# Minimally and Maximally Abstract Retrenchments

C. Jeske, R. Banach
Computer Science Dept., Manchester University, Manchester, M13 9PL, U.K.
jeskec@cs.man.ac.uk,banach@cs.man.ac.uk

**Abstract.** The drawbacks of using refinement alone in the construction of specifications from simple abstract models is used as the spur for the introduction of retrenchment — a method based on the main ideas of refinement but one which is more liberal in character. The basics of the retrenchment mechanism are reviewed in preparation for exploring its integration with refinement. The particular aspect of integration investigated in this paper is the factorisation of a retrenchment step from an abstract to a concrete model into a refinement followed by a retrenchment. The objective is to engineer a system which is at the level of abstraction of the concrete model, but is refinable from the abstract one. The construction given here solves the problem in a universal manner, there being a canonical factorisation of the original retrenchment into an I/O-filtered refinement to the universal system followed by a retrenchment. The universal property arises from the fact that the refinement component of any similar factorisation is refinable to the universal system. An idempotence property supports the claim that the construction is at the correct level of abstraction. A synopsis of an earlier result which factorised a retrenchment step into a canonical retrenchment to a universal system followed by a refinement is presented. A refinement relationship is then shown to exist between the two universal systems. Finally, the consequences of including termination criteria are briefly explored.
**Keywords.** Refinement, Retrenchment, Integration.

## 1 Introduction

Refinement is a well established technique for the formal construction of software. Of the many variants which can be found in the literature, the refinement calculus as exemplified by Back and von Wright [2] and Morgan [14], is concerned with the application of correctness-preserving transformations which convert an abstract and possibly non-executable specification of some program into efficient executable code. Alternatively, in refinement methods like B [1], VDM [11] and Z [18], the notion of refinement is less calculational. Here, each refinement step involves writing a yet more concrete version of the original specification, typically by the addition of more detail into the development. To ensure that the result of each step meets the requirements of the original specification, the developer is required to show that certain proof obligations (POs) hold at each stage. It is this latter process for which the term refinement will be used in the remainder of this paper. For a tutorial on refinement calculi and methods consult [12].

The ability to steadily incorporate detail has resulted in refinement being used not only in the implementation of specifications but also as a tool in the construction of the specifications themselves. However, when the method is applied to realistic systems, it often suffers from shortcomings which at best make the development of the specification awkward, and at worst can mean that large parts of the process cannot be controlled by the refinement mechanism. In particular, if we look at the construction of systems

which are entirely discrete in character, we sometimes find that the use of refinement imposes an unnatural decomposition on the development, with the technique obstructing the gradual construction of the specification in an intelligible and clear manner. For problems which start life as models expressed using continuous mathematics, the situation is more severe. The need to move from the continuous to the discrete domain means refinement is usually unable to relate most of the models in the development, thus excluding large parts of the process from the advantages of a formal approach. In the light of these difficulties, a more flexible specification tool called retrenchment was first put forward in [4], and further developed in [5, 6, 7]. Although based on the main ideas of refinement, it is more liberal in character, and is intended to be able to relate pairs of models that arise naturally in the design process, or occur as a result of standard engineering methods, and which cannot be related by refinement.

The ability to accommodate accepted engineering practice is an important one. To expect engineers to abandon tried and tested procedures is unrealistic and any attempts to impose an unnatural development strategy are likely to result in rejection. Unfortunately, attempts to shoehorn established design methods into the framework of the refinement mechanism often result in just such a strategy. The aim of retrenchment is therefore not to disrupt current discipline but to enable specifications to be developed in a manner which is in harmony with established engineering or designer intuition.

The more liberal nature of retrenchment arises from its less demanding POs, whose structure has been influenced by the intention to facilitate the smooth integration of retrenchment with its refinement precursor. The integration of these two techniques is the topic of this paper, which builds on the work in [3], and examines a further way in which a refinement and retrenchment step can be combined.

Let us suppose we have a retrenchment from an abstract to a concrete model. Given such a step, the work carried out in [3] addressed the problem of finding a system which reflected at the abstract level the detail which had been incorporated into the concrete model. The solution presented was a universal construction at the level of abstraction of the abstract model, but which was refinable to the concrete one. Following on from this result, the goal of the work presented in this paper is to construct a system which is at the level of abstraction of the concrete model but is refinable from the abstract one. The solution given here is a universal system which is at the required level of abstraction.

Such constructions serve to integrate retrenchment and refinement and show that the two techniques are not incompatible with one another. For those charged with the responsibility of building system specifications, this issue is an important one. To accept retrenchment into the fold of formal development methods, they need to be secure in the knowledge that the use of retrenchment will not split the development process into incompatible and irreconcilable paths.

The remainder of this paper is organised as follows. Section 2 touches on the drawbacks of using refinement as the sole means to develop detailed specifications from simple abstract models. This serves as the motivation for retrenchment, the basics of which

are presented in Section 3. Section 4 defines I/O-filtered refinements, a particular form of refinement which facilitates its integration with retrenchment. The heart of this paper is Section 5, which looks at the canonical factorisation of a retrenchment step into a refinement followed by a retrenchment. That this construction is idempotent is shown in Section 6. Section 7 briefly outlines an earlier result which looked at factorising a retrenchment step into a canonical retrenchment followed by a refinement. Section 8 considers an alternative approach to requirements validation. Section 9 shows that there is a refinement relationship between the intermediate systems constructed in the two factorisations. In the penultimate section the results presented so far are extended to a total correctness setting. Finally, Section 11 concludes.

*Notation*. In the sequel we will view systems from a set theoretic and relational viewpoint, which we discuss using a logical meta-notation. Thus a predicate *is* just a notation for a set etc.

## 2 Some Drawbacks of Refinement

We shall use the forward simulation method of proving refinement, since almost all applications of refinement are of this kind. For a comprehensive discussion of both forward and backward simulation see [10].

We introduce a small example which nicely illustrates the kinds of problems that can be experienced when using refinement as the sole tool for constructing a specification. Consider a system whose state $u$ is a set of NATs, and which has an operation $AddEl(i)$ to add an element $i$ to the set. Its description (in a suitable syntactic framework) will be our idealised abstract model. At a more concrete level, we model sets of NATs by injective sequences of NATs, so $u = \{1, 2, 3\}$ would correspond to $w = [1, 2, 3]$, or $w = [2, 1, 3]$, or to any of the other four possible serialisations. It follows that the retrieve relation, $G(u, w)$, which associates corresponding abstract and concrete states, has the form $u = \mathrm{ran}(w)$. At this more concrete level we also wish to take account of a practical aspect of the system and limit the maximum length of the sequence to 10. Thus not all sets have concrete representations. The concrete $AddEl(i)$ operation must test the length of the sequence and for the prior presence of $i$. If the length of the sequence is less than 10, we can model the addition of a new element by, for example, appending the new integer to the end of the sequence. But what are we to do when the length of the sequence is at its maximum? Clearly, appending a new element in this case is not an option since this would violate the bound on the length of $w$. Now, for the concrete $AddEl(i)$ to be a refinement of the abstract one the following PO must hold:

$$G(u, w) \wedge stp_{AddEl_C}(w, i, w') \Rightarrow (\exists\, u' \bullet stp_{AddEl_A}(u, i, u') \wedge G(u', w')) \qquad (2.1)$$

where $stp_{AddEl_C}(w, i, w')$ is the transition or step relation of the concrete $AddEl$ (similarly for the abstract one), and primes indicate the after-state. Examining (2.1) we see that whenever the concrete operation can make a step, the abstract operation must be able to make some matching step. Thus simply doing nothing, i.e. modelling this situation with a *skip*, is not an alternative since an element will still be added at the abstract level. The

only other plausible choice would be for the concrete operation to output an error message, but this would also fail because (2.1) does not allow a change in the I/O signature. Even if the concrete operation did something else within its constraints, it could not tie up with the abstract operation, which would produce a set of cardinality 11.

The above demonstrates one very simple situation in which refinement fails to adequately describe a desirable development step. A similar but more detailed example can be found in [8]. For systems involving continuous variables the shortcomings of refinement are discussed in [15] which looks at the specification of a program for dose calculation in radiotherapy. In the following section we introduce retrenchment, and show how it effortlessly overcomes the problem encountered above with refinement.

# 3 Retrenchment

To make progress we will need to talk about abstract and concrete systems, which we shall call *Abs* and *Conc* respectively. Each system will be described by a state space and a set of operation names. Individual operations will be defined by a transition or step relation and will have their own input and output spaces. For the present we shall work in a partial correctness framework, thus the transition relations will only consist of steps which initialise and terminate successfully. Refer once more to [10] for one approach to partial and total correctness in a relational setting.

For *Abs* the state space will be denoted by $\mathsf{U}$ with typical element $u$. $\mathsf{Ops}_\mathrm{A}$ will be the set of operation names with $Op_\mathrm{A}$ designating a typical operation. For each $Op_\mathrm{A}$ the input and output spaces will be $\mathsf{I}_{Op_\mathrm{A}}$ and $\mathsf{O}_{Op_\mathrm{A}}$ with $i$ and $o$ representing typical elements respectively. We will miss out the subscripts on $i$ and $o$, since it will be clear from the context as to the $Op_\mathrm{A}$ to which they belong. A typical $Op_\mathrm{A}$ transition will be depicted by $u$ -($i$, $Op_\mathrm{A}$, $o$)-› $u'$, where $u$ and $u'$ are the before- and after- states, and $i$ and $o$ are the input and output values. The set of such steps will form the transition or step relation $stp_{Op_\mathrm{A}}(u, i, u', o)$. $Init_\mathrm{A}(u')$ is the predicate that will represent the initialisation operation which will set the state to some initial value $u'$.

The set up for the concrete system *Conc* parallels that for *Abs*. The state space will be $\mathsf{W}$ with typical element $w$. Inputs and outputs are $k \in \mathsf{K}$ and $q \in \mathsf{Q}$ respectively. The operation names are $Op_\mathrm{C} \in \mathsf{Ops}_\mathrm{C}$, and for each $Op_\mathrm{C}$ the transition relation is $stp_{Op_\mathrm{C}}(w, k, w', q)$, with typical transition $w$ -($k$, $Op_\mathrm{C}$, $q$)-› $w'$. The initialisation operation is $Init_\mathrm{C}(w')$.

Unlike with refinement, *Conc* is permitted to have additional operations and so $\mathsf{Ops}_\mathrm{A} \subseteq \mathsf{Ops}_\mathrm{C}$. Corresponding abstract and concrete states will be related by the familiar retrieve relation $G(u, w)$. In addition, for each $Op_\mathrm{A} \in \mathsf{Ops}_\mathrm{A}$, there is a within relation $P_{Op}(i, k, u, w)$ and a concedes relation $C_{Op}(u', w', o, q; i, k, u, w)$. These combine to give the proof obligations for retrenchment. First, there is the initialisation PO (Init PO), which is the same as for refinement. Thus:

$$Init_\mathrm{C}(w') \Rightarrow (\exists\, u' \bullet Init_\mathrm{A}(u') \wedge G(u', w')) \tag{3.1}$$

Second, each pair of related operations must satisfy the operation PO (Op PO):

$$G(u, w) \wedge P_{Op}(i, k, u, w) \wedge stp_{Op_C}(w, k, w', q) \Rightarrow$$
$$(\exists\, u', o \bullet stp_{Op_A}(u, i, u', o) \wedge (G(u', w') \vee C_{Op}(u', w', o, q; i, k, u, w))) \quad (3.2)$$

Notice that the within relation is concerned with both before-states and input values and so enables a change in input signature as well as mixing of input and state information. This allows, for example, information which formed part of the state at one level, to be remodelled as input at the next. What is more, appearing as a conjunct in the antecedent, the within relation strengthens the retrieve relation in before-states and so can be used to restrict the relationship between adjoining levels of abstraction. The concedes relation is mainly concerned with after-states and output values, but may involve before-states and inputs for greater flexibility; the punctuation used in $C_{Op}$ is intended to emphasise this view. Thus, in particular, the concedes relation allows change of output signature and mingling of output and state. More importantly, it admits weakening of the retrieve relation in after-states by being a disjunct to it, and it is especially this aspect which allows the expression of non-refinement like behaviour. The added flexibility provided by the within and concedes relations therefore introduces the possibility of relating models which cannot be related by refinement.

Let us return to the example in Section 2 and see how the proposed development step can be captured formally by the retrenchment mechanism. We have $\mathsf{Ops}_A = \mathsf{Ops}_C = \{Init, AddEl\}$, $\mathsf{U} = \mathbb{P}(\mathsf{NAT})$, $\mathsf{W} = \mathsf{iseq}(\mathsf{NAT})$, $\mathsf{I}_{AddEl} = \mathsf{K}_{AddEl} = \mathsf{NAT}$, $\mathsf{O}_{AddEl} = \emptyset$, and $\mathsf{Q}_{AddEl}$ will depend on how we define the concrete *AddEl* operation. In the simple case in which $AddEl_C$ performs a *skip* when the length of the sequence is at maximum, $\mathsf{Q}_{AddEl} = \emptyset$, otherwise it contains an overflow message; we concentrate on the former.

To describe the *skip* case fully, we need to give the within and concedes relations for *AddEl*($i$). Some examples follow, the first three in order of increasing strength:

1. $P_1 = \mathit{true}\ ,\ C_1 = (i \notin u \Rightarrow w' = w)$
2. $P_2 = \mathit{true}\ ,\ C_2 = (i \notin u \wedge w' = w)$
3. $P_3 = \mathit{true}\ ,\ C_3 = (|u| = 10 \wedge i \notin u \wedge w' = w)$
4. $P_4 = (|u| \leq 10)\ ,\ C_4 = (i \notin u \wedge w' = w)$
5. $P_5 = (|u| < 10 \vee (|u| = 10 \wedge i \in u))\ ,\ C_5 = \mathit{false}$

Notice how examples 4 and 5 use the within clause to exclude certain situations from consideration. In example 5 for instance, the antecedent of the Op PO will be false when $|u| = 10 \wedge i \notin u$, and thus the PO will hold trivially, permitting us to define different behaviour for the concrete system: it can do a *skip* and not add an eleventh element to the sequence. In the first three examples, the concedes clause is used to allow the concrete system to exhibit different behaviour. All three clauses will be true if the concrete system performs a *skip* when the eleventh element is added in the abstract system. Note also that example 5 describes fewer corresponding pairs of steps than the others, since the cases for which $|u| = 10 \wedge i \notin u$ are not in the scope of the relationship between the two models. In fact the triviality of the concedes relation here means that example

5 is describing a particular kind of refinement, and shows that by suitably restricting one's focus a refinement can often be found lurking inside a retrenchment. This point of view is explored in [6, 7].

Retrenchment is not the only approach in the literature which attempts to address the restrictive nature of refinement. For example, work in [9, 19] extends refinement by permitting a change in I/O signature across a development step. Other proposals include Liu's notion of evolution [13]. Development steps which cannot be described by refinement, are incorporated by demanding that the pre- and post- conditions of the abstract specification in the development step be semantically equivalent to subformulae of the pre- and post- conditions at the more concrete level. Now, since $Q$ is a subformula of $(P \wedge Q) \vee (P \wedge \neg Q)$, and the latter is equivalent to $P$, we can go from any $Q$ to any $P$ and so can relate any two models. Evolution, like retrenchment, is a weakening of refinement. This too is true of Smith's approach which he calls realisation [16, 17]. In [16] Smith voices concerns similar to our own. A development route which for pragmatic or methodological reasons starts out with an idealised abstract model, often cannot relate all the intermediate steps using refinement. Smith defines realisation rules which allow a specification to be transformed to one with different functionality in such a way that properties of the resulting specification can be derived from those of the transformed one.

# 4 I/O-Filtered Refinements

Here we define a particular type of refinement, I/O-filtered refinement, which allows for the easier integration of retrenchment and refinement steps. I/O-filtered refinement allows for a change in I/O signature thus reflecting a feature available in retrenchment but prohibited in classical refinement.

We will use the abstract and concrete systems *Abs* and *Conc* set up earlier. For an I/O-filtered refinement $\mathsf{Ops}_\mathrm{A} = \mathsf{Ops}_\mathrm{C}$, and the relationship between the two levels *Abs* and *Conc* is specified by the retrieve relation $G(u, w)$, and for each $Op \in \mathsf{Ops}_\mathrm{A}$, a within relation $R_{Op}(i, k)$ and a nevertheless relation $V_{Op}(o, q)$. All three relations are constrained to be total on their first components and surjective on their second; i.e. they are relations $S$ satisfying firstly $\forall a \exists b \bullet S(a, b)$ and secondly $\forall b \exists a \bullet S(a, b)$. Notice that unlike in retrenchment, these relations set apart the states, inputs and outputs. The Init PO for I/O-filtered refinement is the same as (3.1), while the Op PO has the form:

$$\begin{aligned} &G(u, w) \wedge R_{Op}(i, k) \wedge stp_{Op_\mathrm{C}}(w, k, w', q) \Rightarrow \\ &\quad (\exists u', o \bullet stp_{Op_\mathrm{A}}(u, i, u', o) \wedge G(u', w') \wedge V_{Op}(o, q)) \end{aligned} \tag{4.1}$$

Observe that the structure of this PO corresponds more closely to that found in retrenchment, with operation inputs being dealt with in the antecedent while outputs appear in the consequent. However, note that in contrast to the retrenchment case, the relation involving outputs appears as a conjunct.

# 5 Minimally Abstract Retrenchments

In this section we tackle the problem of decomposing the retrenchment from *Abs* to *Conc* into a refinement followed by a retrenchment. The objective is to engineer a system *Univ*, which is at the level of abstraction of the concrete model, but is refinable from the abstract one. That *Univ* is indeed at the desired level is supported by showing that, within a class of systems, *Univ* has appropriate universal properties. Specifically, that for any system *Xtra* in the class achieving a similar factorisation, there is an I/O-filtered refinement from *Xtra* to *Univ*. We shall also show that for any other system *Univ** which has the same properties as *Univ*, *Univ* and *Univ** are mutually I/O-filtered interrefinable. This interrefinability demonstrates an equivalence between the two systems. These ideas are made precise in Theorem 5.1 below.

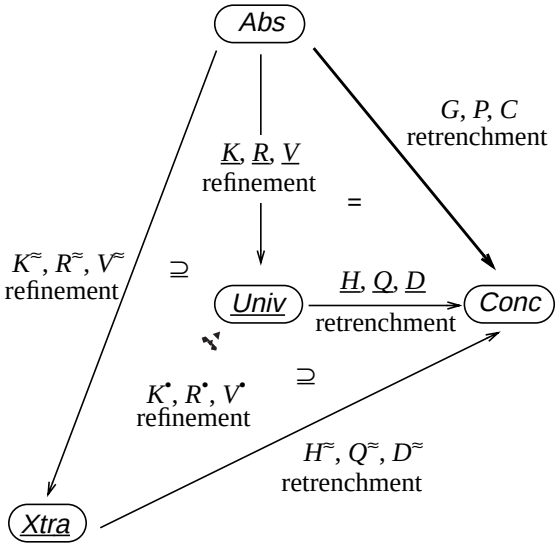


Figure 5.1: Canonical factorisation of the retrenchment from *Abs* to *Conc* into a refinement followed by a retrenchment.

**Theorem 5.1** Let there be a retrenchment as above from *Abs* to *Conc*. Then (see Figure 5.1):

(1) There is a universal system *Univ*, which satisfies properties (U1) to (U5) below, such that there is an I/O-filtered refinement from *Abs* to *Univ* and a retrenchment from *Univ* to *Conc* whose composition is the given retrenchment.

(2) Whenever there is a system *Xtra* which satisfies (X1) to (X5) below, and there is an I/O-filtered refinement from *Abs* to *Xtra* and a retrenchment from *Xtra* to *Conc* whose composition is the given retrenchment, then there is an I/O-filtered

refinement from *Xtra* to *Univ* such that $\underline{K} \Rightarrow K^{\sim};K^{\bullet}$, $\underline{R} \Rightarrow R^{\sim};R^{\bullet}$, $\underline{V} \Rightarrow V^{\sim};V^{\bullet}$, and $H^{\sim} \Rightarrow K^{\bullet};\underline{H}$, $Q^{\sim} \Rightarrow (K^{\bullet} \wedge R^{\bullet});\underline{Q}$, $D^{\sim} \Rightarrow (K^{\bullet\prime} \wedge V^{\bullet} \wedge R^{\bullet} \wedge K^{\bullet});\underline{D}$.

(3) Whenever a system *Univ** has properties (1) and (2) above of *Univ*, then the transitions of *Univ* and *Univ** are mutually I/O-filtered interrefinable.

To prove part (1) of the theorem, we begin by describing the construction of *Univ*. The operation names set of *Univ* is $\mathsf{Ops}_{\underline{U}}$ with elements $Op_{\underline{U}}$. The state space is $\underline{\mathsf{V}}$ with elements $\underline{v}$, inputs are $\underline{j} \in \underline{\mathsf{J}}$ and outputs $\underline{p} \in \underline{\mathsf{P}}$. These are all assembled from the *Abs* and *Conc* systems as follows.

Firstly $\mathsf{Ops}_{\underline{U}} = \mathsf{Ops}_{A}$ as *Univ* is refinable from *Abs*. The spaces are $\underline{\mathsf{V}} = \mathsf{U} \times \mathsf{W}$, $\underline{\mathsf{J}} = \mathsf{I} \times \mathsf{K}$ and $\underline{\mathsf{P}} = \mathsf{O} \times \mathsf{Q}$. We can now give the transitions of *Univ*. For each operation $Op_{\underline{U}}$ a typical transition is $\underline{v}$ -($\underline{j}$, $Op_{\underline{U}}$, $\underline{p}$)-› $\underline{v}'$ or more explicitly:

$$(u, w) \text{-}((i, k), Op_{\underline{U}}, (o, q))\text{-›} (u', w') \tag{5.1}$$

iff $u, i, u', o, w, k, w', q$ satisfy:

$$G(u, w) \wedge P_{Op}(i, k, u, w) \wedge stp_{Op_A}(u, i, u', o) \wedge stp_{Op_C}(w, k, w', q) \wedge (G(u', w') \vee C_{Op}(u', w', o, q; i, k, u, w)) \tag{5.2}$$

Finally, the initialisation operation $Init_{\underline{U}}(\underline{v}')$ assigns any value $(u', w')$ to $\underline{v}'$ such that $Init_A(u') \wedge Init_C(w') \wedge G(u', w')$ holds.

Next, we need to set up the I/O-filtered refinement from *Abs* to *Univ* and the retrenchment from *Univ* to *Conc*. This is done by defining the constituent relations and proving that the relevant POs are satisfied. We will then show that the composition of this I/O-filtered refinement and new retrenchment gives the original retrenchment.

The data for the I/O-filtered refinement consists of the retrieve, within and nevertheless relations. The retrieve relation is:

$$\underline{K}(u, \underline{v}) \;=\; (\underline{v} = (u, w)) \tag{5.3}$$

For an operation *Op* the within relation is:

$$\underline{R}_{Op}(i, \underline{j}) \;=\; (\underline{j} = (i, k)) \tag{5.4}$$

and the nevertheless relation is:

$$\underline{V}_{Op}(o, \underline{p}) \;=\; (\underline{p} = (o, q)) \tag{5.5}$$

The constituent relations of an I/O-filtered refinement must be total and surjective and it is evident that these relations satisfy this requirement.

We now need to show that the refinement POs hold for the above definitions. The Init PO is

$$Init_{\underline{U}}(\underline{v}') \;\Rightarrow\; (\exists\, u' \bullet Init_A(u') \wedge \underline{K}(u', \underline{v}')) \tag{5.6}$$

Assuming the antecedent we have $Init_{\underline{U}}(\underline{v}')$ which, by definition, assigns a value $(u', w')$ to $\underline{v}'$ such that $Init_A(u')$ holds. For this assignment $\underline{K}(u', \underline{v}')$ also holds, and so for any initial $\underline{v}'$ there is a $u'$ as required.

For the Op PO we have to show that:

$$\underline{K}(u, \underline{v}) \wedge \underline{R}_{Op}(i, \underline{j}) \wedge stp_{Op_{\underline{U}}}(\underline{v}, \underline{j}, \underline{v}', \underline{p}) \Rightarrow$$
$$(\exists\, u', o \bullet stp_{Op_A}(u, i, u', o) \wedge \underline{K}(u', \underline{v}') \wedge \underline{V}_{Op}(o, \underline{p})) \quad (5.7)$$

Let us assume the antecedents. Thus we have $stp_{Op_{\underline{U}}}(\underline{v}, \underline{j}, \underline{v}', \underline{p})$, the definition of which gives, amongst other things, $stp_{Op_A}(u, i, u', o)$ where $\underline{v}' = (u', w')$ and $\underline{p} = (o, q)$. For these values $\underline{K}(u', \underline{v}')$ and $\underline{V}_{Op}(o, \underline{p})$ both hold and therefore the consequent holds. We are done.

The retrenchment is defined by the retrieve, within and concedes relations. The retrieve relation is:

$$\underline{H}(\underline{v}, w) \;=\; (\underline{v} = (u, w) \wedge G(u, w)) \quad (5.8)$$

For an operation $Op$ the within relation is:

$$\underline{Q}_{Op}(\underline{j}, k, \underline{v}, w) \;=\; (\underline{j} = (i, k) \wedge \underline{v} = (u, w) \wedge P_{Op}(i, k, u, w)) \quad (5.9)$$

and the concedes relation is defined as:

$$\underline{D}_{Op}(\underline{v}', w', \underline{p}, q; \underline{j}, k, \underline{v}, w) \;=\; (\underline{v}' = (u', w') \wedge \underline{p} = (o, q) \wedge \underline{j} = (i, k) \wedge$$
$$\underline{v} = (u, w) \wedge C_{Op}(u', w', o, q; i, k, u, w)) \quad (5.10)$$

We now turn to the matter of showing that the retrenchment POs are valid. This time the Init PO is:

$$Init_C(w') \;\Rightarrow\; (\exists\, \underline{v}' \bullet Init_{\underline{U}}(\underline{v}') \wedge \underline{H}(\underline{v}', w')) \quad (5.11)$$

We assume $Init_C(w')$. For this initial $w'$, Init PO (3.1) of the original retrenchment says that there is an initial $u'$ such that $Init_A(u')$ and $G(u', w')$ hold. By combining this $u'$ with the $w'$, we get a $\underline{v}'$ for which $Init_{\underline{U}}(\underline{v}')$ holds. Then, using (5.8), we obtain $\underline{H}(\underline{v}', w')$. This completes the proof.

For the operation PO we have to show:

$$\underline{H}(\underline{v}, w) \wedge \underline{Q}_{Op}(\underline{j}, k, \underline{v}, w) \wedge stp_{Op_C}(w, k, w', q) \Rightarrow$$
$$(\exists\, \underline{v}', \underline{p} \bullet stp_{Op_{\underline{U}}}(\underline{v}, \underline{j}, \underline{v}', \underline{p}) \wedge (\underline{H}(\underline{v}', w') \vee \underline{D}_{Op}(\underline{v}', w', \underline{p}, q; \underline{j}, k, \underline{v}, w))) \quad (5.12)$$

We assume the antecedents. Using the definitions of $\underline{H}(\underline{v}, w)$ and $\underline{Q}_{Op}(\underline{j}, k, \underline{v}, w)$ we obtain $G(u, w)$ and $P_{Op}(i, k, u, w)$ where $\underline{v} = (u, w)$ and $\underline{j} = (i, k)$. These, and $stp_{Op_C}(w, k, w', q)$, make up the antecedents of the *Abs* to *Conc* retrenchment Op PO (3.2), so we know there are $u'$, $o$ such that $stp_{Op_A}(u, i, u', o) \wedge (G(u', w') \vee C_{Op}(u', w', o, q; i, k, u, w))$ holds. We now have all the pieces we need to use (5.2) to get $stp_{Op_{\underline{U}}}(\underline{v}, \underline{j}, \underline{v}', \underline{p})$, with $\underline{v}' = (u', w')$ and $\underline{p} = (o, q)$.

We are left with showing $\underline{H}(\underline{v}', w') \vee \underline{D}_{Op}(\underline{v}', w', \underline{p}, q; \underline{j}, k, \underline{v}, w)$. Now, $G(u', w') \vee C_{Op}(u', w', o, q; i, k, u, w)$ is true. So first suppose $G(u', w')$ holds. Then since $\underline{v}' = (u', w')$, (5.8) says $\underline{H}(\underline{v}', w')$ is true and so $\underline{H}(\underline{v}', w') \vee \underline{D}_{Op}(\underline{v}', w', \underline{p}, q; \underline{j}, k, \underline{v}, w)$ must also be true. If on the other hand $C_{Op}(u', w', o, q; i, k, u, w)$ is true, then as $\underline{v}' = (u', w')$, $\underline{p} = (o, q)$, $\underline{j} = (i, k)$ and $\underline{v} = (u, w)$, (5.10) gives $\underline{D}_{Op}(\underline{v}', w', \underline{p}, q; \underline{j}, k, \underline{v}, w)$ from which we once more obtain $\underline{H}(\underline{v}', w') \vee \underline{D}_{Op}(\underline{v}', w', \underline{p}, q; \underline{j}, k, \underline{v}, w)$. This completes the proof.

Finally we need to define the composition of the I/O-filtered refinement and retrenchment just constructed. We define the composition to be a retrenchment for which the component relations are constructed as follows. The retrieve relation is:

$$G(u, w) \;=\; (\underline{K};\underline{H})(u, w) \;=\; (\exists\, \underline{v} \bullet \underline{K}(u, \underline{v}) \wedge \underline{H}(\underline{v}, w)) \tag{5.13}$$

The within relation has the form:

$$\begin{aligned} &P_{Op}(i, k, u, w) \;=\; ((\underline{K} \wedge \underline{R}_{Op});\underline{Q}_{Op})(i, k, u, w) \;= \\ &\quad (\exists\, \underline{v}, \underline{j} \bullet \underline{K}(u, \underline{v}) \wedge \underline{R}_{Op}(i, \underline{j}) \wedge \underline{Q}_{Op}(\underline{j}, k, \underline{v}, w)) \end{aligned} \tag{5.14}$$

and the concedes relation is defined as:

$$\begin{aligned} &C_{Op}(u', w', o, q; i, k, u, w) \;= \\ &\quad ((\underline{K}' \wedge \underline{V}_{Op} \wedge \underline{R}_{Op} \wedge \underline{K});\underline{D}_{Op})(u', w', o, q; i, k, u, w) \;= \\ &\qquad (\exists\, \underline{v}', \underline{p}, \underline{j}, \underline{v} \bullet \underline{K}(u', \underline{v}') \wedge \underline{V}_{Op}(o, \underline{p}) \wedge \underline{R}_{Op}(i, \underline{j}) \wedge \underline{K}(u, \underline{v}) \wedge \\ &\qquad \underline{D}_{Op}(\underline{v}', w', \underline{p}, q; \underline{j}, k, \underline{v}, w)) \end{aligned} \tag{5.15}$$

It is easy to show that these definitions do indeed recover the relations of the original retrenchment. Consider the definition for $G(u, w)$. (5.8) and (5.3) fix $\underline{v}$ to $(u, w)$ for which (5.8) holds only if $G$ holds. Similar arguments apply to the equalities for $P$ and $C$.

We now state and prove properties (U1) to (U5) of *Univ* mentioned in Theorem 5.1. Thus:

$$G(u, w) \wedge \underline{K}(u, (u, w)) \;\Rightarrow\; \underline{H}((u, w), w) \tag{U1}$$

$$P_{Op}(i, k, u, w) \wedge \underline{K}(u, (u, w)) \wedge \underline{R}_{Op}(i, (i, k)) \;\Rightarrow\; \underline{Q}_{Op}((i, k), k, (u, w), w) \tag{U2}$$

$$\begin{aligned} &C_{Op}(u', w', o, q; i, k, u, w) \wedge \underline{K}(u', (u', w')) \wedge \underline{V}_{Op}(o, (o, q)) \wedge \\ &\underline{R}_{Op}(i, (i, k)) \wedge \underline{K}(u, (u, w)) \;\Rightarrow \\ &\quad \underline{D}_{Op}((u', w'), w', (o, q), q; (i, k), k, (u, w), w) \end{aligned} \tag{U3}$$

$$\begin{aligned} &(G(u', w') \vee C_{Op}(u', w', o, q; i, k, u, w)) \wedge \\ &(\underline{H}((u', w'), w') \vee \underline{D}_{Op}((u', w'), w', (o, q), q; (i, k), k, (u, w), w)) \;\Rightarrow \\ &\quad \underline{K}(u', (u', w')) \end{aligned} \tag{U4}$$

$$\begin{aligned} &stp_{Op_C}(w, k, w', q) \wedge G(u, w) \wedge P_{Op}(i, k, u, w) \wedge \\ &stp_{Op_A}(u, i, u', o) \wedge (G(u', w') \vee C_{Op}(u', w', o, q; i, k, u, w)) \wedge \\ &\underline{H}((u, w), w) \wedge \underline{Q}_{Op}((i, k), k, (u, w), w) \wedge stp_{Op_{\underline{U}}}((u, w), (i, k), (u', w'), (o, q)) \wedge \\ &(\underline{H}((u', w'), w') \vee \underline{D}_{Op}((u', w'), w', (o, q), q; (i, k), k, (u, w), w)) \;\Rightarrow \\ &\quad \underline{V}_{Op}(o, (o, q)) \end{aligned} \tag{U5}$$

To demonstrate that these properties hold is easy. (U1) to (U3) are an immediate consequence of (5.8) to (5.10) respectively. (U4) and (U5) are both trivially true since $\underline{K}(u', (u', w'))$ and $\underline{V}_{Op}(o, (o, q))$ hold by definition. It will become clear why these properties are of interest later. This completes the proof for part (1) of the theorem.

We move on to part (2). Assume an I/O-filtered refinement from *Abs* to *Xtra* given by retrieve relation $K^{\sim}$, within relations $R^{\sim}_{Op}$, and nevertheless relations $V^{\sim}_{Op}$; and a retrenchment from *Xtra* to *Conc* given by retrieve relation $H^{\sim}$, within relations $Q^{\sim}_{Op}$, and concedes relations $D^{\sim}_{Op}$. Let the state, input and output spaces of *Xtra* be given by $v^{\sim} \in \mathsf{V}^{\sim}, j^{\sim} \in \mathsf{J}^{\sim}, p^{\sim} \in \mathsf{P}^{\sim}$. Let $Init_{\underline{X}}$ and $stp_{Op_{\underline{X}}}$ be the initialisation and step predicates for *Xtra*. Lastly, let *Xtra* have properties (X1) to (X5):

$$G(u, w) \wedge K^{\sim}(u, v^{\sim}) \;\Rightarrow\; H^{\sim}(v^{\sim}, w) \tag{X1}$$

$$P_{Op}(i, k, u, w) \wedge K^{\sim}(u, v^{\sim}) \wedge R^{\sim}_{Op}(i, j^{\sim}) \;\Rightarrow\; Q^{\sim}_{Op}(j^{\sim}, k, v^{\sim}, w) \tag{X2}$$

$$\begin{array}{l} C_{Op}(u', w', o, q; i, k, u, w) \wedge K^{\sim}(u', v^{\sim\prime}) \wedge V^{\sim}_{Op}(o, p^{\sim}) \wedge \\ R^{\sim}_{Op}(i, j^{\sim}) \wedge K^{\sim}(u, v^{\sim}) \;\Rightarrow \\ \quad D^{\sim}_{Op}(v^{\sim\prime}, w', p^{\sim}, q; j^{\sim}, k, v^{\sim}, w) \end{array} \tag{X3}$$

$$\begin{array}{l} (G(u', w') \vee C_{Op}(u', w', o, q; i, k, u, w)) \wedge \\ (H^{\sim}(v^{\sim\prime}, w') \vee D^{\sim}_{Op}(v^{\sim\prime}, w', p^{\sim}, q; j^{\sim}, k, v^{\sim}, w)) \;\Rightarrow \\ \quad K^{\sim}(u', v^{\sim\prime}) \end{array} \tag{X4}$$

$$\begin{array}{l} stp_{Op_C}(w, k, w', q) \wedge G(u, w) \wedge P_{Op}(i, k, u, w) \wedge \\ stp_{Op_A}(u, i, u', o) \wedge (G(u', w') \vee C_{Op}(u', w', o, q; i, k, u, w)) \wedge \\ H^{\sim}(v^{\sim}, w) \wedge Q^{\sim}_{Op}(j^{\sim}, k, v^{\sim}, w) \wedge stp_{Op_{\underline{X}}}(v^{\sim}, j^{\sim}, v^{\sim\prime}, p^{\sim}) \wedge \\ (H^{\sim}(v^{\sim\prime}, w') \vee D^{\sim}_{Op}(v^{\sim\prime}, w', p^{\sim}, q; j^{\sim}, k, v^{\sim}, w)) \;\Rightarrow \\ \quad V^{\sim}_{Op}(o, p^{\sim}) \end{array} \tag{X5}$$

To prove part (2), we must show that there is an I/O-filtered refinement from *Xtra* to *Univ*. To this end we now define relations $K^{\bullet}$, $R^{\bullet}_{Op}$, $V^{\bullet}_{Op}$, indicate that they satisfy the stated inclusions, and prove that they are the retrieve, within and nevertheless relations of the desired refinement. Thus:

$$K^{\bullet}(v^{\sim}, \underline{v}) \;=\; (\exists\, u \bullet \underline{K}(u, \underline{v}) \wedge K^{\sim}(u, v^{\sim})) \tag{5.16}$$

$$R^{\bullet}_{Op}(j^{\sim}, \underline{j}) \;=\; (\exists\, i \bullet \underline{R}_{Op}(i, \underline{j}) \wedge R^{\sim}_{Op}(i, j^{\sim})) \tag{5.17}$$

$$V^{\bullet}_{Op}(p^{\sim}, \underline{p}) \;=\; (\exists\, o \bullet \underline{V}_{Op}(o, \underline{p}) \wedge V^{\sim}_{Op}(o, p^{\sim})) \tag{5.18}$$

Observe that $K^{\bullet}$ is total and surjective since both $\underline{K}$ and $K^{\sim}$ are total and surjective. Similarly, $R^{\bullet}_{Op}$ and $V^{\bullet}_{Op}$ are also total and surjective. Given the above definitions, establishing that $\underline{K} \Rightarrow K^{\sim};K^{\bullet}$, $\underline{R} \Rightarrow R^{\sim};R^{\bullet}$, $\underline{V} \Rightarrow V^{\sim};V^{\bullet}$, and $H^{\sim} \Rightarrow K^{\bullet};\underline{H}$, $Q^{\sim} \Rightarrow (K^{\bullet} \wedge R^{\bullet});\underline{Q}$, $D^{\sim} \Rightarrow (K^{\bullet\prime} \wedge V^{\bullet} \wedge R^{\bullet} \wedge K^{\bullet});\underline{D}$ all hold is straightforward and details have therefore been omitted.

We move on to the initialisation PO. We need to show

$$Init_{\underline{U}}(\underline{v}') \;\Rightarrow\; (\exists\, v^{\sim\prime} \bullet Init_{\underline{X}}(v^{\sim\prime}) \wedge K^{\bullet}(v^{\sim\prime}, \underline{v}')) \tag{5.19}$$

We assume the antecedent, and so from the definition of $Init_{\underline{U}}(\underline{v}')$ we know $Init_{C}(w')$ and $G(u', w')$ both hold with $\underline{v}' = (u', w')$. Now, as $Init_{C}(w')$ is true, the PO for the retrenchment from *Xtra* to *Conc*,

$$Init_{C}(w') \Rightarrow (\exists\, v^{\sim\prime} \bullet Init_{\underline{X}}(v^{\sim\prime}) \wedge H^{\sim}(v^{\sim\prime}, w')) \tag{5.20}$$

gives $Init_{\underline{X}}(v^{\sim\prime})$, this being one of the desired consequents, and also $H^{\sim}(v^{\sim\prime}, w')$. This, together with the $G(u', w')$ above, gives $K^{\sim}(u', v^{\sim\prime})$ by (X4), and thus since $\underline{K}(u', \underline{v}')$ obviously holds, (5.16) says the other consequent $K^{\bullet}(v^{\sim\prime}, \underline{v}')$ must also be true. We are done.

All that remains is to show the operation PO:

$$\begin{aligned} &K^{\bullet}(v^{\sim}, \underline{v}) \wedge R^{\bullet}_{Op}(j^{\sim}, \underline{j}) \wedge stp_{Op_{\underline{U}}}(\underline{v}, \underline{j}, \underline{v}', \underline{p}) \Rightarrow \\ &\quad (\exists\, v^{\sim\prime}, p^{\sim} \bullet stp_{Op_{\underline{X}}}(v^{\sim}, j^{\sim}, v^{\sim\prime}, p^{\sim}) \wedge K^{\bullet}(v^{\sim\prime}, \underline{v}') \wedge V^{\bullet}_{Op}(p^{\sim}, \underline{p})) \end{aligned} \tag{5.21}$$

is satisfied. As usual we assume the antecedents. Therefore $stp_{Op_{\underline{U}}}(\underline{v}, \underline{j}, \underline{v}', \underline{p})$ is true and so (5.2) says $G(u, w)$, $P_{Op}(i, k, u, w)$ and $stp_{Op_{C}}(w, k, w', q)$, where $\underline{v} = (u, w)$, $\underline{j} = (i, k)$, $\underline{v}' = (u', w')$ and $\underline{p} = (o, q)$, are true as well. The antecedent $K^{\bullet}(v^{\sim}, \underline{v})$ gives $K^{\sim}(u, v^{\sim})$, and since we have $G(u, w)$, (X1) implies that $H^{\sim}(v^{\sim}, w)$ holds. The remaining antecedent $R^{\bullet}_{Op}(j^{\sim}, \underline{j})$ gives $R^{\sim}_{Op}(i, j^{\sim})$, which together with the $P_{Op}(i, k, u, w)$ and the $K^{\sim}(u, v^{\sim})$ we already have furnish $Q^{\sim}_{Op}(j^{\sim}, k, v^{\sim}, w)$ via (X2).

So we have established that $H^{\sim}(v^{\sim}, w)$ and $Q^{\sim}_{Op}(j^{\sim}, k, v^{\sim}, w)$ both hold. Combining these with the $stp_{Op_{C}}(w, k, w', q)$ mentioned above, gives the antecedents of the *Xtra* to *Conc* retrenchment:

$$\begin{aligned} &H^{\sim}(v^{\sim}, w) \wedge Q^{\sim}_{Op}(j^{\sim}, k, v^{\sim}, w) \wedge stp_{Op_{C}}(w, k, w', q) \Rightarrow \\ &\quad (\exists\, v^{\sim\prime}, p^{\sim} \bullet stp_{Op_{\underline{X}}}(v^{\sim}, j^{\sim}, v^{\sim\prime}, p^{\sim}) \wedge (H^{\sim}(v^{\sim\prime}, w') \vee \\ &\qquad D^{\sim}_{Op}(v^{\sim\prime}, w', p^{\sim}, q; j^{\sim}, k, v^{\sim}, w))) \end{aligned} \tag{5.22}$$

From this we obtain one of the consequents we want, namely $stp_{Op_{\underline{X}}}(v^{\sim}, j^{\sim}, v^{\sim\prime}, p^{\sim})$, and in addition $H^{\sim}(v^{\sim\prime}, w') \vee D^{\sim}_{Op}(v^{\sim\prime}, w', p^{\sim}, q; j^{\sim}, k, v^{\sim}, w)$. Next, we notice that the definition of $stp_{Op_{\underline{U}}}(\underline{v}, \underline{j}, \underline{v}', \underline{p})$ also gives $G(u', w') \vee C_{Op}(u', w', o, q; i, k, u, w)$. Therefore, because we have $H^{\sim\prime} \vee D^{\sim}_{Op}$, we can use (X4) to conclude that $K^{\sim}(u', v^{\sim\prime})$ is true. Then, as $\underline{K}(u', \underline{v}')$ also holds, we can get $K^{\bullet}(v^{\sim\prime}, \underline{v}')$ by applying (5.16).

Our last task is to demonstrate that the final consequent, $V^{\bullet}_{Op}(p^{\sim}, \underline{p})$, holds. From the preceding steps we already know that $G(u, w) \wedge P_{Op}(i, k, u, w)$, $stp_{Op_{C}}(w, k, w', q)$, $G(u', w') \vee C_{Op}(u', w', o, q; i, k, u, w)$, $H^{\sim}(v^{\sim}, w) \wedge Q^{\sim}_{Op}(j^{\sim}, k, v^{\sim}, w)$, $stp_{Op_{\underline{X}}}(v^{\sim}, j^{\sim}, v^{\sim\prime}, p^{\sim})$ and $H^{\sim}(v^{\sim\prime}, w') \vee D^{\sim}_{Op}(v^{\sim\prime}, w', p^{\sim}, q; j^{\sim}, k, v^{\sim}, w)$ all hold. Recalling that $stp_{Op_{\underline{U}}}(\underline{v}, \underline{j}, \underline{v}', \underline{p})$ also gives $stp_{Op_{A}}(u, i, u', o)$, provides the missing piece we need to be able to use (X5) and so get $V^{\sim}_{Op}(o, p^{\sim})$. From here it is a simple step to obtain $V^{\bullet}_{Op}(p^{\sim}, \underline{p})$, since (5.18) says that all we now require is $\underline{V}(o, \underline{p})$ which, because $\underline{p} = (o, q)$, definitely holds. We are done. We have now proved part (2) of the theorem.

It remains to prove part (3). Given that *Univ** has the same properties as *Univ*, and noting that properties (U1) to (U5) of *Univ* correspond to (X1) to (X5) of *Xtra*, means

there will be an I/O-filtered refinement from *Univ* to *Univ** and an I/O-filtered refinement from *Univ** to *Univ*. This completes the proof of Theorem 5.1. ☺

Note that the mutual interrefinability in part (3) is not isomorphism in the conventional set theoretical sense; the mutual interrefinability established here amounts to a much looser notion of equivalence of systems.

To conclude this section let us examine the structure of *Univ* in the context of our running example. The state space of the *Univ* is the Cartesian product of the state spaces of the abstract and concrete systems. Thus individual states will be ordered pairs of sets and sequences, e.g. ({2, 4, 6}, [2, 4, 6]} and ({9, 12}, [5]).

As for the transitions of the universal system, (5.2) tells us that these will consist of corresponding pairs of abstract and concrete transitions, i.e. those for which $G$, $P_{Op}$ and $G' \vee C_{Op}$ hold. Thus for nonboundary cases, *Univ* will undergo transitions which look like:

$$(\{2, 4, 6\}, [2, 4, 6]) \text{ -}((8, 8), \mathit{AddEl})\text{-}\rightarrow (\{2, 4, 6, 8\}, [2, 4, 6, 8])$$

For the boundary steps on the other hand (where the concrete operation performs a *skip*), instances look like:

$$(\{1..10\}, [1..10]) \text{ -}((11, 11), \mathit{AddEl})\text{-}\rightarrow (\{1..11\}, [1..10])$$

and we see clearly why the general product structure is needed for the universal state space; evidently [1..10] can never be a serialisation of {1..11}.

# 6 Idempotence

One of the aims in the preceding section was to construct *Univ* so that it was at the level of abstraction of *Conc* but refinable from *Abs*. We showed that for any other similar factorisation, the refinement component was refinable to *Univ*. In this section we provide further evidence indicating that *Univ* is at a low enough level. Specifically, we show that applying the universal construction to the *Univ* to *Conc* retrenchment itself, yields a system, *UUniv*, which in essence is no different from that of *Univ*.

Based on (5.1), which defines the transitions of the universal system, the transitions of *UUniv* look like $((u, w), w)$ -$(((i, k), k), Op_{\underline{U}U}, ((o, q), q))$-› $((u', w'), w')$ for which

$$\underline{H}((u, w), w) \wedge \underline{Q}_{Op}((i, k), k, (u, w), w) \wedge stp_{Op_{\underline{U}}}((u, w), (i, k), (u', w'), (o, q)) \wedge$$
$$stp_{Op_C}(w, k, w', q) \wedge (\underline{H}((u', w'), w') \vee \underline{D}_{Op}((u', w'), w', (o, q), q; (i, k), k, (u', w), w))$$

must hold. Substituting for $\underline{H}$, $\underline{Q}_{Op}$, $\underline{D}_{Op}$, $stp_{Op_{\underline{U}}}$ in the above gives (5.2) which defines the *Univ* transitions $(u, w)$ -$((i, k), Op_{\underline{U}}, (o, q))$-› $(u', w')$. Thus the set of *UUniv* transitions is isomorphic to those of *Univ* since having an extra copy of the concrete transition in *UUniv* is isomorphic to having just the one copy in the corresponding *Univ* transition.

Note that the refinements from *Abs* to *Univ* and *Univ* to *UUniv* will compose to give a refinement from *Abs* to *UUniv*. Thus by Theorem 5.1 *Univ* and *UUniv* are equivalent

and so at the same level of abstraction. All this supports our claim that *Univ* is at a low enough level.

# 7 Maximally Abstract Retrenchments

Here we reproduce some material from [3] in order to lay the necessary groundwork for subsequent sections. The concern in loc. cit. was to construct a universal system *Univ* which was at the level of *Abs* but was refinable to *Conc* (see Figure 7.1). The universality of the construction arises from the fact that there is a refinement from *Univ* to the

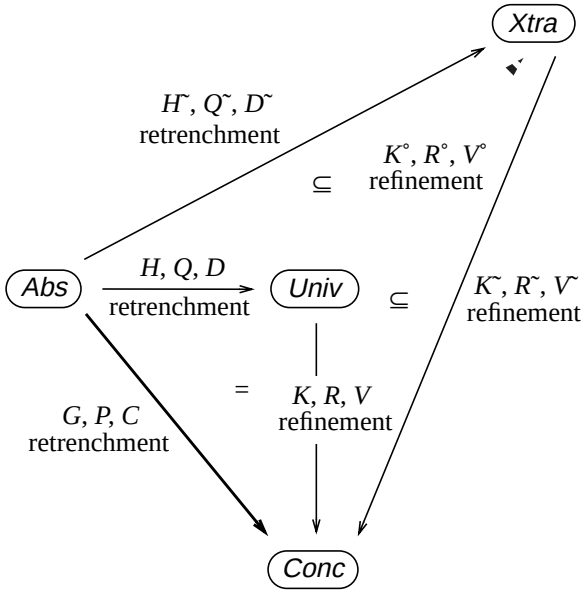


Figure 7.1: Canonical factorisation of the retrenchment from *Abs* to *Conc* into a retrenchment followed by a refinement.

retrenchment component of any similar factorisation. Theorem 7.1 below describes these ideas in more detail.

**Theorem 7.1** Let there be a retrenchment from the *Abs* to *Conc*. Then

(1) There is a universal system *Univ* such that there is a retrenchment from *Abs* to *Univ* and an I/O-filtered refinement from *Univ* to *Conc* whose composition is the given retrenchment.

(2) Whenever there is a system *Xtra* and a retrenchment from *Abs* to *Xtra* and an I/O-filtered refinement from *Xtra* to *Conc* whose composition is the given retrenchment, then there is an I/O-filtered refinement from *Univ* to the concrete core bound

transitions of *Xtra*[1]; such that $H^{\check{}} \Rightarrow H;K^{\circ}$, $Q^{\check{}} \Rightarrow Q;R^{\circ} \wedge K^{\circ}$, $D^{\check{}} \Rightarrow D;K^{\circ\prime} \wedge V^{\circ} \wedge R^{\circ} \wedge K^{\circ}$, and such that $K \Rightarrow K^{\circ};K^{\check{}}$, $R \Rightarrow R^{\circ};R^{\check{}}$, $V \Rightarrow V^{\circ};V^{\check{}}$.

(3) Whenever a system *Univ** has properties (1) and (2) above of *Univ*, then the concrete core bound transitions of *Univ* and *Univ** are mutually I/O-filtered interrefinable.

We now describe the structure of *Univ*. The operation names set of *Univ* is $\mathsf{Ops}_U$ with elements $Op_U$. The state space is $\mathsf{V}$ with elements $v$, inputs are $j \in \mathsf{J}$ and outputs $p \in \mathsf{P}$. These are all constructed from *Abs* and *Conc* as follows.

Firstly, $\mathsf{Ops}_U = \mathsf{Ops}_C$. The spaces are $\mathsf{V} = \mathsf{U} \times \mathsf{W}$, $\mathsf{J} = \mathsf{I} \times \mathsf{K}$ and $\mathsf{P} = \mathsf{O} \times \mathsf{Q}$. To give the transitions of *Univ* we observe that $\mathsf{Ops}_U$ decomposes as $\mathsf{Ops}_U = \mathsf{Ops}_A \cup (\mathsf{Ops}_U - \mathsf{Ops}_A)$.

For an operation $Op_U \in (\mathsf{Ops}_U - \mathsf{Ops}_A)$, we have a transition $v$ -$(j, Op_U, p)$-› $v'$ or more explicitly:

$$(u, w) \text{-}((i, k), Op_U, (o, q))\text{-›} (u', w') \tag{7.1}$$

for arbitrary values $u, i, o, u', w, k, q, w'$; so the non-$Op_A$ transitions of *Univ* form a universal relation.

For an $Op_A \in \mathsf{Ops}_A$, a transition is $v$ -$(j, Op_A, p)$-› $v'$ or:

$$(u, w) \text{-}((i, k), Op_A, (o, q))\text{-›} (u', w') \tag{7.2}$$

iff $u, i, o, u', w, k, q, w'$ satisfy:

$$\begin{aligned} &G(u, w) \wedge P_{Op}(i, k, u, w) \Rightarrow \\ &\quad stp_{Op_A}(u, i, u', o) \wedge (G(u', w') \vee C_{Op}(u', w', o, q; i, k, u, w)) \end{aligned} \tag{7.3}$$

Last, $Init_U(v')$ in *Univ* assigns any value $(u', w')$ to $v'$ such that $Init_A(u') \wedge G(u', w')$ holds.

Finally, we give the data for the retrenchment from *Abs* to *Univ* and the I/O-filtered refinement from *Univ* to *Conc*. The component relations of the retrenchment were defined as follows. The retrieve relation had the form:

$$H(u, v) \;=\; (v = (u, w) \wedge G(u, w)) \tag{7.4}$$

For each operation $Op$ the within relation was:

$$Q_{Op}(i, j, u, v) \;=\; (j = (i, k) \wedge v = (u, w) \wedge P_{Op}(i, k, u, w)) \tag{7.5}$$

while the concedes relation looked like:

$$\begin{aligned} &D_{Op}(u', v', o, p; i, j, u, v) \;= \\ &\quad (v' = (u', w') \wedge p = (o, q) \wedge j = (i, k) \wedge v = (u, w) \wedge \\ &\quad C_{Op}(u', w', o, q; i, k, u, w)) \end{aligned} \tag{7.6}$$

1. The concrete core bound transitions of *Xtra* (resp. *Univ*, *Univ**) are those transitions which are related by the *Abs* to *Xtra* (resp. *Univ*, *Univ**) retrenchment to transitions in *Abs*.

For the I/O-filtered refinement, the retrieve relation $K$, and for each $Op$, the within relation $R_{Op}$ and nevertheless relation $V_{Op}$, were defined as projection functions onto the second component:

$$K(v, w) \;=\; (v = (u, w)) \tag{7.7}$$

$$R_{Op}(j, k) \;=\; (j = (i, k)) \tag{7.8}$$

$$V_{Op}(p, q) \;=\; (p = (o, q)) \tag{7.9}$$

# 8 Requirements Validation

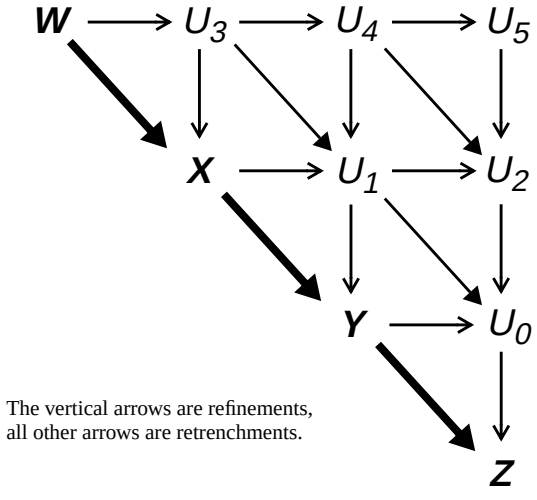


Figure 8.1: Requirements Validation. Lifting the system $Z$ to the level of abstraction of $W$.

The constructions discussed in this paper present an alternative approach for validating that all the requirements of a system have been taken into account. Let us suppose we have a development consisting of a series of retrenchment steps from a system $W$ to a system $Z$ as shown in Figure 8.1. Then it is possible to decompose the retrenchment from $Y$ to $Z$ and construct a system $U_0$ which is at the level of abstraction of $Y$ but expresses the detail in $Z$. Similarly we can construct new systems $U_1$ and $U_3$ which are at the level of $X$ and $W$ respectively. Now, the refinement from $U_1$ to $Y$ and the retrenchment from $Y$ to $U_0$ compose to give a retrenchment from $U_1$ to $U_0$. This new retrenchment can itself be factorised to give the system $U_2$ which will be at the level of $X$. Continuing this process will eventually result in the construction of $U_5$, which expresses the detail introduced in $Z$, but is itself at the abstract level of $W$. Since refinements compose, the POs of the single refinement from $U_5$ to $Z$ can be used to validate that all the requirements of the starting system $W$ have been fulfilled. In [3], the author argued that

although it is unlikely that developers would be disposed to carrying out requirements validation in this way, such a possibility admits the opportunity of being able to view the problem from a different perspective which is always beneficial.

# 9 The Other Diagonal

In this section we show that the systems *Univ* and *Univ* are related by an I/O-filtered refinement for the cases where $\mathsf{Ops}_{\mathrm{U}} = \mathsf{Ops}_{\underline{\mathrm{U}}}$ (see Figure 9.1). First we define the re-

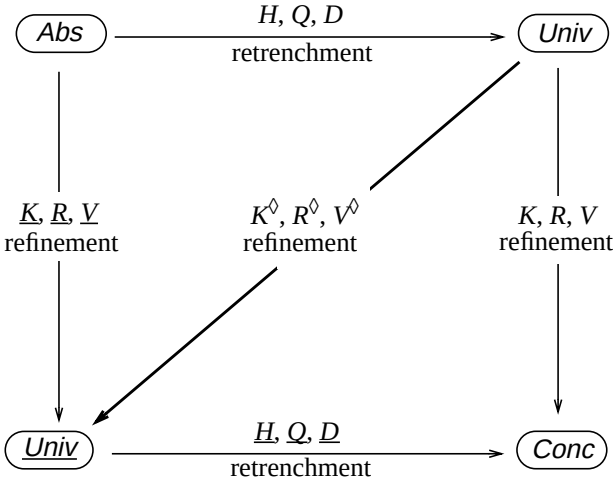

Figure 9.1: A refinement from *Univ* to *Univ*.

lations $K^{\Diamond}$, $R^{\Diamond}{}_{Op}$ and $V^{\Diamond}{}_{Op}$, and then show that they are the retrieve, within and nevertheless relations of the required refinement. Thus:

$$K^{\Diamond}(v, \underline{v}) \;=\; (v = \underline{v}) \tag{9.1}$$

$$R^{\Diamond}{}_{Op}(j, \underline{j}) \;=\; (j = \underline{j}) \tag{9.2}$$

$$V^{\Diamond}{}_{Op}(p, \underline{p}) \;=\; (p = \underline{p}) \tag{9.3}$$

and we see that they are all both total and surjective since each one is a relation between corresponding ordered pairs.

As usual we commence with the Init PO:

$$Init_{\underline{\mathrm{U}}}(\underline{v}') \;\Rightarrow\; (\exists\, v' \bullet Init_{\mathrm{U}}(v') \wedge K^{\Diamond}(v', \underline{v}')) \tag{9.4}$$

Suppose the antecedent holds. From the definition of $Init_{\underline{\mathrm{U}}}(\underline{v}')$ we know $Init_{\mathrm{A}}(u') \wedge G(u', w')$, where $\underline{v}' = (u', w')$, holds. Setting $v' = \underline{v}'$, it follows from the definition of $Init_{\mathrm{U}}(v')$ that it holds too. Lastly, for this assignment, from (9.1) we also know that $K^{\Diamond}(v', \underline{v}')$ is true. Hence for any initial $\underline{v}'$ there is a $v'$ as required.

Next, we focus on the Op PO, namely:

$$K^{\Diamond}(v, \underline{v}) \wedge R^{\Diamond}{}_{Op}(j, \underline{j}) \wedge stp_{Op_{\underline{U}}}(\underline{v}, \underline{j}, \underline{v}', \underline{p}) \Rightarrow$$
$$(\exists\, v', p \bullet stp_{Op_U}(v, j, v', p) \wedge K^{\Diamond}(v', \underline{v}') \wedge V^{\Diamond}{}_{Op}(p, \underline{p})) \tag{9.5}$$

Assume the antecedents. Knowing $stp_{Op_{\underline{U}}}(\underline{v}, \underline{j}, \underline{v}', \underline{p})$, (5.2) says that $G(u, w) \wedge P_{Op}(i, k, u, w)$ and $stp_{Op_A}(u, i, u', o)$ and $G(u', w') \vee C_{Op}(u', w', o, q; i, k, u, w)$ all hold, with $\underline{v}' = (u', w')$, $\underline{p} = (o, q)$, $\underline{j} = (i, k)$ and $\underline{v} = (u, w)$. Furthermore, from $K^{\Diamond}$ and (9.1) we get $v = \underline{v}$, while $R^{\Diamond}$ and (9.2) gives $j = \underline{j}$. Now let $v' = \underline{v}'$ and $p = \underline{p}$, then we obtain $stp_{Op_U}(v, j, v', p)$ via (7.3). Finally, for the given assignments, $K^{\Diamond}(v', \underline{v}')$ follows from (9.1) and $V^{\Diamond}{}_{Op}(p, \underline{p})$ from (9.3). Therefore there are $v', p$ for which the consequent of (9.5) is true. We are done. ☺

Notice that the direction of the refinement supports our assertion that ***Univ*** is more concrete in character than *Univ*. Moreover, there is no refinement in the other direction, since it is not possible to derive $stp_{Op_C}$, and thus $stp_{Op_{\underline{U}}}$, from the assumptions we would have in that case. This is exactly as it should be, for otherwise the systems would be equivalent and so could not be at different levels of abstraction. Also, the presence of $stp_{Op_C}(w, k, w', q)$ only in the definition of ***Univ*** transitions, can be regarded as a further expression of ***Univ***'s more concrete nature.

# 10 Termination

So far we have considered refinement steps which preserve partial correctness. To extend our discussion to the total correctness arena, we now assume that for each operation *Op* there is a termination predicate $trm_{Op}$ (in before-states and inputs), which holds for those steps for which termination is guaranteed. In the majority of refinement methodologies, the structure of the POs is such that $trm_{Op}$ for an abstract operation implies $trm_{Op}$ for the corresponding concrete operation. The upshot of this is that the inclusion of the termination aspects of operations ensures the preservation of properties between levels of abstraction. In retrenchment the direction of the $trm_{Op}$ dependency is reversed allowing a retrenchment step to only preserve chosen properties of the more abstract model. Returning to our original retrenchment from ***Abs*** to ***Conc***, we therefore find that the termination PO has the form:

$$G(u, w) \wedge P_{Op}(i, k, u, w) \wedge trm_{Op_C}(w, k) \Rightarrow trm_{Op_A}(u, i) \tag{10.1}$$

To include termination aspects in the construction given in Section 5, we augment the structure of ***Univ*** with the termination predicate:

$$trm_{Op_{\underline{U}}}(\underline{v}, \underline{j}) = (\exists\, u, i \bullet \underline{K}(u, \underline{v}) \wedge \underline{R}_{Op}(i, \underline{j})) \tag{10.2}$$

and this must satisfy the corresponding refinement and retrenchment termination POs:

$$\underline{K}(u, \underline{v}) \wedge \underline{R}_{Op}(i, \underline{j}) \wedge trm_{Op_A}(u, i) \Rightarrow trm_{Op_{\underline{U}}}(\underline{v}, \underline{j}) \tag{10.3}$$

$$\underline{H}(\underline{v}, w) \wedge \underline{Q}_{Op}(\underline{j}, k, \underline{v}, w) \wedge trm_{Op_C}(w, k) \Rightarrow trm_{Op_{\underline{U}}}(\underline{v}, \underline{j}) \tag{10.4}$$

That this is indeed the case is easy to establish. Taking (10.3) first and assuming the antecedents, we have a $u$ and an $i$ such that $\underline{K}(u, \underline{v}) \wedge \underline{R}_{Op}(i, \underline{j})$ holds. Therefore $trm_{Op_{\underline{U}}}(\underline{v}, \underline{j})$ holds and we are done. Proceeding on to (10.4), from the antecedents $\underline{H}(\underline{v}, w)$ and $\underline{Q}_{Op}(\underline{j}, k, \underline{v}, w)$, using (5.8) and (5.9), we know $\underline{v} = (u, w)$ and $\underline{j} = (i, k)$. For these values $\underline{K}(u, \underline{v})$ and $\underline{R}_{Op}(i, \underline{j})$ clearly hold, so we can apply (10.2) to get $trm_{Op_{\underline{U}}}(\underline{v}, \underline{j})$.

All that remains is to show that the termination PO for the refinement from $\underline{Xtra}$ to $\underline{Univ}$ holds. The PO says:

$$K^{\bullet}(v\tilde{}, \underline{v}) \wedge R^{\bullet}{}_{Op}(j\tilde{}, \underline{j}) \wedge trm_{Op_{\underline{X}}}(v\tilde{}, j\tilde{}) \;\Rightarrow\; trm_{Op_{\underline{U}}}(\underline{v}, \underline{j}) \tag{10.5}$$

Given the antecedents, $K^{\bullet}(v\tilde{}, \underline{v})$ and (5.16) gives $\underline{K}(u, \underline{v})$ with $\underline{v} = (u, w)$, while $R^{\bullet}{}_{Op}(j\tilde{}, \underline{j})$ and (5.17) provide $\underline{R}_{Op}(i, \underline{j})$ with $\underline{j} = (i, k)$. Since $\underline{K}$ and $\underline{R}_{Op}$ hold, (10.2) is satisfied, giving $trm_{Op_{\underline{U}}}(\underline{v}, \underline{j})$. We are done.

To complete the picture, we will also demonstrate that the termination PO for the refinement from $Univ$ to $\underline{Univ}$ holds. The PO in this case reads:

$$K^{\Diamond}(v, \underline{v}) \wedge R^{\Diamond}{}_{Op}(j, \underline{j}) \wedge trm_{Op_{\underline{U}}}(v, j) \;\Rightarrow\; trm_{Op_{\underline{U}}}(\underline{v}, \underline{j}) \tag{10.6}$$

Once again, to confirm this is straightforward. As usual we assume the antecedents. For the chosen values of $\underline{v}$ and $\underline{j}$, say $(u, w)$ and $(i, k)$, (5.3) and (5.4) say $\underline{K}(u, \underline{v})$ and $\underline{R}_{Op}(i, \underline{j})$ are true, which means $trm_{Op_{\underline{U}}}(\underline{v}, \underline{j})$ must also be true. ☺

Space restrictions prevent us from developing these results further, but we have laid the foundations from which a more detailed analysis of the termination aspects can be made. Finally, we take this opportunity to make a correction to the earlier work on maximally abstract retrenchments [3], and note that in Section 8 the termination PO for the refinement from $Univ$ to $Xtra$ can only be established by restricting the universality result to situations in which the relations $K\tilde{}$, $R\tilde{}_{Op}$ and $V\tilde{}_{Op}$ of the refinement from $Xtra$ to $Conc$ are functions. The main result presented in that paper is not affected by the introduction of this constraint.

# 11 Conclusion

This paper has shown how a retrenchment step can be factorised into a canonical I/O-filtered refinement to a universal system followed by a retrenchment. Within the class of systems whose properties we stated, the universal character of the aforementioned system arises from the fact that for all other similar factorisations whose intermediate system belongs to the same class, there is a refinement from the intermediate system to the universal model. The objective was for the universal system to be at the concrete level, and in support of this claim, in addition to the universality result, we showed that the construction of the universal system was idempotent.

Furthermore, given the maximal and minimal factorisations discussed in this paper, we proved in Section 9 that for a given retrenchment step, there was also an I/O-filtered refinement from the universal system at the abstract level to the universal system at the concrete one (if we assume $Conc$ has no additional operations, i.e. $\mathsf{Ops}_A = \mathsf{Ops}_C$). Fi-

nally, we briefly explored the effect of extending the partial correctness framework to a total correctness setting by including guaranteed termination predicates in our models.

The principal purpose of all this activity is to integrate retrenchment with refinement, and this has been achieved to the degree outlined in the preceding sections. It is important to demonstrate that refinement and retrenchment work well together, thus encouraging the joint use of the two techniques in the construction of complex industrial scale specifications. Moreover, the adoption of retrenchment into the developer's armoury will enable formality to be introduced earlier in the development hierarchy than is possible by the use of refinement alone.

# References


[1] Abrial J. R. *The B-Book: Assigning Programs to Meanings.* C.U.P., 1996.

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

[3] Banach R. Maximally Abstract Retrenchments. In *Proc. IEEE ICFEM-00*, pages 133-142. IEEE Computer Society Press, 2000.

[4] Banach R., Poppleton M. Retrenchment: An Engineering Variation on Refinement. In Bert (ed.), *Proc. B-98*, LNCS **1393**, 129-147. Springer 1998. See also UMCS Technical Report UMCS-99-3-2, `http://www.cs.man.ac.uk/cstechrep`

[5] Banach R., Poppleton M. Retrenchment and Punctured Simulation. In Araki, Galloway, Taguchi (eds.), *Proc. IFM-99*, pages 457-476. Springer 1999.

[6] Banach R., Poppleton M. Sharp Retrenchment, Modulated Refinement and Simulation. *Formal Aspects of Computing*, **11**, 498-540, 1999.

[7] Banach R., Poppleton M. Retrenchment, Refinement and Simulation. In Bowen, Dunne, Galloway, King (eds.), *Proc. ZB-00*, LNCS **1878**, 304-323. Springer, 2000.

[8] Banach R., Poppleton M. Engineering and Theoretical Underpinnings of Retrenchment. To be submitted. Available at: `http://www.cs.man.ac.uk/~banach/some.pubs/Retrench.Underpin.ps.gz`

[9] Boiten E., Derrick J. IO-Refinement in Z. In *Third BCS-FACS Northern Formal Methods Workshop*. Ilkley, U.K. 1998.

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

[11] Jones C. B. *Systematic Software Development Using VDM*, 2nd ed. Prentice Hall, 1990.

[12] Litteck H.J., Wallis P.J.L. Refinement Methods and Refinement Calculi. *Software Engineering Journal* **7**(3), 219-229, 1992.

[13] Liu S. Evolution: A More Practical Approach than Refinement for Software Development. In *Proc. ICECCS-97*, pages 142-151. IEEE, 1997.

[14] Morgan C. *Programming from Specifications*, 2nd ed. Prentice Hall, 1994.

[15] Poppleton M., Banach R. Retrenchment: Extending Refinement for Continuous and Control Systems. In *Proc. IWFM'00*, Springer Electronic Workshop in Computer Science Series, `http://ewic.org.uk/ewic`. Springer, 2000.

[16] Smith G. Stepwise Development from Ideal Specifications. In Edwards (ed.), *Aus. Comp. Sci. Conf.* **22**(1), 227-233. IEEE Computer Society, 2000.

[17] Smith G., Fidge C. Incremental Development of Real-Time Requirements: The Light Control Case Study. JUCS **6**, 704-730, 2000.

[18] Spivey J. M. *The Z Notation: A Reference Manual*, 2nd ed. Prentice Hall, 1993.

[19] Stepney S., Cooper D., Woodcock J. More Powerful Z Data Refinement: Pushing the State of the Art in Industrial Refinement. In Bowen, Fett, Hinchey (eds.), *Proc. ZUM'98*, LNCS **1493**, 284-307. Springer, 1998.