Under consideration for publication in Formal Aspects of Computing

Retrenchment and Promotion in Z

Richard Banach[1], Michael Poppleton[2], Czeslaw Jeske[1] and Susan Stepney[3]

[1]Department of Computer Science, University of Manchester,
Manchester M13 9PL, UK,
{banach,cj}@cs.man.ac.uk,
[2]School of Electronics and Computer Science, University of Southampton,
Highfield, Southampton SO17 1BJ, UK,
mrp@ecs.soton.ac.uk,
[3]Department of Computer Science, University of York,
Heslington, York YO10 5DD ,
susan.stepney@cs.york.ac.uk

Abstract. Promotion, a familiar data structuring mechanism in Z, is reviewed. Retrenchment, a generalization of classical data refinement, is reviewed and presented in Z. A theory of the promotion of retrenchments in Z is developed, which supports a variety of requirements scenarios and demonstrates that promotion is also a useful tool in the requirements engineering toolkit of retrenchment. This amplifies its utility in the pure refinement arena, when refinement and retrenchment are made to interwork. A simple case study of promoted retrenchment is presented to illustrate the theory.

Keywords: Formal Specification, Mondex, Promotion, Refinement, Retrenchment.

1. Introduction

In Z, promotion [WD96, DB01, Lup91] is a specification mechanism that enables a succinct description of structured states and their operations. A specification of an individual system component is given, and separately, the specification of a module containing a collection of instances of that component is also given. Any module-level operation concerning only one contained component, is then defined in terms of the component-level operation, in a generic manner. Thus, under the constraint of no dependency on other components, the component-level specification simply factors through the module level.

Many application domains require the hierarchic structuring of collections of components within containers, for example a bank account (in the context of the bank, which contains many such accounts), or an aircraft (in the context of the air traffic control model containing many aircraft). Moreover this structuring concept is part of the object-oriented philosophy. Promotion has been used in real industrial applications, e.g. the Mondex purse [SCW00], and a version has been defined in the B Language [ABDM00].

Correspondence and offprint requests to: Richard Banach, School of Computer Science, University of Manchester, Oxford Road, Manchester, M13 9PL, U.K. email: banach@cs.man.ac.uk

A refinement theory for promotion has been proposed [WD96,DB01,Lup91]. Refinement has long been established as a major formal method for correctness-preserving software development [BvW89, BvW98, dRE98, DB01]. The robustness and reliability of the refinement technique come from the relatively strong, precise conditions that have to hold before a refinement can be asserted between two models of a system. A *retrieve*, or *abstraction* relation is defined between the data types of the first, more abstract model, and the second, more concrete and algorithmic model. The retrieve relation defines how the concrete data type represents the abstract one. Essentially, refinement requires that any concrete behaviour is *simulated by* some corresponding abstract behaviours, by ensuring that the retrieve relation is preserved by the refinement. The Mondex project [SCW00] eloquently demonstrated the use of promotion in combination with refinement.

In [DB01] the authors define conditions that allow the simple factoring of component-level refinement through the promoted level. However, many situations in which developers of high consequence and complex systems might wish to use refinement, feature a series of models whose desired relationships (desired that is, from an engineering perspective) do not satisfy the exacting simulation conditions for refinement. Examples of such model combinations are easily found in modern distributed systems, e.g. [Hen02]. In the face of this, developers either abandon any attempt to use refinement, contenting themselves with less formal but more flexible (and unfortunately less rigorous) techniques, or they deliberately choose to reduce verisimilitude in modelling, perhaps by working with models that represent reality less faithfully than might otherwise be desired.

This gives rise to the need for a less demanding formal development notion to describe such situations, and to address this need, the retrenchment approach [BP98, BP99b, BPJS07] was introduced. Essentially, retrenchment provides a formal way to describe the relationship between two models where simulation breaks down. Whilst lacking the guarantees that refinement offers (since one cannot have both the rigour of refinement and the flexibilty that developers crave), retrenchment is nonetheless a formal relationship between models, and thus allows formal statements to be made about scenarios in which such a level of rigour would otherwise be unavailable. Retrenchment achieves this goal by weakening the downward simulation [dRE98] proof obligation (PO) of refinement in a way that enables it to express relevant facts about the more general situations envisaged. This weakening involves introducing extra relations concerning the two models into the PO (whereas in refinement, only the retrieve relation parameterizes the PO via the models in this way). These extra relations (which we will call the *retrenchment data*) are the *within*, *output* and *concedes* relations, whose job it is to accommodate the lack of precise adherence to the refinement PO criteria. A few examples of situations that have been examined in a retrenchment setting include distributed systems (alluded to in [PB03]), the construction of feature-oriented specifications with inconsistency [BP03], and a control engineering example [PB02] which describes the inherently approximate and varying relationship between continuous- and discrete-time models.

The utility of retrenchment as a formal tool, particularly when it interworks with refinement [Jes05, BJP08], makes it natural to ask how retrenchment and promotion interact. Moreover, there is a requirements engineering (RE) case for this combination. We indicated above two RE scenarios requiring retrenchment: the combination of inconsistent requirements in feature engineering, and the approximate nature of modelling continuous in discrete time. In any application where a notion of approximate or nonsimulating representation is required, componentwise compositionality is still just as useful for the development method as when rigorous simulability holds. We might describe the ideal behaviour of a component in an abstract model, and a more realistic representation of that component in a more implementable concrete model. We might then describe the relation between the models as a retrenchment. We would then need the retrenchment (just as it is with refinement) to be piecewise composable, when the specified components are collected into larger aggregates. Since promotion is a major constructor of composed specifications, it should be possible to promote a retrenchment in an analogous way to that in which a refinement is promoted. However, when the promotion of retrenchment is considered, a number of technical issues arise in the detailed interaction of the retrenchment and promotion techniques, that are novel when compared to the corresponding situation with refinement. It is our intention to clarify these issues in this paper.

Consider the refinement of a collection of similar objects. If the refinement of a single object is well understood, then it is most natural that the refinement of the collection simply replicates the refinement of an individual, and this makes the promotion of refinement of a collection go relatively straightforwardly. However, if one considers the *retrenchment* of a collection of similar objects, then there are more possibilities. Since, under retrenchment, not all the objects need to 'behave' (in the sense of maintaining a refinement relationship with the corresponding abstraction), all, some, or none may be doing so at any moment, and the promotion of a retrenchment may be designed to say various things about these possibilities. Strong promotion insists that at the beginning of a step, all elements of the collection are still refining; weak promotion merely insists some are. One can be more specific about which elements of the collection are still refining: this is precise promotion. In addition to these variants, one can vary how many elements of the collection the several relations in the retrenchment data actually speak about. All of this will be explored below.

While some of the retrenchment promotion techniques described in this paper have already been used by the authors in existing case studies on the use of retrenchment within the context of the Mondex Electronic Purse [BPJS05, BJPS05, BJPS06, BJPS07], the treatments there have, for lack of space, necessarily been rather brief and partial. The present paper serves the need for a more unified and comprehensive account.

The rest of the paper proceeds as follows. In Section 2 we review promotion, and the refinement of promoted specifications, demonstrated in a running example. Section 3 presents retrenchment in Z. Section 4 demonstrates a promotion of the retrenchment of the example. Motivated by corresponding requirements engineering concerns, Sections 4 and 5 then present some of the different notions of the promotion of retrenchments just mentioned. Strong and weak promotion, closely analogous to the promotion of refinements, occur in Section 4, while precise promotion, a more radical departure from refinement promotion, is described in Section 5. Section 6 concludes.

2. Promotion in Z

In [DB01], the authors give a succinct encapsulation of the idea of promotion:

> The purpose of promotion is to find an elegant way of composing specifications in order to build multiple indexed instances of a single component. To do so the component is described as a local state together with operations acting on that state. A global state is defined which consists of multiple instances of this local state together with global operations defined in terms of the local operations and a special promotion schema.

We now present the Z schema construction of promotion —first generically and then via a small running example— and then we review refinement and the promotion of refinement.

If the local state is given by schema A, then the global (or world) state schema *AWorld* is defined as an indexing function from an index set *Ind* to A. To enable concise world-level description of an operation on a single copy of the local state, the promotion framing schema ΦAOp is defined. ΦAOp contains both a global state schema *AWorld* and a local state schema A, and also an input parameter $Ai?$ of type *Ind*, identifying the required local state component for access or update. The expression θA identifies the target A element $Af(Ai?)$ through the index function Af with the local state binding A. The final predicate ensures that all components other than $Af(Ai?)$ remain unchanged.

AWorld

$$Af : Ind \pfun A$$

$$AWorldOp \mathrel{\widehat=} \exists \Delta A \bullet \Phi AOp \land AOp$$

ΦAOp

$$\Delta AWorld$$
$$\Delta A$$
$$Ai? : Ind$$

$$Ai? \in \operatorname{dom} Af$$
$$\theta A = Af(Ai?)$$
$$Af' = Af \oplus \{Ai? \mapsto \theta A'\}$$

ΦAOp is generic insofar as it allows the mechanical definition of the world-level operation *AWorldOp* corresponding to a local operation *AOp*, without itself constraining the behaviour of that local operation in any way (thus permitting us to use *AOp* itself, as a component of *AWorldOp*, to enforce the required behaviour). The same Φ schema shape, but acting on local state $\Delta LState$, gives the framing schema $\Phi LStateOp$, a convention we use for the rest of the paper.

We now give a small running example. The local state is schema *ABag*, a bag of naturals, with two local state operations *AInc* to add a number, and *ASum* to give the sum of the bag elements (returned by Σ):

ABag

$$Avals : \operatorname{bag} \mathbb{N}$$

AInc

$$\Delta ABag$$
$$Ain? : \mathbb{N}$$

$$Avals' = Avals \uplus [\![Ain?]\!]$$

ASum

$$\Xi ABag$$
$$Aans! : \mathbb{N}$$

$$Aans! = \Sigma Avals$$

Following the template *AWorldOp* for a world-level operation, and utilising the relevant framing schema $\Phi ABagOp$ we can define:

$$AWorldInc \mathrel{\widehat=} \exists \Delta ABag \bullet \Phi ABagOp \land AInc \qquad AWorldSum \mathrel{\widehat=} \exists \Delta ABag \bullet \Phi ABagOp \land ASum$$

Note for the reader: In this paper we present various generic formulations in the schema calculus for the promotion of refinements and retrenchments. The abstract global state will be described as *AWorld*, with *AWorld* refined to *CWorld*,

alternatively with *AWorld* retrenched to *DWorld*. The framing schemas are $\Phi ALStateOp$, $\Phi CLStateOp$, $\Phi DLStateOp$ respectively. These formulations are interpreted in the running example above using global states $AWorld$, $CWorld$, $DWorld$, local states $ABag$, $CBag$, $DBag$ etc. Also, to clearly indicate which model is in focus, other schema component variables will be prefixed A, C, D where they pertain to the three models (or worlds) respectively.

2.1. Promotion and Refinement in Z

We give the standard 'contract' semantics of data refinement in Z as per the presentation in [CSW02], which gives abstract/concrete input and output mappings.

Take an abstract model given by the ADT $(A, AInit, \{(AOp, AI_{Op}, AO_{Op}) \mid Op \in \mathsf{Ops}\})$ with state A, initial state $AInit$, and for each operation AOp, input and output spaces AI_{Op}, AO_{Op}. Take a concrete model $(C, CInit, \{(COp, CI_{Op}, CO_{Op}) \mid Op \in \mathsf{Ops}\})$. Further assume retrieve relation $R : [A;\ C]$ between the two state spaces, and for each operation pair Op, input and output mapping relations $RI_{Op} : [AI_{Op};\ CI_{Op}]$ and $RO_{Op} : [AO_{Op};\ CO_{Op}]$. Data refinement is given by three proof obligations (POs), called *initialization*, *applicability* and *correctness* respectively:

$$\forall C' \bullet CInit \Rightarrow \exists A' \bullet AInit \land R' \tag{1}$$

$$\forall A;\ AI_{Op};\ C;\ CI_{Op} \bullet R \land RI_{Op} \land \mathrm{pre}\, AOp \Rightarrow \mathrm{pre}\, COp \tag{2}$$

$$\forall A;\ AI_{Op};\ C;\ CI_{Op};\ C';\ CO_{Op} \bullet R \land RI_{Op} \land \mathrm{pre}\, AOp \land COp \Rightarrow \exists A';\ AO_{Op} \bullet AOp \land R' \land RO_{Op} \tag{3}$$

We will not need to consider the applicability PO further in this paper for two reasons: (i) in our running example, it will always be trivial by the totality of the example's operations, (ii) in retrenchment it is best to subsume all matters connected with operation applicability between the two models using retrenchment's within relation. Not only is this is a generally adequate approach, but it also works well with regard to retrenchment/refinement interworking; see [BPJS07, Jes05]. More details appear in Section 3.

The following concrete 'bag' model represents a bag by recording only the bag sum, which is adequate for the needed operations *Inc* and *Sum*. The obvious refinement between the abstract and concrete bags is via the retrieve relation R_{AC}:

CBag

$Ctot : \mathbb{N}$

CInc

$\Delta CBag$
$Cin? : \mathbb{N}$

$Ctot' = Ctot + Cin?$

CSum

$\Xi CBag$
$Cans! : \mathbb{N}$

$Cans! = Ctot$

R_{AC}

$ABag;\ CBag$

$Ctot = \Sigma Avals$

We assume $Ain? = Cin?$, and $Aans! = Cans!$ by definition. Given suitable initializations, it is easy to see that the POs (1)-(3) are validated under retrieve relation R_{AC}. The concrete promotion is routine:

CWorld

$Cf : Ind \pfun CBag$

$CWorldInc \mathrel{\widehat=} \exists \Delta CBag \bullet \Phi CBagOp \land CInc$

$CWorldSum \mathrel{\widehat=} \exists \Delta CBag \bullet \Phi CBagOp \land CSum$

$\Phi CBagOp$

$\Delta CWorld$
$\Delta CBag$
$Ci? : Ind$

$Ci? \in \mathrm{dom}\, Cf$
$\theta CBag = Cf(Ci?)$
$Cf' = Cf \oplus \{Ci? \mapsto \theta CBag'\}$

Looking forward to a refinement between the promotions, in [DB01] the problem is tackled by identifying conditions which are sufficient for a relation between the abstract and concrete local state spaces to act as a retrieve relation for a refinement that distributes through the promotions to the global states. We employ a simple instance of this: the global retrieve relation simply asserts the local one for all components, a state of affairs which covers the majority of practically arising cases. Furthermore, we restrict ourselves in this paper to this simple case, because even in this

simple context, a number of new issues of interest arise in the context of retrenchment which can be dealt with in a clear manner.

The component level input and output information is reused directly at world level; thus the input and output mapping relations are simply

$$RI^P_{Op} \mathrel{\widehat=} RI_{Op} \wedge \left[Ai?, Ci? : Ind \mid Ai? = Ci? \right] \qquad RO^P_{Op} \mathrel{\widehat=} RO_{Op}$$

Given a retrieve relation R from A to C, the promoted retrieve relation R^P [DB01] from *AWorld* to *CWorld*, and its instantiation R^P_{AC} in the example are:

R^P

$$AWorld;\ CWorld$$
$$\begin{array}{l} \operatorname{dom} Cf = \operatorname{dom} Af \\ \forall n : \operatorname{dom} Cf \bullet \exists R \bullet \theta A = Af(n) \wedge \theta C = Cf(n) \end{array}$$

R^P_{AC}

$$AWorld;\ CWorld$$
$$\begin{array}{l} \operatorname{dom} Cf = \operatorname{dom} Af \\ \forall n : \operatorname{dom} Cf \bullet Cf(n).Ctot = \Sigma Af(n).Avals \end{array}$$

It is easy to see that the POs (1)-(3) are validated with this choice of R^P_{AC} as retrieve relation for the two worlds of bags.

3. Retrenchment in Z

As indicated in the introduction, retrenchment is motivated by the need at times to evolve system designs in a more flexible way than is permitted by refinement alone. A classic paradigm for this arises when the idealised and abstract modelling uses unbounded data domains (such as pure Peano natural numbers), whereas a more accurate model would need to use a bounded domain (corresponding for example to an actual computer's limited precision numbers). The correspondence between pure naturals and bounded numbers can appear invisible provided all numerical quantities remain far from the built in limit, but of course the correspondence breaks down if the limit is ever reached. And whereas refinement would need to be always sensitive to this discrepancy, no matter even if it is never encountered in any real life use of the system, retrenchment, with its greater flexibility, can accommodate the stated discrepancy, for example by utilizing retrenchment's concedes relations.

The concedes relation of retrenchment gives two methodological benefits: (i) non-refining behaviour is brought within the world of formal discourse, with its verification capabilities of formal and tool-supported reasoning, and (ii) validation activities which lie outside the formal structure are supported, such as probabilistic reasoning about refining vs. non-refining behaviour. In this manner concedes relations provide a mechanism that both captures the failure of refinement on the formal side, and at the same time offers a focus for extra-formal validation: the concessions thus act as a bridge between the formal and non-formal worlds.

In fact retrenchment maximizes this role for providing bridges between the formal and non-formal worlds by furnishing not only a concedes option for the main operation PO consequent, but also other avenues, embodied in the within and output relations. These are all intended to allow designers the greatest possible latitude for eloquence regarding the way the strict tenets of refinement might not be met in any given situation. Moreover, this eloquence is more than just an indulgence, since perspicuity of the design and its evolution can be just as important an ingredient in its validation with respect to users' requirements, as the assurance coming from verification of the formal structure; see [BPJS07].

Let us now describe retrenchment in the Z context continuing with the notational conventions established above. Since we are allowing the system to evolve, and the evolution is more likely than not to imply the incorporation of more and lower level detail in the model (and such detail is more likely than not to require bespoke operations to help manage it which have no sensible place in the more abstract model), we allow the concrete model in a retrenchment to possess operations not present in the abstract one.

For the operations, there are the abstract and concrete state spaces A, D, and the corresponding per-operation I/O spaces: abstract AI_{Op}, AO_{Op} and concrete DI_{Op}, DO_{Op}. Next there are the relations that embody the retrenchment itself. As for refinement, we have a *retrieve* relation $R : \left[A;\ D \right]$ between the state spaces. On a per-operation basis, we then have the within, output and concedes relations. The *within* relation is between the input-state spaces $W_{Op} : \left[AI_{Op};\ A;\ DI_{Op};\ D \right]$. The *output* and *concedes* relations are defined over both full input-state-output frames with types $O_{Op};\ C_{Op} : \left[AI_{Op};\ A;\ A';\ AO_{Op};\ DI_{Op};\ D;\ D';\ DO_{Op} \right]$.

As for refinement, a number of POs define a retrenchment between two models. We define the retrenchment of abstract ADT $(A, AInit, \{(AOp, AI_{Op}, AO_{Op}) \mid Op \in \mathsf{Ops}\})$ by concrete ADT $(D, DInit, \{(DOp, DI_{Op}, DO_{Op}) \mid Op \in$

Ops$\}$). The first PO (*initialisation*) is exactly the same as for refinement (1). For the common operations, the second (*correctness*), is analogous to refinement correctness (3):

$$\forall D' \bullet DInit \Rightarrow \exists A' \bullet AInit \wedge R' \tag{4}$$

$$\forall A;\ AI_{Op};\ D;\ DI_{Op};\ D';\ DO_{Op} \bullet R \wedge W_{Op} \wedge DOp \Rightarrow \exists A';\ AO_{Op} \bullet AOp \wedge ((R' \wedge O_{Op}) \vee C_{Op}) \tag{5}$$

Some comment on these is in order now. The POs (4)-(5) essentially define a *transition* semantics for retrenchment, which is analogous to a partial correctness viewpoint. The original formulation of retrenchment [BP98, BP99b] within the context of the B-Method [Abr96] takes *applicability* aspects into account more explicitly,[1]. However experience (especially [Jes05]) has shown that it is in fact most profitable to subsume any such matters within the within relation — in essence, provided the within relation, when restricted to the relevant variables, is at least as strong as any applicability criterion that might exist for the abstract or concrete models, then smooth interworking between refinement and retrenchment results. See [Jes05, BPJS07] for more details.

This policy on applicability gives a useful degree of requirements perspective expressiveness to a retrenchment between two models, given that there may be other purposes in relating two models by a retrenchment than the traditional reification towards code. For example, one could consider the resolution of inconsistent requirements viewpoints [BP03] — such a scenario is entirely in keeping with our earlier remarks about retrenchment embodying a much more intimate connection between requirements and the technicalities of the formalism than is the case for refinement.

The significance of the retrenchment data W_{Op}, O_{Op}, C_{Op} in (5) is as follows. The within clause W_{Op} specifies the subset of the before-state-input frame within which the retrenchment relation between the (abstract and concrete representatives of the operation *Op* of the two) models is defined; it describes the relation between inputs; and allows transfer of information between input and state. For example, the information contained in an abstract input element might move into a concrete state element or vice versa.

The concedes clause C_{Op} allows the description of the state of affairs that obtains when the retrieve relation is not reestablished by the joint action of *AOp* and *COp*, in a disjunctive context (although R, C_{Op} are by no means mutually exclusive). This may describe a weaker approximation of A' by C'; it may be the reaching of a restart checkpoint (in the concrete world say, allowing a reset operation, perhaps not present at the abstract level, to subsequently reestablish a simulable state of affairs); or the linkage of concrete error information to specific abstract state values. The type of C_{Op} includes the before-state-input frame in order to permit access to history information. In any event C_{Op} allows state-output mutability in the same way that W_{Op} allows state-input mutability.

Finally, the output clause O_{Op} provides the counterpart after-state-output mutability and access to history information, for the case where the retrieve relation is reestablished, conjunctively with R'.

We illustrate retrenchment in our running example by retrenching an individual abstract bag *ABag* to a concrete 'bag' *DBag* for which numbers are restricted by a finite bound, i.e. an incarnation of *ABag* taking on board some real world limitations.

$BN \mathrel{\widehat=} 0..BIGNUM$

DBag

$Dtot : BN$

DInc

$\Delta DBag$
$Din? : BN$

$Dtot + Din? \leq BIGNUM \Rightarrow Dtot' = Dtot + Din?$
$Dtot + Din? > BIGNUM \Rightarrow Dtot' = Dtot$

DSum

$\Xi DBag$
$Dans! : BN$

$Dans! = Dtot$

If we posit the natural retrieve relation between the abstract and concrete spaces *ABag*, *DBag*, namely the natural one inherited from equality on all numbers up to *BIGNUM*:

R_{AD}

$ABag;\ DBag$

$Dtot = \Sigma Avals$

[1] In the B-Method a distinction is made between operation availability expressed via guards, and operation availability expressed via preconditions, so a direct comparison between the B formulation and any putative Z formulation taking applicability more directly into account has limited mileage.

then it is clear that a refinement fails to hold between an *ABag* and a *DBag* as soon as the summed contents of the *ABag* exceed *BIGNUM*. However such a situation is relatively easy to capture using a retrenchment.

Here is the retrenchment setup. Firstly the retrieve relation for the enterprise is R_{AD} as just given, avoiding any need to distort the naturally arising one.[2] The other relations of the retrenchment come on an operation by operation basis. For *DInc* we can choose:

W_{Inc}

$ABag;\ DBag$
$Ain? : \mathbb{N};\ Din? : BN$

$Din? = Ain?$

O_{Inc}

$\Delta ABag;\ \Delta DBag$

---

true

C_{Inc}

$\Delta ABag;\ \Delta DBag$
$Ain? : \mathbb{N};\ Din? : BN$

$Dtot + Din? > BIGNUM$
$Dtot' = Dtot$
$AVals \sqsubseteq AVals'$
$Dtot' \leq \Sigma Avals'$

while for *DSum* we can choose:

W_{Sum}

$ABag;\ DBag$

---

true

O_{Sum}

$\Delta ABag;\ \Delta DBag$
$Aout! : \mathbb{N};\ Dout! : BN$

$Dout! = Aout!$

C_{Sum}

$\Delta ABag;\ \Delta DBag$
$Aout! : \mathbb{N};\ Dout! : BN$

---

false

Note that because *Sum* is an enquiry-only operation and the retrenchment operation PO demands that the retrieve relation holds in the antecedent, the retrieve relation cannot but continue to hold in the consequent, allowing us to trivialise the *Sum* concession.

Notice also the use of the word 'choose' above. When one is less constrained in the description of a development step as happens in retrenchment (compared that is with refinement), there is usually considerably more latitude in the choice of what properties of the development step to formalise within the retrenchment data. For example, consider C_{Inc} above, which describes skipping in the *DBag* when *BIGNUM* is exceeded: we could *choose* to add a disjunct to describe the fact that $Dtot'$ remains equal to $\Sigma Avals'$ (i.e. the retrieve relation is reestablished) when *BIGNUM* is not exceeded.

This design choice is closely related to retrenchment's more intimate connection with requirements engineering, as it represents a commitment from the user perspective or domain perspective to certain aspects of the relationship between the two models. Since this is part of the design process, it comes with an element of risk which is absent in the use of refinement alone. In refinement, provided the refinement is valid, the concrete system will conform (in a sense derived from the specific notion of refinement in use) to the abstract one; therefore all the design risk attaches to the abstract model (this assumes that we can ignore 'useless' refinements which do not improve efficiency, such as having the concrete model be two disjoint copies of the abstract model acting in tandem; a legal refinement but an utterly pointless one). In retrenchment, since the model genuinely evolves, the design risk spreads further, being shared between the abstract model, the concrete model, and the retrenchment data that relate them.

4. The Promotion of Retrenchments in Z

We turn to the construction of the *Dworld* from the individual *D* level components. The details are:

[2] Because of the extreme simplicity of our example, and in particular the fact that there are no operations to remove items from the *ABag*, or to decrease *Dtot*, we could have managed with a rather artificial retrieve relation that related all naturals greater than or equal to *BIGNUM* to the *BIGNUM* of *BN*. However this strategy collapses as soon as one contemplates the indicated operations for removal or decrease.

DWorld

$Df : Ind \pfun DBag$

$DWorldInc \mathrel{\widehat=}$
$\quad \exists \Delta DBag \bullet \Phi DBagOp \land DInc$
$DWorldSum \mathrel{\widehat=}$
$\quad \exists \Delta DBag \bullet \Phi DBagOp \land DSum$

ΦDBagOp

$\Delta DWorld$
$\Delta DBag$
$Di? : Ind$

$Di? \in \operatorname{dom} Df$
$\theta DBag = Df(Di?)$
$Df' = Df \oplus \{Di? \mapsto \theta DBag'\}$

We note that this is exactly the same as was done for the *A* and *C* worlds. We now consider the promotion of the *A* to *D* retrenchment. This turns out to be a process with a number of possible viewpoints depending on how requirements engineering considerations interact with the technical details. In broad terms, the process is analogous to the promotion of refinement: each of the retrenchment data predicates on the local state is promoted to a predicate on the global state, in a manner similar to that in which the local retrieve relation *R* is promoted to the global R^P.

However there are significant differences between promotion of refinement and promotion of retrenchment. We recall that R^P required the abstract/concrete component pair at *every* index *n* to satisfy *R*. This is unsurprising, seeing how for refinement, every operation must reestablish *R* for all component pairs. In retrenchment, the presence of the concedes 'escape clause' indicates that there will be more options to consider: a particular step of some operation may reestablish *R*, or it may not. Given that this is the case, what then is the promoted retrenchment retrieve relation to be, and how should we design the generic schemas for the retrenchment data? It turns out that there are several different sensible answers, each of which says something different about the composed system: each possible answer can be seen as addressing a different requirements scenario.

We will describe three variants of the promotion of a retrenchment: strong promotion, weak promotion, and precise promotion. The next subsection introduces strong promotion, with two subvariants, focused and inclusive. The following subsection introduces weak promotion, also with two analogous corresponding subvariants. The resulting four notions are broadly comparable to retrenchment versions of refinement promotion in that they enjoy similar structural features. Precise promotion, as we will see, raises a number of novel questions regarding the nature of promotion, and will be dealt with in Section 5.

In strong retrenchment promotion the retrenchment between *AWorld* and *DWorld* stays as close as possible to the refinement picture. In particular the promoted retrieve relation is the same as for the promotion of refinement. The focused and inclusive subvariants differ as to whether or not they speak about the out-of-focus components of the system in the world output and concedes relations of the retrenchment: the focused subvariant refers only to the component in focus, while the inclusive subvariant refers to all components.

Recognizing that in a promoted world of retrenchments some components may already have conceded while others are still retrieving, weak promotion of retrenchment replaces the universal quantification over components in the strong promoted retrieve relation by an existential quantification. Again there are two subvariants, the focused and inclusive ones. The focused subvariant refers only to the component in focus, and thus leads to output and concedes relations that are identical to those for focused strong promotion. The inclusive weak subvariant differs from the inclusive strong subvariant in what can be claimed regarding the non-highlighted components, and so gives a genuinely different retrenchment.

4.1. Strong Promotion

The promoted retrieve relation R^{PS} (for *strong* promotion of the retrenchment) is precisely the same as the R^P of refinement promotion. The world level within relation ties together the correspondingly indexed abstract and concrete components with an instance of the individual within relation. The three retrenchment relations also reuse input/output information from the component level, just as the promoted input/output mappings did for refinement. The within relation just reuses input information from W_{Op}.

R^{PS}

AWorld; *DWorld*

$\operatorname{dom} Df = \operatorname{dom} Af$
$\forall n : \operatorname{dom} Df \bullet \exists R \bullet$
$\quad \theta A = Af(n) \land \theta D = Df(n)$

W^{PS}_{Op}

AWorld; *DWorld*
AI_{Op}; DI_{Op}; $Ai?, Di? : Ind$

$Ai? = Di?$
$\exists A, D \bullet W_{Op} \land$
$\quad \theta A = Af(Ai?) \land \theta D = Df(Di?)$

The after-state schemas of a strong promotion describe what happens after the step has occurred. The component in focus has either succeeded in reestablishing the retrieve relation R or has conceded; the others continue to retrieve. The latter follows from the fact that the promotion framing schemas specify that all components other than the one in focus do not change state, and so, because they are hypothesised to retrieve in the before-state in the antecedent of the PO, cannot but continue to do so in the after-state, in the PO consequent.

Since R is implicitly reestablished for all but the component in focus, the most succinct retrenchment predicates $O_{Op}^{PSF}, C_{Op}^{PSF}$ (for *strong* promotion, *focused* variant) describing the after-state situation, only talk about that component: *either* the local output clause O_{Op} is reestablished for component $Ai?$ (along with R^{PS}), *or* local concedes clause C_{Op} is established for component $Ai?$:

O_{Op}^{PSF}

$\Delta AWorld;\ \Delta DWorld$
$AO_{Op};\ DO_{Op};\ AI_{Op};\ DI_{Op}$
$Ai?, Di? : Ind$

$\exists\, \Delta A, \Delta D \bullet O_{Op} \land$
$\quad \theta A = Af(Ai?) \land \theta D = Df(Di?) \land$
$\quad \theta A' = Af'(Ai?) \land \theta D' = Df'(Di?)$

C_{Op}^{PSF}

$\Delta AWorld;\ \Delta DWorld$
$AO_{Op};\ DO_{Op};\ AI_{Op};\ DI_{Op}$
$Ai?, Di? : Ind$

$\exists\, \Delta A, \Delta D \bullet C_{Op} \land$
$\quad \theta A = Af(Ai?) \land \theta D = Df(Di?) \land$
$\quad \theta A' = Af'(Ai?) \land \theta D' = Df'(Di?)$

Even though we have implicit R-reestablishment on components out of focus, it may nevertheless be useful when doing automated verification, to state this fact explicitly in the after-state schemas: this yields schema $OCOthers^{PS}$. Its use gives rise to the *inclusive* variant of strong promotion:

$$O_{Op}^{PSI} \mathrel{\widehat=} O_{Op}^{PSF} \land OCOthers^{PS}$$

$$C_{Op}^{PSI} \mathrel{\widehat=} C_{Op}^{PSF} \land OCOthers^{PS}$$

$OCOthers^{PS}$

$\Delta AWorld;\ \Delta DWorld$
$Ai?, Di? : Ind$

$\forall\, n : \mathrm{dom}\, Df \setminus \{Di?\} \bullet$
$\quad \exists\, \Delta A, \Delta D \bullet R \land R' \land$
$\quad \theta A = Af(n) \land \theta D = Df(n) \land$
$\quad \theta A' = Af'(n) \land \theta D' = Df'(n)$

Let us instantiate the above generic constructions to our running example. Here is the retrieve relation.

R_{AD}^{PS}

$AWorld;\ DWorld$

$\mathrm{dom}\, Df = \mathrm{dom}\, Af$
$\forall\, n : \mathrm{dom}\, Df \bullet Df(n).Dtot = \Sigma Af(n).Avals$

The operation-specific relations for operation *Inc* are:

W_{Inc}^{PS}

$AWorld;\ DWorld$
$Ain? : \mathbb{N};\ Din? : BN$
$Ai?, Di? : Ind$

$Din? = Ain?$
$Di? = Ai?$

O_{Inc}^{PSI}

$\Xi AWorld;\ \Xi DWorld$

---

true

C_{Inc}^{PSI}

$\Delta AWorld;\ \Delta DWorld$
$Ain? : \mathbb{N};\ Din? : BN$

$\mathrm{dom}\, Df = \mathrm{dom}\, Af$
$Df(Di?).Dtot + Din? > BIGNUM$
$Df'(Di?).Dtot = Df(Di?).Dtot$
$Af(Ai?).Avals \sqsubseteq Af'(Ai?).Avals$
$Df'(Di?).Dtot \leq \Sigma Af'(Ai?).Avals$
$\forall\, n : \mathrm{dom}\, Df \setminus \{Di?\} \bullet$
$\quad Df(n).Dtot = \Sigma Af(n).Avals \land$
$\quad Df'(n).Dtot = \Sigma Af'(n).Avals$

and the retrenchment of operation *Sum* is strongly promoted as follows:

W^{PS}_{Sum}

$AWorld;\ DWorld$
$Ai?, Di? : Ind$

$Di? = Ai?$

C^{PSI}_{Sum}

false

O^{PSI}_{Sum}

$\Xi AWorld;\ \Xi DWorld$
$Aout! : \mathbb{N};\ Dout! : BN$

$Dout! = Aout!$
$\forall n : \mathrm{dom}\, Df \setminus \{Di?\} \bullet$
$\quad Df(n).Dtot = \Sigma Af(n).Avals \land$
$\quad Df'(n).Dtot = \Sigma Af'(n).Avals$

The above evidently featured the inclusive output and concedes relations. The focused ones merely don't contain the $(\forall n : \mathrm{dom}\ldots)$ clauses. Note the use of the within relations to refer to the index of the bag which is the focus of the operation in question, a role that cannot be played by the retrieve relation given its duty to relate the global state spaces.

4.2. Weak Promotion

In the weak variant it is merely claimed that at least one concrete component retrieves to an abstract one. This affects the retrieve and within relations which become:

R^{PW}

$AWorld;\ DWorld$

$\mathrm{dom}\, Df = \mathrm{dom}\, Af$
$\exists n : \mathrm{dom}\, Df \bullet \exists R \bullet$
$\quad \theta A = Af(n) \land \theta D = Df(n)$

W^{PW}_{Op}

$AWorld;\ DWorld$
$AI_{Op};\ DI_{Op};\ Ai?, Di? : Ind$

$Ai? = Di?$
$\exists A, D \bullet R \land W_{Op} \land$
$\quad \theta A = Af(Ai?) \land \theta D = Df(Di?)$

Note how the different quantification in the retrieve relation has an impact on the within relation. There must now somehow be a tieup between some anonymous local component claimed to validate the retrieve relation (via the existential quantification $\exists n : \mathrm{dom}\, Df$), and the specific component (whichever it is) described by the free index variable of the promotion. The correct tieup is accomplished, by repeating the local retrieve relation R within the body of the within relation. The conjunction of the two relations in the PO antecedent is then able to select the correct component from the retrieve relation, guided by the free index variable of the within relation. This is an instance of a common phenomenon arising when considering compositions of retrenchments, given how the quantifications are arranged in the retrenchment operation PO [BJP08].

The output and concedes relations for the focused subvariant are:

O^{PWF}_{Op}

$\Delta AWorld;\ \Delta DWorld$
$AO_{Op};\ DO_{Op};\ AI_{Op};\ DI_{Op}$
$Ai?, Di? : Ind$

$\exists \Delta A, \Delta D \bullet O_{Op} \land$
$\quad \theta A = Af(Ai?) \land \theta D = Df(Di?) \land$
$\quad \theta A' = Af'(Ai?) \land \theta D' = Df'(Di?)$

C^{PWF}_{Op}

$\Delta AWorld;\ \Delta DWorld$
$AO_{Op};\ DO_{Op};\ AI_{Op};\ DI_{Op}$
$Ai?, Di? : Ind$

$\exists \Delta A, \Delta D \bullet C_{Op} \land$
$\quad \theta A = Af(Ai?) \land \theta D = Df(Di?) \land$
$\quad \theta A' = Af'(Ai?) \land \theta D' = Df'(Di?)$

In the inclusive subvariant, we can make the claim given in the schema $OCOthers^{PW}$, leading to the following output and concedes relations:

$$O^{PWI}_{Op} \mathrel{\widehat=} O^{PWF}_{Op} \land OCOthers^{PW}$$

$$C^{PWI}_{Op} \mathrel{\widehat=} C^{PWF}_{Op} \land OCOthers^{PW}$$

$OCOthers^{PW}$

$\Delta AWorld;\ \Delta DWorld$
$Ai?, Di? : Ind$

$\forall n : \operatorname{dom} Df \setminus \{Di?\} \bullet \exists \Delta A, \Delta D \bullet$
$\quad \theta A = Af(n) \land \theta D = Df(n) \land$
$\quad \theta A' = Af'(n) \land \theta D' = Df'(n)$
$\quad \Rightarrow (R \Rightarrow R')$

The after-state relations for inclusive weak promotion give information about the component in focus indexed by $Di?$, and claim that the retrieve relation is preserved for those other components that happen to validate the retrieve relation before the promoted operation.

For the running example this all gives:

R^{PW}_{AD}

$AWorld;\ DWorld$

$\operatorname{dom} Df = \operatorname{dom} Af$
$\exists n : \operatorname{dom} Df \bullet Df(n).Dtot = \Sigma Af(n).Avals$

W^{PW}_{Inc}

$AWorld;\ DWorld$
$Ain? : \mathbb{N};\ Din? : BN$
$Ai?, Di? : Ind$

$Din? = Ain?$
$Di? = Ai?$
$Df(Di?).Dtot = \Sigma Af(Ai?).Avals$

O^{PWI}_{Inc}

$\Xi AWorld;\ \Xi DWorld$

---

true

C^{PWI}_{Inc}

$\Delta AWorld;\ \Delta DWorld$
$Ain? : \mathbb{N};\ Din? : BN$

$\operatorname{dom} Df = \operatorname{dom} Af$
$Df(Di?).Dtot + Din? > BIGNUM$
$Df'(Di?).Dtot = Df(Di?).Dtot$
$Af(Ai?).Avals \sqsubseteq Af'(Ai?).Avals$
$Df'(Di?).Dtot \leq \Sigma Af'(Ai?).Avals$
$\forall n : \operatorname{dom} Df \setminus \{Di?\} \bullet$
$\quad Df(n).Dtot = \Sigma Af(n).Avals$
$\quad \Rightarrow Df'(n).Dtot = \Sigma Af'(n).Avals$

and

W^{PW}_{Sum}

$AWorld;\ DWorld$
$Ai?, Di? : Ind$

$Di? = Ai?$
$Df(Di?).Dtot = \Sigma Af(Ai?).Avals$

O^{PWI}_{Sum}

$\Xi AWorld;\ \Xi DWorld$
$Aout! : \mathbb{N};\ Dout! : BN$

$Dout! = Aout!$
$\forall n : \operatorname{dom} Df \setminus \{Di?\} \bullet$
$\quad Df(n).Dtot = \Sigma Af(n).Avals$
$\quad \Rightarrow Df'(n).Dtot = \Sigma Af'(n).Avals$

C^{PWI}_{Sum}

false

We observe that the (inclusive subvariants of) strong and weak promoted retrenchment data are related as follows (which in turn depends on assuming nonempty domains for all variables, in order to avoid the usual anomalies):

$$R^{PS} \Rightarrow R^{PW} \qquad O^{PSI}_{Op} \Rightarrow O^{PWI}_{Op} \qquad O^{PSF} \Leftrightarrow O^{PWF}$$
$$R^{PS} \land W^{PS}_{Op} \Rightarrow W^{PW}_{Op} \qquad C^{PSI}_{Op} \Rightarrow C^{PWI}_{Op} \qquad C^{PSF} \Leftrightarrow C^{PWF}$$

5. Precise Promotion

Noting that the existential quantification of weak promotion is vague about which components are still retrieving, and which have already conceded and are no longer in the retrieve relation, in this section we investigate a further variant

of retrenchment promotion, precise promotion, which tracks this information[3] via an additional world level 'good' variable.

The desire to track the information via a world level good variable quickly raises the question: which world? This arises since the retrieve relation (as well as the concedes and other relations of a retrenchment) are defined to be *joint* properties of the abstract and concrete models, thus are not the exclusive possession of either.

In this paper we focus on having the good variable in the concrete world, i.e. it is a *Dgood* variable. We favour this since when a concrete component concedes, and thus (usually) falls outside the remit of the retrieve relation, there might well be concrete operations that help 'reset' some appropriate aspect of the concrete system state, potentially enabling the retrieve relation to be recovered for that component; see e.g. [BP99a]. Such 'reset' operations can easily be enhanced during the promotion to the *Dworld*, with suitable updates to the *Dgood* variable, which bring the errant component back into the retrieving fold in the wider multicomponent context, and the (potentially proper) inclusion of abstract operation names into concrete operation names is intended to comfortably facilitate such scenarios. However, it should be noted that such reset operations, being absent from the abstract system, have no part in the retrenchment data (due to the absence of the corresponding abstract operations), and therefore, what we say below about the management of the 'good' variable, since it is restricted purely to the retrenchment data, will apply *mutatis mutandis* if the 'good' variable lives in the abstract rather than the concrete world.

If it is the case that the 'reset' operations need to be enhanced during promotion to include updates to the *Dgood* variable, then the operations common to both abstract and concrete systems (i.e. those for which there *is* retrenchment data) must also be similarly enhanced, in order that the *Dgood* variable stays accurate whenever any particular step taken by any particular component causes that component's status to change.

Therefore, for precise promotion to work, we need to be able to infer without confusion, whether a concrete operation step reestablishes the retrieve relation, or whether it concedes; this on the basis of the concrete step alone. Such a separation property does not hold automatically for retrenchments, so must be imposed as an axiom. The following axiom, which is assumed to hold for each common operation *Op*, does the trick:

$$DEstRet^{PP}_{DOp} \wedge DEstCon^{PP}_{DOp} \Leftrightarrow \text{false} \tag{6}$$

where

$$DEstRet^{PP}_{DOp} \mathrel{\widehat=} D;\ DI_{Op};\ D';\ DO_{Op} \mid DOp\ \wedge (\exists A;\ AI_{Op};\ A';\ AO_{Op} \bullet R \wedge W_{Op} \wedge AOp \wedge (R' \wedge O_{Op}))$$

$$DEstCon^{PP}_{DOp} \mathrel{\widehat=} D;\ DI_{Op};\ D';\ DO_{Op} \mid DOp\ \wedge (\exists A;\ AI_{Op};\ A';\ AO_{Op} \bullet R \wedge W_{Op} \wedge AOp \wedge C_{Op})$$

$$DNotEstRet^{PP}_{DOp} \mathrel{\widehat=} D;\ DI_{Op};\ D';\ DO_{Op} \mid DOp\ \wedge \neg(\exists A;\ AI_{Op};\ A';\ AO_{Op} \bullet R \wedge W_{Op} \wedge AOp \wedge (R' \wedge O_{Op}))$$

The axiom (6) enables us to determine, on the basis of the information in a concrete step alone, whether the step is capable of establishing the retrieve relation or the concession (but never both). Thus a step of $DWorldOp^{PP}$ is: either able to be simulated in such a way that the retrieve relation is reestablished, in which case we assume it is appropriate to allow its component index $Di?$ to remain in $Dgood'$ (on the assumption that $Di? \in Dgood$ to start with); or the step establishes the concession, or it is not simulable at all (note that $DNotEstRet^{PP}_{DOp}$ must subsume $DEstCon^{PP}_{DOp}$, since $DNotEstRet^{PP}_{DOp}$ is the negation of $DEstRet^{PP}_{DOp}$, and $DEstCon^{PP}_{DOp}$ is asserted disjoint from $DEstRet^{PP}_{DOp}$), in which case we deem it inappropriate for $Di?$ to be in $Dgood'$.

Armed with this, we can give the generic enhancement needed for the concrete world versions of common operations:

$DWorld^{PP}$

$DWorld$
$Dgood : \mathbb{P}\, Ind$

$Dgood \subseteq \operatorname{dom} Df$

ΦDOp

$\Delta DWorld$
ΔD
$Di? : Ind$

$Di? \in \operatorname{dom} Df$
$\theta D = Df(Di?)$
$Df' = Df \oplus \{Di? \mapsto \theta D'\}$

$$DWorldOp^{PP} \mathrel{\widehat=} \exists \Delta D \bullet \Phi DOp \wedge DOp \wedge (DEstRet^{PP}_{DOp} \Rightarrow Dgood' = Dgood) \wedge (DNotEstRet^{PP}_{DOp} \Rightarrow Dgood' = Dgood \setminus \{Di?\})$$

[3] One can imagine further variations on the promotion theme that track other aspects of the two systems, but we will not consider them.

We can give now the appropriate world level retrieve and within relations:

$$R^{PP}$$
$$AWorld;\ DWorld^{PP}$$
$$\mathrm{dom}\, Df = \mathrm{dom}\, Af$$
$$\forall n \in Dgood \bullet \exists R \bullet \theta A = Af(n) \land \theta D = Df(n)$$

$$W^{PP}_{Op}$$
$$AWorld;\ DWorld^{PP}$$
$$AI_{Op};\ DI_{Op};\ Ai?, Di? : Ind$$
$$Ai? = Di? \land Di? \in Dgood$$
$$\exists A, D \bullet W_{Op} \land \theta A = Af(Ai?) \land \theta D = Df(Di?)$$

Note how the tieup between the components denoted by *Dgood* and the specific component referred to by *Di*? is handled in the within relation.

For the output and concedes relations, we can again draw up focused and inclusive subvariants:

$$O^{PPF}_{Op}$$
$$\Delta AWorld;\ \Delta DWorld^{PP}$$
$$AO_{Op};\ DO_{Op};\ AI_{Op};\ DI_{Op}$$
$$Ai?, Di? : Ind$$
$$Di? \in Dgood'$$
$$\exists \Delta A, \Delta D \bullet O_{Op} \land \theta A = Af(Ai?) \land \theta D = Df(Di?) \land \theta A' = Af'(Ai?) \land \theta D' = Df'(Di?)$$

$$C^{PPF}_{Op}$$
$$\Delta AWorld;\ \Delta DWorld^{PP}$$
$$AO_{Op};\ DO_{Op};\ AI_{Op};\ DI_{Op}$$
$$Ai?, Di? : Ind$$
$$Di? \notin Dgood'$$
$$\exists \Delta A, \Delta D \bullet C_{Op} \land \theta A = Af(Ai?) \land \theta D = Df(Di?) \land \theta A' = Af'(Ai?) \land \theta D' = Df'(Di?)$$

$$O^{PPI}_{Op} \mathrel{\widehat{=}} O^{PPF}_{Op} \land OCOthers^{PP}$$

$$C^{PPI}_{Op} \mathrel{\widehat{=}} C^{PPF}_{Op} \land OCOthers^{PP}$$

$$OCOthers^{PP}$$
$$\Delta AWorld;\ \Delta DWorld^{PP}$$
$$Ai?, Di? : Ind$$
$$\forall n \in Dgood' \setminus \{Di?\} \bullet \exists \Delta A, \Delta D \bullet \theta A = Af(n) \land \theta D = Df(n) \land \theta A' = Af'(n) \land \theta D' = Df'(n) \land R \land R'$$

At this point we could consider enhancing the inclusions stated at the end of Section 4.2, between the inclusive subvariants of the various promoted retrenchment data, to incorporate the retrenchment data of the precise variant, and we might anticipate that it falls in a position intermediate between the strong and weak variants. However the precision inherent in the *Dgood* variable means that many of the resulting implications have to be fortified with specific additional hypotheses, and the results are fewer in number and are rather inelegant. We leave them to the motivated reader.

Returning to our running example, we show the promoted retrenchment data explicitly:

$$R^{PP}_{AD}$$
$$AWorld;\ DWorld^{PP}$$
$$\mathrm{dom}\, Df = \mathrm{dom}\, Af$$
$$\forall n : Dgood \bullet Df(n).Dtot = \Sigma Af(n).Avals$$

W^{PP}_{Inc}

$AWorld;\ DWorld^{PP}$
$Ain? : \mathbb{N}$
$Din? : BN$
$Ai?, Di? : Ind$

$Din? = Ain?$
$Di? = Ai?$
$Di? \in Dgood$

O^{PPI}_{Inc}

$\Xi AWorld;\ \Xi DWorld^{PP}$

$Di? \in Dgood'$

C^{PPI}_{Inc}

$\Delta AWorld;\ \Delta DWorld^{PP}$
$Ain? : \mathbb{N}$
$Din? : BN$

$Di? \notin Dgood'$
$\mathrm{dom}\, Df = \mathrm{dom}\, Af$
$Df(Di?).Dtot + Din? > BIGNUM$
$Df'(Di?).Dtot = Df(Di?).Dtot$
$Af(Ai?).Avals \sqsubseteq Af'(Ai?).Avals$
$Df'(Di?).Dtot \leq \Sigma Af'(Ai?).Avals$
$\forall n : Dgood \setminus \{Di?\} \bullet$
$\quad Df(n).Dtot = \Sigma Af(n).Avals \land$
$\quad Df'(n).Dtot = \Sigma Af'(n).Avals$

W^{PPI}_{Sum}

$AWorld;\ DWorld^{PP}$
$Ai?, Di? : Ind$

$Di? = Ai?$
$Di? \in Dgood$

C^{PPI}_{Sum}

false

O^{PSI}_{Sum}

$\Xi AWorld;\ \Xi DWorld^{PP}$
$Aout! : \mathbb{N}$
$Dout! : BN$

$Di? \in Dgood'$
$Dout! = Aout!$
$\forall n : Dgood \setminus \{Di?\} \bullet$
$\quad Df(n).Dtot = \Sigma Af(n).Avals \land$
$\quad Df'(n).Dtot = \Sigma Af'(n).Avals$

6. Conclusion

In the previous sections we started by reviewing Z promotion and how it interacts with Z refinement. This interaction is particularly clean in the simple case where the component retrieve relation just replicates over all components of the promotion. We went on to introduce a Z formulation of retrenchment, and then considered how that interacts with the promotion mechanism, restricting all the while to the simple replicated retrieve relation case. The discussion of Section 4 made clear that retrenchment opens up a number of new aspects for promotion, absent from the promotion of refinement. These primarily centre around which components are still continuing to retrieve, and which might already have conceded, and the extent to which the promoted retrenchment wishes to acknowledge the latter and to be explicit about them. We indicated that these promotion options reflected different requirements scenarios. Examination of further requirements scenarios, through a significant application of these techniques, would be expected to reveal further design drivers on the promotion of retrenchment. An example concerned with system functionality would be the promotion of component-level interactions. An example concerned with system security would centre on the need for redundancy, or fault-tolerance, in operations in a lossy world.

Strong promotion focuses on the all-retrieving case, and while asserting a high degree of adherence to refinement desiderata, ceases to be applicable as soon as the first component has conceded (since the all-components-retrieving condition has ceased to hold). By contrast, weak promotion, while asserting something much weaker as regards refinement desiderata, remains applicable right up to the concession of the last still-retrieving component. The focused and inclusive variants of these differ in whether the after-state relations choose to make any claim about the out-of-focus components.

Precise promotion, as its name suggests, attempts a more incisive quantification of which components are still retrieving and which have already conceded. In fact the objective of being more precise about this throws up a host of methodological issues, which were largely sidestepped in Section 5 by the assumption of the conveniently simplifying axiom (6). The issues centre round who ‘owns’ the information about the components, how it gets updated, and what the meaning of this information is. In this paper we kept to the simplest possible scenario, namely of holding the information in a concrete world variable, and of updating it according to whether the component in focus is able to reestablish the retrieve relation or not. (An alternative but similar formulation arises if we hold the information in an abstract world variable.)

The significance of a world variable such as the *Dgood* variable arises through the inductive structure of an exe-

cution. Given a concrete execution, provided all steps correctly update *Dgood* (which is possible in the light of (6), and provided all concrete operations which *do not* correspond to abstract ones also update *Dgood* correctly), then the value of *Dgood* at any point will indeed be the set of components of the *DWorld* that are simulated by suitable abstract components.[4]

The introduction of the *Dgood* variable engenders a certain breach with the philosophy of retrenchment: separation of concerns in specification. The flexibility afforded by the disjunctive PO is intended to allow the specifier of a model to concentrate on the meaning of that model —be it some functionality viewpoint, security, or performance— without being forced to incorporate any features of a different model of the development chain by the need for simulability. The decision to record, in the concrete world (via *Dgood*), the degree of conformance of abstract and concrete components, forces the specifier of the concrete world to describe its behaviour partly in terms of the abstract one, as expressed in operation $DWorldOp^{PP}$. However we can observe that this intrusion of conformance data into model data is rather mild. We just add a new variable to the model, whose value depends on existing model variables, but whose value does not affect existing model variables. This is somewhat reminiscent of a superposition refinement [BS96, Kat93].

The traditional territory of retrenchment has been to explore the 'grey area' on the boundary of the domain of definition of a refinement relation, and to incorporate the non-refining region into the realm of formal reasoning. Normally this happens without the use of additional variables in either of the two models in question; however precise promotion forces the inclusion of at least one such variable (*Dgood*). Still, this is not the first venture into notions of the degree of approximation of a refinement by a retrenchment necessitating the use of additional variables: [PB02] proposed *evolving* retrenchment, where the extra data, modelling the varying degree of approximation over a sequence of operation steps, was simply a meta-variable. That is, the retrenchment PO had an extra free variable denoting the degree of precision. The precise promotion of retrenchment as we have presented it, goes further and incorporates the accuracy information in a genuine (concrete) model variable. However the meta- route would be a feasible alternative way to go.

These difficulties are only compounded when nondeterminism features heavily in the two systems, potentially leading to imprecision in the updates of the 'good' variable, especially if we weaken the assumptions made in Section 5. Ironically, such difficulties can be considerably eased, at least on a technical level, by reorienting the retrenchment to go in a backward rather than forward direction. In backward retrenchment, as in backward refinement, one in a sense knows what happens next before one has to say anything about the state of affairs beforehand, so one can track the values of the 'good' variable backwards in time a great deal more easily than in the forward direction, and under weaker assumptions. We will pursue these ideas elsewhere.

References

[ABDM00] N. Aguirre, J. Bicarregui, T. Dimitrakos, and T. Maibaum. Towards dynamic population management of abstract machines in the B method. In D. Bert, J.P. Bowen, S. King, and M. Waldén, editors, *Third International Conference of B and Z Users*, volume 2651 of *LNCS*, pages 528–546, Turku, Finland, June 2000. Springer.

[Abr96] J.-R. Abrial. *The B-Book: Assigning Programs to Meanings*. Cambridge University Press, 1996.

[BJP08] R. Banach, C. Jeske, and M. Poppleton. Composition mechanisms for retrenchment. *J. Log. Alg. Prog.*, 2008. to appear.

[BJPS05] R. Banach, C. Jeske, M. Poppleton, and Stepney S. Retrenching the purse: Finite exception logs, and validating the small. In M. Hinchey, editor, *Proc. IEEE/NASA SEW30-06*, pages 234–245, 2005.

[BJPS06] R. Banach, C. Jeske, M. Poppleton, and Stepney S. Retrenching the purse: Hashing injective clear codes, and security properties. In *Proc. IEEE ISOLA-06*, 2006. to appear.

[BJPS07] R. Banach, C. Jeske, M. Poppleton, and Stepney S. Retrenching the purse: The balance enquiry quandary, and generalised and (1,1) forward refinements. *Fund. Inf.*, 77:29–69, 2007.

[BP98] R. Banach and M. Poppleton. Retrenchment: An engineering variation on refinement. In D. Bert, editor, *2nd International B Conference*, volume 1393 of *LNCS*, pages 129–147, Montpellier, France, April 1998. Springer.

[BP99a] R. Banach and M. Poppleton. Retrenchment and punctured simulation. In K. Araki, A. Galloway, and K. Taguchi, editors, *Proc. IFM'99:Integrated Formal Methods 1999*, pages 457–476, University of York, June 1999. Springer.

[BP99b] R. Banach and M. Poppleton. Sharp retrenchment, modulated refinement and simulation. *Formal Aspects of Computing*, 11:498–540, 1999.

[BP03] R. Banach and M. Poppleton. Retrenching partial requirements into system definitions: A simple feature interaction case study. *Requirements Engineering Journal*, 8(2), 2003. 22pp.

[BPJS05] R. Banach, M. Poppleton, C. Jeske, and Stepney S. Retrenching the purse: Finite sequence numbers, and the tower pattern. In J. Fitzgerald, I. Hayes, and A. Tarlecki, editors, *Proc. FM-06, LNCS*, pages 382–398, 2005.

[BPJS07] R. Banach, M. Poppleton, C. Jeske, and Stepney S. Engineering and theoretical underpinnings of retrenchment. *Sci. Comp. Prog.*, 67:301–329, 2007.

[BS96] R.J.R. Back and K. Sere. Superposition refinement of reactive systems. *Formal Aspects of Computing*, 8(3):324–346, 1996.

[4] For this to carry through properly, we would need the *punctured simulations* of [BP99a].

[BvW89] R. J. R. Back and J. von Wright. Refinement calculus part I: Sequential nondeterministic programs. In J.W. de Bakker, W.-P. de Roever, and G. Rozenberg, editors, *Proc. REX Workshop, Stepwise Refinement of Distributed Systems*, volume 430 of *LNCS*, pages 42–66. Springer, 1989.

[BvW98] R. J. R. Back and J. von Wright. *Refinement Calculus: A Systematic Introduction*. Springer, 1998.

[CSW02] D. Cooper, S. Stepney, and J. Woodcock. Derivation of Z refinement proof rules. Technical Report YCS-2002-347, University of York, 2002.

[DB01] J. Derrick and E. Boiten. *Refinement in Z and Object-Z*. FACIT. Springer, 2001.

[dRE98] W.-P. de Roever and K. Engelhardt. *Data Refinement: Model-Oriented Proof Methods and their Comparison*. Cambridge University Press, 1998.

[Hen02] P. Henderson. Reasoning about asynchronous behaviour in distributed systems. In *Proc. Eighth IEEE Int. Conf. on Engineering of Complex Computer Systems (ICECCS02)*, pages 17–24, Greenbelt, Maryland, December 2002. IEEE Computer Society Press.

[Jes05] C. Jeske. *Algebraic Theory of Retrenchment and Refinement*. PhD thesis, School of Computer Science, University of Manchester, 2005.

[Kat93] S. Katz. A superimposition control construct for distributed systems. *ACM TPLAN*, 15(2):337–356, April 1993.

[Lup91] P. J. Lupton. Promoting forward simulation. In *Proceedings of the Fifth Annual Z User Meeting and Z User Workshop*, pages 27–49. Springer, 1991.

[PB02] M. Poppleton and R. Banach. Controlling control systems: An application of evolving retrenchment. In D. Bert, J.P. Bowen, M.C. Henson, and K. Robinson, editors, *Proc. ZB2002: Formal Specification and Development in Z and B*, volume 2272 of *LNCS*, Grenoble, France, January 2002. Springer.

[PB03] M. Poppleton and R. Banach. Structuring retrenchments in B by decomposition. In K. Araki, S. Gnesi, and D. Mandrioli, editors, *Proc. FME2003: Formal Methods*, volume 2805 of *LNCS*, pages 814–833, Pisa, Italy, September 2003. Springer.

[SCW00] S. Stepney, D. Cooper, and J. Woodcock. An electronic purse: Specification, refinement and proof. Technical Report PRG-126, Oxford University Computing Laboratory, 2000.

[WD96] J. Woodcock and J. Davies. *Using Z: Specification, Refinement and Proof*. Prentice-Hall, 1996.