Logic Journal of the IGPL*, 11(3):319–352, 2003.

[15] M. Deutsch and M. C. Henson. Four Theories for Backward Simulation Data-Refinement. In T. Muntean and K. Sere, editors, *RCS'03 – 2nd International Workshop on Refinement of Critical Systems: Methods, Tools and Developments*, bo Academi, Turku – Finland, June 2003.

[16] M. Deutsch, M. C. Henson, and S. Reeves. An analysis of total correctness refinement models for partial relation semantics I. *Logic Journal of the IGPL*, 11(3):287–317, 2003.

[17] M. Deutsch, M. C. Henson, and S. Reeves. Modular reasoning in Z: scrutinising monotonicity and refinement. *University of Essex, technical report CSM-407 (under consideration of FACJ)*, December 2003.

[18] M. Deutsch, M. C. Henson, and S. Reeves. Operation Refinement and Monotonicity in the Schema Calculus. In *ZB 2003 [8]*, pages 103–126, 2003.

[19] A. Diller. *Z: An Introduction to Formal Methods*. J. Wiley and Sons, 2nd edition, 1994.

[20] C. Fischer. How to Combine Z with a Process Algebra. In *ZUM '98 [6]*, pages 5–23, 1998.

[21] J. Grundy. *A Method of Program Refinement*. PhD thesis, University of Cambridge, 1993.

[22] I. J. Hayes, C. B. Jones, and J. E. Nicholls. Understanding the differences between VDM and Z. *Technical Report UMCS-93-8-1. Department of Computer Science, University of Manchester*, August 1993.

[23] M. C. Henson and S. Reeves. Revising Z: I - logic and semantics. *Formal Aspects of Computing*, 11(4):359–380, 1999.

[24] M. C. Henson and S. Reeves. Revising Z: II - logical development. *Formal Aspects of Computing*, 11(4):381–401, 1999.

[25] M. C. Henson and S. Reeves. Investigating Z. *Logic and Computation*, 10(1):43–73, 2000.

[26] M. C. Henson and S. Reeves. Program Development and Specification Refinement in the Schema Calculus. In *ZB 2000 [5]*, pages 344–362, 2000.

[27] M. C. Henson and S. Reeves. A logic for schema-based program development. *Formal Aspects of Computing*, 15(1):84–99, 2003.

[28] M. C. Henson, S. Reeves, and J. P. Bowen. Z logic and its consequences. *Computing and Informatics*, 22(4):381–415, 2003.

[29] C.A.R Hoare and J. He. *Unifying Theories of Programming.* Prentice Hall International, 1998.

[30] C. B. Jones. *Systematic Software Development using VDM.* Prentice Hall International, 2nd edition, 1990.

[31] M. B. Josephs. Specifying Reactive Systems in Z. *Technical Monograph PRG-TR-19-91. Oxford University Computing Laboratory*, 1991.

[32] R. Miarka, E. A. Boiten, and J. Derrick. Guards, Preconditions, and Refinement in Z. In *ZB 2000 [5]*, pages 286–303, 2000.

[33] C. C. Morgan. The specification statement. *ACM Transactions on Programming Languages and Systems*, 10:403–419, 1988.

[34] S. Prehn and W. J. Toetenel, editors. *VDM '91 – Formal Software Development, 4th International Symposium of VDM Europe, Noordwijkerhout, The Netherlands, October 21-25, 1991, Proceedings, Volume 2: Tutorials*, volume 552 of *Lecture Notes in Computer Science*. Springer-Verlag, 1991.

[35] A. W. Roscoe. *The Theory and Practice of Concurrency*. Prentice Hall, 1998.

[36] B. Strulo. How Firing Conditions Help Inheritance. In J. P. Bowen and M. G. Hinchey, editors, *ZUM '95: The Z Formal Specification Notation*, volume 967 of *Lecture Notes in Computer Science*, pages 264–275. Springer-Verlag, 1995.

[37] I. Toyn, editor. *Z Notation: Final Committee Draft, CD 13568.2*. Z Standards Panel, 1999.

[38] J. C. P. Woodcock. An Introduction to Refinement in Z. In *VDM '91 (volume 2) [34]*, pages 96–117, 1991.

[39] J. C. P. Woodcock. The Refinement Calculus. In *VDM '91 (volume 2) [34]*, pages 80–95, 1991.

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

A Specification Logic - A Synopsis

In this appendix, we will revise the specification logic underlying our investigation, settling our notational conventions in the process. The reader may wish to consult [25], [28] and [16] for a more leisurely treatment of our notational and meta-notational conventions.

Our analysis takes place in the "Church-style" version of the Z-logic due to Henson and Reeves, namely $\mathcal{Z_C}$ (*e.g.* [23,24,25]), and a simple conservative extension of that, we name $\mathcal{Z}_\mathcal{C}^\perp$ (*e.g.* [16,17]). This provides a convenient basis, in particular a satisfactory logical account of the schema calculus of Z, as it is normally understood, upon which the present work can be formalised.

A.1 The $\mathcal{Z_C}$ Specification Logic

$\mathcal{Z_C}$ is a typed theory in which the types of higher-order logic are extended with *schema types* whose values are unordered, label-indexed tuples called *bindings*. For example, if the T_i are types and the $\mathtt{z}_i$ are labels (constants) then:

$$[\cdots \mathtt{z}_i : T_i \cdots]$$

is a (schema) type. Values of this type are bindings, of the form:

$$\langle\!|\ \cdots \mathtt{z}_i \Rrightarrow t_i \cdots\ |\!\rangle$$

where the term t_i has type T_i. *Binding selection*, written $t.\mathtt{x}$, is axiomatised so that, for example:

$$\langle\!|\ \mathtt{x} \Rrightarrow 2, \mathtt{y} \Rrightarrow 3\ |\!\rangle.\mathtt{x} = 2$$

Selection generalises so that $t.P$ denotes the predicate P in which each observation $\mathtt{x}$ is replaced by $t.\mathtt{x}$. *Filtered* bindings play a major role in the schema calculus. Such terms have the form $t \upharpoonright T$ and are axiomatised so that, for example:

$$\langle\!|\ \mathtt{x} \Rrightarrow 2, \mathtt{y} \Rrightarrow 3\ |\!\rangle \upharpoonright [\mathtt{x} : \mathbb{N}] = \langle\!|\ \mathtt{x} \Rrightarrow 2\ |\!\rangle$$

The symbols $\preceq$, $\curlywedge$, $\curlyvee$ and $-$ denote the *schema subtype* relation, and the operations of *schema type intersection* and (compatible) *schema type union* and *schema type subtraction*. Every type in $\mathcal{Z_C}$ has a corresponding *carrier set*. This is formed by *closing* the carrier for the type in question (*e.g.* $\mathbb{N} =_{df} \{z^{\mathbb{N}} \mid true\}$) under the cartesian product, power type and schema type operations. [18] Therefore, the following axiom is admissible and is, thus, incorporated within the system:

$$\frac{}{t^T \in T}\ (T)$$

[18] 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 we discussed in section 2.1, 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 write the type of such operation schemas as $\mathbb{P}(T^{in} \curlyvee T^{out'})$ where T^{in} is the type of the "before" sub-binding (state) and $T^{out'}$ is the type of the "after" 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 types), with obvious introduction and elimination rules, by means of:

Definition A.1 $C_0 \star C_1 =_{df} \{z_0 \star z_1 \mid z_0 \in C_0 \wedge z_1 \in C_1\}$

The same restriction obviously applies here: the types of the sets involved must be *disjoint*. In this way, reasoning in Z becomes no more complex than reasoning with binary relations.

We introduce two notational conventions in order to avoid the repeated use of filtering in the context of membership and equality propositions.

Definition A.2 Let $T_1 \preceq T_0$. $\quad t^{T_0} \mathrel{\dot\in} C^{\mathbb{P}\,T_1} =_{df} t \restriction T_1 \in C$

Definition A.3 Let $T_1 \preceq T_0$ or $T_0 \preceq T_1$.

$$t_0^{T_0} \doteq t_1^{T_1} =_{df} t_0 \restriction (T_0 \curlywedge T_1) = t_1 \restriction (T_0 \curlywedge T_1)$$

In [25], the authors 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:

$$\begin{array}{ll} (i) & S_0^{\mathbb{P}\,T_0} \vee S_1^{\mathbb{P}\,T_1} =_{df} \{z^{T_0 \curlyvee T_1} \mid z \mathrel{\dot\in} S_0 \vee z \mathrel{\dot\in} S_1\} \\ (ii) & S_0^{\mathbb{P}\,T_0} \wedge S_1^{\mathbb{P}\,T_1} =_{df} \{z^{T_0 \curlyvee T_1} \mid z \mathrel{\dot\in} S_0 \wedge z \mathrel{\dot\in} S_1\} \end{array}$$

respectively define schema *disjunction* and schema *conjunction*.

Finally, we need the concept of *extreme specifications*. There are only two possible extreme specifications in Z: *True* (sometimes also known as *chance*, *e.g.* [21, ch.3]) which comprises *everything* and *Chaos* which comprises *nothing*. We define these in our logical framework as follows:

Definition A.4 (i) $True =_{df} [T \mid true]$ $\quad (ii)$ $Chaos =_{df} [T \mid false]$

B The $\mathcal{Z}_{\mathcal{C}}^{\perp}$ Specification Logic - A Conservative Extension of $\mathcal{Z}_{\mathcal{C}}$

The only modification we need to make in $\mathcal{Z}_{\mathcal{C}}^{\perp}$ is to include the new distinguished terms which are explicitly needed in the various lifted-totalisation semantics. Specifically: the types of $\mathcal{Z}_{\mathcal{C}}$ are extended to include terms $\perp^T$ for every type T. There are, additionally, a number of axioms which ensure that all the new $\perp^T$ values interact properly, *e.g.*

$$\frac{}{\perp^{[\mathtt{z_0}:\mathtt{T_0}\cdots\mathtt{z_n}:\mathtt{T_n}]} = \langle\!| \, \mathbf{z}_0 \Rrightarrow \perp^{T_0} \cdots \mathbf{z}_n \Rrightarrow \perp^{T_n} |\!\rangle}$$

In other words, $\perp^{[\mathtt{z_0}:\mathtt{T_0}\cdots\mathtt{z_n}:\mathtt{T_n}]}.\mathbf{z}_i = \perp^{T_i}$ $(0 \leq i \leq n)$. Note that this is the *only* axiom concerning distinguished bindings; hence, binding construction is *non-strict* with respect to the $\perp^T$ values.

Finally, the extension of $\mathcal{Z}_{\mathcal{C}}^{\perp}$ which introduces schemas as sets of bindings and the various operators of the schema calculus is undertaken as usual (see *e.g.* [25]), 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 $\perp^T$ values:

Definition B.1 *Natural carriers* for each type are defined by closing: *e.g.* $\mathbb{N} =_{df} \{z^{\mathbb{N}} \mid z \neq \perp^{\mathbb{N}}\}$ under the type forming operations (*i.e.* cartesian product, power type and schema type).

Naturally, the following elimination rule is derivable for natural carriers:

Proposition B.2

$$\frac{t \in T}{t \neq \perp} \ (NatCar^-)$$

□

As a result, the schema calculus is *hereditarily* $\perp$*-free*:

Definition B.3 [Semantics for Atomic Schemas]

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

Note that this definition draws bindings from the *natural carrier* of the type T. As a consequence, writing $t(\perp)$ for a binding satisfying $t.\mathtt{x} = \perp$ for some observation $\mathtt{x}$, we have:

Lemma B.4

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

We proved (in [16]) that the $\mathcal{Z}_{\mathcal{C}}^{\perp}$ core is *conservative* over the $\mathcal{Z}_{\mathcal{C}}$ core, and (in [17]) that the schema calculus in $\mathcal{Z}_{\mathcal{C}}^{\perp}$ *preserves the meaning* of the schema calculus in $\mathcal{Z}_{\mathcal{C}}$.