Composition Mechanisms for Retrenchment

R. Banach[a], C. Jeske[a], M. Poppleton[b]
[a]Computer Science Dept., Manchester University, Manchester, M13 9PL, U.K.
[b]Electronics and Computer Science, University of Southampton, Southampton, SO17 1BJ, U.K.
banach@cs.man.ac.uk , cjeske@cs.man.ac.uk , mrp@ecs.soton.ac.uk

Abstract. Retrenchment is a flexible model evolution formalism that arose as a reaction to the limitations imposed by refinement, and for which the proof obligations feature additional predicates for accommodating design data. Composition mechanisms for retrenchment are studied. Vertical, horizontal, dataflow, parallel and fusion compositions are described. Of particular note are the means by which the additional predicates compose. It is argued that all of the compositions introduced are associative, and that they are mutually coherent. Composition of retrenchment with refinement, so important for the smooth interworking of the two techniques, is discussed. Decomposition, allowing finer grained retrenchments to be extracted from a single large grained retrenchment, is also investigated.

Keywords. Retrenchment, Refinement, Composition, Decomposition, Requirements Engineering, Model Evolution.

1 Introduction

As a design and development technique, specific incarnations of model based refinement (see eg. [de Roever and Engelhardt (1998)] for a survey) can sometimes fall short of what is desired, as regards treating individual requirements issues in an ideal way. Retrenchment was introduced as a means for addressing such issues, in particular allowing them to be treated in a formal manner whilst at the same time not interfering with a perhaps over-idealised refinement development. In [Banach et al. (2007)] the authors gave a comprehensive and broadly based overview of retrenchment. Background and context (such as what has just been hinted at here) were extensively discussed, some key issues that arise with retrenchment were described, and some case studies were explored. We will not repeat all that here. Instead, this paper is concerned with a key technical topic, composition, and the objective of this paper is to set out definitive technical results on compositions of retrenchments, treating retrenchment purely as a mathematical theory, and leaving aside for contemplation in other places (eg. in loc cit and elsewhere), consideration of its fitness for purpose for any particular system development objective.

The heart of retrenchment is the operation proof obligation (PO), which demands that the relationship between corresponding operations at adjacent levels of abstraction be put into a particular first order shape. The shape is a judicious heuristic adaptation of commonly occurring shapes for (conventional model based) refinement, got by enriching the latter with additional relations, these being intended to permit additional design flexibility. The particular choice of first order shape is also designed to allow some interworking between the refinement and retrenchment techniques, based purely on their PO shapes.

Focusing on these additional relations, retrenchment becomes a particular data structure, being characterised by four pieces of data: the retrieve relation G, and on a per-operation basis, the within, output and concedes relations, P_{Op}, O_{Op}, C_{Op}. This is in contrast with refinement, which can be characterised in terms of data principally by G, (though a fairer comparison might be with I/O versions of refinement which have relations also for inputs and outputs; see eg. [Derrick and Boiten (2001)]). The richness of the retrenchment data structure, and the unrestricted nature of the various relations that comprise it, give great scope for expressing non-trivial properties of the related systems by incorporating suitable facts into these relations. Accordingly, there is considerable systems engineering interest in knowing how the information in the G, P_{Op}, O_{Op}, C_{Op} belonging to component retrenchments can combine to give properties of a larger development. Thus we want to see how the various pieces of retrenchment data transform under different notions of composition, raising questions of compatibility and associativity.

This paper defines a number of notions of composition and shows that the questions just posed can be answered positively. Two things are worth emphasising here. The first is that notions of composition for retrenchment do not come preordained, but are a matter for definition. Especially with retrenchment, even when one considers a fixed 'kind' of composition, it is possible to come up with more than one definition, and different definitions enjoy different properties. In this paper we will restrict attention to composition mechanisms that are based on straightforward propositional considerations; these definitions give the easiest route to coherence and associativity. (Alternative definitions, relying increasingly on semantic input, and giving more focused system descriptions, but being more challenging as regards associativity, have been explored for vertical composition in [Banach and Jeske (2002)].)

The second thing is that for every choice of composition mechanism, there are two tasks to attend to. One must show that the mechanism is sound, i.e. it yields a retrenchment, assuming its ingredients were themselves valid retrenchments, and, as already noted, one must show associativity, since a composition mechanism that does not associate is a significantly different beast from one that does. (At minimum, when contemplating a composition of several entities whose composition law is not associative, one must be very clear about what the different association orders are saying about the whole, whether generically or on a case by case basis.)

The rest of this paper is as follows. In Section 2 we recall the retrenchment POs, and the corresponding simulation relation (the latter being concerned with simulation properties between pairs of individual steps). The next few sections are concerned with specific composition mechanisms. Here, the plan is the same for each style of composition; the section starts with a statement of what the form of composition is about, then the principal result is given in outline form, and some discussion of it follows, finally the result is restated in detail and proved. Section 3 covers vertical composition, the composition of development stages; the main soundness proposition is proved in detail here, allowing subsequent proofs to be sketched more briefly. Section 4 covers horizontal composition, the sequential composition of entire operations. Section 5 covers dataflow composition, in which I/O rather than state plays the dominant role. Section 6 covers synchronous parallel composition. Section 7 covers the asynchronous parallel case. Section 8 covers fusion composition, allowing the combination of different retrenchments between the same pair of systems. Section 9 examines associativity and related issues of coherence. It is worth delaying a discussion of associativity to this point in order to take advantage of common aspects of the preceding composition mechanisms. Section 10 covers the composition of retrenchments with refinements, an essential ingredient in retrenchment/refinement interworking — technically, this combines the features of a stronger composition and a degenerate vertical composition. Section 11 considers decomposition, not so much as a direct converse to the preceding material but as a way of extracting more precise operation evolution information from more coarse grained retrenchment data; this is related to preceding mechanisms as appropriate. Up to this point the paper concentrates on the technical details of the mechanisms involved. Section 12 broadens the context and briefly indicates application areas in which these various techniques can be of benefit. Section 13 concludes.

2 Retrenchment: POs and Simulation

In this section we give our basic definitions and notations. We deal with a pair of systems in a development hierarchy, an abstract system *Abs* and a concrete one *Conc*, to be related by a retrenchment. The abstract system has a set of operation names $\mathsf{Ops}_{\mathrm{A}}$, with typical element Op_{A}. An operation Op_{A} works on the abstract state space U having typical element u (the before-state), and on an input space $\mathsf{I}_{Op_{\mathrm{A}}}$ with typical element i. Op_{A} will produce an after-state typically written u' and once more in U, and an output o drawn from an output space $\mathsf{O}_{Op_{\mathrm{A}}}$. Initial states satisfy the predicate $Init_{\mathrm{A}}(u')$. We work in a transition system framework, so an operation Op_{A} is given by its transition or step relation consisting of steps u -(i, Op_{A}, o)-› u'. The set of these steps forms the relation $stp_{Op_{\mathrm{A}}}(u, i, u', o)$. Aggregating over all of $\mathsf{Ops}_{\mathrm{A}}$, we obtain $stp_{\mathrm{A}} = \bigcup_{Op_{\mathrm{A}} \in \mathsf{Ops}_{\mathrm{A}}} stp_{Op_{\mathrm{A}}}$, which is the complete transition relation for the *Abs* system, and where the union is necessarily disjoint since the relevant Op_{A} name is part of every execution step.

An execution fragment of the *Abs* system is a finite or infinite sequence of contiguous steps, written [u_0 -(i_0, $Op_{\mathrm{A},0}$, o_1)-› u_1 -(i_1, $Op_{\mathrm{A},1}$, o_2)-› u_2 …], and drawn from stp_{A}. An execution fragment such that $Init_{\mathrm{A}}(u_0)$ holds is called an execution sequence. An abstract state u is reachable, iff it is the last state of some execution sequence.

At the concrete level we have a similar setup. The operation names are $Op_{\mathrm{C}} \in \mathsf{Ops}_{\mathrm{C}}$. States are $v \in \mathsf{V}$, inputs $j \in \mathsf{J}_{Op_{\mathrm{C}}}$, outputs $p \in \mathsf{P}_{Op_{\mathrm{C}}}$. Initial states satisfy $Init_{\mathrm{C}}(v')$. Transitions are v -(j, Op_{C}, p)-› v', elements of the step relation $stp_{Op_{\mathrm{C}}}(v, j, v', p)$.

2.1 Proof Obligations

Given the above context, a(n output) retrenchment from *Abs* to *Conc* is defined by three facts. Firstly, $\mathsf{Ops}_{\mathrm{A}} \subseteq \mathsf{Ops}_{\mathrm{C}}$, i.e. to each abstract operation there corresponds a concrete operation which we will assume has the same name. The inclusion can be proper so the converse need not hold.[1] Secondly, we have a collection of relations as follows: there is a retrieve relation $G(u, v)$ between abstract and concrete state spaces; and there is a family of within, output, and concedes relations for each $Op_{\mathrm{A}} \in \mathsf{Ops}_{\mathrm{A}}$: $P_{Op}(i, j, u, v)$, $O_{Op}(o, p; u', v', i, j, u, v)$ and $C_{Op}(u', v', o, p; i, j, u, v)$ respectively.

[1] This confirms that the ‘A’ and ‘C’ subscripts on operation names are meta level tags. We suppress them when it is convenient to do so and it does not cause confusion.

These relations are over the variables shown, i.e. the within relations involve the inputs and before-states, while the output and concedes relations involve predominantly the outputs and after-states, though inputs and before-states can also feature if required (the semicolon cosmetically separating these additional possibilities). The relations are collectively referred to as the retrenchment data. Note that we suppress the 'A' and 'C' subscripts on *Op* in these relations since they concern both levels of abstraction equally. Thirdly, a collection of properties (the proof obligations or POs) must hold. The initial states must satisfy:

$$Init_{\mathrm{C}}(v') \;\Rightarrow\; (\exists\, u' \bullet Init_{\mathrm{A}}(u') \wedge G(u', v')) \tag{2.1}$$

and for every corresponding operation pair Op_{A} and Op_{C}, the abstract and concrete step relations must satisfy the operation PO:

$$\begin{aligned}&G(u, v) \wedge P_{Op}(i, j, u, v) \wedge stp_{Op_{\mathrm{C}}}(v, j, v', p) \;\Rightarrow\\ &\quad (\exists\, u', o \bullet stp_{Op_{\mathrm{A}}}(u, i, u', o) \wedge ((G(u', v') \wedge O_{Op}(o, p; u', v', i, j, u, v)) \vee\\ &\qquad C_{Op}(u', v', o, p; i, j, u, v)))\end{aligned} \tag{2.2}$$

In [Banach et al. (2007)] the contrast between primitive retrenchment (which has no output relations O_{Op}) and output retrenchment (which does, as here) was discussed at length, underlining the algebraic utility of the latter. In this paper, we will use the output form, noting that all the results obtained, translate to the primitive form by folding in the universal relation true for all occurrences of output relations O_{Op}. Henceforth we will refer to output retrenchment as just retrenchment.

2.2 The Simulation Relation

For an $Op_{\mathrm{A}} \in \mathsf{Ops}_{\mathrm{A}}$, an important counterfoil to the operation PO is the operation's simulation relation. This holds for an abstract step u -(i, Op_{A}, o)-› u' and a corresponding concrete step v -(j, Op_{C}, p)-› v', the two steps being in simulation, iff:

$$\begin{aligned}&G(u, v) \wedge P_{Op}(i, j, u, v) \wedge stp_{Op_{\mathrm{C}}}(v, j, v', p) \wedge stp_{Op_{\mathrm{A}}}(u, i, u', o) \wedge\\ &((G(u', v') \wedge O_{Op}(o, p; u', v', i, j, u, v)) \vee C_{Op}(u', v', o, p; i, j, u, v))\end{aligned} \tag{2.3}$$

holds. We write this succinctly as (u -(i, Op_{A}, o)-› u') Σ^1 (v -(j, Op_{C}, p)-› v'), where the retrenchment data, G, P_{Op}, O_{Op}, C_{Op}, are understood. Strict simulation, written (u -(i, Op_{A}, o)-› u') $\Sigma^{\mathrm{S}1}$ (v -(j, Op_{C}, p)-› v'), folds $C_{Op} = \mathsf{false}$ into (2.3).

In the retrenchment context, the simulation relation is best approached as something to be calculated in an ad hoc manner. In particular, since all the relations involved in (2.2) are in principle partial, and the consequents of the operation POs contain C_{Op} disjunctively while the antecedents contain P_{Op} conjunctively, the prospects for sequentially composing steps in simulation via a normal inductive technique are greatly reduced.

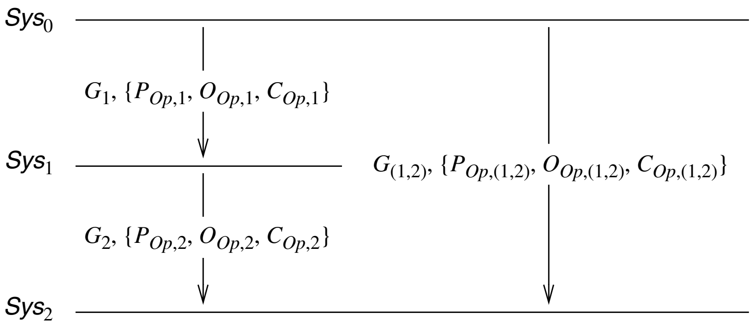
Thus, given a pair of steps s in *Abs* and t in *Conc* which satisfy (2.3), then s may or may not have a step s' that can immediately follow it. If it has, then such an s' may or may not be simulable. And if there is such a simulable s' simulated by t' say, there is no guarantee that any such t' can be concatenated with t to form an execution fragment. One can just as well apply the same reasoning starting with t instead of s. And both arguments can be run backwards for predecessors of s and t. Simulation clearly becomes a much more complex phenomenon than in refinement.

Evidently, deriving a stepwise simulation result, stating that each concrete execution sequence [v_0 -(j_0, $Op_{\mathrm{C},0}$, p_1)-> v_1 -(j_1, $Op_{\mathrm{C},1}$, p_2)-> v_2 …] has an abstract execution sequence [u_0 -(i_0, $Op_{\mathrm{A},0}$, o_1)-> u_1 -(i_1, $Op_{\mathrm{A},1}$, o_2)-> u_2 …] with each pair of corresponding steps in simulation, becomes impractical by conventional means. Proof techniques more directly aimed at termination, such as finite inductions controlled by a decreasing variant function with values in a well founded set, are more likely to yield positive results for those fragments of behaviour that are in simulation.

Note that (2.3) treats *Abs* and *Conc* symmetrically, in contrast with the asymmetric nature of the PO (2.2). For systems *Abs* and *Conc* in retrenchment, the two formulations are equivalent in the sense that whenever the antecedents of the PO are valid, then the corresponding simulation relation can be demonstrated (by definition); and conversely if the simulation relation holds, then the antecedents of the PO obviously do also. As with refinement, the PO is mainly a means of establishing the simulation relation, and the 'don't care' interpretation, when the antecedents of the PO implication are false, is of little interest.

3 Vertical Composition

Suppose we have an abstract system *Abs*, which is transmuted via a retrenchment to a concrete system *Conc*, and that *Conc* is in turn transmuted via a further retrenchment to a (say) implementation system *Imp*. If we assume the granularity of the individual transitions in these models does not change, how are *Abs* and *Imp* related? On the tacit (though by no means mandatory) assumption that a retrenchment is accompanied by a move towards implementability, and depicting such moves vertically, the relationship between *Abs* and *Imp* is a retrenchment which is a composition of the previous two, their vertical composition. If the retrenchment data for the first retrenchment are subscripted '1' and for the second '2', we will subscript the composed data '(1,2)'. Fig. 1 illustrates this, for a more symmetric notation.



Sys_0
G_1, $\{P_{Op,1}$, $O_{Op,1}$, $C_{Op,1}\}$
Sys_1
$G_{(1,2)}$, $\{P_{Op,(1,2)}$, $O_{Op,(1,2)}$, $C_{Op,(1,2)}\}$
G_2, $\{P_{Op,2}$, $O_{Op,2}$, $C_{Op,2}\}$
Sys_2

Fig. 1. Vertical composition.

In outline, the vertically composed retrenchment data are as follows:

$$G_{(1,2)} \equiv G_1 \fatsemi G_2 \tag{3.1}$$

$$P_{Op,(1,2)} \equiv (G_1 \wedge P_{Op,1}) \fatsemi (G_2 \wedge P_{Op,2}) \tag{3.2}$$

$$O_{Op,(1,2)} \equiv O_{Op,1} \fatsemi O_{Op,2} \tag{3.3}$$

$$C_{Op,(1,2)} \equiv (G'_1 \wedge O_{Op,1} \fatsemi C_{Op,2}) \vee (C_{Op,1} \fatsemi G'_2 \wedge O_{Op,2}) \vee (C_{Op,1} \fatsemi C_{Op,2}) \tag{3.4}$$

In (3.1)-(3.4) the forward relational composition $\fatsemi$ is via the relevant variables of the intermediate system. Thus the composed retrieve relation is straightforwardly the composition of the two retrieves; likewise for the composed output relation. The composed within relation is the composition of the two withins, but strengthened by the composed retrieve. Lastly the composed concession has the most complex form: either the after-state retrieve and output relations for the first retrenchment, composed with the concession for the second holds; or the converse holds; or the composition of the two concessions holds.

If one regards the concession as somehow capturing ‘exceptional’ behaviour, then the composed concession has a shape that we might expect: in the presence of two system development steps, either the first alone might have an exception, or the second alone might, or both might. On the other hand, to have ‘non-exceptional’ behaviour, we would need both steps to be non-exceptional, as the composed output relation demands. Next is a precise statement of the composition.

Proposition 3.1 Let Sys_0 (with variables u_0, i_0, o_0) be retrenched to Sys_1 (with variables u_1, i_1, o_1) using G_1, $\{P_{Op,1}, O_{Op,1}, C_{Op,1} \mid Op \in \mathsf{Ops}_0\}$, and Sys_1 be retrenched to Sys_2 (with variables u_2, i_2, o_2) using G_2, $\{P_{Op,2}, O_{Op,2}, C_{Op,2} \mid Op \in \mathsf{Ops}_1\}$. Then Sys_0 is retrenched to Sys_2 using retrieve, within, and concedes relations $G_{(1,2)}$, $\{P_{Op,(1,2)}, O_{Op,(1,2)}, C_{Op,(1,2)} \mid Op \in \mathsf{Ops}_0\}$, where:

$$G_{(1,2)}(u_0, u_2) \equiv [\exists\, u_1 \bullet G_1(u_0, u_1) \wedge G_2(u_1, u_2)] \tag{3.5}$$

$$\begin{aligned}
&P_{Op,(1,2)}(i_0, i_2, u_0, u_2) \equiv \\
&\quad [\exists\, u_1, i_1 \bullet G_1(u_0, u_1) \wedge G_2(u_1, u_2) \wedge \\
&\qquad P_{Op,1}(i_0, i_1, u_0, u_1) \wedge P_{Op,2}(i_1, i_2, u_1, u_2)]
\end{aligned} \tag{3.6}$$

$$\begin{aligned}
&O_{Op,(1,2)}(o_0, o_2; u'_0, u'_2, i_0, i_2, u_0, u_2) \equiv \\
&\quad [\exists\, u'_1, o_1, u_1, i_1 \bullet \\
&\qquad O_{Op,1}(o_0,o_1;u'_0,u'_1,i_0,i_1,u_0,u_1) \wedge O_{Op,2}(o_1,o_2;u'_1,u'_2,i_1,i_2,u_1,u_2)]
\end{aligned} \tag{3.7}$$

$$\begin{aligned}
&C_{Op,(1,2)}(u'_0, u'_2, o_0, o_2; i_0, i_2, u_0, u_2) \equiv \\
&\quad [\exists\, u'_1, o_1, u_1, i_1 \bullet \\
&\qquad (G_1(u'_0, u'_1) \wedge O_{Op,1}(o_0, o_1; u'_0, u'_1, i_0, i_1, u_0, u_1) \wedge \\
&\qquad\quad C_{Op,2}(u'_1, u'_2, o_1, o_2; i_1, i_2, u_1, u_2)) \vee \\
&\qquad (C_{Op,1}(u'_0, u'_1, o_0, o_1; i_0, i_1, u_0, u_1) \wedge \\
&\qquad\quad G_2(u'_1, u'_2) \wedge O_{Op,2}(o_1, o_2; u'_1, u'_2, i_1, i_2, u_1, u_2)) \vee \\
&\qquad (C_{Op,1}(u'_0, u'_1, o_0, o_1; i_0, i_1, u_0, u_1) \wedge \\
&\qquad\quad C_{Op,2}(u'_1, u'_2, o_1, o_2; i_1, i_2, u_1, u_2))]
\end{aligned} \tag{3.8}$$

Proof. To show we have a retrenchment, we must show that the POs for the composed retrenchment follow from the POs for the individual ones: the initialisation PO, $\mathit{Init}(u'_2) \Rightarrow (\exists\, u'_0 \bullet \mathit{Init}(u'_0) \wedge G_{(1,2)}(u'_0, u'_2))$, follows immediately by composing the individual initialisation POs.

For the operation PO, let us assume (3.5), (3.6), and a step u_2 -(i_2, Op, o_2)-› u'_2 for some operation Op of Sys_2 such that $Op \in \mathsf{Ops}_0$. Now (3.5) and (3.6) imply there are u_1, i_1 such that $G_2(u_1, u_2) \wedge P_{Op,2}(i_1, i_2, u_1, u_2)$ holds, so the second retrenchment implies that there are u'_1, o_1 for which there is a Sys_1 step u_1 -(i_1, Op, o_1)-› u'_1, for which $(G_2(u'_1, u'_2) \wedge O_{Op,2}(o_1, o_2; \ldots)) \vee C_{Op,2}(u'_1, u'_2, o_1, o_2; \ldots)$ holds. Using the u_0, i_0 from (3.5), (3.6), and the step u_1 -(i_1, Op, o_1)-› u'_1, we repeat the argument to deduce the existence of u'_0, o_0 and a Sys_0 step u_0 -(i_0, Op, o_0)-› u'_0 for which $(G_1(u'_0, u'_1) \wedge O_{Op,1}(o_0, o_1; \ldots)) \vee C_{Op,1}(u'_0, u'_1, o_0, o_1; \ldots)$ holds. So from (3.5), (3.6), and the step u_2 -(i_2, Op, o_2)-› u'_2, we have deduced the u_0 -(i_0, Op, o_0)-› u'_0 step such that $(\exists\, u'_1, o_1, u_1, i_1 \bullet ((G'_1 \wedge O_{Op,1}) \vee C_{Op,1}) \wedge ((G'_2 \wedge O_{Op,2}) \vee C_{Op,2}))$ holds. A little boolean algebra turns $((G'_1 \wedge O_{Op,1}) \vee C_{Op,1}) \wedge ((G'_2 \wedge O_{Op,2}) \vee C_{Op,2})$ into $((\underline{G}'_{(1,2)} \wedge \underline{O}_{Op,(1,2)}) \vee \underline{C}_{Op,(1,2)})$, where $\underline{G}'_{(1,2)}$, $\underline{O}_{Op,(1,2)}$ and $\underline{C}_{Op,(1,2)}$ are $G'_{(1,2)}$, $O_{Op,(1,2)}$ and $C_{Op,(1,2)}$ without their existential quantifications. And since $(\exists \ldots \bullet (A \clubsuit B)) \Rightarrow (\exists \ldots \bullet A) \clubsuit (\exists \ldots \bullet B)$ where $\clubsuit \in \{\wedge, \vee\}$, we infer that $(\exists \ldots \bullet ((\underline{G}'_{(1,2)} \wedge \underline{O}_{Op,(1,2)}) \vee \underline{C}_{Op,(1,2)}))$ implies $((G'_{(1,2)} \wedge O_{Op,(1,2)}) \vee C_{Op,(1,2)})$. So the operation PO is valid for the composed retrenchment with vertical composition defined by (3.5)-(3.8). ☺

4 Horizontal Composition

Suppose we have abstract and concrete systems Abs and Conc, which are related by a retrenchment given by the usual data. Suppose we have an abstract operation $Op_{\mathrm{A},1}$ followed by another abstract operation $Op_{\mathrm{A},2}$ (which is to say that their relational composition $Op_{\mathrm{A},1} ; Op_{\mathrm{A},2}$ is non-empty). Suppose that $Op_{\mathrm{A},1}$ is retrenched to $Op_{\mathrm{C},1}$ and that $Op_{\mathrm{A},2}$ is retrenched to $Op_{\mathrm{C},2}$. Under what conditions is $Op_{\mathrm{A},1} ; Op_{\mathrm{A},2}$ retrenched to $Op_{\mathrm{C},1} ; Op_{\mathrm{C},2}$, and in particular, what are the appropriate retrenchment data for these compound operations? That is the problem addressed by the horizontal composition of retrenchments.[2] See Fig. 2.

$$u \text{ -}(i_1, Op_{\mathrm{A},1}, o_1)\text{-› } \underline{u} \text{ -}(i_2, Op_{\mathrm{A},2}, o_2)\text{-› } u'$$

$$v \text{ -}(j_1, Op_{\mathrm{C},1}, p_1)\text{-› } \underline{v} \text{ -}(j_2, Op_{\mathrm{C},2}, p_2)\text{-› } v'$$

Fig. 2. Horizontal composition.

The discussion in Section 2.2 argued that the retrenchment PO was closely related to the appropriate simulation relation, and moreover, that the nature of retrenchment precluded the naive sequential composition of retrenchment simulation squares. Therefore the reader should be alert to the possibility that the agenda of horizontal

[2] So unlike vertical composition, it does not directly build a third retrenchment between systems out of two existing retrenchments between systems. Rather it is related to simulation, as we shall see. The two operations discussed could even come from two different retrenchments sharing the same state spaces.

composition of retrenchments is fraught with danger. In this section we give a composition law featuring a construction strong enough to exclude the dangerous cases.

If the retrenchment data for the first retrenchment are subscripted '1' and for the second '2', we will subscript the composed data '$(Op,1;Op,2)$'. In outline, the horizontally composed retrenchment data are as follows:

$$G_{(Op,1;Op,2)} \equiv G \tag{4.1}$$

$$P_{(Op,1;Op,2)} \equiv P_{Op,1} \wedge G \wedge \mathrm{wp}(\Sigma^1(Op_1), (G \wedge P_{Op,2})) \tag{4.2}$$

$$O_{(Op,1;Op,2)} \equiv O_{Op,1} \fatsemi O_{Op,2} \tag{4.3}$$

$$C_{(Op,1;Op,2)} \equiv (G'_1 \wedge O_{Op,1} \fatsemi C_{Op,2}) \vee (C_{Op,1} \fatsemi G'_2 \wedge O_{Op,2}) \vee (C_{Op,1} \fatsemi C_{Op,2}) \tag{4.4}$$

In (4.1)-(4.4), we see that the output and concedes relations have the same form as in (3.1)-(3.4); however this time, the forward relational composition $\fatsemi$ concerns the intermediate abstract and concrete state variables shared between the two consecutive operations (as the common after-state of the first and before-state of the second). The retrieve relation is just the common one. The most complex ingredient of the composed retrenchment data is the within relation. This asserts not only the within relation of the first operation (together with the retrieve relation), but also the weakest precondition (on the before-states and inputs of the first operation, and defined properly in (4.6) below) that guarantees that both the abstract and concrete first operation yield results that are certain to fall into the within and retrieve relation for the second operation. The detailed results now follow, including some further elaborations.

Proposition 4.1 Let $Op_{\mathrm{A},1}$, (with variables u, i_1, o_1), be retrenched to $Op_{\mathrm{C},1}$, (with variables v, j_1, p_1), via $G(u, v)$, $P_{Op,1}(i_1, j_1, u, v)$, $O_{Op,1}(o_1, p_1; u', v', i_1, j_1, u, v)$, $C_{Op,1}(u', v', o_1, p_1; i_1, j_1, u, v)$, and let $Op_{\mathrm{A},2}$, (with variables u, i_2, o_2), be retrenched to $Op_{\mathrm{C},2}$, (with variables v, j_2, p_2), via $G(u, v)$, $P_{Op,2}(i_2, j_2, u, v)$, $O_{Op,2}(o_2, p_2; u', v', i_2, j_2, u, v)$, $C_{Op,2}(u', v', o_2, p_2; i_2, j_2, u, v)$.

Then $Op_{\mathrm{A},1};Op_{\mathrm{A},2}$ (with variables u, $[i_1, i_2]$, $[o_1, o_2]$), is retrenched to $Op_{\mathrm{C},1};Op_{\mathrm{C},2}$ (with variables v, $[j_1, j_2]$, $[p_1, p_2]$), via $G(u, v)$ and:

$$\begin{aligned} &P_{(Op,1;Op,2)}([i_1, i_2], [j_1, j_2], u, v) \equiv \\ &\quad [P_{Op,1}(i_1, j_1, u, v) \wedge G(u, v) \wedge \\ &\qquad \mathrm{wp}(\Sigma^1(Op_1), (G(u, v) \wedge P_{Op,2}(i_2, j_2, u, v)))] \end{aligned} \tag{4.5}$$

where:

$$\begin{aligned} &\mathrm{wp}(\Sigma^1(Op_1), (G(u, v) \wedge P_{Op,2}(i_2, j_2, u, v))) \equiv \\ &\quad \{(i_1, j_1, u, v) \mid (\forall\, \underline{u}, o_1, \underline{v}, p_1 \bullet \\ &\qquad\qquad (u \;\text{-}(i_1, Op_{\mathrm{A},1}, o_1)\text{->}\; \underline{u})\ \Sigma^1\ (v \;\text{-}(j_1, Op_{\mathrm{C},1}, p_1)\text{->}\; \underline{v}) \\ &\qquad\qquad\quad \Rightarrow G(\underline{u}, \underline{v}) \wedge P_{Op,2}(i_2, j_2, \underline{u}, \underline{v}))\} \end{aligned} \tag{4.6}$$

$$\begin{aligned} &O_{(Op,1;Op,2)}([o_1, o_2], [p_1, p_2]; u', v', [i_1, i_2], [j_1, j_2], u, v) \equiv \\ &\quad [\exists\, \underline{u}, \underline{v} \bullet O_{Op,1}(o_1, p_1; \underline{u}, \underline{v}, i_1, j_1, u, v) \wedge O_{Op,2}(o_2, p_2; u', v', i_2, j_2, \underline{u}, \underline{v})] \end{aligned} \tag{4.7}$$

$$\begin{aligned} &C_{(Op,1;Op,2)}(u', v', [o_1, o_2], [p_1, p_2]; [i_1, i_2], [j_1, j_2], u, v) \equiv \\ &\quad [\exists\, \underline{u}, \underline{v} \bullet \\ &\qquad (G(\underline{u}, \underline{v}) \wedge O_{Op,1}(o_1, p_1; \underline{u}, \underline{v}, i_1, j_1, u, v) \wedge \\ &\qquad\qquad C_{Op,2}(u', v', o_2, p_2; i_2, j_2, \underline{u}, \underline{v})) \vee \\ &\qquad (C_{Op,1}(\underline{u}, \underline{v}, o_1, p_1; i_1, j_1, u, v) \wedge \end{aligned}$$

$$\begin{aligned} & G(u', v') \wedge O_{Op,2}(o_2, p_2;\, u', v', i_2, j_2, \underline{u}, \underline{v})) \vee \\ & (C_{Op,1}(\underline{u}, \underline{v}, o_1, p_1;\, i_1, j_1, u, v) \wedge C_{Op,2}(u', v', o_2, p_2;\, i_2, j_2, \underline{u}, \underline{v}))] \end{aligned} \qquad (4.8)$$

Proof sketch. We abbreviate $P_{(Op,1;Op,2)}$ to $P_{(1;2)}$ etc. Suppose we have a transition of $Op_{\mathrm{C},1}\,;Op_{\mathrm{C},2}$, $v \text{ -}(j_1, Op_{\mathrm{C},1}, p_1)\text{-›} \underline{v} \text{ -}(j_2, Op_{\mathrm{C},2}, p_2)\text{-›} v'$, and suppose that $G(u, v) \wedge P_{(1;2)}([i_1, i_2], [j_1, j_2], u, v)$ holds. Since $P_{(1;2)}$ implies $P_{Op,1}(i_1, j_1, u, v) \wedge G(u, v)$, we can use the operation PO for Op_1 and $v \text{ -}(j_1, Op_{\mathrm{C},1}, p_1)\text{-›} \underline{v}$ in the usual way, and get $u \text{ -}(i_1, Op_{\mathrm{A},1}, o_1)\text{-›} \underline{u}$, for which $(G(\underline{u}, \underline{v}) \wedge O_{Op,1}(o_1, p_1; \dots)) \vee C_{Op,1}(\underline{u}, \underline{v}, o_1, p_1; \dots)$ holds. But this means that $(u \text{ -}(i_1, Op_{\mathrm{A},1}, o_1)\text{-›} \underline{u})\ \Sigma^1\ (v \text{ -}(j_1, Op_{\mathrm{C},1}, p_1)\text{-›} \underline{v})$ holds, so by (4.5) and (4.6), $G(\underline{u}, \underline{v}) \wedge P_{Op,2}(i_2, j_2, \underline{u}, \underline{v})$ holds also. So the antecedents for the operation PO for Op_2 and $\underline{v} \text{ -}(j_2, Op_{\mathrm{C},2}, p_2)\text{-›} v'$ hold, which gives us $\underline{u} \text{ -}(i_2, Op_{\mathrm{A},2}, o_2)\text{-›} u'$, for which $(G(u', v') \wedge O_{Op,2}(o_2, p_2; \dots)) \vee C_{Op,2}(u', v', o_2, p_2; \dots)$ holds. We now combine the consequents of the two POs, much as in Proposition 3.1. ☺

If we employ strict simulation Σ^{S1} instead of Σ^1 in (4.6), which gives:

$$\begin{aligned} & \mathrm{wp}(\Sigma^{\mathrm{S1}}(Op_1), (G(u, v) \wedge P_{Op,2}(i_2, j_2, u, v))) \equiv \\ & \quad \{(i_1, j_1, u, v) \mid (\forall\ \underline{u}, o_1, \underline{v}, p_1 \bullet \\ & \qquad (u \text{ -}(i_1, Op_{\mathrm{A},1}, o_1)\text{-›} \underline{u})\ \Sigma^{\mathrm{S1}}\ (v \text{ -}(j_1, Op_{\mathrm{C},1}, p_1)\text{-›} \underline{v}) \\ & \qquad\quad \Rightarrow G(\underline{u}, \underline{v}) \wedge P_{Op,2}(i_2, j_2, \underline{u}, \underline{v}))\} \end{aligned} \qquad (4.9)$$

then Proposition 4.1 simplifies.

Corollary 4.2 Let Σ^1 be replaced by Σ^{S1} in Proposition 4.1. Then the composition reduces to (4.5) (with $\mathrm{wp}(\Sigma^{\mathrm{S1}}(Op_1), \dots)$ instead of $\mathrm{wp}(\Sigma^1(Op_1), \dots)$), (4.7), and:

$$\begin{aligned} & C_{(Op,1;Op,2)}(u', v', [o_1, o_2], [p_1, p_2];\, [i_1, i_2], [j_1, j_2], u, v) \equiv \\ & \quad [\exists\ \underline{u}, \underline{v} \bullet (G(\underline{u}, \underline{v}) \wedge O_{Op,1}(o_1, p_1;\, \underline{u}, \underline{v}, i_1, j_1, u, v) \wedge \\ & \qquad C_{Op,2}(u', v', o_2, p_2;\, i_2, j_2, \underline{u}, \underline{v}))] \end{aligned} \qquad (4.10)$$

Proposition 4.1 can be viewed as yielding a small stepwise simulation result (cf. Section 2.2), except that in stepwise simulation, one wants to assume at the outset the concrete execution fragment and attendant hypotheses, including in particular a suitable within relation, while in Proposition 4.1, the composed within relation emerges as a joint property calculated from the two retrenched operations via (4.5). So in Proposition 4.1 $P_{(1;2)}$ is not a part of the antecedent of the main inference, but a part of the consequent.

Proposition 4.1 enforces some strong conditions via the $(\forall\ \Sigma^{\mathrm{S1}} \Rightarrow G \wedge P_{Op,2})$ structure in (4.6). If we seek to weaken these, the prospects are limited. Looking to the operation PO structure, suppose we replace $\mathrm{wp}(\Sigma^1(Op_1), \dots)$ in (4.5) by:

$$\begin{aligned} & \mathrm{PO}(Op_1, (G(u, v) \wedge P_{Op,2}(i_2, j_2, u, v))) \equiv \\ & \quad \{(i_1, j_1, u, v) \mid (\forall\ \underline{v}, p_1 \bullet \exists\ \underline{u}, o_1 \bullet \\ & \qquad (u \text{ -}(i_1, Op_{\mathrm{A},1}, o_1)\text{-›} \underline{u})\ \Sigma^1\ (v \text{ -}(j_1, Op_{\mathrm{C},1}, p_1)\text{-›} \underline{v}) \\ & \qquad\quad \wedge G(\underline{u}, \underline{v}) \wedge P_{Op,2}(i_2, j_2, \underline{u}, \underline{v}))\} \end{aligned} \qquad (4.11)$$

The resulting law of horizontal composition can be proved sound, since the proof of Proposition 4.1 can be suitably modified. However it is not associative, as illustrated in the following counterexample adapted from [Poppleton (2001)].

Counterexample 4.3 Let *Abs* be retrenched to *Conc*, where both state spaces are $\{a, b\}$, there is no I/O, and there are three common operations, Op_1, Op_2, Op_3. Consider Fig. 3. The transitions for Op_1, Op_2, Op_3 are as shown. In particular Op_3 is a skip in both systems, Op_2 is functional in both systems, and the only nondeterminism occurs in the *Abs* Op_1. The vertical lines show G which is the identity on $\{a, b\}$; and for all three operations, $G = P_{Op_i}$. The only nontrivial concession is for Op_2 on state b, where it says that the *Abs* transition can go to a, whereas the *Conc* transition skips.

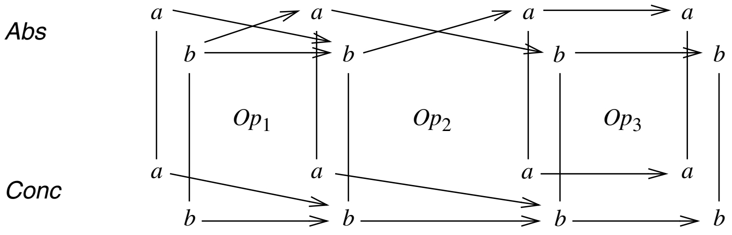


Fig. 3. Weaker horizontal composition counterexample.

We now find the following. Firstly, $P_{(2;3)}$ is just $\{(a, a)\}$ since the *Abs* and *Conc* transitions of Op_2 from b do not jointly land within $P_{Op,3}$; hence $P_{(1;(2;3))}$ is empty since a is not in the range of the *Conc* Op_1.

Secondly, $P_{(1;2)}$ is $\{(a, a), (b, b)\}$ since (b, b) is in $P_{Op,2}$ and both *Abs* and *Conc* Op_1 map both a and b to b; hence (b, b) is in $P_{((1;2);3)}$ since $(Op_1 ; Op_2)$ in both *Abs* and *Conc* skips on b, and (b, b) is in $P_{Op,3}$. Thus $P_{(1;(2;3))}$ differs from $P_{((1;2);3)}$. ☺

If we weaken further by making the $\forall\ \underline{v}, p_1$ in (4.11) existential too, then soundness itself fails, as the reader can readily check.

5 Dataflow Composition

Dataflow composition is an adaptation of horizontal composition in which the role of the state is eliminated. Instead of identifying the after-state of the first operation with the before-state of the second operation, the output of the first operation is identified with the input of the second operation, as in pipelining.

Thus in dataflow composition, the I/O plays the role of state. Abstract steps look like $*$ -(i, Op_{A}, o)-› $*$, where $*$ is the only element of a dummy one-point state space, or even more simply (i, Op_{A}, o). See Fig. 4. This obviates the need for any retrieve relation $G(u, v)$. Now a small subtlety emerges. In the output retrenchment operation PO, (2.2), it is clear that G needs to default to true. However in the primitive retrenchment operation PO, given by setting the output relation to true in (2.2), setting G to true also, makes the PO (close to) vacuous, since, aside from a joint reachability criterion, the consequent of the PO is unable to assert anything. On the other hand defaulting G to false also trivialises the PO since now the whole antecedent becomes false. Thus for primitive retrenchment, we have to default G to true in conjunctive contexts, and to false in disjunctive ones. (Of course only the former occur in the output retrenchment PO.)

If the retrenchment data for the first retrenchment are subscripted '1' and for the second '2', we will subscript the composed data '$(Op,1\text{›}Op,2)$'. The absence of state also makes it convenient to *assume* that the outputs of the first pair of operations fall into the within relation of the second pair, rather than to calculate sufficient conditions for that to be the case — so that is the perspective from which the results on dataflow composition are designed. Taking care to plug O into I as stated, we get a simplified form of Proposition 4.1.

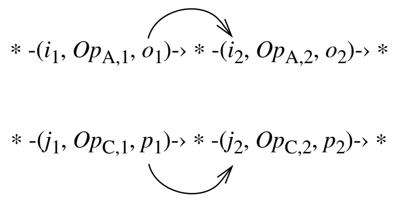

* -(i_1, $Op_{\mathrm{A},1}$, o_1)-› * -(i_2, $Op_{\mathrm{A},2}$, o_2)-› *
* -(j_1, $Op_{\mathrm{C},1}$, p_1)-› * -(j_2, $Op_{\mathrm{C},2}$, p_2)-› *

Fig. 4. Dataflow composition.

Proposition 5.1 Let $Op_{\mathrm{A},1}$, (with variables i_1, o_1), be retrenched to $Op_{\mathrm{C},1}$, (with variables j_1, p_1), via $P_{Op,1}(i_1, j_1)$, $O_{Op,1}(o_1, p_1; i_1, j_1)$, $C_{Op,1}(o_1, p_1; i_1, j_1)$, and let $Op_{\mathrm{A},2}$, (with variables i_2, o_2), be retrenched to $Op_{\mathrm{C},2}$, (with variables j_2, p_2), via $P_{Op,2}(i_2, j_2)$, $O_{Op,2}(o_2, p_2; i_2, j_2)$, $C_{Op,2}(o_2, p_2; i_2, j_2)$. Suppose that:

$$(O_{Op,1}(o_1, p_1; i_1, j_1) \vee C_{Op,1}(o_1, p_1; i_1, j_1)) \wedge (o_1 = i_2) \wedge (p_1 = j_2) \;\Rightarrow\; P_{Op,2}(i_2, j_2) \tag{5.1}$$

Then $Op_{\mathrm{A},1} ; Op_{\mathrm{A},2}$ (with variables i_1, o_2), is retrenched to $Op_{\mathrm{C},1} ; Op_{\mathrm{C},2}$ (with variables j_1, p_2), via:

$$P_{(Op,1\text{›}Op,2)}(i_1, j_1) \;\equiv\; [P_{Op,1}(i_1, j_1)] \tag{5.2}$$

$$O_{(Op,1\text{›}Op,2)}(o_2, p_2; i_1, j_1) \equiv [\exists\, a, c \bullet O_{Op,1}(a, c; i_1, j_1) \wedge O_{Op,2}(o_2, p_2, a, c)] \tag{5.3}$$

$$\begin{aligned} &C_{(Op,1\text{›}Op,2)}(o_2, p_2; i_1, j_1) \;\equiv \\ &\quad [\exists\, a, c \bullet (O_{Op,1}(a, c; i_1, j_1) \wedge C_{Op,2}(o_2, p_2, a, c)) \vee \\ &\qquad\qquad (C_{Op,1}(a, c; i_1, j_1) \wedge O_{Op,2}(o_2, p_2, a, c)) \vee \\ &\qquad\qquad (C_{Op,1}(a, c; i_1, j_1) \wedge C_{Op,2}(o_2, p_2, a, c))] \end{aligned} \tag{5.4}$$

Proof sketch. This follows very much the structure of Proposition 4.1; (5.1) ensures that a more complex composed within relation like (4.5) is not needed, and reasoning about states is replaced by reasoning about I/O, and the one point rule. ☺

Note that the primitive version of the above behaves well, in that when O is erased, (5.4) reduces to just $C_{Op,1} \wedge C_{Op,2}$.

6 Synchronous Parallel Composition

In synchronous parallel composition, two separate retrenchments between two separate pairs of systems, with some identifiable operation name pairs (see Fig. 5), but with separate state and I/O spaces, are brought together in lockstep. The state space is a cartesian product (as are the I/O spaces for identifiable pairs), and identically named operations from the two systems each act in their own component of the product.

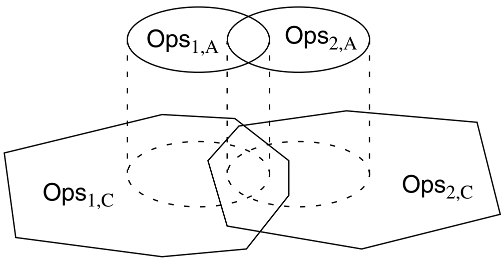

$\mathsf{Ops}_{1,\mathrm{A}}$
$\mathsf{Ops}_{2,\mathrm{A}}$
$\mathsf{Ops}_{1,\mathrm{C}}$
$\mathsf{Ops}_{2,\mathrm{C}}$

Fig. 5. Synchronous parallel composition.

If the retrenchment data for the first retrenchment are subscripted '1' and for the second '2', we will subscript the composed data '(1|2)'. In outline, the basic idea is simple. Operation names common to both systems are defined to work in lockstep, while operation names exlusive to one or other system are defined to work in lockstep with the identity on the othe system. For the abstract systems we can thus write this as:

$$\begin{array}{lll} Op_{(1|2),\mathrm{A}} \equiv & Op_{1,\mathrm{A}} \wedge Op_{2,\mathrm{A}} & \text{if } Op \in \mathsf{Ops}_{1,\mathrm{A}} \cap \mathsf{Ops}_{2,\mathrm{A}} \\ & Op_{1,\mathrm{A}} \wedge \mathrm{id}_{2,\mathrm{A}} & \text{if } Op \in \mathsf{Ops}_{1,\mathrm{A}} - \mathsf{Ops}_{2,\mathrm{A}} \\ & \mathrm{id}_{1,\mathrm{A}} \wedge Op_{2,\mathrm{A}} & \text{if } Op \in \mathsf{Ops}_{2,\mathrm{A}} - \mathsf{Ops}_{1,\mathrm{A}} \end{array} \qquad (6.1)$$

with a similar definition for the concrete case. Initialisation of the composition is just the joint initialisation of the two components. In outline, the composed retrenchment data for the composition turns out to be:

$$G_{(1,2)} \equiv G_1 \wedge G_2 \qquad (6.2)$$

$$\begin{array}{lll} P_{Op,(1|2)} \equiv & P_{Op,1} \wedge P_{Op,2} & \text{if } Op \in \mathsf{Ops}_{1,\mathrm{A}} \cap \mathsf{Ops}_{2,\mathrm{A}} \\ & P_{Op,1} & \text{if } Op \in \mathsf{Ops}_{1,\mathrm{A}} - \mathsf{Ops}_{2,\mathrm{A}} \\ & P_{Op,2} & \text{if } Op \in \mathsf{Ops}_{2,\mathrm{A}} - \mathsf{Ops}_{1,\mathrm{A}} \end{array} \qquad (6.3)$$

$$\begin{array}{lll} O_{Op,(1|2)} \equiv & O_{Op,1} \wedge O_{Op,2} & \text{if } Op \in \mathsf{Ops}_{1,\mathrm{A}} \cap \mathsf{Ops}_{2,\mathrm{A}} \\ & O_{Op,1} & \text{if } Op \in \mathsf{Ops}_{1,\mathrm{A}} - \mathsf{Ops}_{2,\mathrm{A}} \\ & O_{Op,2} & \text{if } Op \in \mathsf{Ops}_{2,\mathrm{A}} - \mathsf{Ops}_{1,\mathrm{A}} \end{array} \qquad (6.4)$$

$$\begin{array}{lll} C_{Op,(1|2)} \equiv & \multicolumn{2}{l}{(G'_1 \wedge O_{Op,1} \wedge C_{Op,2}) \vee (C_{Op,1} \wedge G'_2 \wedge O_{Op,2}) \vee (C_{Op,1} \wedge C_{Op,2})} \\ & & \text{if } Op \in \mathsf{Ops}_{1,\mathrm{A}} \cap \mathsf{Ops}_{2,\mathrm{A}} \\ & C_{Op,1} & \text{if } Op \in \mathsf{Ops}_{1,\mathrm{A}} - \mathsf{Ops}_{2,\mathrm{A}} \\ & C_{Op,2} & \text{if } Op \in \mathsf{Ops}_{2,\mathrm{A}} - \mathsf{Ops}_{1,\mathrm{A}} \end{array} \qquad (6.5)$$

The above is easy to understand. The composed retrieve relation is just the two component retrieve relations, each acting in its own state space. For operation names common to both systems, the remaining retrenchment data have a form that is by now familiar from previous compositions, while for exclusive operation names, the retrenchment data is just that for the system in question, since retrenchments for identity operations need not be other than trivial. The following detailed result is now easy to prove.

Proposition 6.1 Let $\mathsf{Sys}_{1,\mathrm{A}}$ (with variables u_1, i_1, o_1) be retrenched to $\mathsf{Sys}_{1,\mathrm{C}}$ (with variables v_1, j_1, p_1) using G_1, $\{P_{Op,1}, O_{Op,1}, C_{Op,1} \mid Op \in \mathsf{Ops}_{1,\mathrm{A}}\}$, and $\mathsf{Sys}_{2,\mathrm{A}}$ (with variables u_2, i_2, o_2) be retrenched to $\mathsf{Sys}_{2,\mathrm{C}}$ (with variables v_2, j_2, p_2) using G_2, $\{P_{Op,2}, O_{Op,2}, C_{Op,2} \mid Op \in \mathsf{Ops}_{2,\mathrm{A}}\}$. Let $\mathsf{Sys}_{(1|2),\mathrm{A}}$ have state variable (u_1, u_2) valued in the appropriate product space, initial states satisfying:

$$Init_{(1|2),\mathrm{A}}(u'_1, u'_2) \;\equiv\; [Init_{1,\mathrm{A}}(u'_1) \wedge Init_{2,\mathrm{A}}(u'_2)] \tag{6.6}$$

and let the operations of $\mathsf{Sys}_{(1|2),\mathrm{A}}$ be given by:

$$\begin{aligned}&(u_1, u_2)\;\text{-}((i_1, i_2), Op, (o_1, o_2))\text{->}\;(u'_1, u'_2) \quad \text{iff}\\ &\qquad u_1\;\text{-}(i_1, Op, o_1)\text{->}\;u'_1 \wedge u_2\;\text{-}(i_2, Op, o_2)\text{->}\;u'_2 \wedge\\ &\qquad Op \in \mathsf{Ops}_{1,\mathrm{A}} \cap \mathsf{Ops}_{2,\mathrm{A}}\end{aligned} \tag{6.7}$$

$$\begin{aligned}&(u_1, u_2)\;\text{-}(i_1, Op, o_1)\text{->}\;(u'_1, u'_2) \quad \text{iff}\\ &\qquad u_1\;\text{-}(i_1, Op, o_1)\text{->}\;u'_1 \wedge u_2 = u'_2 \wedge Op \in \mathsf{Ops}_{1,\mathrm{A}} - \mathsf{Ops}_{2,\mathrm{A}}\end{aligned} \tag{6.8}$$

$$\begin{aligned}&(u_1, u_2)\;\text{-}(i_2, Op, o_2)\text{->}\;(u'_1, u'_2) \quad \text{iff}\\ &\qquad u_1 = u'_1 \wedge u_2\;\text{-}(i_2, Op, o_2)\text{->}\;u'_2 \wedge Op \in \mathsf{Ops}_{2,\mathrm{A}} - \mathsf{Ops}_{1,\mathrm{A}}\end{aligned} \tag{6.9}$$

Likewise let $\mathsf{Sys}_{(1|2),\mathrm{C}}$ have state variable (v_1, v_2), with initial states satisfying:

$$Init_{(1|2),\mathrm{C}}(v'_1, v'_2) \;\equiv\; [Init_{1,\mathrm{C}}(v'_1) \wedge Init_{2,\mathrm{C}}(v'_2)] \tag{6.10}$$

and operations given by:

$$\begin{aligned}&(v_1, v_2)\;\text{-}((j_1, j_2), Op, (p_1, p_2))\text{->}\;(v'_1, v'_2) \quad \text{iff}\\ &\qquad v_1\;\text{-}(j_1, Op, p_1)\text{->}\;v'_1 \wedge v_2\;\text{-}(j_2, Op, p_2)\text{->}\;v'_2 \wedge\\ &\qquad Op \in \mathsf{Ops}_{1,\mathrm{C}} \cap \mathsf{Ops}_{2,\mathrm{C}}\end{aligned} \tag{6.11}$$

$$\begin{aligned}&(v_1, v_2)\;\text{-}(j_1, Op, p_1)\text{->}\;(v'_1, v'_2) \quad \text{iff}\\ &\qquad v_1\;\text{-}(j_1, Op, p_1)\text{->}\;v'_1 \wedge v_2 = v'_2 \wedge Op \in \mathsf{Ops}_{1,\mathrm{C}} - \mathsf{Ops}_{2,\mathrm{C}}\end{aligned} \tag{6.12}$$

$$\begin{aligned}&(v_1, v_2)\;\text{-}(j_2, Op, p_2)\text{->}\;(v'_1, v'_2) \quad \text{iff}\\ &\qquad v_1 = v'_1 \wedge v_2\;\text{-}(j_2, Op, p_2)\text{->}\;v'_2 \wedge Op \in \mathsf{Ops}_{2,\mathrm{C}} - \mathsf{Ops}_{1,\mathrm{C}}\end{aligned} \tag{6.13}$$

Then $\mathsf{Sys}_{(1|2),\mathrm{A}}$ is retrenched to $\mathsf{Sys}_{(1|2),\mathrm{C}}$ using retrieve, within, output and concedes relations $G_{(1|2)}$, $\{P_{Op,(1|2)}, O_{Op,(1|2)}, C_{Op,(1|2)} \mid Op \in \mathsf{Ops}_{1,\mathrm{A}} \cup \mathsf{Ops}_{2,\mathrm{A}}\}$, where:

$$G_{(1|2)}((u_1, u_2), (v_1, v_2)) \;\equiv\; [G_1(u_1, v_1) \wedge G_2(u_2, v_2)] \tag{6.14}$$

and if $Op \in \mathsf{Ops}_{1,\mathrm{A}} \cap \mathsf{Ops}_{2,\mathrm{A}}$ then:

$$\begin{aligned}&P_{Op,(1|2)}((i_1, i_2), (j_1, j_2), (u_1, u_2), (v_1, v_2)) \;\equiv\\ &\qquad [P_{Op,1}(i_1, j_1, u_1, v_1) \wedge P_{Op,2}(i_2, j_2, u_2, v_2)]\end{aligned} \tag{6.15}$$

$$\begin{aligned}&O_{Op,(1|2)}((o_1, o_2), (p_1, p_2); (u'_1, u'_2), (v'_1, v'_2), (i_1, i_2), (j_1, j_2), (u_1, u_2), (v_1, v_2)) \;\equiv\\ &\qquad [O_{Op,1}(o_1, p_1; u'_1, v'_1, i_1, j_1, u_1, v_1) \wedge\\ &\qquad\qquad O_{Op,2}(o_2, p_2; u'_2, v'_2, i_2, j_2, u_2, v_2)]\end{aligned} \tag{6.16}$$

$$\begin{aligned}&C_{Op,(1|2)}((u'_1, u'_2), (v'_1, v'_2), (o_1, o_2), (p_1, p_2); (i_1, i_2), (j_1, j_2), (u_1, u_2), (v_1, v_2)) \;\equiv\\ &\qquad [(G_1(u'_1, v'_1) \wedge O_{Op,1}(o_1, p_1; u_1, v_1, i_1, j_1, u_1, v_1) \wedge\end{aligned}$$

$$\begin{aligned}&\quad C_{Op,2}(u'_2, v'_2, o_2, p_2; i_2, j_2, u_2, v_2)) \vee \\ &(C_{Op,1}(u'_1, v'_1, o_1, p_1; i_1, j_1, u_1, v_1) \wedge \\ &\quad G_2(u'_2, v'_2) \wedge O_{Op,2}(o_2, p_2; u'_1, v'_1, i_2, j_2, u_2, v_2)) \vee \\ &(C_{Op,1}(u'_1, v'_1, o_1, p_1; i_1, j_1, u_1, v_1) \wedge \\ &\quad C_{Op,2}(u'_2, v'_2, o_2, p_2; i_2, j_2, u_2, v_2))] \end{aligned} \tag{6.17}$$

and if $Op \in (\mathsf{Ops}_{1,\mathrm{A}} - \mathsf{Ops}_{2,\mathrm{A}})$ then:

$$P_{Op,(1\|2)}(i_1, j_1, (u_1, u_2), (v_1, v_2)) \equiv [P_{Op,1}(i_1, j_1, u_1, v_1)] \tag{6.18}$$

$$\begin{aligned}&O_{Op,(1\|2)}(o_1, p_1; (u'_1, u'_2), (v'_1, v'_2), i_1, j_1, (u_1, u_2), (v_1, v_2)) \equiv \\ &\quad [O_{Op,1}(o_1, p_1; u'_1, v'_1, i_1, j_1, u_1, v_1)]\end{aligned} \tag{6.19}$$

$$\begin{aligned}&C_{Op,(1\|2)}((u'_1, u'_2), (v'_1, v'_2), o_1, p_1; i_1, j_1, (u_1, u_2), (v_1, v_2)) \equiv \\ &\quad [C_{Op,1}(u'_1, v'_1, o_1, p_1; i_1, j_1, u_1, v_1)]\end{aligned} \tag{6.20}$$

and if $Op \in (\mathsf{Ops}_{2,\mathrm{A}} - \mathsf{Ops}_{1,\mathrm{A}})$ then:

$$P_{Op,(1\|2)}(i_2, j_2, (u_1, u_2), (v_1, v_2)) \equiv [P_{Op,2}(i_2, j_2, u_2, v_2)] \tag{6.21}$$

$$\begin{aligned}&O_{Op,(1\|2)}(o_2, p_2; (u'_1, u'_2), (v'_1, v'_2), i_2, j_2, (u_1, u_2), (v_1, v_2)) \equiv \\ &\quad [O_{Op,2}(o_2, p_2; u'_2, v'_2, i_2, j_2, u_2, v_2)]\end{aligned} \tag{6.22}$$

$$\begin{aligned}&C_{Op,(1\|2)}((u'_1, u'_2), (v'_1, v'_2), o_2, p_2; i_2, j_2, (u_1, u_2), (v_1, v_2)) \equiv \\ &\quad [C_{Op,2}(u'_2, v'_2, o_2, p_2; i_2, j_2, u_2, v_2)]\end{aligned} \tag{6.23}$$

Proof sketch. The initialisation PO is trivial. Besides that, the cases covered by (6.18)-(6.23) are just the individual retrenchments with some superfluous variables. The lockstep case given by (6.15)-(6.17), has algebra much the same as that in Proposition 3.1: in Proposition 3.1 conjunction is used to combine the consequents at the two levels of the composition, while here, conjunction is used to combine the consequents for the two components in lockstep, which, moreover, is simpler, requiring no quantification. ☺

7 Asynchronous Parallel Composition

Given the preceding, it is not hard to imagine a notion of *asynchronous* parallel composition that works like the $(\mathsf{Ops}_{1,\mathrm{A}} - \mathsf{Ops}_{2,\mathrm{A}})$ and $(\mathsf{Ops}_{2,\mathrm{A}} - \mathsf{Ops}_{1,\mathrm{A}})$ parts of Proposition 6.1, provided $\mathsf{Ops}_{1,\mathrm{A}} \cap \mathsf{Ops}_{2,\mathrm{A}} = \varnothing$. However this misses a laxer variant, that tolerates one or other of the subsystems being outside the needed antecedents of Proposition 6.1 which feature the *conjunction* of the component retrieve relations. The variant we explore here features the *disjunction*.

For asynchronous parallel composition if the retrenchment data for the first retrenchment are subscripted ‘1’ and for the second ‘2’, we will subscript the composed data ‘(1+2)’. In outline, the retrenchment data for asynchronous parallel composition are:

$$G_{(1+2)} \equiv G_1 \vee G_2 \tag{7.1}$$

$$\begin{aligned}P_{Op,(1+2)} \equiv\ & G_1 \wedge P_{Op,1} && \text{if } Op \in \mathsf{Ops}_{1,\mathrm{A}} - \mathsf{Ops}_{2,\mathrm{A}} \\ & G_2 \wedge P_{Op,2} && \text{if } Op \in \mathsf{Ops}_{2,\mathrm{A}} - \mathsf{Ops}_{1,\mathrm{A}}\end{aligned} \tag{7.2}$$

$$\begin{aligned}O_{Op,(1+2)} \equiv\ & O_{Op,1} && \text{if } Op \in \mathsf{Ops}_{1,\mathrm{A}} - \mathsf{Ops}_{2,\mathrm{A}} \\ & O_{Op,2} && \text{if } Op \in \mathsf{Ops}_{2,\mathrm{A}} - \mathsf{Ops}_{1,\mathrm{A}}\end{aligned} \tag{7.3}$$

$$C_{Op,(1+2)} \equiv \begin{cases} C_{Op,1} & \text{if } Op \in \mathsf{Ops}_{1,\mathrm{A}} - \mathsf{Ops}_{2,\mathrm{A}} \\ C_{Op,2} & \text{if } Op \in \mathsf{Ops}_{2,\mathrm{A}} - \mathsf{Ops}_{1,\mathrm{A}} \end{cases} \tag{7.4}$$

The detailed result is easily proved.

Proposition 7.1 Let $\mathit{Sys}_{1,\mathrm{A}}$ (with variables u_1, i_1, o_1) be retrenched to $\mathit{Sys}_{1,\mathrm{C}}$ (with variables v_1, j_1, p_1) using G_1, $\{P_{Op,1}, O_{Op,1}, C_{Op,1} \mid Op \in \mathsf{Ops}_{1,\mathrm{A}}\}$, and $\mathit{Sys}_{2,\mathrm{A}}$ (with variables u_2, i_2, o_2) be retrenched to $\mathit{Sys}_{2,\mathrm{C}}$ (with variables v_2, j_2, p_2) using G_2, $\{P_{Op,2}, O_{Op,2}, C_{Op,2} \mid Op \in \mathsf{Ops}_{2,\mathrm{A}}\}$. Suppose $\mathsf{Ops}_{1,\mathrm{A}} \cap \mathsf{Ops}_{2,\mathrm{A}} = \varnothing$.

Let $\mathit{Sys}_{(1+2),\mathrm{A}}$ have state variable (u_1, u_2) valued in the appropriate product space, initial states satisfying $\mathit{Init}_{(1+2),\mathrm{A}}(u'_1, u'_2)$ given by the right hand side of (6.6), and let the operations of $\mathit{Sys}_{(1+2),\mathrm{A}}$ be given by (6.8), (6.9). Likewise let $\mathit{Sys}_{(1+2),\mathrm{C}}$ have state variable (v_1, v_2), initial states satisfying $\mathit{Init}_{(1+2),\mathrm{C}}(v'_1, v'_2)$ given by the right hand side of (6.10), and let the operations of $\mathit{Sys}_{(1+2),\mathrm{C}}$ be given by (6.12), (6.13).

Then $\mathit{Sys}_{(1+2),\mathrm{A}}$ is retrenched to $\mathit{Sys}_{(1+2),\mathrm{C}}$ using retrieve, within, output and concedes relations $G_{(1+2)}$, $\{P_{Op,(1+2)}, O_{Op,(1+2)}, C_{Op,(1+2)} \mid Op \in \mathsf{Ops}_{1,\mathrm{A}} \cup \mathsf{Ops}_{2,\mathrm{A}}\}$, where:

$$G_{(1+2)}((u_1, u_2), (v_1, v_2)) \equiv [G_1(u_1, v_1) \vee G_2(u_2, v_2)] \tag{7.5}$$

and if $Op \in (\mathsf{Ops}_{1,\mathrm{A}} - \mathsf{Ops}_{2,\mathrm{A}})$ then:

$$P_{Op,(1+2)}(i_1, j_1, (u_1, u_2), (v_1, v_2)) \equiv [G_1(u_1, v_1) \wedge P_{Op,1}(i_1, j_1, u_1, v_1)] \tag{7.6}$$

with $O_{Op,(1+2)}$ and $C_{Op,(1+2)}$ given by the right hand sides of (6.19), (6.20), and if $Op \in (\mathsf{Ops}_{2,\mathrm{A}} - \mathsf{Ops}_{1,\mathrm{A}})$ then:

$$P_{Op,(1+2)}(i_2, j_2, (u_1, u_2), (v_1, v_2)) \equiv [G_2(u_2, v_2) \wedge P_{Op,2}(i_2, j_2, u_2, v_2)] \tag{7.7}$$

with $O_{Op,(1+2)}$ and $C_{Op,(1+2)}$ given by the right hand sides of (6.22), (6.23).

8 Fusion Composition

When developing a complex system, it may happen that different retrenchment relationships between the same two system models may arise. Recalling that for either $\clubsuit \in \{\wedge, \vee\}$, $A \Rightarrow B$ and $C \Rightarrow D$ implies $A \clubsuit C \Rightarrow B \clubsuit D$, yields a strategy for combining such different retrenchments about the same pair of abstract and concrete systems. See Fig. 6. Both $\clubsuit$ give two results worth noting, depending on whether or not the two retrenchments share the same retrieve relation. The more straightforward disjunctive case is given first.

If the retrenchment data for the first retrenchment are subscripted '1' and for the second '2', we will subscript the composed data '(1∨2)'. In outline, the retrenchment data for disjunctive fusion composition is as follows:

$$G_{(1\vee 2)}(u, v) \equiv G_1 \vee G_2 \tag{8.1}$$

$$P_{Op,(1\vee 2)} \equiv (G_1 \vee P_{Op,2}) \wedge (P_{Op,1} \vee G_2) \wedge (P_{Op,1} \vee P_{Op,2}) \tag{8.2}$$

$$O_{Op,(1\vee 2)} \equiv (G_1 \vee O_{Op,2}) \wedge (O_{Op,1} \vee G_2) \wedge (O_{Op,1} \vee O_{Op,2}) \tag{8.3}$$

$$C_{Op,(1\vee 2)} \equiv C_{Op,1} \vee C_{Op,2} \tag{8.4}$$

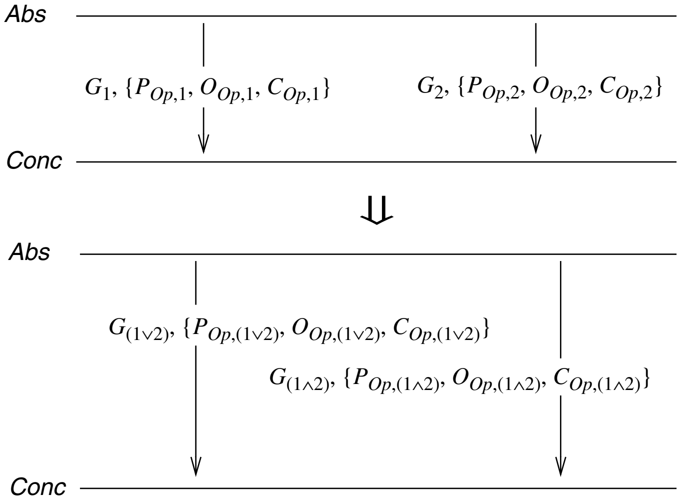

Abs
G_1, $\{P_{Op,1}, O_{Op,1}, C_{Op,1}\}$
G_2, $\{P_{Op,2}, O_{Op,2}, C_{Op,2}\}$
Conc
$\Downarrow$
Abs
$G_{(1\vee 2)}$, $\{P_{Op,(1\vee 2)}, O_{Op,(1\vee 2)}, C_{Op,(1\vee 2)}\}$
$G_{(1\wedge 2)}$, $\{P_{Op,(1\wedge 2)}, O_{Op,(1\wedge 2)}, C_{Op,(1\wedge 2)}\}$
Conc

Fig. 6. Conjunctive and disjunctive fusion compositions.

In (8.1)-(8.4) we see echoes of the familiar shapes that feature in retrenchment composition notions, except back to front. This time it is the concession that has the simple shape, whereas the within and output relations have the tripartite shape; all this is due to the disjunctive combination of the retrieve relations in (8.1). If the two retrieve relations happen to be the same, the more complex formulae simplify: the first two conjuncts in (8.2) and (8.3) are erased.

Proposition 8.1 Let *Abs* be retrenched to *Conc* using G_1, $\{P_{Op,1}, O_{Op,1}, C_{Op,1} \mid Op \in \mathsf{Ops}_\mathrm{A}\}$ (with the usual variables). Let *Abs* also be retrenched to *Conc* using G_2, $\{P_{Op,2}, O_{Op,2}, C_{Op,2} \mid Op \in \mathsf{Ops}_\mathrm{A}\}$ (with the usual variables). Then *Abs* is retrenched to *Conc* also via $G_{(1\vee 2)}$ and $\{P_{Op,(1\vee 2)}, O_{Op,(1\vee 2)}, C_{Op,(1\vee 2)} \mid Op \in \mathsf{Ops}_\mathrm{A}\}$ where:

$$G_{(1\vee 2)}(u, v) \equiv [G_1(u, v) \vee G_2(u, v)] \tag{8.5}$$

$$\begin{aligned} P_{Op,(1\vee 2)}(i, j, u, v) \equiv \\ &[(G_1(u, v) \vee P_{Op,2}(i, j, u, v)) \wedge \\ &(P_{Op,1}(i, j, u, v) \vee G_2(u, v)) \wedge \\ &(P_{Op,1}(i, j, u, v) \vee P_{Op,2}(i, j, u, v))] \end{aligned} \tag{8.6}$$

$$\begin{aligned} O_{Op,(1\vee 2)}(o, p; u', v', i, j, u, v) \equiv \\ &[(G_1(u', v') \vee O_{Op,2}(o, p; u', v', i, j, u, v)) \wedge \\ &(O_{Op,1}(o, p; u', v', i, j, u, v) \vee G_2(u', v')) \wedge \\ &(O_{Op,1}(o, p; u', v', i, j, u, v) \vee O_{Op,2}(o, p; u', v', i, j, u, v))] \end{aligned} \tag{8.7}$$

$$C_{Op,(1\vee 2)}(u', v', o, p; i, j, u, v) \equiv \\ \quad [C_{Op,1}(u', v', o, p; i, j, u, v) \vee C_{Op,2}(u', v', o, p; i, j, u, v)] \tag{8.8}$$

Proof sketch. The disjunction of the PO antecedents for the two retrenchments gives (8.5), (8.6). The disjunction of the consequents combines the existential quantifications, and then the abstract steps via distributivity, after which boolean algebra gives (8.7), (8.8). ☺

If the two retrenchments share the same retrieve relation, this simplifies as stated above.

Corollary 8.2 Let $G_1 = G_2$ in Proposition 8.1. Then the composition reduces to:

$$P_{Op,(1\vee 2)}(i, j, u, v) \equiv [P_{Op,1}(i, j, u, v) \vee P_{Op,2}(i, j, u, v)] \tag{8.9}$$

$$O_{Op,(1\vee 2)}(o, p; u', v', i, j, u, v) \equiv \\ \quad [O_{Op,1}(o, p; u', v', i, j, u, v) \vee O_{Op,2}(o, p; u', v', i, j, u, v)] \tag{8.10}$$

$$C_{Op,(1\vee 2)}(u', v', o, p; i, j, u, v) \equiv \\ \quad [C_{Op,1}(u', v', o, p; i, j, u, v) \vee C_{Op,2}(u', v', o, p; i, j, u, v)] \tag{8.11}$$

Note that we could happily replace one or more of the disjunctions in (8.6) by either disjunct alone or by conjunctions of them as these just strengthen the antecedents of the relevant PO. Similarly for the RHS of (8.9).

The conjunctive fusion of retrenchments has a more familiar form; we subscript the composed data '(1∧2)'. Here it is in outline form:

$$G_{(1\wedge 2)} \equiv G_1 \wedge G_2 \tag{8.12}$$

$$P_{Op,(1\wedge 2)} \equiv P_{Op,1} \wedge P_{Op,2} \tag{8.13}$$

$$O_{Op,(1\wedge 2)} \equiv O_{Op,1} \wedge O_{Op,2} \tag{8.14}$$

$$C_{Op,(1\wedge 2)} \equiv (G_1 \wedge O_{Op,1} \wedge C_{Op,2}) \vee (C_{Op,1} \wedge G_2 \wedge O_{Op,2}) \vee (C_{Op,1} \wedge C_{Op,2}) \tag{8.15}$$

However (8.12)-(8.15) come with a caveat on joint simulability. Here are the details, starting with a preliminary definition.

Definition 8.3 Let *Abs* be retrenched to *Conc* using G_1, $\{P_{Op,1}, O_{Op,1}, C_{Op,1} \mid Op \in \mathsf{Ops}_\mathrm{A}\}$ (with the usual variables). Let *Abs* also be retrenched to *Conc* using G_2, $\{P_{Op,2}, O_{Op,2}, C_{Op,2} \mid Op \in \mathsf{Ops}_\mathrm{A}\}$ (with the usual variables). If for any $Op \in \mathsf{Ops}_\mathrm{A}$ and u, v, i, j, v', p, such that:

$$G_1(u, v) \wedge G_2(u, v) \wedge P_{Op,1}(i, j, u, v) \wedge P_{Op,2}(i, j, u, v) \wedge stp_{Op_\mathrm{C}}(v, j, v', p) \tag{8.16}$$

holds, we have:

$$\varnothing \neq \{(u', o) \mid (u \text{ -}(i, Op_{\mathrm{A},1}, o)\text{-› } u') \ \Sigma^1 \ (v \text{ -}(j, Op_{\mathrm{C},1}, p)\text{-› } v')\} \cap \\ \quad \{(u', o) \mid (u \text{ -}(i, Op_{\mathrm{A},2}, o)\text{-› } u') \ \Sigma^1 \ (v \text{ -}(j, Op_{\mathrm{C},2}, p)\text{-› } v')\} \tag{8.17}$$

then we say the two retrenchments from *Abs* to *Conc* are close to cosimulating.

Proposition 8.4 Let *Abs* be retrenched to *Conc* using G_1, $\{P_{Op,1}, O_{Op,1}, C_{Op,1} \mid Op \in \mathsf{Ops}_\mathrm{A}\}$ (with the usual variables). Let *Abs* also be retrenched to *Conc* using G_2, $\{P_{Op,2}, O_{Op,2}, C_{Op,2} \mid Op \in \mathsf{Ops}_\mathrm{A}\}$ (with the usual variables). Suppose the two retrenchments from *Abs* to *Conc* are close to cosimulating. Then *Abs* is retrenched to *Conc* also via $G_{(1\wedge 2)}$ and $\{P_{Op,(1\wedge 2)}, O_{Op,(1\wedge 2)}, C_{Op,(1\wedge 2)} \mid Op \in \mathsf{Ops}_\mathrm{A}\}$ where:

$$G_{(1\wedge 2)}(u, v) \equiv [G_1(u, v) \wedge G_2(u, v)] \tag{8.18}$$

$$P_{Op,(1\wedge 2)}(i, j, u, v) \equiv [P_{Op,1}(i, j, u, v) \wedge P_{Op,2}(i, j, u, v)] \tag{8.19}$$

$$\begin{aligned} &O_{Op,(1\wedge 2)}(o, p; u', v', i, j, u, v) \equiv \\ &\quad [O_{Op,1}(o, p; u', v', i, j, u, v) \wedge O_{Op,2}(o, p; u', v', i, j, u, v)] \end{aligned} \tag{8.20}$$

$$\begin{aligned} &C_{Op,(1\wedge 2)}(u', v', o, p; i, j, u, v) \equiv \\ &\quad [(G_1(u', v') \wedge O_{Op,1}(o, p; u', v', i, j, u, v) \wedge C_{Op,2}(u', v', o, p; i, j, u, v)) \vee \\ &\quad (C_{Op,1}(u', v', o, p; i, j, u, v) \wedge G_2(u', v') \wedge O_{Op,2}(o, p; u', v', i, j, u, v)) \vee \\ &\quad (C_{Op,1}(u', v', o, p; i, j, u, v) \wedge C_{Op,2}(u', v', o, p; i, j, u, v))] \end{aligned} \tag{8.21}$$

Proof sketch. The conjunction of the PO antecedents for the two retrenchments gives (8.18), (8.19). The conjunction of the consequents exploits the close to cosimulation of the two retrenchments to deduce that whenever the conjunction of PO antecedents holds for some u, v, i, j, v', p, a joint witnessing u', o can be found for the two PO consequents. Hence the conjunction of existential quantifications can be combined into a single quantification, after which boolean algebra gives (8.20), (8.21). ☺

Note that the restriction of the above to the same retrieve relation case, merely removes the need for (8.18), and simplifies (8.21) in the obvious way, so we do not quote it separately.

The above is, up to a point, reminiscent of the schema calculus of Z, the detailed differences hinging on the precise lexical mechanisms used to identify (and to keep distinct) various pieces of the two components, as well as the fact that the Z schema calculus has a wider remit anyway. One can make this analogy more extensive, and thereby bring the present mechanism closer to synchronous parallel composition too, by allowing state (or other) spaces to overlap[3] rather than coincide exactly, and allowing the sets of operation names to overlap rather than coincide exactly. The notational ramifications of this are rather cumbersome so we do not go into details. And whereas the disjunctive version of such a generalised fusion can be carried through relatively straightforwardly, the conjunctive version involves the kind of additional constraints we saw in Proposition 8.4.

9 Associativity, Coherence, Stronger Compositions

With any notion of composition comes the issue of associativity. Once soundness is proved, associativity reduces to an algebraic problem of performing the composition two ways and checking the equivalence of the outcomes. We see that in most of the cases dealt with above, the structural forms of the compositions are very similar, differing in which variables are identified or not, and which of the identified ones are quantified over. The symmetry of the expressions derived is a big help in showing associativity. We give the treatment of some typical cases, leaving the rest as obvious generalisations.

[3] This means viewing each state (or other) space as a cartesian product, and identifying common factors in the overlapping spaces, (on the basis that these cartesian factors carry the values of lexically identical variables of the two systems, for example).

Consider (5.2), which defines $P_{(Op,1 \rangle Op,2)}$ as $P_{Op,1}$. Since the first element of a sequence is the same regardless of the assembly order of the sequence, associativity follows. Similar arguments yield:

Proposition 9.1 The compositions in (5.2), (6.18)-(6.23), (7.6), (7.7), are associative. ☺

Consider (8.5), which defines $G_{(1 \vee 2)}$ as $(G_1 \vee G_2)$. Substituting this into $G_{((1 \vee 2) \vee 3)} = (G_{(1 \vee 2)} \vee G_3)$ yields $((G_1 \vee G_2) \vee G_3)$, equivalent to the other association order, so associativity follows. The same holds if $\vee$ is replaced by $\wedge$, and if the variables that occur in the various predicates are made distinct, and/or quantified over in the ways that occur above, since the fact that we deal with *distinct* systems/retrenchments etc., enables us to avoid any bound variable capture problems.

Proposition 9.2 The compositions in (3.5)-(3.7), (4.7), (5.3), (6.6), (6.10), (6.14)-(6.16), (7.5), (8.5), (8.8)-(8.11), (8.18)-(8.20), are associative. ☺

Consider (8.21). Suppressing the $\wedge$'s to save space, this defines $C_{Op,(1 \wedge 2)}$ as $(G_1 O_{Op,1} C_{Op,2} \vee C_{Op,1} G_2 O_{Op,2} \vee C_{Op,1} C_{Op,2})$. Since various G's and O's occur in this, associativity for $C_{Op,(1 \wedge 2)}$ depends on the composition laws for the G's and O's, given by (8.18), (8.20). Substituting these, and $C_{Op,(1 \wedge 2)}$, into $C_{Op,((1 \wedge 2) \wedge 3)}$ as given by $(G_{(1 \wedge 2)} O_{Op,(1 \wedge 2)} C_{Op,3} \vee C_{Op,(1 \wedge 2)} G_3 O_{Op,3} \vee C_{Op,(1 \wedge 2)} C_{Op,3})$ yields, after some working:

$$\begin{aligned} & C_{Op,((1 \wedge 2) \wedge 3)} = \\ & \quad [G_1 O_{Op,1} G_2 O_{Op,2} C_{Op,3} \vee G_1 O_{Op,1} C_{Op,2} G_3 O_{Op,3} \vee C_{Op,1} G_2 O_{Op,2} G_3 O_{Op,3} \\ & \quad \vee G_1 O_{Op,1} C_{Op,2} C_{Op,3} \vee C_{Op,1} G_2 O_{Op,2} C_{Op,3} \vee C_{Op,1} C_{Op,2} G_3 O_{Op,3} \\ & \quad \vee C_{Op,1} C_{Op,2} C_{Op,3}] \end{aligned} \tag{9.1}$$

which is easily seen to be symmetric in the indices 1, 2, 3. Therefore the other association order will yield the same result. As previously, the use of distinct and/or quantified variables for the cases that occur in previous sections will not spoil associativity. Neither will the interchange of $\wedge$ and $\vee$, nor cases where the retrieve relation is the same or absent.

Proposition 9.3 The compositions in (3.8), (4.8), (4.10), (5.4), (8.6), (8.7), (8.21), are associative. ☺

The above covers everything except (4.5). However it is not hard to see by explicit calculation that the two association orders for (4.5) yield:

$$\begin{aligned} & P_{((1;2);3)} = P_{((1;2);3)} = \\ & \quad [G_1 P_{Op,1} \wedge \mathrm{wp}(\Sigma^1(Op_1), G_2 P_{Op,2}) \wedge \\ & \qquad \mathrm{wp}(\Sigma^1(Op_1), \mathrm{wp}(\Sigma^1(Op_2), G_3 P_{Op,3}))] \end{aligned} \tag{9.2}$$

which relies on the compositionality and associativity of the wp set transformer.

Proposition 9.4 The composition in (4.5) is associative. ☺

With associativity covered for each of the compositions, there arises the additional question of whether the different composition methods cohere. In other words if two systems are combined using one technique, and the result combined with a third system using another technique, is the answer equivalent to doing the second composition earlier and the first composition later? In view of the structural similiarity of the composition laws in all the cases examined, and the inevitable disjointness of the variables quantified over in different compositions, we claim the answer is affirmative, at least up to natural isomorphisms such as the one needed to identify $((u_1, u_2), u_3)$ with $(u_1, (u_2, u_3))$ in Section 6.

In each of the compositions treated, boolean algebra was the guiding light, and led to an easy treatment of associativity. However, the presence of the disjunction in the retrenchment PO consequent, and the use of the distributive law in forming the composed concedes relations, can lead to a rapid proliferation of cases, usually in a composed C (cf. (9.1)). Many of these need not contain useful facts about the systems of interest. (Their presence is innocuous provided *some* top level disjunct of C contains valid information whenever C is needed.) By judicious strengthening of the output and concedes relations with information from the PO antecedent (in effect bringing the PO closer to the simulation relation Σ^1) these effects can be controlled. However associativity (and thus inevitably coherence in general) become more difficult issues. For pure vertical composition, these matters have been studied in some depth in [Banach and Jeske (2002), Banach and Poppleton (2003)]. Given the structural similarities between the various compositions, the theory of stronger compositions for the other composition techniques will be similar.

10 Composition of Retrenchment with Refinement

One of the goals of retrenchment is to coexist smoothly and fruitfully with refinement, so that a development process can get the benefits of both: the strength of reasoning control offered by refinement, together with the expressivity of model evolution offered by retrenchment. For this we formulate a notion of refinement whose structure is in sympathy with the POs adopted for retrenchment. Refinement will thus be characterised by a forward simulation criterion, namely by the usual initialisation PO (2.1), and the following operation PO:

$$\begin{aligned}&G(u, v) \land In_{Op}(i, j) \land stp_{Op_C}(v, j, v', p) \;\Rightarrow\\ &\qquad(\exists\, u', o \bullet stp_{Op_A}(u, i, u', o) \land G(u', v') \land Out_{Op}(o, p))\end{aligned} \tag{10.1}$$

This is convenient, as with some mild additional assumptions on In_{Op}, Out_{Op}, the definition can be brought close to other refinement definitions. We assume for compatibility, that the sets of operations at the abstract and concrete levels are in 1-1 correspondence.

One can now ask how do such refinements and retrenchments compose? In the case of output retrenchment, one can view (10.1) as a degenerate retrenchment PO with null concession and suitably restricted P_{Op} and O_{Op}, and then use vertical composition. However this approach does not extend to primitive retrenchment (which has no O_{Op} relations), or to many other forms, differing in the shape of the defining PO, that can be imagined. We want a composition policy with refinement that extends to all variants of retrenchment. Therefore we proceed as follows.

One can see a refinement characterised by (2.1) and (10.1), as providing relations $G(u, v)$, $In_{Op}(i, j)$, $Out_{Op}(o, p)$ between the state spaces, input spaces, output spaces respectively, at the two levels. One can view these as a translation mechanism for mapping any predicate W in the abstract (resp. concrete) world to the concrete (resp. abstract) one: we just take the relational image of W through an appropriate cartesian product of $G(u, v)$, $In_{Op}(i, j)$, $Out_{Op}(o, p)$ relations. So we use a copy of $G(u, v)$ for each occurrence of a state variable in W, a copy of $In_{Op}(i, j)$ for each occurrence of an input variable in W, a copy of $Out_{Op}(o, p)$ for each occurrence of an output variable in W. We use this to fuel the definition of a retrenchment/refinement composition.

Proposition 10.1 Let there be a retrenchment from *Abs* to *Conc* (using the usual variables) given by G_{T}, $\{P_{Op,\mathrm{T}}, O_{Op,\mathrm{T}}, C_{Op,\mathrm{T}} \mid Op \in \mathsf{Ops}_{\mathrm{A}}\}$. Let there be a refinement from *Conc* to *Imp* (the 'implementation' system, with variables w, k, q) given by G_{F}, $\{In_{Op,\mathrm{F}}, Out_{Op,\mathrm{F}} \mid Op \in \mathsf{Ops}_{\mathrm{C}}\}$. Then there is a retrenchment from *Abs* to *Imp* given by $G_{(\mathrm{T,F})}$, $\{P_{Op,(\mathrm{T,F})}, O_{Op,(\mathrm{T,F})}, C_{Op,(\mathrm{T,F})} \mid Op \in \mathsf{Ops}_{\mathrm{A}}\}$, where:

$$G_{(\mathrm{T,F})}(u, w) \equiv [\exists\, v \bullet G_{\mathrm{T}}(u, v) \land G_{\mathrm{F}}(v, w)] \tag{10.2}$$

$$\begin{aligned}&P_{Op,(\mathrm{T,F})}(i, k, u, w) \equiv \\ &\quad [\exists\, v, j \bullet G_{\mathrm{T}}(u, v) \land G_{\mathrm{F}}(v, w) \land P_{Op,\mathrm{T}}(i, j, u, v) \land In_{Op,\mathrm{F}}(j, k)]\end{aligned} \tag{10.3}$$

$$\begin{aligned}&O_{Op,(\mathrm{T,F})}(o, q; u', w', i, k, u, w) \equiv \\ &\quad [\exists\, v', p, v, j \bullet O_{Op,\mathrm{T}}(o, p; u', v', i, j, u, v) \land \\ &\qquad Out_{Op,\mathrm{F}}(p, q) \land G_{\mathrm{F}}(v', w') \land In_{Op,\mathrm{F}}(j, k) \land G_{\mathrm{F}}(v, w)]\end{aligned} \tag{10.4}$$

$$\begin{aligned}&C_{Op,(\mathrm{T,F})}(u', w', o, q; i, k, u, w) \equiv \\ &\quad [\exists\, v', p, v, j \bullet C_{Op,\mathrm{T}}(u', v', o, p; i, j, u, v) \land \\ &\qquad G_{\mathrm{F}}(v', w') \land Out_{Op,\mathrm{F}}(p, q) \land In_{Op,\mathrm{F}}(j, k) \land G_{\mathrm{F}}(v, w)]\end{aligned} \tag{10.5}$$

Proposition 10.2 Let there be a refinement from *Pre* (the 'preliminary' system, with variables t, h, n) to *Abs* given by G_{F}, $\{In_{Op,\mathrm{F}}, Out_{Op,\mathrm{F}} \mid Op \in \mathsf{Ops}_{\mathrm{A}}\}$. Let there be a retrenchment from *Abs* to *Conc* (using the usual variables) given by G_{T}, $\{P_{Op,\mathrm{T}}, O_{Op,\mathrm{T}}, C_{Op,\mathrm{T}} \mid Op \in \mathsf{Ops}_{\mathrm{A}}\}$. Then there is a retrenchment from *Pre* to *Conc* given by $G_{(\mathrm{F,T})}$, $\{P_{Op,(\mathrm{F,T})}, O_{Op,(\mathrm{F,T})}, C_{Op,(\mathrm{F,T})} \mid Op \in \mathsf{Ops}_{\mathrm{A}}\}$, where:

$$G_{(\mathrm{F,T})}(t, v) \equiv [\exists\, u \bullet G_{\mathrm{F}}(t, u) \land G_{\mathrm{T}}(u, v)] \tag{10.6}$$

$$\begin{aligned}&P_{Op,(\mathrm{F,T})}(h, j, t, v) \equiv \\ &\quad [\exists\, u, i \bullet G_{\mathrm{F}}(t, u) \land G_{\mathrm{T}}(u, v) \land In_{Op,\mathrm{F}}(h, i) \land P_{Op,\mathrm{T}}(i, j, u, v)]\end{aligned} \tag{10.7}$$

$$\begin{aligned}&O_{Op,(\mathrm{F,T})}(n, p; t', v', h, j, t, v) \equiv \\ &\quad [\exists\, u', o, u, i \bullet Out_{Op,\mathrm{F}}(n, o) \land G_{\mathrm{F}}(t', u') \land In_{Op,\mathrm{F}}(h, i) \land G_{\mathrm{F}}(t, u) \land \\ &\qquad O_{Op,\mathrm{T}}(o, p; u', v', i, j, u, v)]\end{aligned} \tag{10.8}$$

$$\begin{aligned}&C_{Op,(\mathrm{F,T})}(u', w', o, q; i, k, u, w) \equiv \\ &\quad [\exists\, u', o, u, i \bullet G_{\mathrm{F}}(t', u') \land Out_{Op,\mathrm{F}}(n, o) \land In_{Op,\mathrm{F}}(h, i) \land G_{\mathrm{F}}(t, u) \land \\ &\qquad C_{Op,\mathrm{T}}(u', v', o, p; i, j, u, v)]\end{aligned} \tag{10.9}$$

Proof sketches. The proofs of the above follow the style of Proposition 3.1. Initialisation is trivial. Then one starts with the lower system, exploits the relevant PO to assert the relevant property of a transition of the intermediate system, and then proceeds to exploit the other PO. Since there is only one component retrenchment, the forms (10.2)-(10.9) emerge from a top level case analysis, rather than needing boolean algebra. ☺

With the basics established, let us restrict to output retrenchment, and, taking a refinement to be a degenerate retrenchment with false concession, compare the compositions (10.2)-(10.9) with the vertical composition of Section 3. We see that the forms (10.2)-(10.9) differ slightly from those in (3.5)-(3.8) when a false concession is folded in to the latter. While the retrieve relations and within relations compose identically (overlooking the different signatures of P_{Op} and In_{Op}), the formula for the output relation features additional occurrences of $G_F{}' \wedge In_{Op,F} \wedge G_F$ (in both (10.4) and (10.8)) compared with (3.7), and the formula for the concedes relation features additional occurrences of $In_{Op,F} \wedge G_F$ (in both (10.5) and (10.9)) compared with (3.8). These additional occurrences of elements from the PO antecedent justify viewing the present compositions as stronger vertical compositions. Note that the strengthenings are as benign as can be, in that both strengthened and unstrengthened forms of (10.4), (10.5) and (10.8), (10.9) lead to the same simulation relation for the combined retrenchment/refinement. Note furthermore that because both (3.5)-(3.8) and (10.2)-(10.9) are sound, we have a choice of composition for these cases, underlining again that compositions are a matter for definition.

The heterogeneous compositions we have defined, indicate that associativity is really a coherence issue here. We note that the compositions (10.2)-(10.9) are all pure relational compositions, which points to easy associativity. Thus a refinement composed with a retrenchment composed with a further refinement, reduces to two successive bouts of relational composition for each constituent relation. Moreover a retrenchment composed with a refinement composed with a further retrenchment, can be seen as yielding a concession like (9.1), but with occurrences[4] of $G_2{}'O_{Op,2}$ strengthened by $In_{Op,2}G_2$, and terms containing $C_{Op,2}$ erased; this for either association order.

Proposition 10.3 The compositions (10.2)-(10.9) of retrenchments with refinements are associatively coherent.

11 Decomposition

The counterpart of composition is decomposition. One can search for conditions that capture the inverses of all the constructions given above. However it is more natural to look for decompositions based on the likely uses of retrenchment, in particular on its potential for being 'like refinement except round the edges' (see [Banach et al. (2007)]). This points to decomposing operations and retrenchment data round different parts of their activity, primarily by partitioning the operations' domains in appropriate ways.

Proposition 11.1 Let there be a retrenchment from *Abs* to *Conc* (using the usual variables) given by G, $\{P_{Op}, O_{Op}, C_{Op} \mid Op \in \mathsf{Ops}_A\}$. Suppose for $Op \in \mathsf{Ops}_A$:

$$\begin{aligned}\mathrm{dom}(Op_A) &= \{(u,i) \mid \exists\, u', o \bullet stp_{Op_A}(u,i,u',o)\}\\ &= a_{Op,1} \cup a_{Op,2} \cup \ldots \cup a_{Op,K_{Op}} \qquad (11.1)\\ \mathrm{dom}(Op_C) &= \{(v,j) \mid \exists\, v', p \bullet stp_{Op_C}(v,j,v',p)\}\\ &= c_{Op,1} \cup c_{Op,2} \cup \ldots \cup c_{Op,L_{Op}} \qquad (11.2)\end{aligned}$$

Let $Op_{A,k}$ and $Op_{C,l}$ be names for suboperations with step relations as follows:

$$stp_{Op_{A,k}} = a_{Op,k} \lhd stp_{Op_A} \quad ; \quad stp_{Op_{C,l}} = c_{Op,l} \lhd stp_{Op_C} \qquad (11.3)$$

where $\lhd$ is domain restriction. Then:

[4] The after-state prime introduced for clarity.

$$stp_{Op_{\mathrm{A}}} = \bigcup\nolimits_{1 \le k \le K_{Op}} stp_{Op_{\mathrm{A},k}} \quad ; \quad stp_{Op_{\mathrm{C}}} = \bigcup\nolimits_{1 \le l \le L_{Op}} stp_{Op_{\mathrm{C},l}} \tag{11.4}$$

Let:

$$P_{Op,kl} = \{(i, j, u, v) \in P_{Op} \mid (u, i) \in a_{Op,k}, (v, j) \in c_{Op,l}\} \tag{11.5}$$

$$\begin{aligned} O_{Op,kl} &= \{(o, p; \ldots) \in O_{Op} \mid stp_{Op_{\mathrm{A},k}}(u, i, u', o), stp_{Op_{\mathrm{C},l}}(v, j, v', p)\} \\ &\subseteq O^{+}{}_{Op,kl} \subseteq O_{Op} \end{aligned} \tag{11.6}$$

$$\begin{aligned} C_{Op,kl} &= \{(u', v', o, p; \ldots) \in C_{Op} \mid stp_{Op_{\mathrm{A},k}}(u, i, u', o), stp_{Op_{\mathrm{C},l}}(v, j, v', p)\} \\ &\subseteq C^{+}{}_{Op,kl} \subseteq C_{Op} \end{aligned} \tag{11.7}$$

Then for all $1 \le k \le K_{Op}$, $1 \le l \le L_{Op}$:

(1) $Op_{\mathrm{A},k}$ is retrenched to $Op_{\mathrm{C},l}$ via G, $P_{Op,kl}$, $O_{Op,kl}$, $C_{Op,kl}$.

(2) $Op_{\mathrm{A},k}$ is retrenched to $Op_{\mathrm{C},l}$ via G, $P_{Op,kl}$, $O^{+}{}_{Op,kl}$, $C^{+}{}_{Op,kl}$.

(3) $Op_{\mathrm{A},k}$ is retrenched to $Op_{\mathrm{C},l}$ via G, $P_{Op,kl}$, O_{Op}, C_{Op}.

Proof sketch. That (11.4) holds is immediate. That (1)-(3) hold follows from the original operation PO for *Op*. ☺

Note that the unions in (11.1), (11.2) need not be disjoint, though the disjoint case is highly relevant to a decomposition strategy. A disjunctive fusion composition converse to Proposition 11.1 is worth recording.

Proposition 11.2 Let there be a retrenchment from *Abs* to *Conc* (with the usual variables), but using a naming convention that groups operations into families of suboperations belonging to a main operation name, and allowing retrenchment data between arbitrary suboperations of main operation names' abstract and concrete families. Thus the abstract suboperation names are $\{Op_{\mathrm{A},k} \mid 1 \le k \le K_{Op}, Op \in \mathsf{Ops}_{\mathrm{A}}\}$, and the relevant concrete ones are $\{Op_{\mathrm{C},l} \mid 1 \le l \le L_{Op}, Op \in \mathsf{Ops}_{\mathrm{A}}\}$. The retrenchment itself is given by G, and $\{P_{Op,kl}, O_{Op,kl}, C_{Op,kl} \mid 1 \le k \le K_{Op}, 1 \le l \le L_{Op}, Op \in \mathsf{Ops}_{\mathrm{A}}\}$, so that each individual suboperation PO holds for all $1 \le k \le K_{Op}$, $1 \le l \le L_{Op}$, $Op \in \mathsf{Ops}_{\mathrm{A}}$. For $Op \in \mathsf{Ops}_{\mathrm{A}}$, define operations Op_{A}, Op_{C} by:

$$stp_{Op_{\mathrm{A}}} = \bigcup\nolimits_{1 \le k \le K_{Op}} stp_{Op_{\mathrm{A},k}} \quad ; \quad stp_{Op_{\mathrm{C}}} = \bigcup\nolimits_{1 \le l \le L_{Op}} stp_{Op_{\mathrm{C},l}} \tag{11.8}$$

Then for all $1 \le k \le K_{Op}$, $1 \le l \le L_{Op}$:

(1) Op_{A} is retrenched to $Op_{\mathrm{A},k}$ via $G = \mathrm{id}_{\mathsf{U}}$, $P_{Op} = \mathrm{id}_{\mathsf{U} \times \mathsf{I}}$, $O_{Op} = \mathrm{id}_{\mathsf{U} \times \mathsf{I} \times \mathsf{U} \times \mathsf{O}}$, $C_{Op} = \varnothing$.

(2) $Op_{\mathrm{A},k}$ is retrenched to Op_{A} via $G = \mathrm{id}_{\mathsf{U}}$, $P_{Op} = \mathrm{id}_{\mathrm{dom}(Op_{\mathrm{A},k}) - \bigcup\{\mathrm{dom}(Op_{\mathrm{A},k'}) | k' \ne k\}}$, $O_{Op} = \mathrm{id}_{\mathsf{U} \times \mathsf{I} \times \mathsf{U} \times \mathsf{O}}$, $C_{Op} = \varnothing$.

(3) Analogous results for the concrete (sub)operations.

(4) Op_{A} is retrenched to Op_{C} (in terms of the original data) via
G, $P_{Op} = P_{Op,kl} \rhd (\mathrm{dom}(Op_{\mathrm{C},l}) - \bigcup\{\mathrm{dom}(Op_{\mathrm{C},l'}) \mid l' \ne l\})$, $O_{Op,kl}$, $C_{Op,kl}$

(5) For $Op \in \mathsf{Ops}_{\mathrm{A}}$, defining:

$$P_{Op} = \bigcup\nolimits_{1 \le k \le K_{Op}, 1 \le l \le L_{Op}} P_{Op,kl} \tag{11.9}$$

$$O_{Op} = \bigcup\nolimits_{1 \le k \le K_{Op}, 1 \le l \le L_{Op}} O_{Op,kl} \tag{11.10}$$

$$C_{Op} = \bigcup\nolimits_{1 \le k \le K_{Op}, 1 \le l \le L_{Op}} C_{Op,kl} \tag{11.11}$$

then Op_{A} is retrenched to Op_{C} via G, P_{Op}, O_{Op}, C_{Op}.

Proof sketch. Claims (1), (2), (3) reduce to obvious refinements. For a given *Op*, *k*, *l*, claim (4) is a composition according to Proposition 10.1 and Proposition 10.2, of the refinement in (1), the given *k*, *l* retrenchment, and the concrete version of the refinement in (2). Claim (5) follows by disjunctive reasoning, as in Corollary 8.2. ☺

12 Application Areas

In preceding sections, we focused on the technical details of a variety of composition mechanisms for retrenchments. In this one, we look outwards, to outline the utility of these mechanisms in the system engineering context. We do not deal with applications in detail, which would unbalance the present paper, rather we talk about applicability in general terms and point to more detailed work elsewhere.

Vertical composition hardly needs justification of course, as the idea of developing a system by proceeding from the highest level abstraction towards implementation via incremental stages is such an old one. From a system engineering point of view, the most salient point as regards propositionally driven retrenchment composition, is the potentially rapid proliferation of top level disjuncts in composed concessions, of which many can be redundant, as noted in Section 9. The previously cited [Banach and Jeske (2002), Banach and Poppleton (2003)] explore this in some detail and offer appropriate remedies. The decompositions treated in Section 11 can also help to counteract this proliferation, by subdividing operations' concessions into finer grained pieces that can be judiciously recombined to avoid 'junk'.

Horizontal composition, and its close ally the simulation relation, are intriguingly different in retrenchment as compared with refinement, particularly as regards loss of standard inductive reasoning. The horizontal composition result that we proved, squeezes the permitted departure points for composed operations, via a fairly stringent composed within relation.

Since horizontal compositions of the simulation relation can hold even if the departure points do not fall in the permitted squeezed area, an ad hoc approach for understanding how different parts of two systems in a retrenchment relationship are able to simulate one another offers the most productive way through the simulation landscape. Finite inductions, such as are used to establish loop termination for example, come closest to replacing the standard inductions for horizontal reasoning of refinement. This, and the other results in Section 4 illuminate rather well the nature of the territory between provable horizontal composition and the simulation relation.

Being a natural outgrowth of horizontal composition, dataflow composition combines neatly with synchronous parallel composition to give a flexible mechanism for composing development/evolution steps for subcomponents, into a development/evolution step for the system as a whole (for a single pair of abstraction layers, and at the semantic level). It is not hard to see that if the inputs and outputs of subcomponents are suitably factorised, the subinputs and suboutputs can be connected up at will to form a wide variety of dataflow networks.[5] The technique is most easily applied when the graph of subcomponents is acyclic. The application of this to eg. cir-

[5] We did not pursue this extension in Sections 5 and 6 to avoid notational clutter.

cuit design, is not hard to imagine, and has been exploited using the simulation relations of the composed retrenchments for fault tree extraction [Banach and Cross (2004), Banach and Bozzano (2006), Banach and Bozzano (2007a), Banach and Bozzano (2007b)], the last of these showing that the cyclic case can be handled unproblematically too. There is certainly no reason why other analyses of the simulation relations of composed retrenchments should not also yield fruitful outcomes.

The asynchronous parallel composition we sketched finds application in the development of combinations of independent units of functionality, thus being related to promotion in Z terminology [Spivey (1992), Potter et al. (1996)]. The antecedent of its PO allows for the possibility that some of the components have already passed into a non-simulable condition, unlike the synchronous version.

Fusion composition is as already noted, reminiscent of the schema calculus of Z (cf. [Woodcock and Davies (1996)], Chapter 17) but adapted for retrenchment. Due to its more focused remit in dealing with the relationship between transition systems, there is a less visible need for an analogue of schema negation; i.e. how interesting can it be to say that there is *not* a retrenchment relationship between two steps when retrenchment is already so flexible? A further application of fusion composition arises in viewpoint composition, in which different retrenchments between two systems focus on different aspects of their relationship.

The compositions of retrenchments with refinements enable a number of system development scenarios to be cast as generic algebraic problems, which can be solved once and for all. For example, suppose a system Abs is refined to an implementation Ref, and subsequently the definition of Abs is evolved to accommodate new requirements, giving a retrenched system Ret. Can one do the necessary refinement of Ret to get a new implementation automatically? The affirmative answer to this question and others like it appears in [Jeske (2005)]. Other relevant works are [Banach (2000), Jeske and Banach (2002)]. These constructions, the technical details of which can get surprisingly arduous, all rest on the compositions of retrenchments with refinements studied in Section 10.

The technical points noted in the last three paragraphs were all key ingredients in the creation of the *Tower Pattern*, and its application to the Mondex Electronic Purse development [Stepney et al. (2000)]. The tower, introduced in the context of Mondex in [Banach et al. (2005)] and further exploited for other Mondex requirements issues in [Banach et al. (2006a), Banach et al. (2006b)] is the ‘applications nickname’ given to the various square completion problems solved in [Jeske (2005)]. For Mondex, the detailed application of these concepts needed to be reinforced by the other ideas because of the structure of the original Mondex development.

Finally, the decomposition mechanisms described in the previous section open the door to capturing many aspects of finegrained requirements reasoning via a selection of retrenchments. In [Poppleton and Banach (2003)] decomposition is combined with the algebraic techniques just highlighted, to show how a spectrum of requirements issues, falling beyond the usual scope of refinement, can be both expressed and formally related to one another.

13 Conclusions

In the preceding sections we introduced a variety of composition mechanisms for retrenchments, and examined the interaction with (a convenient form of) refinement, as well as looking at decomposition via partitions of operations' domains. It is important to explore a number of these mechanisms in the mutual context that they create for each other, as composition for retrenchments only rarely reduces to simple composition of relations. Thus while some components of a retrenchment composition combine by simple relational composition, others combine using a $C_{(1,2)} = (G_1 C_2 \vee C_1 G_2 \vee C_1 C_2)$ shape, and there are other possibilities too, as we saw. The main issues raised, concern associativity and coherence, and these are best dealt with under a common umbrella. Fortunately, the shapes we adopted behave well as regards associativity and coherence, within the propositionally based strategy for composition pursued in this paper.

The latter remark underlines the fact that the choice of a law of composition is exactly that: a choice. Different choices can lead to different properties, as the discussion of retrenchment/refinement composition showed to a small extent, and which has been much more extensively explored for vertical composition in [Banach and Jeske (2002)]. In fact other kinds of retrenchment than the form investigated in this paper can throw up different criteria that influence the range of choices available for defining laws of composition. Finally, in the last section, we hope to have sketched enough evidence to convinced the reader that all the composition mechanisms that we investigated have worthwhile applications to system engineering.

References

Banach R. (2000); Maximally Abstract Retrenchments. *in*: Proc. IEEE ICFEM-00, 133-142.

Banach R., Bozzano M. (2006); Retrenchment, and the Generation of Fault Trees for Static, Dynamic and Cyclic Systems. *in*: Proc. SAFECOMP-06, Gorski (ed.), LNCS **4166**, 127-141, Springer.

Banach R., Bozzano M. (2007a); The Mechanical Generation of Fault Trees for Reactive Systems I: Combinational Circuits. *submitted*.

Banach R., Bozzano M. (2007b); The Mechanical Generation of Fault Trees for Reactive Systems II: Clocked and Feedback Circuits. *submitted*.

Banach R., Cross R. (2004); Safety Requirements and Fault Trees using Retrenchment. *in*: Proc. SAFECOMP-04, Heisel, Liggesmeyer, Wittmann (eds.), LNCS **3219**, 210-223, Springer.

Banach R., Jeske C. (2002); Stronger Compositions for Retrenchments, and Feature Engineering. *submitted*.

Banach R., Poppleton M. (2003); Retrenching Partial Requirements into System Definitions: A Simple Feature Interaction Case Study. Req. Eng. J. **8**, 266-288.

Banach R., Poppleton M., Jeske C., Stepney S. (2005); Retrenching the Purse: Finite Sequence Numbers, and the Tower Pattern. *in*: Proc. FM-05, Fitzgerald, Hayes, Tarlecki (eds.), LNCS **3582**, 382-398, Springer.

Banach R., Jeske C., Poppleton M., Stepney S. (2006a); Retrenching the Purse: Finite Exception Logs, and Validating the Small. *in*: Proc. IEEE/NASA SEW30-06, 234-245.

Banach R., Jeske C., Poppleton M., Stepney S. (2006b); Retrenching the Purse: Hashing Injective CLEAR Codes, and Security Properties. *in*: Proc. IEEE ISOLA-06, *to appear*.

Banach R., Poppleton M., Jeske C., Stepney S. (2007); Engineering and Theoretical Underpinnings of Retrenchment. Sci. Comp. Prog. **67**, 301-329.

de Roever W-P., Engelhardt K. (1998); Data Refinement: Model-Oriented Proof Methods and their Comparison. Cambridge University Press.

Derrick J., Boiten E. (2001); Refinement in Z and Object-Z, Foundations and Advanced Applications. FACIT, Springer.

Jeske C. (2005); Algebraic Theory of Retrenchment and Refinement. PhD. Thesis Manchester University Department of Computer Science.

Jeske C., Banach R. (2002); Minimally and Maximally Abstract Retrenchments. *in*: Proc. IFM-02, Butler, Petre, Sere (eds.), LNCS **2335**, 380-399, Springer.

Poppleton M. R. (2001); Formal Methods for Continuous Systems: Liberalising Refinement in B. PhD. Thesis Manchester University Department of Computer Science.

Poppleton M. R., Banach R. (2003); Structuring Retrenchments in B by Decomposition. *in*: Proc. FME-03, Araki, Gnesi, Mandrioli (eds.), LNCS **2805**, 814-833, Springer.

Potter B., Sinclair J., Till D. (1996); An Introduction to Formal Specification and Z. Prentice Hall.

Spivey J. M. (1992); The Z Notation: A Reference Manual. (2nd. ed.) Prentice Hall.

Stepney S., Cooper D., Woodcock J. (2000); An Electronic Purse: Specification, Refinement and Proof. Technical Report PRG-126, Oxford University Computing Laboratory.

Woodcock J., Davies J. (1996); Using Z: Specification, Refinement, and Proof. Prentice-Hall.