An analysis of total correctness refinement models for partial relation semantics I

MOSHE DEUTSCH, MARTIN C. HENSON, Department of Computer Science, University of Essex, Wivenhoe Park, Colchester, Essex CO4 3SQ, UK
E-mail: {mdeuts, hensm}@essex.ac.uk

STEVE REEVES, Department of Computer Science, University of Waikato, Private Bag 3105, Hamilton, New Zealand
E-mail: stever@cs.waikato.ac.nz

Abstract
This is the first of a series of papers devoted to the thorough investigation of (total correctness) refinement based on an underlying partial relational model. In this paper we restrict attention to operation refinement. We explore four theories of refinement based on an underlying partial relation model for specifications, and we show that they are all equivalent. This, in particular, sheds some light on the relational completion operator (lifted-totalisation) due to Woodcock which underlies data refinement in, for example, the specification language Z. It further leads to two simple alternative models which are also equivalent to the others.

Keywords: operation refinement, specification language, specification logic

1 Introduction
In this paper we provide a thorough investigation of total correctness operation refinement (that is the degenerate case of data refinement in which simulations are identity functions) where the underlying semantics of specifications is given in terms of partial relations. A major example of such a semantics is the specification language Z, in which a specification denotes a set of bindings which can be construed to be a partial relation between input sub-bindings and output sub-bindings (see [18], [11], [6] and [8] for accounts of Z logic and semantics along these lines). Woodcock, in the book [18] (chapter 16), describes a general account of data refinement in which a relational operator (which we will call the Woodcock-completion) plays the crucial underlying role. This paper can be construed, at one level, to be providing a detailed exposition and mathematical account of that relational completion operation by providing a systematic and comprehensive comparison with other possible approaches. In addition, since the theme of combining formal methods is now an important re-

\footnote{1}{Now that we have established that we are dealing with total correctness, we will drop this qualification: unless we specify otherwise, we are interested only in total correctness refinement.}

\footnote{2}{[5] (chapter 8) also introduces a similar idea.}
In order that our general analysis of partial relations can be extended to Z in particular (which will permit in due course an investigation of the schema calculus) we will base our mathematical investigation on the logic $Z_C$ described in [11]. This is an extension of higher order logic with schema types and filtered bindings which can be extended conservatively to cover the key features of Z. We will need, in fact, a theory extension of $Z_C$ which includes additional constants and axioms in order to provide an account of the Woodcock-completion. The base logic and the theory extension, are presented for completeness in appendix A.

Our investigation is essentially proof-theoretic. We have found that the connections and relationships between various theories of refinement can be most clearly demonstrated in this mathematical style. Even where the underlying notions of refinement are model-theoretic, we will nevertheless express them as theories: they are characterised by sets of introduction and elimination rules. This permits a particularly straightforward strategy for demonstrating that theories are equivalent, or related, which is quite general. This strategy is explained in section 4 below.

The paper is structured in the following way. We first revise Z logic, setting some notation and outlining this important application area for the partial relation semantics. We then move on, in section 3, to provide four theories of refinement, which are prima facie distinct, based on various alternative semantics for specifications: the first is based on sets of implementations, the second on properties of preconditions and postconditions, the third on a relational completion operator, and the last on weakest preconditions. In section 4 we go on to demonstrate that all four of these theories are equivalent. We are led to introduce further models in sections 5 and 6 which are motivated by our earlier results. In addition to our conclusions, acknowledgements and references, we conclude with an appendix setting out the mathematical basis for the entire investigation.

2 Preliminaries

In this first section we will revise a little Z logic, settling some notational conventions in the process.

In [11] 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. [7], [18]). In this paper we will use the meta-variable $U$ (with decorations) to range over operation schema. As an example, consider the operation schema:

\[\text{Example: Operation Schema} \]

Unlike our earlier papers, [11] attempts to formalise Z as it is informally understood. Further and additional detail, including a summary of the ideas introduced here, can be found in appendix A and in [11]. The appendix covers, in addition to a description of the core logic $Z_C$ and the theory $Z_C^\perp$, a section demonstrating that $Z_C^\perp$ is conservative over $Z_C$. Our objective is to make the appendix here sufficiently complete so that it can serve to support the other papers in this series in the future. All the mathematics in the main body of this paper takes place in the extended theory $Z_C^\perp$ which is described in the appendix. In certain circumstances (e.g. sections 3.2 and 3.4 below) some theories can be formalised in the core logic $Z_C$; since this is technically significant, we will indicate the fact explicitly.
2. **PRELIMINARIES**

$E_{x_0}$

\[
\begin{align*}
\text{x, x': N} \\
\text{x = 0} \\
\text{x' < 10}
\end{align*}
\]

$E_{x_0}$ has the type $\mathbb{P}[x : \mathbb{N}, x' : \mathbb{N}]$, and is understood to be a set of bindings of schema type $[x : \mathbb{N}, x' : \mathbb{N}]$. Recall that unprimed labels (such as $x$) are understood to be observations of the state before the operation, whereas primed labels (such as $x'$) are observations of the state afterwards. Each operation schema $U$ will have a type of the form $\mathbb{P} \cdot T$ where $T$ is a schema type. The type $T$ can, additionally, always be partitioned as the (compatible) union of its input (or before) type $T^\text{in}$, and its output (or after) type $T^\text{out}$. That is, $T = T^\text{in} \uplus T^\text{out}$. For the schema $E_{x_0}$ we have $T^\text{in} = [x : \mathbb{N}]$ and $T^\text{out} = [x' : \mathbb{N}]$. In this paper, since we are only dealing with operation refinement, we can assume that all operation schemas have the type $\mathbb{P} \cdot T$ where $T = T^\text{in} \uplus T^\text{out}$. With this in place we can omit the type superscripts in most places in the sequel.

**Definition 2.1** (semantics for atomic schemas)

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

Note that this definition draws bindings from the natural carrier of the type $T$. As a consequence, writing $t(\perp)$ for any binding which projects an instance of the constant $\perp$ (that is: $t.x = \perp$ for some observation $x$), we have:

**Lemma 2.2**

\[
\frac{t(\perp) \in U}{false}
\]

The bindings $\langle x \mapsto 0, x' \mapsto n \rangle$, where $n < 10$, are all elements of $E_{x_0}$. In fact there are no other elements in this case. We can introduce the idea of the precondition of the schema (domain of the relation the schema denotes) to express the partiality involved.

**Definition 2.3** Let $T^\text{in} \preceq V$.

\[
\text{Pre } U (x^V) =_{df} \exists z \in U \cdot x =_{T^\text{in}} z
\]

**Proposition 2.4** The following introduction and elimination rules are immediately derivable for preconditions:

\[
\frac{t_0 \in U}{t_0 =_{T^\text{in}} t_1}
\]

\[
\frac{t \in U \quad y \in U \cdot y =_{T^\text{in}} t \vdash P}{P}
\]

where $y$ is fresh.

---

5 In the technical development it is usual to write the operation schema horizontally, with a vertical line separating the declarations from the predicate.
An analysis of total correctness refinement models for partial relation semantics

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, $T^w$). In this sense operation schemas denote partial relations.

Naturally, an immediate question will arise: 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 A basic analysis of refinement

We begin by introducing four distinct notions of refinement based on four 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 Woodcock-completion which is the \textit{de facto} standard for Z.

3.1 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. We do this, via a standard expedient of introducing a special value $\perp$, which might represent unwelcome behaviour. Such behaviour might be non-termination but it need not, and nothing we shall do with this value commits us to that interpretation. We shall pronounce $\perp$ the \textit{abortive} value. In [18] the authors refer to it as \textit{undefined} which is, we feel, unfortunate in the context of partial relations; we will have much more to say about this below in section 4.4. In order to introduce this value into the analysis, the technical development below takes place in the extended theory $ZC\perp$.

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 \textit{implementation}. To signal this interpretation we shall in fact write this judgement as:

$$g \in U$$

to be pronounced “$g$ implements (is an implementation of) $U$”.

Now an operation schema has been modelled as a partial relation, so we need to consider how an implementation behaves outside the precondition of the schema (domain of the underlying relation). There are degrees of freedom, but we shall, in this paper, permit what we call \textit{chaotic models}. More exactly, we understand silence in the specification to be permission for an implementation to behave in any arbitrary manner,

---

6If we were not interested in total correctness, we could simply take the subset relation on the partial relations as refinement, and we would then obtain a very well-behaved theory which also makes the schema calculus fully monotonic with respect to refinement but necessarily treats preconditions as triggers (firing conditions).

7There is an obvious alternative, based on \textit{abortive models} or what in [10] is effectively the \textit{partial model}; we will say a little more in section 4.4 and it will be investigated in our future work.
including the abortive behaviour \(⊥\). In other respects we will expect a model (an implementation) to produce a result which is in the relation whenever it is supplied with an input inside the precondition. This leads to the following definition of the modelling (implementing) relation.

**Definition 3.1**

\[
g \in_f U = \text{df} (\forall z \in T^{in}_\perp \cdot \text{Pre} \ U \ z \Rightarrow z \ast (g \ z)' \in U) \land g \in T^{in}_\perp \rightarrow T^{out}_\perp
\]

Then we can prove the following.

**Proposition 3.2** The following introduction and elimination rules are derivable:

\[
\frac{z \in T^{in}_\perp, \text{Pre} \ U \ z \vdash z \ast (g \ z)' \in U \ g \in T^{in}_\perp \rightarrow T^{out}_\perp}{g \in_f U} (e^+_f)
\]

where \(z\) is a fresh variable.

\[
\frac{g \in_f U \ \text{Pre} \ U \ t \vdash t \ast (g \ t)' \in U \ g \in T^{in}_\perp \rightarrow T^{out}_\perp}{g \in_f U} (e^-_{f,m})
\]

\[
\frac{g \in_f U \ g \in T^{in}_\perp \rightarrow T^{out}_\perp}{g \in_f U} (e^-_{f,r})
\]

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 3.3**

\[
\hat{U} = \text{df} \{ z \mid z \in_f U \}
\]

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

**Definition 3.4**

\[
U_0 \supseteq_f U_1 = \text{df} \ \hat{U}_0 \subseteq \hat{U}_1
\]

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

### 3.2 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, [16] and as discussed in [13] and [15]. In those contexts we do not so much have an alternative notion of refinement as two sufficient conditions (essentially the premises of the introduction rule in proposition 3.5 below). By adding the two elimination rules we add necessary conditions, and thus formalise an independent theory. There is also a connection with theorem 3.1.2 of [12] (page 77), although that analysis concerns
An analysis of total correctness refinement models for partial relation semantics
the two-predicate designs of the refinement calculus (syntactic preconditions) rather
than the single predicate specifications of a language like Z (logical preconditions).
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 a refinement may 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, named for Mike Spivey, is written $U_0 \sqsubseteq_s U_1$ and
is given by the definition that leads directly to the following rules:

**Proposition 3.5** Let $z$, $z_0$, $z_1$ be fresh variables.

$$
\frac{\text{Pre } U_1 \ z \vdash \text{Pre } U_0 \ z}{U_0 \sqsubseteq_s U_1} (\sqsubseteq_s^+) \\
\frac{U_0 \sqsubseteq_s U_1 \ \text{Pre } U_0 \ t \ \text{Pre } U_1 \ t}{U_0 \sqsubseteq_s U_1 \ \text{Pre } U_0 \ t \ \text{Pre } U_1 \ t} (\sqsubseteq_s^-)
$$

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

### 3.3 $W$-refinement

Our third notion of refinement is taken directly from the literature [18]. It is based
on a relational completion operator due to Woodcock. For notational convenience we
will write $T^*$ for the set $T^*_1 \star T^*_2$.

The *lifted totalisation* of a set of bindings (Woodcock-completion) can be defined as
follows:

**Definition 3.6**

$$
\dot{U} = \text{df } \{z_0 \star z'_1 \in T^* \mid \text{Pre } U \ z_0 \Rightarrow z_0 \star z'_1 \in U\}
$$

**Proposition 3.7** The following introduction and elimination rules are derivable for
lifted totalised sets:

$$
\frac{t_0 \star t'_1 \in T^* \ \text{Pre } U \ t_0 \vdash t_0 \star t'_1 \in U}{t_0 \star t'_1 \in \dot{U}} (\dagger)
$$

and:

$$
\frac{t_0 \star t'_1 \in \dot{U} \ \text{Pre } U \ t_0}{t_0 \star t'_1 \in \dot{U}} (\dagger^-) \quad \frac{t_0 \star t'_1 \in U}{t_0 \star t'_1 \in T^*} (\dagger^-)
$$

$\blacksquare$
Note that it is, sometimes, useful to use the following version of \( (\bullet_0) \) rule, which is based upon disjunction elimination, rather than implication elimination.

**Proposition 3.8**

\[
\begin{aligned}
t_0 \ast t_1' \in U & \quad \neg \text{Pre } U t_0 \vdash P \\
& \quad t_0 \ast t_1' \in U \vdash P \tag{\(\bullet_0\)}
\end{aligned}
\]

**Lemma 3.9** The following are derivable:

\[
\begin{array}{lll}
U \subseteq \bigdot{U} & (i) & \bot \in \bigdot{U} & (ii) & \neg \text{Pre } U t_0 \vdash t_0 \in T_{\text{in}}^\perp \quad t_1' \in T_{\text{out}}^\prime \\
& & t_0 \ast t_1' \in \bigdot{U} & (iii)
\end{array}
\]

**Proof.** (i) is trivial. For (ii), consider the following derivation:

\[
\begin{aligned}
& y \in U \quad (2) \\
\frac{y = \bot \perp}{\bot = \bot \perp} & (2) \\
\frac{\bot \in T^\ast}{\bot \in \bigdot{U}} & (1) \\
\frac{\bot \in \bigdot{U}}{\bot \in \bigdot{U}} & (2)
\end{aligned}
\]

For (iii), consider the following derivation:

\[
\begin{aligned}
& t_0 \in T_{\text{in}}^\perp \quad t_1' \in T_{\text{out}}^\prime \quad \neg \text{Pre } U t_0 \quad \text{Pre } U t_0 \quad (1) \\
& t_0 \ast t_1' \in T^\ast \\
\frac{\bot = \bot \perp}{\bot \ast \bot \perp} & (2) \\
\frac{\bot \ast t_1' \in \bigdot{U}}{t_0 \ast t_1' \in \bigdot{U}} & (1)
\end{aligned}
\]

Lemmas 3.8(i), (ii) and (iii) demonstrate that definition 3.6 is consistent with the intentions described in [18] chapter 16: the underlying partial relation is contained in the completion, the abortive element is present in the relation, and more generally, each value outside the precondition maps to every value in the co-domain of the relation. Furthermore, the following rules, which are derived from lemma 3.8(iii), embody the non-strict lifting with respect to the abortive element and the fact that everything outside the precondition is mapped onto the abortive value (as well as everything else in the co-domain of the relation).

**Corollary 3.10**

\[
\begin{array}{ll}
t_1' \in T_{\text{out}}^\prime & (i) \\
\bot \ast t_1' \in \bigdot{U} & \neg \text{Pre } U t \quad t \in T_{\text{in}}^\perp & (ii) \\
\frac{t \ast \bot \in \bigdot{U}}{t \ast t_1' \in \bigdot{U}} & (i)
\end{array}
\]
An analysis of total correctness refinement models for partial relation semantics I

$W_{\bullet}$-refinement, written $U_0 \sqsubseteq_{w_{\bullet}} U_1$, and named for Jim Woodcock, is defined as follows.

**Definition 3.11**

\[ U_0 \sqsubseteq_{w_{\bullet}} U_1 \iff U_0 \subseteq U_1 \]

Obvious introduction and elimination rules follow from this.

### 3.4 WP-refinement

Our final theory of refinement is that based on a weakest-precondition interpretation. In order to formalise this we begin by introducing a notion of postcondition to complement the precondition we introduced earlier.

**Definition 3.12**

\[ Post U z_0 = \{ z'_1 \mid z_0 \star z'_1 \in U \} \]

Note that this introduces a set, rather than a predicate.

With this in place we can introduce the weakest precondition interpretation of an operation schema. Again, the specified postcondition ($C$ in the definition below) is expressed as a set rather than as a predicate.

**Definition 3.13**

\[ wp U C = \{ z \mid Pre U z \land Post U z \subseteq C \} \]

The reason why we choose to work with sets, rather than predicates, is simply that it casts the technical material in a similar style to the models we introduced earlier for F-refinement and $W_{\bullet}$-refinement, which also construct sets from the underlying partial relations.

**Proposition 3.14** The following introduction and elimination rules for the weakest precondition of $U$ are derivable:

\[
\frac{Pre U t \quad z' \in Post U t \quad z' \in C}{t \in wp U C}
\]

where $z$ is a fresh variable.

A weakest precondition semantics for Z is provided in [3], a paper based on the semantics of Z to be found in the (then) draft Z standard [4] (now superseded by [5]). It would be very interesting to investigate the relationship between the two approaches, but that is beyond the scope of this paper. In passing we note that they provide an interpretation over the syntax of Z (atomic operation schema expressions) whereas we opt for one over the partial relational semantics (sets of bindings). Generality (an interpretation over all schema expressions) is obtained in two significantly different ways: in ours it follows since all schema expressions denote sets of bindings through the semantics; [3] on the other hand relies on the fact that all schema expressions may be written in the form of an atomic schema. That, in turn, relies on the standard equational logic of schemas. We make some further remarks about our semantic approach, and the equational logic, in section 8. These considerations will become even more significant when, in future work, we examine monotonicity issues and what would be necessary to obtain fully monotonic schema calculi for Z.
We can now define WP-refinement.

**Definition 3.15**

\[ \mathcal{U}_0 \sqsupseteq_{wp} \mathcal{U}_1 \equiv_{df} \forall \mathcal{C} \cdot \mathcal{T}^{out} \bullet \wp \mathcal{U}_1 \mathcal{C} \subseteq \wp \mathcal{U}_0 \mathcal{C} \]

**Proposition 3.16** The following introduction and elimination rules for WP-refinement are derivable:

\[ \frac{z \in \wp \mathcal{U}_1 \mathcal{C}}{z \in \wp \mathcal{U}_0 \mathcal{C}} \]  
\[ \frac{\mathcal{U}_0 \sqsupseteq_{wp} \mathcal{U}_1}{\mathcal{T} \in \wp \mathcal{U}_1 \mathcal{C} \subseteq \wp \mathcal{U}_0 \mathcal{C}} \]

where \( z \) and \( \mathcal{C} \) are fresh variables.

**Lemma 3.17**

\[ \frac{\text{Pre } \mathcal{U} \mathcal{T}}{\mathcal{T} \in \wp \mathcal{U} \text{ (Post } \mathcal{U} \mathcal{T})} \]

**Proof.**

\[ \frac{\text{Pre } \mathcal{U} \mathcal{T}}{\mathcal{T} \in \wp \mathcal{U} \text{ (Post } \mathcal{U} \mathcal{T})} \]

Since the reverse direction always holds, we have established that \( \mathcal{T} \in \wp \mathcal{U} \text{ (Post } \mathcal{U} \mathcal{T}) \) and Pre \( \mathcal{U} \mathcal{T} \) are equivalent.

These interpretations will be reminiscent from accounts such as [14], although it is unusual to provide a proof-theoretic presentation in terms of introduction and elimination rules. In addition, in [14], the preconditions and postconditions of specification statements are syntactic (explicitly given), rather than logical (implicitly given) as they are in Z. Note that our interpretation does not involve the \( \bot \) values, and could therefore be formalised in ZC.

**3.5 Review**

What is the relationship between these four notions of refinement? In particular, can an exploration of that question shed any light on why the Woodcock-completion has been defined in just the way it has? What, in particular, is the role of the value \( \bot \)? Why is the lifting process non-strict with respect to the abortive value? We will begin with the first of these questions.

**4 Four equivalent theories**

In this section we demonstrate that our four theories of refinement are all equivalent. In doing this we will see clearly the critical role that the \( \bot \) 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
An analysis of total correctness refinement models for partial relation semantics I

294

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.9

4.1 F-refinement and W•-refinement are equivalent (in Z⁺C + AC)

4.1.1 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•-refinement.

Definition 4.1

\[ U = \{ z \in T^\text{in}_\perp \rightarrow T^\text{out}' \mid z \subseteq \cdot \} \]

Then we have:

Definition 4.2

\[ g \in_r U = \text{df} g \in U \]

And then R-refinement is simply: \( U_0 \sqsupseteq_r U_1 = \text{df} U_0 \subseteq U_1 \) with the usual introduction
and elimination rules.

4.1.2 R-refinement and W•-refinement are equivalent

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

Proposition 4.3 The following rule is derivable:

\[ U_0 \sqsupseteq_r U_1 \quad t \in U_0 \quad t \in U_1 \]

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

An alternative strategy would be to show that a similar property holds for the introduction rule. In the refinement
theories we consider there is only ever a single introduction rule, and this suggests that this might be a more
efficient approach. However, there are as many premises to the introduction rule as there are distinct elimination
rules, so in the end the amount of work involved is essentially the same. Moreover, by considering the elimination
rules separately, we can in some cases (see for example sections 4.2 and 4.4 below) isolate particular properties
and reasons underlying equivalence (or non-equivalence in other circumstances) which highlight particular issues of
interest.

---

9 An alternative strategy would be to show that a similar property holds for the introduction rule. In the refinement
theories we consider there is only ever a single introduction rule, and this suggests that this might be a more
efficient approach. However, there are as many premises to the introduction rule as there are distinct elimination
rules, so in the end the amount of work involved is essentially the same. Moreover, by considering the elimination
rules separately, we can in some cases (see for example sections 4.2 and 4.4 below) isolate particular properties
and reasons underlying equivalence (or non-equivalence in other circumstances) which highlight particular issues of
interest.
4. **FOUR EQUIVALENT THEORIES**

Where \( \delta \) is:

\[
\begin{align*}
U_0 \supseteq_r U_1 & \quad y \in T_1^{\text{out}} \implies y \subseteq U_0 \\
& \quad U_0 \supseteq w \cdot U_1 \\
& \quad y \subseteq U_0 \\
& \quad t \subseteq U_1 \\
& \quad t \in U_1 
\end{align*}
\]

From this we have:

**Theorem 4.4**

\[
\begin{align*}
U_0 \supseteq_r U_1 & \quad U_0 \supseteq w \cdot U_1 
\end{align*}
\]

**Proof.** This follows immediately, by \((\supseteq w \cdot)\), from proposition 4.3.

We now show that \(W\cdot\)-refinement satisfies the R-refinement elimination rule.

**Proposition 4.5**

\[
\begin{align*}
U_0 \supseteq w \cdot U_1 & \quad g \in U_0 \\
& \quad g \in U_1 
\end{align*}
\]

**Proof.**

\[
\begin{align*}
U_0 \supseteq w \cdot U_1 & \quad g \subseteq U_0 \\
& \quad t \in g \\
& \quad U_0 \supseteq w \cdot U_1 \\
& \quad t \in U_0 \\
& \quad g \subseteq U_1 \\
& \quad t \in U_1 \\
& \quad g \in T_1^{\text{in}} \implies T_1^{\text{out}} \\
& \quad g \in U_1 
\end{align*}
\]

**Theorem 4.6**

\[
\begin{align*}
U_0 \supseteq w \cdot U_1 & \quad U_0 \supseteq_r U_1 
\end{align*}
\]

Theorems 4.4 and 4.6 together demonstrate that \(W\cdot\)-refinement and R-refinement are equivalent.

\[\text{\textsuperscript{10}The proofs of such theorems are always automatic by the structural symmetry between introduction and elimination rules. We shall not give them in future.}\]
4.1.3 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 4.7** The following rules are derivable:

\[
\begin{array}{c}
g \in_f U \\
g \subseteq U \\
g \in T_{\downarrow}^{in} \rightarrow T_{\downarrow}^{out}
\end{array}
\]

**Proof.**

\[
\frac{g \in_f U \quad z_0 \cdot z_1' \in g}{z_0 \cdot z_1' \in T^*} \quad \quad \quad \quad \quad \frac{g \in_f U \quad Pre U \ z_0 \quad z_0 \cdot (g \ z_0)' \in U \quad z_0 \cdot z_1' \in U}{z_0 \cdot z_1' \in U} \quad \frac{Pre U \ z_0 \quad z_0 \in T_{\downarrow}^{in} \quad \delta}{z_0 \in T_{\downarrow}^{in}}
\]

where \( \delta \) is:

\[
\frac{g \in_f U \quad z_0 \cdot z_1' \in g}{z_0 \cdot z_1' \in T^*}
\]

The second rule is immediate.

**Theorem 4.8**

\[
\frac{g \in_f U \quad g \subseteq U}{g \in_r U}
\]

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

**Proposition 4.9**

\[
\begin{array}{c}
g \in_r U \\
Pre U \ t \quad t \in T_{\downarrow}^{in} \quad g \in_r U \\
t \cdot (g \ t)' \in U \\
g \in_r U \\
t \cdot (g \ t)' \in g \\
Pre U \ t \quad t \cdot (g \ t)' \in U
\end{array}
\]

**Proof.**

\[
\frac{g \in_r U \quad g \in T_{\downarrow}^{in} \rightarrow T_{\downarrow}^{out}}{t \cdot (g \ t)' \in g} \quad \frac{g \in_r U \quad g \subseteq U}{g \in T_{\downarrow}^{in} \rightarrow T_{\downarrow}^{out}} \quad \frac{g \in_r U \quad t \in T_{\downarrow}^{in}}{t \cdot (g \ t)' \in g}
\]

The second rule is immediate.
4. **FOUR EQUIVALENT THEORIES**

Theorem 4.10

\[
\begin{align*}
g \in r U & \quad \Rightarrow \quad g \in f U
\end{align*}
\]

Then, from 4.8 and 11.10, we see that the two notions of implementation are equivalent. Hence, so are the two notions of refinement.

4.2 \textit{W•}-refinement and S-refinement are equivalent

We begin by showing that \textit{W•}-refinement satisfies the two S-refinement elimination rules. Firstly the rule for preconditions.

**Proposition 4.11** The following rule is derivable:

\[
\begin{array}{c}
U_0 \unlhd_{w+} U_1 \quad \text{Pre } U_1 \; t \\
\hline
\text{Pre } U_0 \; t
\end{array}
\]

**Proof.** Consider the following derivation:

\[
\begin{array}{c}
\neg \text{Pre } U_0 \; t \quad \text{(i)} \\
\hline
\text{Pre } U_1 \; t \\
\hline
\text{false} \quad \text{(ii)} \\
\hline
\text{Pre } U_0 \; t \quad \text{(i)}
\end{array}
\]

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

**Proposition 4.12** The following rule is derivable:

\[
\begin{array}{c}
U_0 \unlhd_{w+} U_1 \quad \text{Pre } U_1 \; t_0 \ast t'_1 \in U_0 \\
\hline
\text{false} \\
\hline
\text{Pre } U_0 \; t_0 \quad \text{(i)}
\end{array}
\]

**Proof.**

\[
\begin{array}{c}
t_0 \ast t'_1 \in U_0 \quad \text{(i)} \\
\hline
\text{Pre } U_1 \; t_0 \\
\hline
\text{Pre } U_0 \; t_0 \quad \text{(i)}
\end{array}
\]

**Theorem 4.13**

\[
U_0 \unlhd_{w+} U_1 \\
U_0 \unlhd_s U_1
\]

\[\square\]
We now show that S-refinement satisfies the $W_\bullet$-elimination rule.

**Proposition 4.14**

\[
\frac{U_0 \succeq_s U_1 \quad t_0 \ast t_1' \in U_0}{t_0 \ast t_1' \in \dot{U}_1}
\]

**Proof.**

\[
\frac{t_0 \ast t_1' \in \dot{U}_0 \quad U_0 \succeq_s U_1 \quad \text{Pre } U_1 t_0}{t_0 \ast t_1' \in \dot{U}_1 \quad \text{Pre } U_0 t_0} \quad (1)
\]

**Theorem 4.15**

\[
\frac{U_0 \succeq_s U_1}{U_0 \succeq_{w_\bullet} U_1}
\]

Theorems 4.13 and 4.15 together establish that the theories of S-refinement and $W_\bullet$-refinement are equivalent.

### 4.3 WP-refinement and S-refinement are equivalent

We begin by showing that WP-refinement satisfies the two S-refinement elimination rules. In these results we will often use the fact that $t_1' \in \text{Post } U t_0$ and $t_0 \ast t_1' \in U$ are equivalent without further comment. First we have the rule for preconditions.

**Proposition 4.16**

\[
\frac{U_0 \succeq_{wp} U_1 \quad \text{Pre } U_1 t}{\text{Pre } U_0 t}
\]

**Proof.** Consider the following derivation:

\[
\frac{\text{Pre } U_1 t \quad U_0 \succeq_{wp} U_1 \quad t \in wp \; U_1 \left(\text{Post } U_1 t\right)}{t \in wp \; U_0 \left(\text{Post } U_1 t\right) \quad \text{Pre } U_0 t \quad (3.17)}
\]

Now the second elimination rule.

**Proposition 4.17** The following rule is derivable:

\[
\frac{U_0 \succeq_{wp} U_1 \quad \text{Pre } U_1 t_0 \quad t_0 \ast t_1' \in U_0}{t_0 \ast t_1' \in U_1}
\]
4. FOUR EQUIVALENT THEORIES

Proof. Consider the following derivation:

\[
\begin{array}{c}
U_0 \preceq \wp U_1 \\
\text{Pre } U_1 \ t_0
\end{array}
\overset{\text{Pre } U_1 \ t_0}{\Rightarrow} \begin{array}{c}
\ t_0 \in \wp U_1 \ (Post \ U_1 \ t_0)
\end{array}
\overset{(4.17)}{\Rightarrow} \begin{array}{c}
t_0 \in \wp U_0 \ (Post \ U_1 \ t_0)
\end{array}
\overset{(1)}{\Rightarrow} \begin{array}{c}
t_0 \star t_1' \in U_1
\end{array}
\]

Theorem 4.18

\[
U_0 \preceq \wp U_1 \quad U_0 \preceq s U_1
\]

We now show that every S-refinement is a WP-refinement.

Proposition 4.19

\[
U_0 \preceq s U_1 \quad t \in \wp U_1 C
\]

Proof. Consider the following derivation:

\[
\begin{array}{c}
U_0 \preceq s U_1 \ t \in \wp U_1 C
\end{array}
\overset{\text{Pre } U_1 \ t}{\Rightarrow} \begin{array}{c}
t \in \wp U_0 C
\end{array}
\overset{(1)}{\Rightarrow} \begin{array}{c}
t \star t' \in U_1
\end{array}
\overset{(1)}{\Rightarrow} \begin{array}{c}
t' \in C
\end{array}
\]

Theorem 4.20

\[
U_0 \preceq s U_1
\]

Theorems 4.18 and 4.20 establish that WP-refinement and S-refinement are equivalent.

4.4 Review

The model of schemas introduced in W•-refinement not only totalises the schema as a set of bindings, it also introduces the \(\bot\) values and extends the domains and codomains accordingly. The totalisation then stipulates chaotic behaviour outside the precondition and additionally for the \(\bot\) values.

Why is it necessary to include the new values? What are the consequences of totalisation without lifting?

In [18], the authors explicitly discuss these questions.\footnote{They call the \(\bot\) value “undefined”, which is perhaps unfortunate since this is also used with reference to values outside the domain of definition of particular schemas. We will continue to refer to call it the “abortive” value.} By way of explanation they consider the particular schema:
An analysis of total correctness refinement models for partial relation semantics

\[ \kappa \]
\[ x, x' : \mathbb{N} \]
\[ x' = 0 \]

This denotes a total constant relation in the model. They then illustrate carefully that lifting ensures that schema composition is strict with respect to chaotic behaviour.\(^\text{12}\)

On the other hand, totalisation without lifting leads to non-strict recovery from chaos. First we introduce the chaotic specification.

**Definition 4.21**

\[ \text{Chaos} =_{df} [T | \text{false}] \]

Now we describe the non-lifted totalisation, by ensuring that the values are drawn only from natural carrier set rather than the extension including the abortive value.

**Definition 4.22**

\[ \overset{\circ}{U} =_{df} \{ z \in T | \text{Pre } U z \Rightarrow z \in U \} \]

**Proposition 4.23 (Woodcock and Davies)**

\[ \begin{align*}
(i) & \quad \overset{\ast}{\text{Chaos}} \overset{\circ}{\kappa} =_{df} \overset{\ast}{\text{Chaos}} \\
(ii) & \quad \overset{\circ}{\text{Chaos}} \overset{\circ}{\kappa} =_{df} \kappa
\end{align*} \]

**Proof.** See [18], page 238.

It should be noted that the second of these results is contingent on the particular choice \( \kappa \): it is not true in general. But this observation notwithstanding, the interpretation of the results requires care. \( \text{Chaos} \) is described as representing undefinedness, or a run-time error being encountered whatever the initial value. This is odd since, in particular, \( \text{Chaos} \) permits the input \( \perp \) to result in a well-defined output (lifting is non-strict with respect to \( \perp \)). \( \text{Chaos} \), as is indicated by our nomenclature, is a relation that permits a chaotic relationship between input and output. It is the relation \( \{ z_0 \ast z'_1 \in T^* | z'_1 = \perp \} \) that would be closer to what they have in mind (since they refer to \( \perp \) as the undefined value). We, for reasons discussed above, will call this \( \text{Abort} \).

Now \( W_{\ast}\)-refinement is defined as the subset relation on the Woodcock-completion. Since \( \text{Chaos} \) is the whole of \( T^* \), every schema refines it. In particular the identity relation refines it, and this is the identity for composition. Hence one has\(^\text{13}\)

\[ \kappa = \text{Identity} \overset{\circ}{\kappa} \supseteq \text{Chaos} \overset{\circ}{\kappa} \]

for any \( \kappa \), and this would not appear to be recovery from run-time error, but a natural consequence of the general permissiveness inherent in \( \text{Chaos} \), indeed, a natural consequence of the fact that the Woodcock-completion is non-strict with respect to the abortive value.

---

\(^{12}\)Note, this is strictness with respect to chaos, not the abortive value.

\(^{13}\)At least when \( \kappa \) is e.g. total, and so when composition can be guaranteed to be monotonic. We will explore this further in our companion papers.
5. **THE NON-LIFTED TOTALISATION**

Our analysis has, on the other hand, provided a very clear mathematical explanation for lifting: with non-lifted totalisation it is not possible to prove proposition 4.11 (which requires explicit use of $\bot$ value). Indeed, we can do better: the following is an explicit counterexample.

**Definition 4.24**

$$\text{True} =_{df} [T \mid \text{true}]$$

**Proposition 4.25**

$$\doveq{\text{True}} = \doveq{\text{Chaos}}$$

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

We have, however, only begun to provide answers to the natural questions that arise. For example, although lifting appears to be necessary, why does it have to be non-strict with respect to $\bot$? Proposition 4.25 also raises a question: why is there a distinction between implicit (Chaos) and explicit (True) permission to behave? Note that in the Woodcock-completion, $\text{True} \neq \text{Chaos}$.

5. **The non-lifted totalisation**

In the previous section we noted the asymmetry between implicit and explicit chaos. Implicit chaos is more extensive, it permits abortive behaviour that explicit chaos does not allow. This asymmetry seems inevitable in order to obtain a reasonable theory of refinement. This is, as we have shown, indeed the case, unless we re-examine the nature of preconditions.

It seemed only natural to identify the notion of the precondition of a schema with the domain of definition of the underlying relation. There is, however, an alternative approach. Instead of taking a value to be in the precondition when it is related to at least one element of the co-domain of the underlying relation, we could take the condition to be that a value is not related to at least one element of the co-domain of the completed relation. This anticipates the idea that a value which is not in the domain of definition of the underlying relation, will be related to all values in the co-domain after the relation is completed: it excludes from the precondition values in the underlying relation which are already related to all values in the co-domain.

Surprisingly, this leads to a theory which can be formalised entirely in $\mathbb{ZC}$ (it does not require lifting at all) and which is equivalent to the theories of the previous section. In this way we show that, for operation refinement at least, that lifting of relations is not necessary if one wishes to establish a relational completion semantics for refinement.

5.1 **Preconditions revisited**

In this section we refer to the standard definition of preconditions as $\text{Pre}_0$ in order to contrast it clearly with the new one.

---

14 One might take S-refinement to establish minimal conditions.
Definition 5.1 Let $T^{in} \leq V$.

$$\text{Pre} U \ z^V =_{df} \exists x_0', x_1' \in T^{out'} \bullet z \vdash T^{in} \ast x_0' \notin U \land z \vdash T^{in} \ast x_1' \in U$$

There is a similarity (but not quite an equivalence) here with the total model described in [10] (page 45). His interpretation of specifications as predicates in that model makes use of a similar concept of precondition to $\text{Pre}_1$, although this is not made explicit.

Proposition 5.2 The following introduction and elimination rules are derivable for preconditions:

\[
\frac{t \ast t_0' \notin U \quad t \ast t_1' \in U \quad t_0' \in T^{out'}}{\text{Pre}_1 U t} \quad (\text{Pre}^+_1)
\]

\[
\frac{\text{Pre}_1 U t \quad t \vdash T^{in} \ast y_0' \notin U, t \vdash T^{in} \ast y_1' \in U, y_0' \in T^{out'} + P}{P} \quad (\text{Pre}^-_1)
\]

where $y_0$ and $y_1$ are fresh variables.

The new notion of preconditions implies the old one.

Lemma 5.3 The following rule is derivable:

\[
\frac{\text{Pre}_1 U t}{\text{Pre}_0 U t}
\]

5.2 $W_0$-refinement

The totalisation (non-lifted) of a set of bindings can be defined as follows:

Definition 5.4

$$\hat{U} =_{df} \{ z_0 \ast z_1' \in T \mid \text{Pre}_1 U z_0 \Rightarrow z_0 \ast z_1' \in U \}$$

Proposition 5.5 The following rules are derivable:

\[
\frac{t_0 \ast t_1' \in U \quad \text{Pre}_1 U t_0 \vdash t_0 \ast t_1' \in U}{t_0 \ast t_1' \in \hat{U}} \quad (\diamond^+) \quad \frac{t_0 \ast t_1' \in \hat{U} \quad \text{Pre}_1 U t_0}{t_0 \ast t_1' \in U} \quad (\diamond^-_0)
\]

\[
\frac{t_0 \ast t_1' \in \hat{U} \quad \text{Pre}_1 U t_0}{t_0 \ast t_1' \in T} \quad (\diamond^-_1)
\]

Notice that the values in this completion range over the natural carrier set of the type $T$.

Lemma 5.6

\[
\frac{U \subseteq \hat{U}}{U \subseteq \hat{U}} \quad (i) \quad \frac{\hat{U} \subseteq U}{U \subseteq U} \quad (ii) \quad \frac{\neg \text{Pre}_0 U t_0 \quad t_0 \in T^{in} \quad t_1' \in T^{out'}}{t_0 \ast t_1' \in \hat{U}} \quad (iii)
\]
5. THE NON-LIFTED TOTALISATION

Proof. For (ii), consider the following derivation:

\[
\begin{array}{c}
t_0 * t'_1 \in U \\
t_0 * t'_1 \not\in U \\
t_0 * t'_1 \not\in U \\
t_0 * t'_1 \in T^* \\
t_0 * t'_1 \in U
\end{array}
\]

where \( \delta \) is:

\[
\begin{array}{c}
t_0 * t'_1 \in U \\
t_0 * t'_1 \not\in U \\
t_0 * t'_1 \not\in U \\
t_0 * y' \in U \\
t'_1 \in T_{out'}
\end{array}
\]

\( W_\diamond \)-refinement is then defined as follows:

Definition 5.7

\[
U_0 \sqsubseteq_{w_\diamond} U_1 =df U_0 \subseteq U_1
\]

Obvious introduction and elimination rules are derivable.

5.3 \( W_\diamond \)-refinement and \( S_1 \)-refinement are equivalent

As we have seen, the abortive value was critical in showing that \( W_\star \)-refinement and \( S \)-refinement are equivalent. Naturally we should assure ourselves that \( W_\diamond \)-refinement and a modified version of \( S \)-refinement are equivalent. Let \( S_1 \)-refinement be \( S \)-refinement, in which all instances of \( Pre_0 \) are replaced by \( Pre_1 \). We will content ourselves by showing that \( W_\diamond \)-refinement satisfies the \( S_1 \)-refinement elimination rule concerning preconditions. The remaining elimination rule, and indeed the other direction of the equivalence proof are not significantly different from the proofs we provided earlier in section 4.2.

Proposition 5.8 The following rule is derivable:

\[
U_0 \sqsubseteq_{w_\diamond} U_1 \quad Pre_1 U_1 t \quad Pre_1 U_0 t
\]
An analysis of total correctness refinement models for partial relation semantics I

Proof. Consider the following derivation:

\[
\begin{array}{c}
\frac{t \ast y'_0 \in U_1}{\text{false}} \quad \frac{t \ast y'_0 \notin U_1}{\text{true}} \quad \frac{t \ast y'_1 \in U_1}{\text{true}} \quad \frac{t \ast y'_1 \notin U_1}{\text{false}}
\end{array}
\]

\[\text{Pre}_1 U_1 t \quad \text{false} (2)\]

Where \(\delta\) is:

\[
\frac{\neg \text{Pre}_1 U_0 t}{t \in T^{in}} \quad \frac{\text{Pre}_1 U_1 t}{y' \in T^{out}} \quad \frac{t \ast y'_0 \in U_0}{t \ast y'_0 \in \tilde{U}_0} (1)
\]

5.4 \(W_\circ\)-refinement and \(W_*\)-refinement are equivalent (in \(Z_C\))

We begin by showing that \(W_\circ\)-refinement satisfies the \(W_*\)-refinement elimination rule, for bindings, which range over the natural carrier set.

**Proposition 5.9** Let \(t_0 \ast t'_1\) be a binding with the property that:

\(t_0 \ast t'_1 \in T\)

Then the following rule is derivable:

\[
\frac{U_0 \sqsupseteq_w U_1}{t_0 \ast t'_1 \in U_0}
\]

Proof. Consider the following derivation:

\[
\begin{array}{c}
\frac{t_0 \ast t'_1 \in U_0}{\text{true}} \quad \frac{t_0 \ast t'_1 \in U_0}{\text{true}} \\
\text{Pre}_1 U_0 t_0 \quad \text{Pre}_1 U_0 t_0 \quad \text{Pre}_1 U_0 t_0 \quad \text{Pre}_1 U_0 t_0 \quad \text{Pre}_1 U_0 t_0 \\
\end{array}
\]

\[t_0 \ast t'_1 \in U_0 (1)\]
**Theorem 5.10** When $W_\ast$-refinement is understood to range over the natural carriers, then, we have:

\[
\begin{align*}
U_0 \sqsupseteq_{w_0} U_1 \\
U_0 \sqsupseteq_{w_\ast} U_1
\end{align*}
\]

Likewise, we show that $W_\ast$-refinement satisfies the $W_\diamond$-refinement elimination rule.

**Proposition 5.11** The following rule is derivable:

\[
\begin{align*}
U_0 \sqsupseteq_{w_\ast} U_1 & \quad t_0 \times t'_1 \in \hat{U}_0 \\
& \quad t_0 \times t'_1 \in \hat{U}_1
\end{align*}
\]

**Proof.** Consider the following derivation:

\[
\begin{align*}
t_0 \times t'_1 & \in \hat{U}_0 \\
& \quad (5.6(ii)) \\
U_0 \sqsupseteq_{w_\ast} U_1 & \quad t_0 \times t'_1 \in \hat{U}_0 \\
& \quad (1) \\
t_0 \times t'_1 & \in \hat{U}_1
\end{align*}
\]

In this case, a term cannot satisfy the premise $t_0 \times t'_1 \in \hat{U}_0$ without belonging to the natural carrier, so in this direction no extra condition is necessary.

**Theorem 5.12**

\[
\begin{align*}
U_0 \sqsupseteq_{w_\ast} U_1 \\
U_0 \sqsupseteq_{w_0} U_1
\end{align*}
\]

For terms ranging over the natural carriers, then, the non-lifted and lifted models are equivalent. Since $W_\diamond$-refinement can be formalised in $Z_C$, and since the carrier sets of $Z_C$ are the natural carriers of $Z_C^0$, we may conclude that we can formalise a form of $W$-refinement in $Z_C$ without moving to the extended theory and introducing lifting. The penalty for this is the need for a novel notion of precondition.

### 6 The strict-lifted totalisation

A second question arising from our review was this: why is it necessary to permit recovery from the abortive value in completing the relation? In order to investigate this we consider a final relational completion in which the relation is lifted and totalised, but is strict with respect to abortive behaviour: $\bot$ maps only to $\bot$. 

Definition 6.1

\[ \hat{U} = \text{df} \{ z_0 \ast z'_1 \in T^* \mid \text{Pre } U \ z_0 \Rightarrow z_0 \ast z'_1 \in U \land z_0 = \bot \Rightarrow z'_1 = \bot \} \]

We obtain obvious introduction and elimination rules, which in this case we will not state explicitly. In addition we have, what are by now, fairly standard properties:

Lemma 6.2

\[
\begin{align*}
(i) & \quad U \subseteq \hat{U} \\
(ii) & \quad \hat{U} \subseteq U \\
(iii) & \quad \bot \in \hat{U}
\end{align*}
\]

\[
\begin{align*}
(iv) & \quad \neg \text{Pre } U \ t \ t \in T_{\text{in}}^* \\
(v) & \quad \neg \text{Pre } U \ t_0 \ t_0 \in T_{\text{in}}^* \ t'_1 \in T_{\text{out}}^* \\
\end{align*}
\]

Notice that in (v) \( t_0 \) ranges over the natural carrier set, rather than the extended carrier.

We now introduce \( W_{\ast} \)-refinement.

Definition 6.3

\[ U_0 \sqsubseteq_{w_{\ast}} U_1 = \text{df} \quad \hat{U}_0 \subseteq \hat{U}_1 \]

Again, we will not state the obvious rules.

7 \( W_{\ast} \)-refinement and \( W_{\ast} \)-refinement are equivalent

In the usual manner, we show that \( W_{\ast} \)-refinement satisfies the elimination rule of \( W_{\ast} \)-refinement.

Proposition 7.1 The following rule is derivable:

\[
\begin{align*}
& U_0 \sqsubseteq_{w_{\ast}} U_1 \quad t_0 \ast t'_1 \in \hat{U}_0 \\
& \quad t_0 \ast t'_1 \in \hat{U}_1
\end{align*}
\]

Proof. Consider the following derivation:

\[
\begin{align*}
& \delta_0 \\
& \vdots \\
& U_0 \sqsubseteq_{w_{\ast}} U_1 \quad t_0 \ast t'_1 \in \hat{U}_0 \\
& t_0 \ast t'_1 \in T^* \quad \text{(1)} \\
& t_0 \ast t'_1 \in \hat{U}_0 \quad \text{(1)} \\
& \text{Pre } \ U_1 \ t_0 \ t_0 \ast t'_1 \in \hat{U}_1 \\
& \quad t_0 \ast t'_1 \in \hat{U}_1 \\
& \quad t_0 \ast t'_1 \in \hat{U}_1 \\
& \quad t_0 \ast t'_1 \in \hat{U}_1
\end{align*}
\]
where $\delta_0$ is:

\[
\begin{array}{c}
\delta_1 \\
\vdots \\
U_0 \equiv_{w_0} U_1 \quad t_0 \ast \bot' \in U_0 \\
\frac{t_0 \ast \bot' \in U_1}{\text{Pre } U_1 t_0} \quad (1)
\end{array}
\]

and where $\delta_1$ is:

\[
\begin{array}{c}
t_0 \ast t_1' \in U_0 \\
\frac{t_0 \ast t_1' \in U_0}{\text{false} \quad (2)}
\end{array}
\]

\[
\begin{array}{c}
t_0 \ast t_1' \in U_0 \\
\frac{t_0 \ast t_1' \in U_0}{\text{false} \quad (2) \quad (6.2(i))}
\end{array}
\]

\[
\begin{array}{c}
t_0 \ast t_1' \in U_0 \\
\frac{t_0 \ast t_1' \in U_0}{\text{false} \quad (6.2(i))}
\end{array}
\]

Notice the use of the second version of rule ($\ast$) (proposition 3.8) in $\delta_0$.

**Theorem 7.2**

\[
\begin{array}{c}
U_0 \equiv_{w_0} U_1 \\
\frac{U_0 \equiv_{w_0} U_1}{U_0 \equiv_{w_1} U_1}
\end{array}
\]

Likewise, we prove that $W_\bullet$-refinement implies $W_\circ$-refinement by proving that $W_\bullet$-refinement satisfies the elimination rule of $W_\circ$-refinement.

**Proposition 7.3** The following rule is derivable:

\[
\begin{array}{c}
U_0 \equiv_{w_\circ} U_1 \\
t_0 \ast t_1' \in U_0 \\
t_0 \ast t_1' \in U_0
\end{array}
\]

**Proof.** Consider the following derivation:

\[
\begin{array}{c}
\delta \\
\vdots \\
t_0 \ast t_1' \in U_0 \\
t_0 \ast t_1' \in U_1 \\
\frac{\text{Pre } U_1 t_0}{t_0 \ast t_1' \in U_1} \quad (1) \\
t_0 \ast t_1' \in U_0 \\
t_0 = \bot' \quad (1)
\end{array}
\]

\[
\begin{array}{c}
t_0 \ast t_1' \in U_1 \\
t_0 \ast t_1' \in U_1 \\
\frac{t_0 = \bot'}{t_1' = \bot'} \quad (1)
\end{array}
\]
where $\delta$ is:

$$
\begin{align*}
\d_0 \ast \d'_1 & \in \bigcup_0 \\
t_0 \ast t'_1 & \in \bigcup_0 \\
\text{(1)} & \\
\text{(4.12)} & \\
\end{align*}
$$

$$
\begin{align*}
U_0 \sqsubseteq w_a U_1 \\
t_0 \ast t'_1 & \in \bigcup_0 \\
\text{Pre } U_1 t_0 & \\
\text{(4.12(i))} & \\
\end{align*}
$$

Theorem 7.4

$$
U_0 \sqsubseteq w_a U_1 \\
U_0 \sqsubseteq w_o U_1 & \\
$$

This concludes the analysis.

8 Conclusions and future work

Refinement in the context of partial relational models of specifications is an important sub-area of refinement in general, not least because the standard interpretation of Z specifications is a partial relational interpretation. There are, as we have demonstrated here, a number of, at least at first sight, distinct notions of refinement in this context. For Z, the standard account is that due to Woodcock and is based on an interesting notion of relational completion which lifts and totalises partial relations. We have shown that this notion is by no means arbitrary, but that there are a number of alternative notions which are equivalent. We have been able to begin to explain some of the mathematical reasons why it is defined in precisely the way it is, particularly why it is necessary to introduce the abortive value $\bot$. Contrary to the argument in [18] the critical reason concerns the behaviour of preconditions, rather than the behaviour of refinement under composition. On the other hand a novel notion of what it means for a value to be in the precondition of a specification leads to a Woodcock-like theory in which the relations need not be lifted. There is, in addition, the possibility of insisting that the abortive value is handled in a strict fashion, also leading to an equivalent theory.

It would have been possible to develop the technical material in this paper purely in terms of abstract partial relations, rather than those induced by Z specifications. Formalisation in $Z_C$, however, makes dealing with Z no more complicated than dealing with abstract partial relations, and it has a further advantage which will become clear in the companion papers to follow: we may smoothly extend our analysis to explore refinement in the context of the schema calculus. The specification language Z has the advantage of permitting large scale specifications to be constructed in a modular and systematic fashion using connectives reminiscent of quantificational logic. The development of Z logic in, for example, [11] shows how this algebra of schema expressions can be realised as a calculus by showing how to interpret the connectives and quantifiers as operations on partial relations which have, in general,
non-equal domains. In order to understand how Z, in particular, handles refinement in the generalised context of the schema calculus, a separate investigation is planned as a sequel to this paper. In particular, monotonicity issues have to be fully explored. It is well-known that the specification forming operations have poor monotonicity properties with respect to refinement in the relational completion semantics, but a thorough investigation, with a careful consideration of all the options, has not to date been undertaken. What we can say already, given the results in this paper, is that there is no likelihood for improved monotonicity properties for Z if its underlying semantics is to remain based on the partial relation model: every refinement theory we have constructed on this basis is equivalent. It is perhaps worth pointing out that although we only gave an explicit semantics for atomic schemas (see definition 2.1) the meta-variables $U$ range over arbitrary sets of bindings of type $T^{\text{in}} \times T^{\text{out}}$. In other words we can plug in the entire semantics of the schema calculus (in the way described in, for example, [11]) beneath the account of operation refinement without affecting the analysis in any way. The consequence of this is, then, that all six theories we have described must suffer from the same defects regarding monotonicity properties. Improving upon this position will be one of the themes underlying our further investigation and we will see how the one way forward is to abandon the standard semantics and its equational logic, in favour of an inequational logic of refinement.

There is a well-known alternative to the notion of refinement which we have investigated in this paper. This is the behavioural approach outlined, for example, in [6]. In this model, preconditions are understood to be firing conditions (or triggers) and may, therefore, not be weakened. A parallel investigation to the one we have described in this paper (with extensions to cover the schema calculus) is possible, and will also be reported in a future paper in this series.

We made comment, in section 3.2, regarding the relationship between S-refinement and a theorem concerning two-predicate designs appearing in [12]. The semantics of these designs involve a termination value (it is written $\text{ok}'$). This suggests an interesting generalisation of the work reported here in which two-predicate designs are represented by total relations involving both signals for termination and non-termination. This model will also be reported in future work.

This paper has concentrated on operation refinement, that is the degenerate case of data refinement in which all simulations are identity functions. This was sufficient for our purposes here, but the deeper notion is data refinement, and there are further aspects to the questions we have raised regarding the Woodcock-completion which do not emerge until data refinement is considered. This is why we emphasised the word "begin" earlier in this concluding section. Operation refinement is sufficiently interesting mathematically to motivate the range of possibilities we have explored here, but there are divergences and surprises to uncover when non-trivial simulations are permitted. This topic will also be covered in a companion paper, making links and comparison between our proof-theoretic approach and the largely semantic approach generally taken in [5].

Our analysis of weakest precondition semantics is here restricted to the atomic schema

---

15 However, one can reasonably argue that, by insulating the schema logic (based on the partial relation interpretation) from the refinement logic (based on any one of the six theories we have described) Z effectively reduces its refinement logic to the atomic schema case. This is because the schema logic permits the derivation of an equational logic which makes all schema expressions equal to an atomic schema.
An analysis of total correctness refinement models for partial relation semantics I

case, and to operation refinement. Some interesting work, developing one or other of
these dimensions, has been undertaken, for example by Woodcock and Cavalcanti [2]
[4], by Groves [4], and in [5]. Formalisation and investigation within our theoretical
setting, of an extension to what we have shown in this paper and with our proof-
theoretic methodology, is both simple and revealing. It will form an additional phase
of our exploration of refinement and we plan to cover the topic in another paper in
this series.

9 Acknowledgements

We would like to thank the EPSRC (grant reference: GR/L57913), the Royal Society
of Great Britain and the New Zealand Foundation for Research, Science and Tech-
ology (grant reference: UOWX0011) for financially supporting this research. Moshe
Deutsch is supported by the British Council through an ORS award.

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 Rob Arthan, Lindsay Groves, Greg Reeve, Ray Turner and Jim Woodcock.

References


tions. Formal Approaches to Computing and Information Technology – FACTT. Springer, May


University, 2000.


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.


1996.
A. THE SPECIFICATION LOGIC \( \mathcal{Z}_C \) AND THE THEORY \( \mathbb{Z}^C \)

A The specification logic \( \mathcal{Z}_C \) and the theory \( \mathbb{Z}^C \)

A.1 \( \mathcal{Z}_C \)

A.1.1 The language of \( \mathcal{Z}_C \)

\( \mathcal{Z}_C \) is an extension of higher order logic with \textit{schema types}. We begin with the types:

\[
T ::= \mathbb{B} | \mathbb{N} | P \times T | T \times T | \ldots \]

We will often permit the meta-variable \( D \) to range over sequences of labels such as \( \ldots z^T, \ldots \) (the order is not important), also writing \( \alpha[D] \), when \( D \) is \( \ldots z^T, \ldots \), for the alphabet set (in the metalanguage) of labels \( \{ \ldots z, \ldots \} \). No label may occur more than once in such a type.

Types of the form \([D]\) are called \textit{schema types}. The symbols \( \subseteq, \lambda \), \( \gamma \) and \( \neg \) denote the schema \textit{subtype} relation, and the operations of \textit{schema type intersection}, (compatible) \textit{schema type union} and \textit{schema type subtraction}.

All categories of the language of \( \mathcal{Z}_C \) must be well-formed with respect to these types.

Next we have the category of \textit{terms}. We assume the existence of a denumerable set of variables for each type \( T \). We use \( t \) as a meta-variable over terms of arbitrary type.

The syntax of terms is then:

\[
t^T ::= x^T \mid t^T \cdot t^T \ldots \mid t^T \times t^T \mid t^T_0 \times t^T_1 \mid t^T_{0 \times 2}
\]

\[
\begin{align*}
[t] & ::= t^2 \mid [D] \mid t^T \cdot \ldots \mid \{ \ldots z \} \mid t^T \\
[t^T_0 \times t^T_1] & ::= (t^T_0, t^T_1) \\
[t^2] & ::= \{ z \mid P \} \\
t^n & ::= \text{True} | \text{False}
\end{align*}
\]

where \([D] \preceq T_2\).

For clarity of presentation we permit the meta-variable \( C \) (etc.) to range over terms of power type, and \( S \) (etc.) to range over sets of schema type. The latter are \textit{schemas}. We let \( U \) (etc.) range over \textit{operation schemas}. These are schemas involving both \textit{before} and \textit{after} observations. We can always write the type of such operation schemas as \( P(T^{in} \cap T^{out}) \) 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 \textit{binding concatenation}, written \( t_0 \bullet t_1 \) 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):

\[
C_0 \bullet C_1 = \{ z_0 \bullet z_1 \mid z_0 \in C_0 \land z_1 \in C_1 \}
\]

The same restriction obviously applies here: the types of the sets involved must be disjoint.

We employ the notation \( b.P \) and \( b.t \) (generalising binding selection) which is adapted from \( \Pi \).

Suppose that \( \{ z_0, \ldots z_n \} \) is the alphabet set of \( t \), then \( t.P \) is \( P[z_0/t.z_0] \cdots [z_n/t.z_n] \).

The formul\ae\ of \( \mathcal{Z}_C \) delineate a typed bounded predicate logic.

\[
P ::= \text{false} \mid t^T = t^T \mid t^T \in C^P T \mid \neg P \mid P \lor P \mid \exists z^T \in C^P T \bullet P
\]

The logic of \( \mathcal{Z}_C \) is classical, so the remaining logical operations are available by definition. We also, as usual, abbreviate \( \neg(t^T \in C^P T) \) to \( t^T \notin C^P T \).

A crucial observation is \textit{unicity of types}: every term and set of \( \mathcal{Z}_C \) has a unique type. We can make great use of this observation. It enables us to remove type decoration in most circumstances.

A.1.2 The logic of \( \mathcal{Z}_C \)

The judgements of the logic have the form \( \Gamma \vdash P \) where \( \Gamma \) is a set of formul\ae. 
The logic is presented as a natural deduction system in sequent form. We shall omit all data (entailment symbol, contexts, type etc.) which remain unchanged by a rule. In the rule \(\frac{P_0}{P_0 \lor P_1} (\lor^+)\), the variable \(y\) may not occur in \(C, P_0, P_1\) nor any other assumption.

\[
\begin{align*}
\frac{P_0}{P_0 \lor P_1} (\lor^+) & \quad \frac{P_1}{P_1 \lor P_1} (\lor^+) & \quad \frac{y \in C \cdot P_0}{P_0 \vdash P_1} (\exists^+) & \quad \frac{y \in C \cdot P_0}{P_0 \vdash P_1} (\forall^+) & \quad \frac{P_0 \vdash P_1 \cdot P_0}{P_1 \vdash P_2} (\exists^+) & \quad \frac{P_1 \vdash P_2}{P_2} (\forall^+) & \quad \frac{P \vdash \top}{P \vdash \bot} (\neg^+) & \quad \frac{P \vdash \bot}{\neg P} (\neg^+) & \quad \frac{\neg \vdash P}{\neg \vdash \neg P} (\neg^+) & \quad \frac{\bot}{P} (\bot^+)
\end{align*}
\]

\[
\begin{align*}
P[z/t] & \quad t \in C \quad \exists z \in C \cdot P \quad \exists z \in C \cdot P_0 & \quad P [z/y] \vdash P_1 \quad (\exists^+) & \quad \exists z \in C \cdot P \quad \exists z \in C \cdot P_0 \vdash P_1 \quad (\forall^+)
\end{align*}
\]

\[
\begin{align*}
\frac{P[z/t]}{z \in P} (\exists^+) & \quad \frac{t = t'}{P[z/t']} (\exists^+) & \quad \frac{t \in \{z \mid P\}}{P[z/t]} (\exists^+)
\end{align*}
\]

where

\[
t_0 \equiv t_1 = \text{def} \forall z \in t_0 \cdot z \in t_1 \land \forall z \in t_0 \cdot z \in t_0
\]

The transitivity of equality and numerous equality congruence rules for the various term forming operations are all derivable in view of rule \((\text{sub})\). In particular, we can prove that set-equality in \(\mathcal{Z}_C\) is extensional.

The following weakening rule is admissible and is incorporated within the system.

\[
\frac{\Gamma \vdash P_1}{\Gamma, P_0 \vdash P_1} (\text{wk})
\]

The carrier sets (with the expected derived rules) are easily defined this system:

\[
T = \{z^T \mid T \text{true}\}
\]

The ambiguity presents no problem, since only types appear as superscripts and only sets appear after the membership relation.

A.1.3 Restricted equality

In many contexts we need to compare bindings over a common restricted type.

**Definition A.1** Let \(T \leq T_0\) and \(T \leq T_1\).

\[
t_0^{T_0} = T \quad t_1^{T_1} = T_1 \quad t_0 \mid T = t_1 \mid T
\]

A restricted form of membership is very useful for establishing the schema calculus. In this case the types are uniquely determined by the syntax and do not appear in the notation.

**Definition A.2** Let \(T_0 \leq T_1\).

\[
t^T_1 \in \mathcal{C}^T_0 = \text{def} \ t \mid T_0 \in C
\]
A. THE SPECIFICATION LOGIC $Z^C$ AND THE THEORY $Z^C_{⊥}$

A.1.4 Functions

We will need total functions over types. These are easily introduced.

**Definition A.3**

\[ T_0 \rightarrow T_1 = \text{df} \{ g \in P(T_0 \times T_1) \mid \text{unicity}(g) \land \text{total}(g) \} \]

Note that functions are modelled as subsets of $T_0 \times T_1$ rather than $T_0 \times T_1$, and that for notational convenience, we let $g$ (etc.) range over terms of type $P(T_0 \times T_1)$. In fact we only do this when such a term is a function.

When $g$ is known to be an element of $T_0 \rightarrow T_1$ and $z_0 \in T_0$, we will write $gz_0$ (as usual) for the unique element $z_1 \in T_1$ such that $z_0 \star z_1 \in g$.

A.2 $Z^C_{⊥}$

In $Z^C_{⊥}$ we introduce new constants (to play the role of “undefined” values) at every type. Thus we postulate new constants $\bot_T$ in every type $T$. There are, additionally, a number of axioms which ensure that all the new $\bot_T$ values interact properly.

\[
\langle | z_0/z_0 | \cdots | z_n/z_n | \rangle = \bot_T \\
\langle \bot_T, \bot_T | T_0 \times T_1 \rangle = \bot_T \\
\{ z^T \mid z = \bot_T \} = \bot_T^T
\]

For example:

\[
\bot_{[z_0/T_0, z_n/T_n]} = \bot_{T_i} (0 \leq i \leq n)
\]

Note that these are the only axioms concerning undefined terms, hence, the term forming constructions are non-strict with respect to the $\bot_T$ values.

In order to facilitate the proper interpretation of schemas we will need to introduce the natural carrier sets which are those sets of elements of types which explicitly exclude the $\bot_T$ values:

**Definition A.4** The natural carriers for each type are defined by closing:

\[
\mathbb{N} = \text{df} \{ z^N \mid z \neq \bot_N \}
\]

and:

\[
\mathbb{B} = \text{df} \{ z^B \mid z \neq \bot_B \}
\]

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

In $Z^C_{⊥}$ the schema calculus is introduced by means of definitions such as:

\[
[S \mid P] = \text{df} \{ z^T \mid z \in S \land z.P \}
\]

for atomic schemas, and:

\[
S_0^P \lor S_1^P T_1 = \text{df} \{ z^T \lor z \mid z \in S_0 \lor z \in S_1 \}
\]

for schema disjunction.

In interpreting the schema calculus in $Z^C_{⊥}$ we systematically modify the definitions so that the values are drawn over the natural carriers rather than the type. For example:

\[
[S \mid P] = \text{df} \{ z \in T \mid z \in S \land z.P \}
\]

defines atomic schemas, and:

\[
S_0^P \lor S_1^P T_1 = \text{df} \{ z \in (\sim T_0 \lor \sim T_1) \mid z \in S_0 \lor z \in S_1 \}
\]

defines schema disjunction.

As a result the schema calculus is hereditarily $\bot$-free, since obviously:
A.3 Conservativity

\[ \mathcal{Z}_C^+ \] is conservative over \( \mathcal{Z}_C \). The presence of projection terms \((t,1, t, z)\) in \( \mathcal{Z}_C^+ \) makes the definition of a simple translation of \( \mathcal{Z}_C^+ \) into \( \mathcal{Z}_C \) difficult. The appropriate approach is to show that there is a translation of \( \mathcal{Z}_C^{\perp-} \) into \( \mathcal{Z}_C \), where \( \mathcal{Z}_C^{\perp-} \) is roughly \( \mathcal{Z}_C^+ \) with the projections removed. Having shown that \( \mathcal{Z}_C^{\perp-} \) is a conservative extension of \( \mathcal{Z}_C \) we can then show that \( \mathcal{Z}_C^+ \) is a conservative (in fact a definitional) extension of \( \mathcal{Z}_C^{\perp-} \). This then completes the task.

A.3.1 The theory \( \mathcal{Z}_C^{\perp-} \)

The language of \( \mathcal{Z}_C^{\perp-} \) is that of \( \mathcal{Z}_C^+ \) with the projection terms removed. The \( \mathcal{Z}_C^{\perp-} \) logic is the logic of \( \mathcal{Z}_C^+ \) with the equality rules for the projection terms removed and the following equality rules (which are derivable in \( \mathcal{Z}_C^+ \) and \( \mathcal{Z}_C \)) added:

\[
\begin{align*}
\top_0 &= t_2 & t_1 &= t_3 \\
\vdots & & t_i &= v_i \\
\langle \cdots z_i \Rightarrow t_i \cdots \rangle &= \langle \cdots z_i \Rightarrow v_i \cdots \rangle 
\end{align*}
\]

A.3.2 \( \mathcal{Z}_C^{\perp-} \) is conservative over \( \mathcal{Z}_C \)

**Definition A.7** We define a mapping from the language of \( \mathcal{Z}_C^{\perp-} \) into that of \( \mathcal{Z}_C \). We omit the type superscripts which can easily be inferred.

\[
\begin{align*}
(\exists z \bullet P)^* &= \exists z \bullet P^* \\
(P_0 \lor P_1)^* &= P_0^* \lor P_1^* \\
(\neg P)^* &= \neg P^* \\
nfalse^* &= P[z/t]^* \\
(t \in \{ z \mid P \})^* &= P[z/t]^* \\
((t_0, t_1) = (t_2, t_3))^* &= (t_0 = t_2)^* \land (t_1 = t_3)^* \\
(\langle \cdots z_i \Rightarrow t_i \cdots \rangle = \langle \cdots z_i \Rightarrow v_i \cdots \rangle)^* &= \cdots \land (t_i = v_i)^* \land \cdots \\
(\{ z \mid P_0 \} = \{ z \mid P_1 \})^* &= (\forall z \bullet P_0 \iff P_1)^* \\
(c = c)^* &= \text{true} \\
&(c = t)^* &= \text{false} \quad (c \text{ a constant}) \\
&(c = \bot)^* &= \text{true} \\
&(c = \top)^* &= \text{false} \quad (t \neq \bot) \\
&(c = \bot)^* &= \text{false} \quad (c \text{ a constant}; c \neq \bot) \\
&(c = \top)^* &= \text{false} \quad (c \text{ a constant})
\end{align*}
\]

Note that if \( P \) is in the language of \( \mathcal{Z}_C \) then \( P^* = P \).

**Proposition A.8** If \( \Gamma \vdash_{\mathcal{Z}_C^{\perp-}} P \) then \( \Gamma^* \vdash_{\mathcal{Z}_C} P^* \)

**Proof.** By induction on the structure of derivations. Since the translation is a direct structural recursion for the most part, the majority of the cases are immediate. We illustrate with one of the new axioms. Consider:

\[
(\bot, \top_0, \bot_1) = \bot_0 \land \top_1
\]

Note simply that \((\bot_0, \bot_1) = \bot_0 \land \bot_1)^* = (\bot, \bot) \land (\bot_1)\) = \text{true} \land \text{true}.

**Corollary A.9** \( \mathcal{Z}_C^{\perp-} \) is conservative over \( \mathcal{Z}_C \).
A. \textbf{THE SPECIFICATION LOGIC $Z_C$ AND THE THEORY $Z_C^\perp$}

A.3.3 $Z_C^\perp$ is conservative over $Z_C^{\perp-}$

In fact $Z_C^\perp$ is a definitional extension of $Z_C^{\perp-}$.

\textbf{Definition A.10} For all types, $T_0$ and $T_1$ we define:

$$1^\mathcal{P}((T_0 \times T_1) \times T_0) = \{ ((z_0, z_1), z_2) \mid T_0 \times T_1 \}$$

Note, that $1$ satisfies totality and unicity so $1 \in (T_0 \times T_1) \rightarrow T_0$

Then we can define the first projection operator of $Z_C^\perp$ (and of course, of $Z_C$).

\textbf{Definition A.11} \hspace{1cm} t^T_0 \times T_1 \cdot 1 = 1^\mathcal{P}((T_0 \times T_1) \times T_0)(t)

Similar definitions can be established for the second projection and for binding projections. With these in place one easily derives the missing $Z_C^\perp$ equality rules for projections.

\textbf{Proposition A.12} $Z_C^\perp$ is conservative over $Z_C^{\perp-}$. \hfill \blacksquare

Received 18th January 2002, revised 10 February 2003.