# Retrenchment and System Properties

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

**Abstract.** Retrenchment, introduced to alleviate the excessively strong demands that refinement sometimes imposes, is most easily applied 'vertically', i.e. to individual execution steps, without regard to the sequential composition of those steps. This paper addresses the issue of how system properties, akin to sets of execution sequences, can be transformed between systems through retrenchment. Property transformers based on simulation relations are introduced, as are transformers of ambivalent properties and constrained property transformers. Their theory is investigated. The simplifications in the context of regular relations are explored. A number of examples in the discrete and continuous domains are examined.
**Keywords.** Retrenchment, Simulation, Simulability, System Property, Ambivalent Property, Property Transformers.

## 1 Introduction

In [Banach et al. (2005)] the authors gave a comprehensive and broadly based overview of the motivations for introducing retrenchment, and set out the main characteristics of the technique. In retrenchment, the properties established for the after-variables and outputs of a pair of abstract and concrete execution steps, is not a straightforward counterpart of the properties assumed for the before-variables and inputs (because of the wider discrepancy allowed between the properties of the two steps in a retrenchment operation PO, as compared with a typical refinement operation PO). Therefore the analysis of how wider system properties transmutate through retrenchment design phases is not trivial, and is closely related to the analysis of the simulation properties that can be established within a retrenchment. This paper is devoted to carrying through the more complex analysis involved.

In contrast to refinement, the properties derivable from the retrenchment operation POs alone are comparatively weak, so a treatment based just on what can be generically derived from them will, in general, not get very far. Accordingly, the strategy adopted in this paper is to work with simulation relations defined in general terms, and to disregard the issue of the extent to which such relations might be directly provable from the operation POs. Since the aim of retrenchment is to cater for situations where the clean compositionality so characteristic of refinement is unachievable, the extent to which these simulability notions might actually hold in a given scenario, must typically be left to ad-hoc reasoning about the case in hand. In this therefore, the situation is as with many aspects of retrenchment: there are swathes of general reasoning that 'do not quite join up together from general principles' and which must therefore in practice be brought together by bespoke means.

To see how this works out in this paper, it is best to summarise the following sections one by one. In Section 2 we set up our basic terminology, including notations for systems and retrenchments. In Section 3 default retrenchments are reviewed and biretrenchments are introduced. The latter are the symmetric counterparts of the conventional asymmetric notions of retrenchment (in which the operation POs are implica-

tions oreinted from concrete to abstract). It turns out that the more flexible arena of retrenchment makes it easy to recast many conventional retrenchments into the symmetric biretrenchment form. This is very convenient, as symmetric notions are much more appropriate to the discussion of system properties, which should not be sensitive to where in the development hierarchy any particular system lives. In Section 4 we introduce the basic notions of execution fragment, multifragment, and simulability that underpin the rest of the paper. The main simulability definition is crafted in such a way that the simulability relationship between two multifragments is unique, which has useful consequences later, particularly as the flexibility of retrenchment means that we cannot avoid consideration of nonsimulating segments between (multi)fragments. Section 5 considers vertical composition of the simulability relation introduced in Section 4. The results here require additional hypotheses to come through in full measure (eg. biretrenchments prove to be particularly useful), but what is and what is not achievable regarding vertical composition influences the way that the definitions of Section 4 are set up, and these points are discussed.

Section 6 applies the preceding to the study of system properties, which are defined essentially as certain sets of multifragments. The set extension of the simulability relation gives rise to simulation property transformers similar to box and diamond operators in a modal algebra (see eg. [Popkorn (1994)]), and the relvant properties of these are developed. In Section 7 a different tack is initiated. Some system properties are essentially expressions of (some aspects of) the structure of a class of systems. For these properties, called ambivalent properties, the property definition ports directly between systems, giving an alternative means of mapping these properties between systems. The two trains of thought come together in Section 8 where constrained property transformers are developed. These allow the nonsimulating segments of multifragments, whose mappings under the simulation transformers are more or less unconstrained, to have these unruly characteristics curtailed. The interaction between the simulation transformers and constraints gives rise to a more complicated version of the theory of Section 6.

Many retrenchment relationships between systems are characterised by straightforward functional or inverse functional relations. These generalise to regular relations, and Section 9 investigates the considerable simplification of the theory of Section 8 that accrues from regularity. The wide applicability of these results is evidenced by the fact that they readily apply to a wide class of default retrenchments. This suggests that many retrenchments may be modified to yield ones falling into the regular class, enabling them thereby to enjoy regular simulation transformers, which in turn can imply a particularly clean interaction with certain system properties.

Section 10 examines three examples in the light of the developed theory. The first considers a retrenchment from multisets to finite sequences, which gives the theory a straightforward discrete vehicle. The second contemplates a simple digital redesign control problem, and typifies the state of affairs in most numerical analysis situations, where the clean structure investigated previously is less applicable. These two examples are adapted from [Banach et al. (2005)]. A third example models the retrenchment of systems that can be viewed as pairs of noninterfering processes. Here the situation demands the amplification of the previously developed idea of ambivalence, and the preservation of noninterference (as demanded by many security properties), can be modelled using the interaction of the new ambivalence concepts and the simulation transformers developed earlier. Section 11 concludes.

## 2 Output Retrenchments

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

Given an execution step, we use the functions: st, in, Op, ou, st′, to return the before-state, input value, operation name, output value, after-state respectively of the step, i.e. for a step $u$ -($i$, $Op_\mathrm{A}$, $o$)-› $u'$, we get $u$, $i$, $Op_\mathrm{A}$, $o$, $u'$ respectively.

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 $v$ -($j$, $Op_\mathrm{C}$, $p$)-› $v'$, elements of the step relation $stp_{Op_\mathrm{C}}(v, j, v', p)$.

In [Banach and Jeske (2002)] the contrast between normal or primitive retrenchment and output retrenchment was discussed at some length, and the algebraic utility of output retrenchment was made apparent, in that output retrenchment allows the properties of outputs of transitions in the 'well behaved' (i.e. refinement-like) cases to be cleanly separated from the corresponding properties in the 'badly behaved' cases. For consistency's sake we will use output retrenchment exlusively in this paper but we will refer to it as just retrenchment for simplicity.

Given the above context, an (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, $P_{Op}(i, j, u, v)$, $O_{Op}(o, p; u', v', i, j, u, v)$ and $C_{Op}(u', v', o, p; i, j, u, v)$ respectively, one of each for each operation $Op_\mathrm{A} \in \mathsf{Ops}_\mathrm{A}$. The within, output, and concedes 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. 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:

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

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.

and for every corresponding operation pair $Op_A$ and $Op_C$, the abstract and concrete step relations must satisfy the operation PO:

$$G(u, v) \wedge P_{Op}(i, j, u, v) \wedge stp_{Op_C}(v, j, v', p) \Rightarrow \\ (\exists\, u', o \bullet stp_{Op_A}(u, i, u', o) \wedge ((G(u', v') \wedge O_{Op}(o, p; u', v', i, j, u, v)) \vee \\ C_{Op}(u', v', o, p; i, j, u, v))) \tag{2.2}$$

It is easy to show that retrenchments compose vertically in an associative manner. Thus if there is a third system, let us call it the executable system *Exe*, and a retrenchment from *Conc* to *Exe* characterised by retrieve, within, output, and concedes relations $H(v, w)$, $Q_{Op}(j, k, v, w)$, $F_{Op}(p, q; \ldots)$, $D_{Op}(v', w', p, q; \ldots)$, then there is a retrenchment from *Abs* to *Exe* for which the relations in an obvious notation are:

$$G_{(A,E)}(u, w) = [\exists\, v \bullet G(u, v) \wedge H(v, w)] \tag{2.3}$$

$$P_{Op,(A,E)}(i, k, u, w) = \\ [\exists\, v, j \bullet G(u, v) \wedge H(v, w) \wedge P_{Op}(i, j, u, v) \wedge Q_{Op}(j, k, v, w)] \tag{2.4}$$

$$O_{Op,(A,E)}(o, q; \ldots) = \\ [\exists\, v', p, v, j \bullet O_{Op}(o, p; \ldots) \wedge F_{Op}(p, q; \ldots)] \tag{2.5}$$

$$C_{Op,(A,E)}(u', w', o, q; \ldots) = \\ [\exists\, v', p, v, j \bullet (G(u, v) \wedge O_{Op}(o, p; \ldots) \wedge D_{Op}(v', w', p, q; \ldots)) \vee \\ (C_{Op}(u', v', o, p; \ldots) \wedge H(v, w) \wedge F_{Op}(p, q; \ldots)) \vee \\ (C_{Op}(u', v', o, p; \ldots) \wedge D_{Op}(v', w', p, q; \ldots))] \tag{2.6}$$

(This is proved by observing that the assumption of (2.4) and a suitable *Exe* step allows the antecedent of the operation PO for the *Conc* to *Exe* retrenchment to be infered, whence the PO can be used to infer a suitable *Conc* step; the procedure is repeated, allowing the conjunction of the respective consequents to derive the soundness of (2.5) and (2.6).) The details are covered in [Banach (2003)], and in [Banach and Jeske (2002)], the latter of which also deals with the fact that stronger notions of composition can be formulated, which in general do not enjoy associativity of vertical composition without additional constraints.

# 3 Default Retrenchments and Biretrenchments

As shown in [Banach and Jeske (2002)], one way of getting a retrenchment for an arbitrary pair of systems is via the default mechanism. With the above notational conventions for two systems *Abs* and *Conc*, suppose we are given a $G(u, v)$, and for each $Op \in \mathsf{Ops}_A \cap \mathsf{Ops}_C$, a $P_{Op}(i, j, u, v)$ and an $O_{Op}(o, p; \ldots)$; where these simply express how state, input and output spaces are related in the two models but need not be more specific regarding properties of *Abs* and *Conc* than that.[2] We define for each operation $Op \in \mathsf{Ops}_A \cap \mathsf{Ops}_C$:

$$P^{Def}_{Op}(i, j, u, v) = \\ (G(u, v) \wedge P_{Op}(i, j, u, v) \wedge \\ (\exists\, u', o, v', p \bullet stp_{Op_A}(u, i, u', o) \wedge stp_{Op_C}(v, j, v', p))) \tag{3.1}$$

2. Henceforth we use the symmetric $Op \in Ops_A \cap Ops_C$, rather than $Op \in Ops_A$, with a view to future flexibility.

$$C^{\mathrm{Def}}{}_{Op}(u', v', o, p; i, j, u, v) =$$
$$(G(u, v) \wedge P_{Op}(i, j, u, v) \wedge stp_{Op_{\mathrm{A}}}(u, i, u', o) \wedge stp_{Op_{\mathrm{C}}}(v, j, v', p) \wedge$$
$$\neg\, (G(u', v') \wedge O_{Op}(o, p; u', v', i, j, u, v)) \tag{3.2}$$

These are the default within and concedes relations for the systems (given $G$ and $\{P_{Op}, O_{Op} \mid Op \in \mathsf{Ops}_{\mathrm{A}} \cap \mathsf{Ops}_{\mathrm{C}}\}$). For these, the following is a trivial corollary of Proposition 3.1 in [Banach and Jeske (2002)]:

**Proposition 3.1** Suppose given two systems *Abs* and *Conc*, and also given $G$ and $\{P_{Op}, O_{Op} \mid Op \in \mathsf{Ops}_{\mathrm{A}} \cap \mathsf{Ops}_{\mathrm{C}}\}$. Then with the default within and concedes relations defined in (3.1) and (3.2), the operation PO:

$$G(u, v) \wedge P^{\mathrm{Def}}{}_{Op}(i, j, u, v) \wedge stp_{Op_{\mathrm{C}}}(v, j, v', p) \;\Rightarrow$$
$$(\exists\, u', o \bullet stp_{Op_{\mathrm{A}}}(u, i, u', o) \wedge ((G(u', v') \wedge O_{Op}(o, p; u', v', i, j, u, v)) \vee$$
$$C^{\mathrm{Def}}{}_{Op}(u', v', o, p; i, j, u, v))) \tag{3.3}$$

is satisfied.

We now define a biretrenchment between two systems *Abs* and *Conc*, by insisting that as well as (2.1) holding, and (2.2) holding for all $Op \in \mathsf{Ops}_{\mathrm{A}} \cap \mathsf{Ops}_{\mathrm{C}}$, we also have for all $Op \in \mathsf{Ops}_{\mathrm{A}} \cap \mathsf{Ops}_{\mathrm{C}}$:

$$G(u, v) \wedge P_{Op}(i, j, u, v) \wedge stp_{Op_{\mathrm{A}}}(u, i, u', o) \;\Rightarrow$$
$$(\exists\, v', p \bullet stp_{Op_{\mathrm{C}}}(v, j, v', p) \wedge ((G(u', v') \wedge O_{Op}(o, p; u', v', i, j, u, v)) \vee$$
$$C_{Op}(u', v', o, p; i, j, u, v))) \tag{3.4}$$

This is the counterpart of (2.2) that interchanges the role of abstract and concrete systems. It is clear that (2.3)-(2.6) will do duty for the associative composition of biretrenchments as well as for conventional retrenchments.

**Proposition 3.2** Suppose a default retrenchment between two systems *Abs* and *Conc* is given. Then it is a biretrenchment.

*Proof.* The definitions (3.1) and (3.2) are symmetrical regarding the abstract and concrete systems. Thus if (3.3) is provable, then so is:

$$G(u, v) \wedge P^{\mathrm{Def}}{}_{Op}(i, j, u, v) \wedge stp_{Op_{\mathrm{A}}}(u, i, u', o) \;\Rightarrow$$
$$(\exists\, v', p \bullet stp_{Op_{\mathrm{C}}}(v, j, v', p) \wedge ((G(u', v') \wedge O_{Op}(o, p; u', v', i, j, u, v)) \vee$$
$$C^{\mathrm{Def}}{}_{Op}(u', v', o, p; i, j, u, v))) \tag{3.5}$$

☺

One consequence of the above is that there is no reason why a 'bespoke retrenchment', purposely created to single out system characteristics of interest to the development, cannot also be crafted as a biretrenchment. All that has to be done is to ensure that sufficient properties are included in the output and concedes relations so that (3.4) is valid for all abstract transitions that satisfy $G \wedge P^{\mathrm{Def}}{}_{Op}$ (besides (2.2) being valid for the concrete ones). This contrasts vividly with the situation in refinement where due to the exclusive reliance on the retrieve relation, a proper reduction of nondeterminism can be described, but nothing can be said about those abstract steps that cannot be simulated.

# 4 Fragments and Simulability

We will be dealing a lot with sequences. If $a = [a_0 \ldots a_n]$ is a finite sequence, then $\mathrm{dom}(a) = \{0 \ldots n\}$ and $\mathrm{dom}^\bullet(a) = \{0 \ldots n-1\}$, both defaulting to $\emptyset$ if $a$ is empty, and with $\mathrm{dom}^\bullet(a)$ defaulting to $\emptyset$ if $a$ has one element. If $a$ is infinite then $\mathrm{dom}(a) = \mathrm{dom}^\bullet(a) = \mathsf{NAT}$. We also define $\mathrm{md}(a) = \max(\mathrm{dom}(a))$ and $\mathrm{md}^\bullet(a) = \max(\mathrm{dom}^\bullet(a))$; these are $\infty$ if $a$ is infinite.

**Definition 4.1** An execution fragment (fragment for short) is a sequence of contiguous execution steps written in the form:

$$\mathcal{S} = [\ u_0 \text{-}(i_0, Op_{\mathrm{A},0}, o_1)\text{-}\!\rightarrow u_1 \text{-}(i_1, Op_{\mathrm{A},1}, o_2)\text{-}\!\rightarrow u_2 \ldots\ ] \tag{4.1}$$

where:

(1) $\mathcal{S}$ may be of zero length: $\mathcal{S} = [\,]$.

(2) $\mathcal{S}$ may be of finite length:

$$\mathcal{S} = [\ u_0 \text{-}(i_0, Op_{\mathrm{A},0}, o_1)\text{-}\!\rightarrow u_1 \ldots\ \ldots u_n \text{-}(i_n, Op_{\mathrm{A},n}, o_{n+1})\text{-}\!\rightarrow u_{n+1}\ ] \tag{4.2}$$

with $n \geq 0$, $\mathrm{dom}(\mathcal{S}) = \{0 \ldots n\}$, $\mathrm{dom}^\bullet(\mathcal{S}) = \{0 \ldots n-1\}$.

(3) $\mathcal{S}$ may be of infinite length, (and $\mathrm{dom}(\mathcal{S}) = \mathrm{dom}^\bullet(\mathcal{S}) = \mathsf{NAT}$).

The $l$'th step of $\mathcal{S}$, $\mathcal{S}[l]$, is the one starting at the $l$'th state i.e. $u_l \text{-}(i_l, Op_{\mathrm{A},l}, o_{l+1})\text{-}\!\rightarrow u_{l+1}$. Thus for each $l \in \mathrm{dom}^\bullet(\mathcal{S})$, $\mathrm{st}'(\mathcal{S}[l]) = \mathrm{st}(\mathcal{S}[l+1])$. Note that $u_0$ does **not** need to satisfy the $Init_{\mathrm{A}}$ property; however if it does, $\mathcal{S}$ is called an initial (execution) fragment.

**Definition 4.2** An execution multifragment (multifragment for short) is a sequence of execution fragments:

$$\mathcal{S} = [\mathcal{S}_0, \mathcal{S}_1, \ldots\ ] \tag{4.3}$$

where:

(1) $\mathcal{S}$ may be of zero length: $\mathcal{S} = [\,]$.

(2) $\mathcal{S}$ may be of finite length: $\mathcal{S} = [\mathcal{S}_0 \ \ldots\ \mathcal{S}_n]$ for some $n \geq 0$; moreover for all $k \in \mathrm{dom}^\bullet(\mathcal{S})$, $\mathcal{S}_k$ is finite, while $\mathcal{S}_n$ may be finite or infinite.

(3) $\mathcal{S}$ may be of infinite length, in which case for all $k \in \mathrm{dom}(\mathcal{S})$, $\mathcal{S}_k$ is finite.

A multifragment may contain concatenable adjacent elements. However if it does not, i.e. if for all $k \in \mathrm{dom}^\bullet(\mathcal{S})$, $\mathrm{st}'(\mathcal{S}_k[\mathrm{md}(\mathcal{S}_k)]) \neq \mathrm{st}(\mathcal{S}_{k+1}[0])$, then it is called curt.

The context will always distinguish whether $\mathcal{S}$ denotes a fragment or a multifragment. The corresponding concrete notions will be denoted by $\mathcal{T}$.

**Definition 4.3** Let $u \text{-}(i, Op_{\mathrm{A}}, o)\text{-}\!\rightarrow u'$ be an abstract step and $v \text{-}(j, Op_{\mathrm{C}}, p)\text{-}\!\rightarrow v'$ a concrete step. Then these steps are in simulation (or the abstract step simulates the concrete step), also written $(u \text{-}(i, Op_{\mathrm{A}}, o)\text{-}\!\rightarrow u')\ \Sigma^1\ (v \text{-}(j, Op_{\mathrm{C}}, p)\text{-}\!\rightarrow v')$, iff we have:

$$\begin{array}{l} G(u, v) \wedge P_{Op}(i, j, u, v) \wedge stp_{Op_{\mathrm{C}}}(v, j, v', p) \wedge stp_{Op_{\mathrm{A}}}(u, i, u', o) \wedge \\ ((G(u', v') \wedge O_{Op}(o, p; u', v', i, j, u, v)) \vee C_{Op}(u', v', o, p; i, j, u, v)) \end{array} \tag{4.4}$$

An abstract (resp. concrete) step is simulable iff there is a concrete (resp. abstract) step such that (4.4) holds for the pair. Otherwise it is nonsimulable. Note that simu-

lability depends not only on the two systems in question, but on the retrenchment relation (as defined by $G$ and the $P_{Op}$, $O_{Op}$, $C_{Op}$) between them.

We extend the simulation notion to abstract and concrete (multi)fragments in the following. First we recall that a segment $t$ of a sequence $s$ is a subsequence 'with no gaps', i.e. it is a subsequence definable by two indexes of $s$, say $a$ and $b$, and consists of all elements $s[k]$ with $a \leq k \leq b$, so $\mathrm{dom}(s[a..b]) = \{a..b\}$. We call this an $(a,b)$-segment of $s$, and write it $s[a..b]$. If $s$ is infinite then $b$ can be $\infty$, and then $\mathrm{dom}(s[a..\infty])$ consists of all finite $k \geq a$. If $a > b$ then the concept of $(a,b)$-segment is undefined, in particular it is convenient to insist that all segments have at least one element. For a segment $s[a..b]$, $\mathrm{md}(s[a..b]) = b$ of course (or $\infty$ if $s[a..b]$ is infinite), but we also have $\mathrm{mind}(s[a..b]) = a$, defining the minimal element of the domain of $s[a..b]$.

**Definition 4.4** Let $\mathcal{S}$ and $\mathcal{T}$ be abstract and concrete fragments. Then the simulability relation $\mathcal{S}\ \Sigma(l_i, l_f, m_i, m_f)\ \mathcal{T}$ holds iff:

(1) $l_f - l_i = m_f - m_i \geq 0$, including the case that $l_f = m_f = \infty$, and $(l_i, l_f)$ and $(m_i, m_f)$ define $(l_i, l_f)$- and $(m_i, m_f)$- segments $\mathcal{S}[l_i..l_f]$ and $\mathcal{T}[m_i..m_f]$ of $\mathcal{S}$ and $\mathcal{T}$ respectively.

(2) For all $k \in \mathrm{dom}(\mathcal{S}[l_i..l_f])$ and $k' \in \mathrm{dom}(\mathcal{T}[m_i..m_f])$, where $k - l_i = k' - m_i$, steps $k$ of $\mathcal{S}$, i.e. $u_k$ -$(i_k, Op_{A,k}, o_{k+1})$-› $u_{k+1}$ and $k'$ of $\mathcal{T}$, i.e. $v_{k'}$ -$(j_{k'}, Op_{C,k'}, p_{k'+1})$-› $v_{k'+1}$ are in simulation.

The segments $\mathcal{S}[l_i..l_f]$ and $\mathcal{T}[m_i..m_f]$ are called simulating segments. $\mathcal{S}[0..l_i-1]$ and $\mathcal{T}[0..m_i-1]$, i.e. the portions of $\mathcal{S}$ and $\mathcal{T}$ before the simulating segments, are called the lead-ins of $\mathcal{S}$ and $\mathcal{T}$; they may be empty. $\mathcal{S}[l_f+1..\mathrm{md}(\mathcal{S})]$ and $\mathcal{T}[m_f+1..\mathrm{md}(\mathcal{T})]$ are correspondingly called the lead-outs of $\mathcal{S}$ and $\mathcal{T}$, also possibly empty. When $\mathcal{S}\ \Sigma(l_i, l_f, m_i, m_f)\ \mathcal{T}$ is understood, we write $^\bullet\mathcal{S}$ to refer to $\mathcal{S}$ with its lead-in removed, and $\mathcal{S}^\bullet$ to refer to $\mathcal{S}$ with its lead-out removed; and in $^\bullet\mathcal{S}^\bullet$ both are removed; N.B. the lead-in and lead-out are undefined if there is no simulating segment of $\mathcal{S}$. Similar considerations apply in the concrete case for $^\bullet\mathcal{T}$, $\mathcal{T}^\bullet$, $^\bullet\mathcal{T}^\bullet$.

When we have $\mathcal{S}\ \Sigma(0, \mathrm{md}(\mathcal{S}), 0, \mathrm{md}(\mathcal{T}))\ \mathcal{T}$, i.e. the simulating segments define a bijection, we say that the simulation is exact, and write $\mathcal{S}\ \Sigma\ \mathcal{T}$ for short. It is clear that if $\mathcal{S}\ \Sigma(l_i, l_f, m_i, m_f)\ \mathcal{T}$ holds for some $l_i, l_f, m_i, m_f$, then $^\bullet\mathcal{S}^\bullet\ \Sigma\ ^\bullet\mathcal{T}^\bullet$ is true.

**Example 4.5** Fig. 1 gives some illustrations of Definition 4.4. In each diagram the arrows are execution steps and the vertical lines depict the parts of the 'in simulation' relationship pertinent to the before-state or after-state. In each diagram $\mathcal{S}$ is shown above $\mathcal{T}$, and the shaded squares and parallelograms indicate that the relevant steps of $\mathcal{S}$ and $\mathcal{T}$ are in simulation, while a cross indicates that this is not the case. Thus Fig. 1.(a) illustrates that $\mathcal{S}\ \Sigma(0,2,0,2)\ \mathcal{T}$, or alternatively $\mathcal{S}\ \Sigma\ \mathcal{T}$. But $\mathcal{S}\ \Sigma(0,1,0,1)\ \mathcal{T}$ and $\mathcal{S}\ \Sigma(1,2,1,2)\ \mathcal{T}$ are also true of Fig. 1.(a), as well as three other statements about short segments. This illustrates that the $\mathcal{S}\ \Sigma(l_i, l_f, m_i, m_f)\ \mathcal{T}$ notion does not insist that there are no steps of $\mathcal{S}$ and $\mathcal{T}$ other than the ones it mentions that are in simulation, i.e. it provides a lower bound to simulability. In the subsequent examples we will not refer to this aspect explicitly, though we return to the issue later. In this light, Fig. 1.(b) illustrates $\mathcal{S}\ \Sigma(0,1,1,2)\ \mathcal{T}$. In Fig. 1.(c), we have that the first steps of $\mathcal{S}$ and $\mathcal{T}$ are in simulation, as are the last steps of $\mathcal{S}$ and $\mathcal{T}$, so we have $\mathcal{S}\ \Sigma(0,0,0,0)\ \mathcal{T}$ and $\mathcal{S}\ \Sigma(2,2,2,2)\ \mathcal{T}$. Similar remarks apply to Fig. 1.(d) where although each step of $\mathcal{S}$ is in simulation with some step of $\mathcal{T}$, the relevant steps of $\mathcal{T}$ do not form a segment.

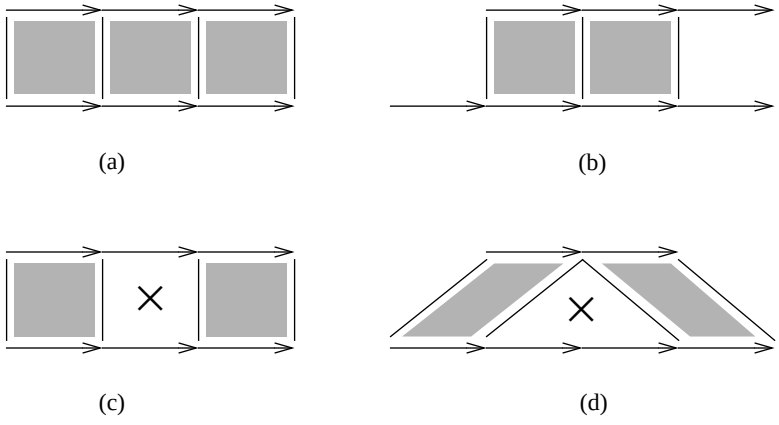



Fig. 1

**Definition 4.6** Let $\mathcal{S}$ and $\mathcal{T}$ be abstract and concrete multifragments. Then the relation $\mathcal{S}\ \Sigma(ll)\ \mathcal{T}$ holds iff $ll$ is a sequence of pairs of of triples of NATs, such that if $ll = [((s_0,l_{0,\mathrm{i}},l_{0,\mathrm{f}}),(t_0,m_{0,\mathrm{i}},m_{0,\mathrm{f}})),((s_1,l_{1,\mathrm{i}},l_{1,\mathrm{f}}),(t_1,m_{1,\mathrm{i}},m_{1,\mathrm{f}})),\ \ldots\ ]$ then:

(1) $\mathrm{dom}(\mathcal{S}) = \{s_k \mid k \in \mathrm{dom}(ll)\}$ and $\mathrm{dom}(\mathcal{T}) = \{t_k \mid k \in \mathrm{dom}(ll)\}$.

(2) For each $k \in \mathrm{dom}^\bullet(ll)$, we have $(s_k,l_{k,\mathrm{i}},l_{k,\mathrm{f}}) < (s_{k+1},l_{k+1,\mathrm{i}},l_{k+1,\mathrm{f}})$ and $(t_k,m_{k,\mathrm{i}},m_{k,\mathrm{f}}) < (t_{k+1},m_{k+1,\mathrm{i}},m_{k+1,\mathrm{f}})$, where $(s_k,l_{k,\mathrm{i}},l_{k,\mathrm{f}}) < (s_{k+1},l_{k+1,\mathrm{i}},l_{k+1,\mathrm{f}})$ iff $(s_k < s_{k+1}) \vee (s_k = s_{k+1} \wedge l_{k,\mathrm{f}} < l_{k+1,\mathrm{i}})$; similarly for $(t_k,m_{k,\mathrm{i}},m_{k,\mathrm{f}}) < (t_{k+1},m_{k+1,\mathrm{i}},m_{k+1,\mathrm{f}})$.

(3) If $k \in \mathrm{dom}^\bullet(ll)$ then $\neg(s_k = s_{k+1} \wedge l_{k,\mathrm{f}} + 1 = l_{k+1,\mathrm{i}} \wedge t_k = t_{k+1} \wedge m_{k,\mathrm{f}} + 1 = m_{k+1,\mathrm{i}})$.

(4) If $k \in \mathrm{dom}(ll)$ then $\mathcal{S}_{s_k}\ \Sigma(l_{k,\mathrm{i}},l_{k,\mathrm{f}},m_{k,\mathrm{i}},m_{k,\mathrm{f}})\ \mathcal{T}_{t_k}$ in the sense of Definition 4.4.

(5) Every simulable step of $\mathcal{S}$ and $\mathcal{T}$ lies in some segment described by $ll$, i.e.:

  (i) For every fragment $\mathcal{S}_s$ in $\mathcal{S}$, for every simulable step $\mathcal{S}_s[j]$ of $\mathcal{S}_s$, there is a $k \in \mathrm{dom}(ll)$, such that $\mathcal{S}_{s_k} = \mathcal{S}_s$ and $l_{k,\mathrm{i}} \le j$ and $j \le l_{k,\mathrm{f}}$.

  (ii) For every fragment $\mathcal{T}_t$ in $\mathcal{T}$, for every simulable step $\mathcal{T}_t[j]$ of $\mathcal{T}_t$, there is a $k \in \mathrm{dom}(ll)$, such that $\mathcal{T}_{t_k} = \mathcal{T}_t$ and $m_{k,\mathrm{i}} \le j$ and $j \le m_{k,\mathrm{f}}$.

In Definition 4.6, clauses (1)-(3) ensure that the triples in $ll$ refer to the fragments in $\mathcal{S}$ and $\mathcal{T}$ in a well defined, complete, appropriately ordered and maximal way. Clauses (4)-(5) say that the triples in $ll$ in fact refer to segments of the fragments which are in simulation, and that there are no simulable steps in $\mathcal{S}$ or $\mathcal{T}$ not captured by $ll$.

Again, the segments defined by $ll$ are called simulating segments. If $\mathcal{S}\ \Sigma(ll)\ \mathcal{T}$, for a fragment $\mathcal{S}_s$ within $\mathcal{S}$, the lead-in of $\mathcal{S}_s$ is the portion of $\mathcal{S}_s$ that precedes the first simulating segment of $\mathcal{S}_s$ mentioned in $ll$, and ${}^\bullet\mathcal{S}_s$ is $\mathcal{S}_s$ with this removed. Likewise the lead-out of $\mathcal{S}_s$ is the portion of $\mathcal{S}_s$ that follows the last simulating segment of $\mathcal{S}_s$ mentioned in $ll$, and $\mathcal{S}_s{}^\bullet$ is $\mathcal{S}_s$ with this portion removed. ${}^\bullet\mathcal{S}_s{}^\bullet$ is $\mathcal{S}_s$ with both of these re-

moved. The notations ${}^{\bullet}\mathcal{S}$, $\mathcal{S}^{\bullet}$, ${}^{\bullet}\mathcal{S}^{\bullet}$ and ${}^{\bullet}\mathcal{T}$, $\mathcal{T}^{\bullet}$, ${}^{\bullet}\mathcal{T}^{\bullet}$ refer to the removal of lead-ins, lead-outs, and both, from all fragments of $\mathcal{S}$ and $\mathcal{T}$.

In multifragments, the lead-ins and lead-outs and breaks in simulation internal to the constituent fragments, are collectively called exceptions.

**Proposition 4.7** Suppose $\mathcal{S}$ and $\mathcal{T}$ are abstract and concrete multifragments, and that $\mathcal{S}\ \Sigma(ll)\ \mathcal{T}$ and $\mathcal{S}\ \Sigma(ll')\ \mathcal{T}$ both hold. Then $ll' = ll$.

*Proof.* The conditions in Definition 4.6 ensure that $ll$ describes a totally ordered bijection between all the simulable steps in $\mathcal{S}$ and all the simulable steps in $\mathcal{T}$. There is only one such totally ordered bijection. ☺

Since the $ll$ in $\mathcal{S}\ \Sigma(ll)\ \mathcal{T}$ is unique we can suppress it when convenient, writing $\mathcal{S}\ \Sigma\ \mathcal{T}$ for short.

**Corollary 4.8** Suppose $\mathcal{S}$ and $\mathcal{T}$ are abstract and concrete multifragments. Then $\mathcal{S}\ \Sigma\ \mathcal{T}$ iff there is a totally ordered bijection between all the simulable steps in $\mathcal{S}$ and all the simulable steps in $\mathcal{T}$, where each element of the bijection pairs a step of $\mathcal{S}$ to a step of $\mathcal{T}$ in simulation with it. If the bijection exists, it is unique.

Fig. 2 illustrates Definition 4.6 using the same conventions as Fig. 1. In Fig. 2.(a) we have $\mathcal{S}\ \Sigma([((0,1,1),(0,2,2)),((0,3,4),(1,0,1))])\ \mathcal{T}$, and Fig. 2.(a) also illustrates non-trivial lead-ins and lead-outs for $\mathcal{S}$ and $\mathcal{T}$, namely $\mathcal{S}_0[0..0]$, $\mathcal{S}_0[5..5]$ and $\mathcal{T}_0[0..1]$, $\mathcal{T}_1[2..2]$. Also ${}^{\bullet}\mathcal{S}^{\bullet}\ \Sigma\ {}^{\bullet}\mathcal{T}^{\bullet}$ fails in Fig. 2.(a) because of $\mathcal{S}_0[2..2]$. In Fig. 2.(b) we have $\mathcal{S}\ \Sigma([((0,0,1),(0,0,1)),((1,0,1),(0,3,4))])\ \mathcal{T}$. In Fig. 2.(c)-(d), we have examples of exactness, so that we can write $\mathcal{S}\ \Sigma\ \mathcal{T}$; eg. in Fig. 2.(d), $ll$ = [((0,0,1),(0,0,1)), ((1,0,1),(0,2,3)),((1,2,3),(1,0,1))]. Fig. 2.(c) shows that abstract and concrete multifragments $\mathcal{S}$ and $\mathcal{T}$ such that $\mathcal{S}\ \Sigma(ll)\ \mathcal{T}$ holds, can be disconnected even taking the $\Sigma(ll)$ relationship into account. In Fig. 2.(e) we have a counterexample to $\mathcal{S}\ \Sigma(ll)\ \mathcal{T}$ since the middle two simulations do not respect clause (2) of Definition 4.6. Finally in Fig. 2.(f), we have $\mathcal{S}\ \Sigma([((0,0,1),(0,0,1)),((1,0,0),(0,2,2)),((1,2,2),(0,4,4)),((1,3,4),(1,0,1))]\ \mathcal{T}$, showing that breaks in simulation are tolerated within multifragments. Consequently, Fig. 1.(c)-(d) can be described using the $\mathcal{S}\ \Sigma(ll)\ \mathcal{T}$ notation, even if not using the $\mathcal{S}\ \Sigma(l_i, l_f, m_i, m_f)\ \mathcal{T}$ one. (Fig. 1.(a)-(b) can be described by either.)

The intention of Definition 4.4 and Definition 4.6 is to capture a notion of simulability sufficiently flexible to encompass the possibilities admitted by retrenchment, and convenient enough to build further theory. Thus the fact that the conditions applicable in the before- and after- states of an execution step in retrenchment are manifestly not the same (by (2.2), and reflected in (4.4)) means that while the situation in Fig. 1.(a) is the ideal, possibilities such as Fig. 1.(b) in which simulability breaks down cannot be excluded, the more so since the concrete system is permitted to have operations not present at the abstract level. Beyond this, the fact that Figs. 1.(c) and 1.(d) are also admitted by Definition 4.4, indicates that Definition 4.4 alone does not cover all the issues regarding simulability between multifragments that need to be addressed. Definition 4.6 is the natural generalisation of Definition 4.4 to multifragments, and covers more general breakdowns in simulation. It contains not only clauses that translate simulability to the multifragment context, but clauses intended to address aspects of 'completeness' that Definition 4.4 does not cover. We refer to the various ways of failing to live up to the exemplary behaviour of Fig. 1.(a) as punctured simulations.[3]

# 5 Vertical Composition

We now look at the vertical composition properties of $\mathcal{S}$ $\Sigma(l_i, l_f, m_i, m_f)$ $\mathcal{T}$ for fragments, and of $\mathcal{S}$ $\Sigma(ll)$ $\mathcal{T}$ for multifragments. Let $\mathcal{U}$ refer to a (multi)fragment of a executable system *Exe*, related by a retrenchment to *Conc* as assumed in (2.3)-(2.6). We say of a simulable *Conc* step, that it is *AC*-simulable if it is in simulation with some abstract step, and that it is *CE*-simulable if it is in simulation with some executable step, depending on which retrenchment we have in mind. Similarly for the other systems and various retrenchments and simulation phraseologies.

**Proposition 5.1** Suppose $\mathcal{S}$, $\mathcal{T}$, $\mathcal{U}$, are fragments of abstract, concrete and executable systems *Abs*, *Conc*, *Exe*. Suppose that we have both $\mathcal{S}$ $\Sigma(l_{a,i}, l_{a,f}, m_{a,i}, m_{a,f})$ $\mathcal{T}$ and $\mathcal{T}$ $\Sigma(l_{b,i}, l_{b,f}, m_{b,i}, m_{b,f})$ $\mathcal{U}$. Then $\mathcal{S}$ $\Sigma(l_{(a,b),i}, l_{(a,b),f}, m_{(a,b),i}, m_{(a,b),f})$ $\mathcal{U}$, according to the 'in simulation' definition stemming from the vertically composed retrenchment defined by (2.3)-(2.6), where:

$$\begin{array}{l} l_{(a,b),i} = \text{If } l_{b,i} - m_{a,i} = k_{l,i} > 0 \text{ Then } l_{a,i} + k_{l,i} \text{ Else } l_{a,i} \text{ Fi} \\ l_{(a,b),f} = \text{If } m_{a,f} - l_{b,f} = k_{l,f} > 0 \text{ Then } l_{a,f} - k_{l,f} \text{ Else } l_{a,f} \text{ Fi} \\ m_{(a,b),i} = \text{If } m_{a,i} - l_{b,i} = k_{m,i} > 0 \text{ Then } m_{b,i} + k_{m,i} \text{ Else } m_{b,i} \text{ Fi} \\ m_{(a,b),f} = \text{If } l_{b,f} - m_{a,f} = k_{m,f} > 0 \text{ Then } m_{b,f} - k_{m,f} \text{ Else } m_{b,f} \text{ Fi} \end{array} \quad (5.1)$$

provided that these define a $(l_{(a,b),i}, l_{(a,b),f})$-segment of $\mathcal{S}$ and a $(m_{(a,b),i}, m_{(a,b),f})$-segment of $\mathcal{T}$.

*Proof.* It is clear that provided (5.1) defines two nonempty segments $\mathcal{S}[l_{(a,b),i}..l_{(a,b),f}]$ and $\mathcal{U}[m_{(a,b),i}..m_{(a,b),f}]$ (which is not certain since (5.1) does not guarantee that $l_{(a,b),i} \leq l_{(a,b),f}$ and $m_{(a,b),i} \leq m_{(a,b),f}$), then these segments are each in simulation with the largest possible common segment of $\mathcal{T}$, given by the intersection of $\mathcal{T}[m_{a,i}..m_{a,f}]$ and $\mathcal{T}[l_{b,i}..l_{b,f}]$, and thus are in *AE*-simulation with each other via (2.3)-(2.6). ☺

**Proposition 5.2** The vertical composition of Proposition 5.1 is associative.

*Proof.* This reduces to the observation that $\max(\max(a,b),c)) = \max(a,\max(b,c))$ (and likewise for min), in the context of constructing the segments of the multiple compositions. ☺

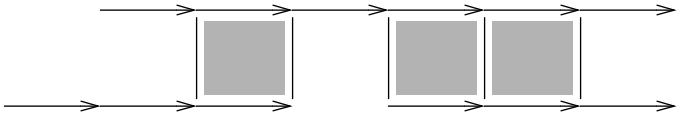
**Proposition 5.3** Suppose $\mathcal{S}$, $\mathcal{T}$, $\mathcal{U}$, are multifragments of systems *Abs*, *Conc*, *Exe*, which are related via biretrenchments captured in the notations used in (2.3)-(2.6). Suppose both $\mathcal{S}$ $\Sigma(ll_a)$ $\mathcal{T}$ and $\mathcal{T}$ $\Sigma(ll_b)$ $\mathcal{U}$ hold, where:

$$\begin{array}{l} ll_a = [((s_{a,0}, l_{a,0,i}, l_{a,0,f}), (t_{a,0}, m_{a,0,i}, m_{a,0,f})), \\ \qquad ((s_{a,1}, l_{a,1,i}, l_{a,1,f}), (t_{a,1}, m_{a,1,i}, m_{a,1,f})), \ \ldots\ ] \\ ll_b = [((s_{b,0}, l_{b,0,i}, l_{b,0,f}), (t_{b,0}, m_{b,0,i}, m_{b,0,f})), \\ \qquad ((s_{b,1}, l_{b,1,i}, l_{b,1,f}), (t_{b,1}, m_{b,1,i}, m_{b,1,f})), \ \ldots\ ] \end{array} \quad (5.2)$$

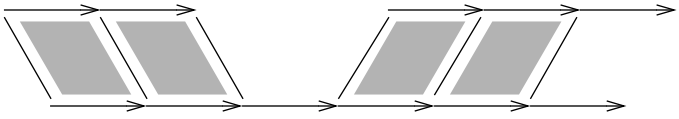
Suppose the sets of *AC*-simulable and *CE*-simulable steps of *Conc* are equal. Let $\underline{\underline{ll}}_{(a,b)}$ be the set of all pairs of triples of the form:

$$((s_{a,j}, l_{(a,b),j,i}, l_{(a,b),j,f}), (t_{b,k}, m_{(a,b),k,i}, m_{(a,b),k,f})) \quad (5.3)$$

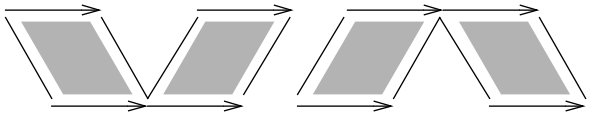
3. In [Banach and Poppleton (1999)], the term punctured simulation was introduced in the context of retrenchments done within the B-Method, and described a more constrained set of situations than is considered in the present paper.



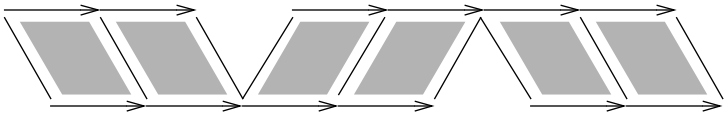
(a)



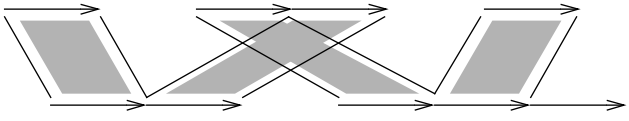
(b)



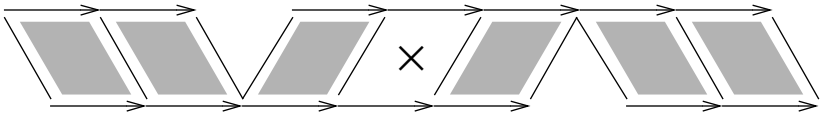
(c)



(d)



(e)



(f)

Fig. 2

such that for each such pair of triples, $\mathcal{S}_{s_{a,j}}\ \Sigma(l_{(a,b),j,i}, l_{(a,b),j,f}, m_{(a,b),k,i}, m_{(a,b),k,f})\ \mathcal{U}_{t_{b,k}}$ holds, where $l_{(a,b),j,i}$, $l_{(a,b),j,f}$, $m_{(a,b),k,i}$, $m_{(a,b),k,f}$ are given by applying the construction in Proposition 5.1 to the simulable segments $\mathcal{S}_{s_{a,j}}[l_{a,j,i}..l_{a,j,f}]$, $\mathcal{U}_{t_{b,k}}[m_{b,k,i}..m_{b,k,f}]$, and a suitable $\mathcal{T}_{t_{c,k'}}[n_{c,k',i}..n_{c,k',f}]$ via (5.1).

Let $\underline{ll}_{(a,b)}$ be the set of all pairs of triples obtained from $\underline{ll}_{(a,b)}$ by amalgamating all continguous pairs of segments, i.e. by replacing two pairs:

$$\begin{array}{l}((s_{a,j}, l_{(a,b),j,i}, l_{(a,b),j,f}), (t_{b,k}, m_{(a,b),k,i}, m_{(a,b),k,f}))\ \text{and}\\ ((s_{a,j'}, l_{(a,b),j',i}, l_{(a,b),j',f}), (t_{b,k'}, m_{(a,b),k',i}, m_{(a,b),k',f}))\ \text{such that}\\ (s_{a,j'} = s_{a,j} \wedge l_{(a,b),j',i} = l_{(a,b),j,f} + 1)\ \wedge\\ (t_{b,k'} = t_{b,k} \wedge m_{(a,b),k',i} = m_{(a,b),k,f} + 1)\end{array} \tag{5.4}$$

by the one pair:

$$((s_{a,j}, l_{(a,b),j,i}, l_{(a,b),j',f}), (t_{b,k}, m_{(a,b),k,i}, m_{(a,b),k',f})) \tag{5.5}$$

and repeating (countably often if necessary) until there are no more continguous pairs. Let $ll_{(a,b)}$ be an ordering of the set $\underline{ll}_{(a,b)}$ where the ordering of the elements is given by:

$$\begin{array}{l}((s_{(a,b),j}, l_{(a,b),j,i}, l_{(a,b),j,f}), (t_{(a,b),k}, m_{(a,b),k,i}, m_{(a,b),k,f}))\ <\\ ((s_{(a,b),j'}, l_{(a,b),j',i}, l_{(a,b),j',f}), (t_{(a,b),k'}, m_{(a,b),k',i}, m_{(a,b),k',f}))\\ \qquad\qquad \Leftrightarrow\\ (s_{(a,b),j} < s_{(a,b),j'}) \vee (t_{(a,b),k} < t_{(a,b),k'}) \vee\\ (s_{(a,b),j} = s_{(a,b),j'} \wedge t_{(a,b),k} = t_{(a,b),k'} \wedge (l_{(a,b),j,f} < l_{(a,b),j',i} \vee m_{(a,b),k,f} < m_{(a,b),k',i}))\end{array} \tag{5.6}$$

Then $\mathcal{S}\ \Sigma(ll_{(a,b)})\ \mathcal{U}$, according to the 'in simulation' definition stemming from the vertically composed retrenchment defined by (2.3)-(2.6), which is a biretrenchment.

*Proof.* That the composed retrenchment defined by (2.3)-(2.6) is a biretrenchment under the circumstances has been pointed out already in Section 3, so we must establish the various clauses of Definition 4.6 with respect to $ll_{(a,b)}$.

Firstly we show that the pairs of segments described by $ll_{(a,b)}$ are in simulation. To see this we note that by Corollary 4.8, both $\mathcal{S}\ \Sigma(ll_a)\ \mathcal{T}$ and $\mathcal{T}\ \Sigma(ll_b)\ \mathcal{U}$ describe order preserving bijections between the *AC*-simulable steps of $\mathcal{S}$ and $\mathcal{T}$ on the one hand and the *CE*-simulable steps of $\mathcal{T}$ and $\mathcal{U}$ on the other. Since the sets of *AC*-simulable steps of *Conc* and the *CE*-simulable steps of *Conc* are equal, the $(m_{a,k,i}, m_{a,k,f})$-segments of $\mathcal{T}$ refered to in the second components of elements of $ll_a$ are exactly the same as the $(l_{b,k,i}, l_{b,k,f})$-segments of $\mathcal{T}$ refered to in the first components of elements of $ll_b$. Therefore each step of $\mathcal{T}$ contained in these segments, existentially witnesses as intermediate transition, the derivation of an *AE*-simulation between the corresponding *Abs* and *Exe* steps, obtained by composing the *AC*-simulation and the *CE*-simulation according to (2.3)-(2.6). This gives a bijection between *Abs* and *Exe* steps at the level of individual simulating steps. Since the relative order of corresponding simulating steps in $\mathcal{S}\ \Sigma(ll_a)\ \mathcal{T}$ and $\mathcal{T}\ \Sigma(ll_b)\ \mathcal{U}$ is the same, the relative order of corresponding simulating steps in the *AE*-simulations will be the same too. So we have a totally ordered bijection between *Abs* and *Exe* simulable steps. The steps aggregate via the construction of Proposition 5.1 into relations between segments of $\mathcal{S}$ and $\mathcal{U}$, $\mathcal{S}_{s_{a,j}}\ \Sigma(l_{(a,b),j,i}, l_{(a,b),j,f}, m_{(a,b),k,i}, m_{(a,b),k,f})\ \mathcal{U}_{t_{b,k}}$, and these comprise $\underline{ll}_{(a,b)}$. The $\mathcal{S}_{s_{a,j}}$ and $\mathcal{U}_{t_{b,k}}$ can be aggregated further into $\underline{ll}_{(a,b)}$ via (5.4) and (5.5), repeatedly applied, perhaps countably many times, until a fixed point is reached. And since the

order on individual steps in $\mathcal{S}$, $\mathcal{T}$, $\mathcal{U}$, is a total order, the resulting segments in $\mathcal{S}$ and $\mathcal{U}$, which are disjoint by construction, inherit total orders too. Since the pairs of these in $\underline{ll}_{(a,b)}$ consist of source and target data in the elements of a bijection, it is not hard to see that (5.6) defines a total order on the pairs, and $\underline{ll}_{(a,b)}$ with this total order is $ll_{(a,b)}$.

Next we consider the non-simulating steps. Let *Op* be an operation name common to the *Abs*, *Conc* and *Exe* systems, and let $u$ -($i$, $Op_A$, $o$)-› $u'$ be an *AC*-nonsimulating abstract step in a segment of $\mathcal{S}$. Suppose there was a step $w$ -($k$, $Op_E$, $q$)-› $w'$ of *Exe* with which it was in simulation via the relations (2.3)-(2.6). Then since $G_{(A,E)}(u, w) \wedge P_{Op,(A,E)}(i, k, u, w)$ is presumed, we can infer $G(u, v) \wedge P_{Op}(i, j, u, v)$ for some $v$ and $j$. Since we have a biretrenchment, the step $u$ -($i$, $Op_A$, $o$)-› $u'$ and these, imply via the biretrenchment operation PO (3.4), that there is some concrete step $v$ -($j$, $Op_C$, $p$)-› $v'$ which is in simulation with $u$ -($i$, $Op_A$, $o$)-› $u'$. This contradicts the *AC*-nonsimulability of $u$ -($i$, $Op_A$, $o$)-› $u'$, which therefore must be *AE*-nonsimulable. A similar argument using the conventional retrenchment operation PO shows that the *CE*-nonsimulating implementation steps are also *AE*-nonsimulable.

Now we have accumulated enough facts to dispose of the various clauses of Definition 4.6 for $\mathcal{S}$ $\Sigma(ll_{(a,b)})$ $\mathcal{U}$. Since $\mathrm{dom}(\mathcal{S}) = \{s_k \mid k \in \mathrm{dom}(ll_a)\}$ and all *AC*-simulable steps of $\mathcal{S}$ occur as *AE*-simulating steps of $\mathcal{S}$ by construction, $\mathrm{dom}(\mathcal{S}) = \{s_k \mid k \in \mathrm{dom}(ll_{(a,b)})\}$ quickly follows; likewise $\mathrm{dom}(\mathcal{T}) = \{t_k \mid k \in \mathrm{dom}(ll_{(a,b)})\}$, and thus clause (1). For clause (2) we observe that splitting a segment into two non-empty ones extends and otherwise preserves the $(s_k, l_{k,i}, l_{k,f}) < (s_{k+1}, l_{k+1,i}, l_{k+1,f})$ ordering, consistent with the indexing, and fusing two contiguous segments in the same fragment of $\mathcal{S}$ in the obvious way shrinks and otherwise preserves the $(s_k, l_{k,i}, l_{k,f}) < (s_{k+1}, l_{k+1,i}, l_{k+1,f})$ ordering, consistent with the indexing. Now the way we have constructed the segments in $\mathcal{S}$ of $ll_{(a,b)}$ is by splitting and fusing the original segments in $\mathcal{S}$ of $ll_a$, and similarly for the segments in $\mathcal{U}$ of $ll_{(a,b)}$ — since we first split them via the construction of Proposition 5.1, and then aggregated the results via (5.4) and (5.5). Both activities thus yield orderings on their elements consistent with the indexing and clause (2) follows. Clause (3) follows because (5.4) and (5.5) and their repetition up to a fixed point ensure there are no pairs contiguous segments left in $ll_{(a,b)}$. Clause (4) holds trivially by construction, while clause (5) holds because $ll_{(a,b)}$ was constructed to represent the composition of two (total and surjective) bijections between all the simulable steps of $\mathcal{S}$, $\mathcal{T}$, and $\mathcal{U}$. ☺

**Proposition 5.4** The vertical composition of Proposition 5.3 is associative.

*Proof.* This essentially reduces to the observation that the construction rests on a composition of bijections. ☺

Proposition 5.3 is an elegant result, but it rested on two assumptions that do not generally hold for an arbitrary pair of simulation relations $\mathcal{S}$ $\Sigma(ll_a)$ $\mathcal{T}$ and $\mathcal{T}$ $\Sigma(ll_b)$ $\mathcal{U}$ between multifragments, namely that the two retrenchments in question were biretrenchments, and that the sets of *AC*-simulable and *CE*-simulable steps of *Conc* were the same. Let us examine what happens when these assumptions are relaxed.

If we have conventional retrenchments rather than biretrenchments, i.e. operation PO (3.4) does not hold, then we cannot deduce the *AE*-nonsimulability of abstract *AC*-nonsimulating steps on the basis of their *AC*-nonsimulability, as we did above. Thus there might be an *AC*-nonsimulating *Abs* step, say $u$ -($i$, $Op_A$, $o$)-› $u'$ which is never-

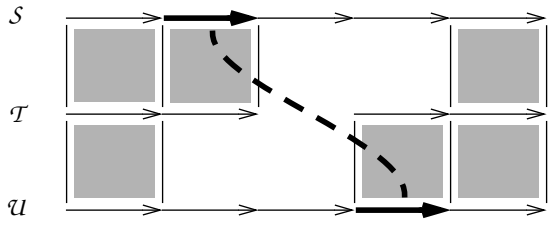



Fig. 3

theless in simulation with some *Exe* step, say $w$ -($k$, $Op_E$, $q$)-› $w'$, even if neither of them is in simulation with any *Conc* step whatsoever. This happens because the composed 'in simulation' relation, the analogue using (2.3)-(2.6) of (4.4) for the composed retrenchment, does not require there to be any intermediate *Conc* step to witness its truth. Note that the *AE*-nonsimulability of *CE*-nonsimulable *Exe* steps continues to hold because the conventional retrenchment operation PO still holds.

The repercussions of the above depend on whether the sets of *AC*-simulable and *CE*-simulable steps of *Conc* are the same in the composition. If they are, then all the *AE*-simulable *Exe* steps in $\mathcal{U}$ are catered for in the bijection of simulable steps given by $ll_{(a,b)}$. Therefore for any *AC*-nonsimulating *Abs* step $u$ -($i$, $Op_A$, $o$)-› $u'$ which occurs in an exception of $\mathcal{S}$ and is in simulation with an *Exe* step $w$ -($k$, $Op_E$, $q$)-› $w'$, either $w$ -($k$, $Op_E$, $q$)-› $w'$ does not occur in $\mathcal{U}$, or if it does, $w$ -($k$, $Op_E$, $q$)-› $w'$ is already in simulation with some other *Abs* step described in $ll_{(a,b)}$. Thus the simulating fragments described by $ll_{(a,b)}$ enjoy a maximality property, despite there possibly being additional *AE*-simulable *Abs* steps in $\mathcal{S}$. In any case we lose clause (5).(i) of Definition 4.6 (though (5).(ii) still holds), and thus we cannot assert $\mathcal{S}$ $\Sigma(ll_{(a,b)})$ $\mathcal{U}$.

If the sets of *AC*-simulable and *CE*-simulable steps of *Conc* are not the same in the composition, then the constructed $ll_{(a,b)}$ may describe a proper subset of a maximal collection of simulating fragments of $\mathcal{S}$ and $\mathcal{U}$ as the next counterexample shows.

**Counterexample 5.5** Consider $\mathcal{S}$, $\mathcal{T}$, and $\mathcal{U}$, illustrated in Fig. 3. Steps 1 and 2 of $\mathcal{U}_0$ are *CE*-nonsimulable and thus *AE*-nonsimulable, and steps 2 and 3 of $\mathcal{S}_0$ are *AC*-nonsimulable, and thus *AE*-nonsimulable if we have biretrenchments. Step 1 of $\mathcal{S}_0$ is in *AC*-simulation with a step of $\mathcal{T}$, but there is no *CE*-simulation to compose it with so it falls outside the domain of the $ll_{(a,b)}$ constructed in Proposition 5.3, which is $[((0,0,0),(0,0,0)), ((0,4,4),(0,4,4))]$; similarly for step 3 of $\mathcal{U}_0$. But there is no fundamental reason why step 1 of $\mathcal{S}_0$ cannot be in *AE*-simulation with step 3 of $\mathcal{U}_0$, and if it is, it leads to $\mathcal{S}$ $\Sigma(ll'_{(a,b)})$ $\mathcal{U}$, where $ll'_{(a,b)} = [((0,0,0),(0,0,0)), ((0,1,1),(0,3,3)), ((0,4,4),(0,4,4))]$, provided of course that steps 2 and 3 of $\mathcal{S}_0$ are *AE*-nonsimulable. If these steps are *AE*-simulable, then we still get a maximality property of the simulation described by $ll'_{(a,b)}$ even though not all of the criteria of Definition 4.6 are met.

The lessons of these points are thus clear. If we lose biretrenchments, a composition may lead to additional *AE*-simulable *Abs* steps because *AC*-nonsimulating *Abs*

steps may become *AE*-simulable. If we lose equality of the *AC*-simulable and *CE*-simulable steps of *Conc*, the ability of $ll_{(a,b)}$ to control all of the steps of $\mathcal{S}$ and $\mathcal{U}$ declared simulable in $ll_a$ and $ll_b$ is forfeited, with the consequence that unpredictable additional simulations between steps of $\mathcal{S}$ and $\mathcal{U}$ may arise, and not just nicely in-order ones as in Fig. 3, as cases like Fig. 2.(e) can easily arise. If we lose both properties, we get both problems, and they can also interact, as the *AE*-simulable *AC*-non-simulable *Abs* steps, may simulate with the *CE*-simulable *Exe* steps liberated from $ll_b$ supervision.

In the end, without the support of the stronger conditions, the bijection in $ll_{(a,b)}$ merely provides a safe lower bound on possible simulating segments in $\mathcal{S}$ and $\mathcal{U}$. What remains of these situations can be salvaged in the following.

**Definition 5.6** The notion $\mathcal{S}\ \Sigma(ll)^{-}\ \mathcal{U}$ is given by Definition 4.6 except that clause (5).(i) is omitted.

**Remark 5.7** Note that this definition permits segments of nonsimulating but nevertheless simulable abstract steps of arbitrary length wihin $\mathcal{S}$. This is not very satisfactory as a concept for capturing simulability, hence we do not give it a high profile. Its only features worthy of note are in the following.

If $p$ is a sequence of pairs $[(p_{0a},p_{0b}), (p_{1a},p_{1b}), \ldots]$, let snd($p$) be the sequence of second projections of elements of $p$, i.e. $[p_{0b}, p_{1b}, \ldots]$.

**Lemma 5.8** Suppose $\mathcal{S}\ \Sigma(ll)^{-}\ \mathcal{U}$ and $\mathcal{S}\ \Sigma(ll')^{-}\ \mathcal{U}$ both hold. Then snd($ll$) = snd($ll'$).

*Proof.* This is an easy unidirectional weakening of Proposition 4.7 . ☺

**Proposition 5.9** Suppose $\mathcal{S}$, $\mathcal{T}$, $\mathcal{U}$, are multifragments of systems *Abs*, *Conc*, *Exe*, which are related via retrenchments captured in the notations used in (2.3)-(2.6). Suppose both $\mathcal{S}\ \Sigma(ll_a)^{-}\ \mathcal{T}$ and $\mathcal{T}\ \Sigma(ll_b)^{-}\ \mathcal{U}$ hold, as in Proposition 5.3. Suppose each *CE*-simulable step of *Conc* is also *AC*-simulable, and let $ll_{(a,b)}$ be constructed as in Proposition 5.3. Then $\mathcal{S}\ \Sigma(ll_{(a,b)})^{-}\ \mathcal{U}$ according to the 'in simulation' definition stemming from the vertically composed retrenchment defined by (2.3)-(2.6).

*Proof.* This is also an easy unidirectional weakening of Proposition 5.3. ☺

The inclusion of the *CE*-simulable steps of *Conc* in the *AC*-simulable ones is perhaps a more arguable scenario than their exact equality (though exact equality is a feature of the retrenchments in [Banach and Poppleton (2003)]). One can imagine the approach to a truly executable system, with all its awkward boundary cases etc., to be via a series of simplified models, each incorporating more of the low level detail than its predecessor, and each thus more distant from the original abstract model than its predecessor; and all this happening in a monotonic manner with respect to inclusion of simulable steps at intermediate models. The composition properties of such a process are neatly captured in Proposition 5.9, though Remark 5.7 considerably diminishes enthusiasm for the $\Sigma(ll)^{-}$ notion. We see that the general results we are able to prove about compositions of simulations between (multi)fragments under retrenchment are not very strong, and so practical cases will often need to be handled by ad hoc means.

**Remark 5.10** The above vertical composition results, although needing additional conditions to hold in the strongest cases, have nevertheless influenced the design of the preceding definitions, in particular Definition 4.6. For instance consider Fig. 2.(c)

which shows a simulation between two multifragments $\mathcal{S} = [\mathcal{S}_0, \mathcal{S}_1, \mathcal{S}_2]$ and $\mathcal{T} = [\mathcal{T}_0, \mathcal{T}_1, \mathcal{T}_2]$, such that $\mathcal{S}_0$ and $\mathcal{S}_1$ simulate with $\mathcal{T}_0$, and $\mathcal{S}_2$ simulates with $\mathcal{T}_1$ and $\mathcal{T}_2$; and where these two pieces of the simulation are completely separate. Now imagine that the two pieces refered to multifragments of three systems, *Abs*, *Conc* and *Exe*, such that the first piece was a simulation between $\mathcal{S} = [\mathcal{S}_0, \mathcal{S}_1]$ and $\mathcal{T} = [\mathcal{T}_0]$, and the second piece was a simulation between $\mathcal{T} = [\mathcal{T}_0]$, and $\mathcal{U} = [\mathcal{U}_0, \mathcal{U}_1]$. Vertically composing them would give a simulation between $\mathcal{S} = [\mathcal{S}_0, \mathcal{S}_1]$ and $\mathcal{U} = [\mathcal{U}_0, \mathcal{U}_1]$. But this would also consist of two disconnected pieces, $\mathcal{S}_0$ simulating $\mathcal{U}_0$, and $\mathcal{S}_1$ simulating $\mathcal{U}_1$. This shows that insisting on stronger connectivity properties in Definition 4.6 would give a concept of simulation that failed to compose vertically even under favourable conditions. Similar remarks explain why 'holes' in simulation as in Fig. 2.(f) are permitted. They can be easily generated by vertical composition of more innocent situations.

# 6 System Properties and Simulation Transformers

In this section, the notions from the preceding sections are applied to show how properties of one system transform through a given retrenchment to yield properties of the other system. Given the rather flexible nature of retrenchment, it is of just as much interest to examine how properties of the concrete system map to the abstract system, as it is to see how abstract properties map to the concrete system; this is despite the fact that the latter perspective is significantly more important when one takes a refinement oriented point of view. Also for the same reason, the previous sections were predominantly built on the symmetric 'in simulation' concept of Definition 4.3, rather than using a more asymmetric notion ensuing from the asymmetry of the retrenchment POs.

We will identify a property of a system with the set of system behaviours that display that property, where a system behaviour is a multifragment.[4] This conception of system behaviour is more generous than in conventional scenarios, the reason being, that since we will use simulability relationships between abstract and concrete systems as the main vehicle for translating properties at one level into properties at the other, and, as we discussed earlier, simulability relationships exhibit various kinds of misbehaviour illustrated in examples, the view that a system behaviour can be a single initial execution fragment is too simplistic.

**Definition 6.1** Suppose we are given a system *Abs*, and let a set of steps $\Sigma(Abs) \subseteq stp_A$ be nominated as the simulable steps of *Abs*. Then we say that a multifragment $\mathcal{S}$ of *Abs* is $\Sigma(Abs)$-curt iff, for all $k, k+d_k \in \mathrm{dom}(\mathcal{S})$ such that fragments $\mathcal{S}_k$ and $\mathcal{S}_{k+d_k}$ both contain simulable steps and $d_k \geq 1$ is the smallest element of $\mathsf{NAT}^+$ for which this holds (for the given $k$), then $\mathrm{st}'(\mathcal{S}_k{}^{\bullet}[\mathrm{md}(\mathcal{S}_k{}^{\bullet})]) \neq \mathrm{st}({}^{\bullet}\mathcal{S}_{k+d_k}[\mathrm{mind}({}^{\bullet}\mathcal{S}_{k+d_k})])$, where the lead-out discarded from $\mathcal{S}_k$ to give $\mathcal{S}_k{}^{\bullet}$ is the longest non-$\Sigma(Abs)$ suffix of $\mathcal{S}_k$, and the lead-in discarded from $\mathcal{S}_{k+d_k}$ to give ${}^{\bullet}\mathcal{S}_{k+d_k}$ is the longest non-$\Sigma(Abs)$ prefix of $\mathcal{S}_{k+d_k}$.

**Definition 6.2** An unfettered property $\mathcal{SS}$ of a system *Abs* is just a set of multifragments of that system.

---

4. We are thus deliberately excluding 'properties' that only concern at most the state, input, and output spaces, which can be discussed via the relations $G$, $P_{Op}$, $O_{Op}$, $C_{Op}$.

**Definition 6.3** A property $\mathcal{SS}$ of a system $\mathit{Abs}$ with respect to a set of simulable steps $\Sigma(\mathit{Abs})$, is an unfettered property for which each multifragment $\mathcal{S} \in \mathcal{SS}$ is both curt and $\Sigma(\mathit{Abs})$-curt.

**Remark 6.4** The intention of Definition 6.3 is to suppress fruitless fragmentation of the multifragments comprising a property. In a sense, a curt and $\Sigma(\mathit{Abs})$-curt multifragment acts as a code for all its more finely fragmented derivatives (including when these are embellished with nontrivial lead-ins and lead-outs and intervening nonsimulating fragments). In fact we can regard a property as encoding a down-closed set of such more finely fragmented derivatives if we regard a single application of step [1] or [2] in Definition 6.5 below as a progression up a partial order, in a rather obvious way.

**Definition 6.5** Let $\mathcal{S}''$ be a multifragment of a system $\mathit{Abs}$ with respect to a set of simulable steps $\Sigma(\mathit{Abs})$. Let $\mathcal{S} = \mathsf{PR}_\mathrm{A}(\mathcal{S}'')$ be the multifragment derived from $\mathcal{S}''$ by the following steps.

[1] Concatenate a pair of concatenable adjacent fragments in $\mathcal{S}''$ (i.e. replace a pair $\mathcal{S}_k$ and $\mathcal{S}_{k+1}$ of $\mathcal{S}''$ such that $\mathrm{st}'(\mathcal{S}_k[\mathrm{md}(\mathcal{S}_k)]) = \mathrm{st}(\mathcal{S}_{k+1}[0])$ by their concatenation). Repeat until no further change is possible, to produce multifragment $\mathcal{S}'$.

[2] In $\mathcal{S}'$, whenever fragments $\mathcal{S}_k$ and $\mathcal{S}_{k+d_k}$ both contain simulable steps, and $d_k \geq 1$ is minimal for the given $k$, and $\mathrm{st}'(\mathcal{S}_k^\bullet[\mathrm{md}(\mathcal{S}_k^\bullet)]) = \mathrm{st}({}^\bullet\mathcal{S}_{k+d_k}[\mathrm{mind}({}^\bullet\mathcal{S}_{k+d_k})])$, then replace $\mathcal{S}_k$ and $\mathcal{S}_{k+d_k}$ by the concatenation of $\mathcal{S}_k^\bullet$ and ${}^\bullet\mathcal{S}_{k+d_k}$, discarding all $\mathcal{S}_{k+m}$ for $0 < m < d_k$. Repeat until no further change is possible, to produce multifragment $\mathcal{S} = \mathsf{PR}_\mathrm{A}(\mathcal{S}'')$.

We write $\mathsf{PR}_\mathrm{A}$ also for the set extension of $\mathsf{PR}_\mathrm{A}$ to unfettered properties.

**Lemma 6.6** For every $\mathcal{S}$, $\mathsf{PR}_\mathrm{A}(\mathcal{S})$ is curt and $\Sigma(\mathit{Abs})$-curt. $\mathsf{PR}_\mathrm{A}$ is a function on multifragments which acts as the identity on curt and $\Sigma(\mathit{Abs})$-curt multifragments. By extension, for every unfettered property $\mathcal{SS}$, $\mathsf{PR}_\mathrm{A}(\mathcal{SS})$ is a property, and thus $\mathsf{PR}_\mathrm{A}$ is a function on unfettered properties which acts as the identity on properties.

*Proof.* That for any $\mathcal{S}$, $\mathsf{PR}_\mathrm{A}(\mathcal{S})$ is curt and $\Sigma(\mathit{Abs})$-curt is clear as step [1] removes all opportunuites for concatenation, and step [2] ensures $\Sigma(\mathit{Abs})$-curtness. Suitably understood, all instances of the steps of the procedure in Definition 6.5 are non-interfering, so their application in any order always leads to an unambiguous result, and thus $\mathsf{PR}_\mathrm{A}$ is a function. If a multifragment is curt and $\Sigma(\mathit{Abs})$-curt to start with, then steps [1] and [2] of Definition 6.5 are both null and $\mathsf{PR}_\mathrm{A}$ acts as the identity. The remainder is by set extension. ☺

**Proposition 6.7** Let $\mathcal{T}$ be a concrete multifragment, $\mathcal{SS}$ be an abstract unfettered property, and $\mathcal{S} \,\Sigma\, \mathcal{T}$ for some $\mathcal{S} \in \mathcal{SS}$, all in the context of a retrenchment which defines the simulable steps. Then $\mathsf{PR}_\mathrm{A}(\mathcal{S}) \,\Sigma\, \mathcal{T}$ where $\mathsf{PR}_\mathrm{A}(\mathcal{S}) \in \mathsf{PR}_\mathrm{A}(\mathcal{SS})$.

*Proof.* Suppose $\mathcal{S} \,\Sigma\, \mathcal{T}$ for some $\mathcal{S} \in \mathcal{SS}$. Then if $\mathcal{S}$ is not curt and $\Sigma(\mathit{Abs})$-curt, the procedure in Definition 6.5 at worst concatenates some fragments and erases some nonsimulable steps and fragments from $\mathcal{S}$, yielding $\mathsf{PR}_\mathrm{A}(\mathcal{S})$. But all the simulable steps of $\mathcal{S}$ survive this process, in the same order (and possibly more concatenated than in $\mathcal{S}$). So $\mathsf{PR}_\mathrm{A}(\mathcal{S}) \,\Sigma\, \mathcal{T}$ is not hard to prove. ☺

All of the above have concrete counterparts, $\mathsf{PR}_C$ being the concrete analogue of $\mathsf{PR}_A$. Below we will continue to mainly just cite the abstract versions of results without comment, assuming the concrete versions tacitly understood.

**Assumption 6.8** From now on we will assume that all abstract multifragments are curt and $\Sigma(\mathit{Abs})$-curt, and all concrete multifragments are curt and $\Sigma(\mathit{Conc})$-curt; all with respect to sets of simulating steps $\Sigma(\mathit{Abs})$ and $\Sigma(\mathit{Conc})$ that should be clear from the context.

Proposition 6.7 and Remark 6.4 above justify this simplification, avoiding the constant need to deal with the $\mathsf{PR}$ functions and with overfragmented multifragments.

**Definition 6.9** Suppose given two systems $\mathit{Abs}$ and $\mathit{Conc}$, and let $\Sigma(\mathit{Abs})$ and $\Sigma(\mathit{Conc})$ be the sets of abstract and concrete simulable steps derived from a retrenchment via Definition 4.3. Let $\mathcal{SS}$ be a property of $\mathit{Abs}$ and $\mathcal{TT}$ be a property of $\mathit{Conc}$ (with respect to $\Sigma(\mathit{Abs})$ and $\Sigma(\mathit{Conc})$ respectively). Then the simulation transformers $[\Sigma]$ and $\langle\Sigma\rangle$ are defined as follows.

$$[\Sigma]\mathcal{SS} = \{\mathcal{T} \mid (\exists\, \mathcal{S} \bullet \mathcal{S} \in \mathcal{SS} \wedge \mathcal{S}\ \Sigma\ \mathcal{T}) \wedge (\forall\, \mathcal{S} \bullet \mathcal{S}\ \Sigma\ \mathcal{T} \Rightarrow \mathcal{S} \in \mathcal{SS})\} \tag{6.1}$$

$$\langle\Sigma\rangle\mathcal{SS} = \{\mathcal{T} \mid (\exists\, \mathcal{S} \bullet \mathcal{S} \in \mathcal{SS} \wedge \mathcal{S}\ \Sigma\ \mathcal{T})\} \tag{6.2}$$

$$[\Sigma]\mathcal{TT} = \{\mathcal{S} \mid (\exists\, \mathcal{T} \bullet \mathcal{T} \in \mathcal{TT} \wedge \mathcal{S}\ \Sigma\ \mathcal{T}) \wedge (\forall\, \mathcal{T} \bullet \mathcal{S}\ \Sigma\ \mathcal{T} \Rightarrow \mathcal{T} \in \mathcal{TT})\} \tag{6.3}$$

$$\langle\Sigma\rangle\mathcal{TT} = \{\mathcal{S} \mid (\exists\, \mathcal{T} \bullet \mathcal{T} \in \mathcal{TT} \wedge \mathcal{S}\ \Sigma\ \mathcal{T})\} \tag{6.4}$$

Note the key role played by Assumption 6.8 in Definition 6.9. Without it the second conjunct in (6.1) would cause $[\Sigma]\mathcal{SS}$ to be empty in general, as for a given $\mathcal{T}$, $\mathcal{S}\ \Sigma\ \mathcal{T}$ does not by any means guarantee that $\mathcal{S}$ is curt and $\Sigma(\mathit{Abs})$-curt, as it needs to be if $\mathcal{S} \in \mathcal{SS}$ is to hold. Likewise for (6.3).

**Proposition 6.10** The following hold for the transformers $\langle\Sigma\rangle$ and $[\Sigma]$.

(1) $\mathcal{SS}_1 \subseteq \mathcal{SS}_2 \Rightarrow \langle\Sigma\rangle\mathcal{SS}_1 \subseteq \langle\Sigma\rangle\mathcal{SS}_2 \wedge [\Sigma]\mathcal{SS}_1 \subseteq [\Sigma]\mathcal{SS}_2$

(2) $[\Sigma]\mathcal{SS} \subseteq \langle\Sigma\rangle\mathcal{SS}$

(3) $\mathcal{TT} \subseteq [\Sigma]\mathcal{SS} \Leftrightarrow \langle\Sigma\rangle\mathcal{TT} \subseteq \mathcal{SS}$

*Proof.* Immediate from the definitions. ☺

The functions $[\Sigma]$ and $\langle\Sigma\rangle$ (interpreted in both abstract-to-concrete and concrete-to-abstract directions) provide the fundamental simulation transformers of properties between the two levels. The purely relational nature of the $\mathcal{S}\ \Sigma\ \mathcal{T}$ simulability relationship means that $[\Sigma]$ and $\langle\Sigma\rangle$ work like fairly standard box and diamond modal operators on a modal algebra, and below we develop some of the natural consequences of this formalism. We only say 'fairly standard' since, unlike many similar situations, we incorporate the clause $(\exists\, \mathcal{S} \bullet \mathcal{S} \in \mathcal{SS} \wedge \mathcal{S}\ \Sigma\ \mathcal{T})$ in (6.1). Its omission would admit into $[\Sigma]\mathcal{SS}$, concrete fragments that are entirely non-simulable. These have no place in the definition of a mapping via simulation of $\mathcal{SS}$. Similarly for (6.3).

In general, we obtain a web of richly interrelated properties by applying arbitrary strings in the alphabet $\{\langle\Sigma\rangle, [\Sigma]\}$ to properties at the abstract and concrete level, with each occurrence of $\langle\Sigma\rangle$ and $[\Sigma]$ interpreted as an abstract-to-concrete or concrete-to-abstract transformer in the only way that makes sense[5].

**Definition 6.11** Let $\mathcal{SS}$ be an abstract property. Then we define $\mathcal{SS}^\bullet$ to be the property $\{\mathcal{S} \mid (\exists\, \mathcal{T} \bullet \mathcal{S} \in \mathcal{SS} \wedge \mathcal{S}\ \Sigma\ \mathcal{T})\}$. If $\mathcal{SS} = \mathcal{SS}^\bullet$, we say that $\mathcal{SS}$ is proper.

Thus $\mathcal{SS}^\bullet$ is $\mathcal{SS}$ with any non-simulating multifragments removed.

**Proposition 6.12** Let $\mathcal{SS}$ be an abstract property.

(1) $\ldots\ (\langle\Sigma\rangle[\Sigma])^3\mathcal{SS} \subseteq (\langle\Sigma\rangle[\Sigma])^2\mathcal{SS} \subseteq \langle\Sigma\rangle[\Sigma]\mathcal{SS} \subseteq \mathcal{SS}^\bullet \subseteq \mathcal{SS}$

(2) $\mathcal{SS}^\bullet \subseteq [\Sigma]\langle\Sigma\rangle\mathcal{SS} \subseteq ([\Sigma]\langle\Sigma\rangle)^2\mathcal{SS} \subseteq ([\Sigma]\langle\Sigma\rangle)^3\mathcal{SS}\ \ldots$

(3) $[\Sigma]\langle\Sigma\rangle[\Sigma]\mathcal{SS} = [\Sigma]\mathcal{SS}$

(4) $\langle\Sigma\rangle[\Sigma]\langle\Sigma\rangle\mathcal{SS} = \langle\Sigma\rangle\mathcal{SS}$

(5) $w[\Sigma]\langle\Sigma\rangle[\Sigma]\mathcal{SS} = w[\Sigma]\mathcal{SS}$, where $w \in \{\langle\Sigma\rangle, [\Sigma]\}^*$

(6) $w\langle\Sigma\rangle[\Sigma]\langle\Sigma\rangle\mathcal{SS} = w\langle\Sigma\rangle\mathcal{SS}$, where $w \in \{\langle\Sigma\rangle, [\Sigma]\}^*$

*Proof.* For (1), $\mathcal{SS}^\bullet \subseteq \mathcal{SS}$ is obvious, after which we argue as follows. Let $\mathcal{S} \in \langle\Sigma\rangle[\Sigma]\mathcal{SS}$. Then by (6.4), for some $\mathcal{T} \in [\Sigma]\mathcal{SS}$, $\mathcal{S}\ \Sigma\ \mathcal{T}$ holds. Now if $\mathcal{T} \in [\Sigma]\mathcal{SS}$ and $\mathcal{S}\ \Sigma\ \mathcal{T}$ holds, then by (6.1), $\mathcal{S} \in \mathcal{SS}$. Also, because $\mathcal{S}\ \Sigma\ \mathcal{T}$ and $\mathcal{S} \in \mathcal{SS}$ both hold, then $\mathcal{S} \in \mathcal{SS}^\bullet$. So $\langle\Sigma\rangle[\Sigma]\mathcal{SS} \subseteq \mathcal{SS}^\bullet$. Since $\langle\Sigma\rangle[\Sigma]\mathcal{SS}$, and hence $(\langle\Sigma\rangle[\Sigma])^n\mathcal{SS}$ are all properties, the cases for $(\langle\Sigma\rangle[\Sigma])^n$ for $n > 1$ follow by induction.

For (2), we have the following. Let $\mathcal{S} \in \mathcal{SS}^\bullet$. Then there is some $\mathcal{T}$ such that $\mathcal{S}\ \Sigma\ \mathcal{T}$; also for any such $\mathcal{T}$, we have $\mathcal{T} \in \langle\Sigma\rangle\mathcal{SS}$. Therefore such a $\mathcal{T}$ witnesses the first clause of $\mathcal{S} \in [\Sigma]\langle\Sigma\rangle\mathcal{SS}$. For the second clause, for the same $\mathcal{S}$, for any $\mathcal{T}$, if $\mathcal{S}\ \Sigma\ \mathcal{T}$ then $\mathcal{T} \in \{\mathcal{T} \mid \mathcal{S} \in \mathcal{SS}^\bullet \wedge \mathcal{S}\ \Sigma\ \mathcal{T}\} \subseteq \langle\Sigma\rangle\mathcal{SS}$, or more briefly $\mathcal{S}\ \Sigma\ \mathcal{T} \Rightarrow \mathcal{T} \in \langle\Sigma\rangle\mathcal{SS}$, and so we deduce $\mathcal{S} \in [\Sigma]\langle\Sigma\rangle\mathcal{SS}$. Altogether we get $\mathcal{SS}^\bullet \subseteq [\Sigma]\langle\Sigma\rangle\mathcal{SS}$. The remaining steps are by induction again.

For (3), we have $\langle\Sigma\rangle[\Sigma]\mathcal{SS} \subseteq \mathcal{SS}$ by (1), so that $[\Sigma]\langle\Sigma\rangle[\Sigma]\mathcal{SS} \subseteq [\Sigma]\mathcal{SS}$ by monotonicity. On the other hand, since $([\Sigma]\mathcal{SS})^\bullet = [\Sigma]\mathcal{SS}$, then $[\Sigma]\mathcal{SS} \subseteq [\Sigma]\langle\Sigma\rangle([\Sigma]\mathcal{SS})$ by the concrete counterpart of (2), giving $[\Sigma]\langle\Sigma\rangle[\Sigma]\mathcal{SS} = [\Sigma]\mathcal{SS}$. The reasoning for (4) is similar. Now (5) and (6) follow by noting that $\langle\Sigma\rangle$ and $[\Sigma]$ are both functions on properties. We are done. ☺

Of particular interest are properties $\mathcal{SS}$ such that $\langle\Sigma\rangle[\Sigma]\mathcal{SS} = \mathcal{SS}$ or $\mathcal{SS} = [\Sigma]\langle\Sigma\rangle\mathcal{SS}$, or better still, when both are true.

**Definition 6.13** Let $\mathcal{SS}$ be an abstract property. Then $\mathrm{core}(\mathcal{SS}) = \langle\Sigma\rangle[\Sigma]\mathcal{SS} \subseteq \mathcal{SS}$, is called the core of $\mathcal{SS}$. If $\mathrm{core}(\mathcal{SS}) = \mathcal{SS}$ then $\mathcal{SS}$ is a core property. Similarly $\mathcal{SS}^\bullet \subseteq \mathrm{rect}(\mathcal{SS}) = [\Sigma]\langle\Sigma\rangle\mathcal{SS}$, gives the rectification of $\mathcal{SS}$. If $\mathrm{rect}(\mathcal{SS}) = \mathcal{SS}$ then $\mathcal{SS}$ is a rectified property. A property is weakly robust iff it is a proper, core and rectified property, i.e. $\langle\Sigma\rangle[\Sigma]\mathcal{SS} = \mathcal{SS} = [\Sigma]\langle\Sigma\rangle\mathcal{SS}$. A property is robust iff it satisfies $\mathcal{SS}^\bullet = \mathcal{SS}$ and $[\Sigma]\mathcal{SS} = \langle\Sigma\rangle\mathcal{SS}$.

Observing that $[\Sigma]\mathcal{SS}$ provides the biggest concrete property $\mathcal{TT}$ such that arbitrary translations (via simulation) of its elements back to the abstract level remain within $\mathcal{SS}$, the core of $\mathcal{SS}$ provides the smallest subproperty of $\mathcal{SS}$ which contains all these arbitrary translations. Similarly $\langle\Sigma\rangle\mathcal{SS}$ provides the biggest concrete property $\mathcal{TT}$ reachable via arbitrary translations of the elements of $\mathcal{SS}$, and the rectification of $\mathcal{SS}$

5. Obviously if we were dealing with more than two levels of abstraction we would have to enrich the $\langle\Sigma\rangle$ and $[\Sigma]$ notations somewhat.

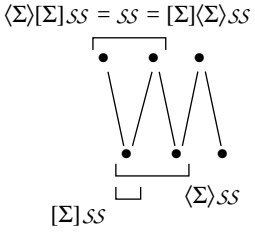



Fig. 4

gives the largest superproperty of $\mathcal{SS}^{\bullet}$ such that arbitrary translations of its elements do not breach $\mathcal{TT}$. Weakly robust and robust properties embody both virtues.

**Proposition 6.14** Let $\mathcal{SS}$ be an abstract property.

(1) $[\Sigma]\mathcal{SS}$ is rectified.

(2) $\langle\Sigma\rangle\mathcal{SS}$ is core.

(3) If $\mathcal{SS}$ is robust, then $\mathcal{SS}$ is weakly robust.

*Proof.* Proposition 6.12.(3)-(4) give (1) and (2). For (3), suppose $\mathcal{SS}$ is robust. Then $[\Sigma]\mathcal{SS} = \langle\Sigma\rangle\mathcal{SS}$, so $\langle\Sigma\rangle[\Sigma]\mathcal{SS} = \langle\Sigma\rangle\langle\Sigma\rangle\mathcal{SS}$. Also, since $\langle\Sigma\rangle[\Sigma]\mathcal{SS} \subseteq \mathcal{SS}^{\bullet}$ by Proposition 6.12.(1), and $\mathcal{SS}^{\bullet} \subseteq \langle\Sigma\rangle\langle\Sigma\rangle\mathcal{SS}$, and $\mathcal{SS}^{\bullet} = \mathcal{SS}$ by robustness, we deduce $\langle\Sigma\rangle[\Sigma]\mathcal{SS} = \mathcal{SS}$. We know that $\mathcal{SS} = \mathcal{SS}^{\bullet} \subseteq [\Sigma]\langle\Sigma\rangle\mathcal{SS}$ by Proposition 6.12.(2), and by robustness, $[\Sigma]\langle\Sigma\rangle\mathcal{SS} = [\Sigma][\Sigma]\mathcal{SS}$ follows from $[\Sigma]\mathcal{SS} = \langle\Sigma\rangle\mathcal{SS}$. So we just have to show that $[\Sigma][\Sigma]\mathcal{SS} \subseteq \mathcal{SS}$. So suppose that $\mathcal{S} \in [\Sigma][\Sigma]\mathcal{SS}$ but $\mathcal{S} \notin \mathcal{SS}$. Then from $\mathcal{S} \in [\Sigma][\Sigma]\mathcal{SS}$ we have $(\exists\, \mathcal{T} \bullet \mathcal{T} \in [\Sigma]\mathcal{SS} \wedge \mathcal{S}\,\Sigma\,\mathcal{T}) = (*)$ and $(\forall\, \mathcal{T} \bullet \mathcal{S}\,\Sigma\,\mathcal{T} \Rightarrow \mathcal{T} \in [\Sigma]\mathcal{SS})$. Suppose $\mathcal{T}$ satisfies $(*)$. But $(*)$ says $\mathcal{T} \in [\Sigma]\mathcal{SS}$, and so by (6.1) we have $\mathcal{S}\,\Sigma\,\mathcal{T} \Rightarrow \mathcal{S} \in \mathcal{SS}$, contradicting $\mathcal{S} \notin \mathcal{SS}$. ☺

**Counterexample 6.15** Fig. 4 shows that the natural converse of Proposition 6.14.(3) does not hold. The blobs represent individual multifragments and the lines the $\Sigma$ relationship. We have $\langle\Sigma\rangle[\Sigma]\mathcal{SS} = \mathcal{SS} = [\Sigma]\langle\Sigma\rangle\mathcal{SS}$ but $[\Sigma]\mathcal{SS} \neq \langle\Sigma\rangle\mathcal{SS}$.

**Proposition 6.16** Let $\mathcal{SS}$ be an abstract property. Then there is a partition of $\mathcal{CMF}_{\mathrm{A}}$, the set of all curt and $\Sigma(\mathit{Abs})$-curt multifragments, and partitions of $\mathcal{CMF}_{\mathrm{C}}$, the set of all curt and $\Sigma(\mathit{Conc})$-curt multifragments, as follows:

(1) $\mathcal{CMF}_{\mathrm{A}} = \mathcal{SS}^{\bullet} \uplus \overline{\mathcal{SS}}^{\bullet} \uplus \mathcal{CMF}_{\mathrm{A}}^{\mathcal{NS}}$

(2) $\mathcal{CMF}_{\mathrm{C}} = [\Sigma]\mathcal{SS} \uplus [\Sigma](\overline{\mathcal{SS}}) \uplus (\langle\Sigma\rangle\mathcal{SS} \cap \langle\Sigma\rangle(\overline{\mathcal{SS}})) \uplus \mathcal{CMF}_{\mathrm{C}}^{\mathcal{NS}}$

where $\overline{\mathcal{SS}}^{\bullet}$ is clearly a property, where $\mathcal{CMF}_{\mathrm{A}}^{\mathcal{NS}} = \{\mathcal{S} \mid \neg(\exists\, \mathcal{T} \bullet \mathcal{S}\,\Sigma\,\mathcal{T})\} \subseteq \mathcal{CMF}_{\mathrm{A}}$, $\mathcal{CMF}_{\mathrm{A}}^{\mathcal{NS}}$ being the nonsimulating subset of $\mathcal{CMF}_{\mathrm{A}}$, and where $\mathcal{CMF}_{\mathrm{A}}^{\mathcal{S}}$ is its complement, and with similar definitions for $\mathcal{CMF}_{\mathrm{C}}^{\mathcal{NS}}$ and $\mathcal{CMF}_{\mathrm{C}}^{\mathcal{S}}$. Abstract complements are taken in $\mathcal{CMF}_{\mathrm{A}}$. Moreover:

(3) If $\mathcal{SS}$ is core, then $\overline{\mathcal{SS}}^{\bullet}$ is rectified.

(4) If $\mathcal{SS}$ is rectified, then $\overline{\mathcal{SS}}^{\bullet}$ is core.

(5) If $\mathcal{SS}$ is robust, then $(\langle\Sigma\rangle\mathcal{SS} \cap \langle\Sigma\rangle(\overline{\mathcal{SS}}^{\bullet})) = \emptyset$ in (2).

*Proof.* For (1), it is clear that $\overline{\mathcal{SS}}^{\bullet}$ is a property and that the union is disjoint. For (2) we note that for any simulable $\mathcal{T}$, one of the following holds. Either $(\forall\, \mathcal{S} \bullet \mathcal{S}\ \Sigma\ \mathcal{T} \Rightarrow \mathcal{S} \in \mathcal{SS})$ holds and $\mathcal{T} \in [\Sigma]\mathcal{SS}$, or $(\forall\, \mathcal{S} \bullet \mathcal{S}\ \Sigma\ \mathcal{T} \Rightarrow \mathcal{S} \in \overline{\mathcal{SS}}^{\bullet})$ holds and $\mathcal{T} \in [\Sigma](\overline{\mathcal{SS}}^{\bullet})$, or neither holds. But in this last case, because $\mathcal{T}$ is simulable, there must be an $\mathcal{S}_1 \in \mathcal{SS}$, and an $\mathcal{S}_2 \in \overline{\mathcal{SS}}^{\bullet}$, such that $\mathcal{S}_1\ \Sigma\ \mathcal{T}$ and $\mathcal{S}_2\ \Sigma\ \mathcal{T}$ both hold, and the decomposition of $\mathcal{CMF}_{\mathrm{C}}$ is clear.

Of course all the $[\Sigma]\mathcal{SS}$, $\langle\Sigma\rangle(\overline{\mathcal{SS}}^{\bullet})$, $\langle\Sigma\rangle\mathcal{SS}$, $[\Sigma](\overline{\mathcal{SS}}^{\bullet})$, figuring in the above are core or rectified, as appropriate, by Proposition 6.14.

To establish (3) we calculate as follows, where $\mathrm{com}_{\mathrm{A}}$ and $\mathrm{com}_{\mathrm{C}}$ are complements in $\mathcal{CMF}_{\mathrm{A}}$ and $\mathcal{CMF}_{\mathrm{C}}$ respectively.

$[\Sigma]\langle\Sigma\rangle(\overline{\mathcal{SS}}^{\bullet}) = [\Sigma]\{\mathcal{T} \mid (\exists\, \mathcal{S} \bullet \mathcal{S} \in \overline{\mathcal{SS}}^{\bullet} \wedge \mathcal{S}\ \Sigma\ \mathcal{T})\}$
$=$ (definitions)
$[\Sigma]\{\mathcal{T} \mid (\exists\, \mathcal{S}, \mathcal{T}' \bullet \mathcal{S} \notin \mathcal{SS} \wedge \mathcal{S}\ \Sigma\ \mathcal{T}' \wedge \mathcal{S}\ \Sigma\ \mathcal{T})\}$
$=$ (PC, set theory)
$[\Sigma]\{\mathcal{T} \mid (\exists\, \mathcal{S} \bullet \mathcal{S} \notin \mathcal{SS} \wedge \mathcal{S}\ \Sigma\ \mathcal{T})\}$
$=$ (PC, set theory)
$[\Sigma]\mathrm{com}_{\mathrm{C}}(\{\mathcal{T} \mid (\forall\, \mathcal{S} \bullet \mathcal{S}\ \Sigma\ \mathcal{T} \Rightarrow \mathcal{S} \in \mathcal{SS})\})$
$=$ (PC, definitions of $[\Sigma]\mathcal{SS}$ and $\mathcal{CMF}_{\mathrm{C}}^{\mathcal{NS}}$)
$[\Sigma](\{\mathcal{T} \mid \mathcal{T} \notin [\Sigma]\mathcal{SS}\} \cup \mathcal{CMF}_{\mathrm{C}}^{\mathcal{NS}})$
$=$ $(\mathcal{T} \in \mathcal{CMF}_{\mathrm{C}}^{\mathcal{NS}} \Rightarrow (\forall\, \mathcal{S} \bullet \neg(\mathcal{S}\ \Sigma\ \mathcal{T})))$
$[\Sigma]\{\mathcal{T} \mid \mathcal{T} \notin [\Sigma]\mathcal{SS}\}$
$=$ (definition of $[\Sigma]$, PC, set theory)
$\{\mathcal{S} \mid \neg((\forall\, \mathcal{T}' \bullet \mathcal{T}' \in [\Sigma]\mathcal{SS} \vee \neg(\mathcal{S}\ \Sigma\ \mathcal{T}')) \vee (\exists\, \mathcal{T}' \bullet \mathcal{S}\ \Sigma\ \mathcal{T}' \wedge \mathcal{T}' \in [\Sigma]\mathcal{SS}))\}$
$=$ (definition of $\langle\Sigma\rangle$, PC, set theory)
$\mathrm{com}_{\mathrm{A}}(\{\mathcal{S} \mid (\forall\, \mathcal{T}' \bullet \mathcal{S}\ \Sigma\ \mathcal{T}' \Rightarrow \mathcal{T}' \in [\Sigma]\mathcal{SS})\} \cup \langle\Sigma\rangle[\Sigma]\mathcal{SS}\ )$
$=$ (definition of $[\Sigma][\Sigma]\mathcal{SS}$ and $\mathcal{CMF}_{\mathrm{A}}^{\mathcal{NS}}$)
$\mathrm{com}_{\mathrm{A}}([\Sigma][\Sigma]\mathcal{SS} \cup \mathcal{CMF}_{\mathrm{A}}^{\mathcal{NS}} \cup \langle\Sigma\rangle[\Sigma]\mathcal{SS})$
$=$ (Proposition 6.10)
$\mathrm{com}_{\mathrm{A}}(\langle\Sigma\rangle[\Sigma]\mathcal{SS}\ \cup \mathcal{CMF}_{\mathrm{A}}^{\mathcal{NS}})$
$=$ (PC, set theory)
$\mathrm{com}_{\mathrm{A}}(\langle\Sigma\rangle[\Sigma]\mathcal{SS}) \cap \mathcal{CMF}_{\mathrm{A}}^{\mathcal{S}} = \mathrm{com}_{\mathrm{A}}(\langle\Sigma\rangle[\Sigma]\mathcal{SS})^{\bullet}$
$=$ ($\mathcal{SS}$ is core)
$\overline{\mathcal{SS}}^{\bullet}$ (6.5)

To establish (4) we must show that if $\mathcal{SS}$ is rectified, then $\langle\Sigma\rangle[\Sigma](\overline{\mathcal{SS}}^{\bullet}) = \overline{\mathcal{SS}}^{\bullet}$.

$\langle\Sigma\rangle[\Sigma](\overline{\mathcal{SS}}^{\bullet}) = \langle\Sigma\rangle\{\mathcal{T} \mid (\exists\, \mathcal{S} \bullet \mathcal{S} \in \overline{\mathcal{SS}}^{\bullet} \wedge \mathcal{S}\ \Sigma\ \mathcal{T}) \wedge (\forall\, \mathcal{S} \bullet \mathcal{S}\ \Sigma\ \mathcal{T} \Rightarrow \mathcal{S} \in \overline{\mathcal{SS}}^{\bullet})\}$
$=$ (definitions)
$\langle\Sigma\rangle\{\mathcal{T} \mid (\exists\, \mathcal{S}, \mathcal{T}' \bullet \mathcal{S} \notin \mathcal{SS} \wedge \mathcal{S}\ \Sigma\ \mathcal{T}' \wedge \mathcal{S}\ \Sigma\ \mathcal{T}) \wedge$
$\qquad (\forall\, \mathcal{S} \bullet \mathcal{S}\ \Sigma\ \mathcal{T} \Rightarrow (\exists\, \mathcal{T}' \bullet \mathcal{S} \notin \mathcal{SS} \wedge \mathcal{S}\ \Sigma\ \mathcal{T}'))\}$
$=$ (PC, set theory)
$\langle\Sigma\rangle\{\mathcal{T} \mid (\exists\, \mathcal{S} \bullet \mathcal{S} \notin \mathcal{SS} \wedge \mathcal{S}\ \Sigma\ \mathcal{T}) \wedge (\forall\, \mathcal{S} \bullet \mathcal{S}\ \Sigma\ \mathcal{T} \Rightarrow \mathcal{S} \notin \mathcal{SS})\}$

$= $ (PC, set theory)
$\langle\Sigma\rangle \mathrm{com}_C(\{\mathcal{T} \mid (\forall\, \mathcal{S} \bullet \mathcal{S}\ \Sigma\ \mathcal{T} \Rightarrow \mathcal{S} \in \mathcal{SS}) \vee (\exists\, \mathcal{S} \bullet \mathcal{S}\ \Sigma\ \mathcal{T} \wedge \mathcal{S} \in \mathcal{SS})\})$
$= $ (PC, definitions of $[\Sigma]\mathcal{SS}$ and $\mathcal{CMF}\ {}^{\mathcal{NS}}_{\mathrm{C}}$)
$\langle\Sigma\rangle \mathrm{com}_C([\Sigma]\mathcal{SS} \cup \mathcal{CMF}\ {}^{\mathcal{NS}}_{\mathrm{C}} \cup \langle\Sigma\rangle\mathcal{SS})$
$= $ ($\mathcal{T} \in \mathcal{CMF}\ {}^{\mathcal{NS}}_{\mathrm{C}} \Rightarrow (\forall\, \mathcal{S} \bullet \neg(\mathcal{S}\ \Sigma\ \mathcal{T}))$, Proposition 6.10)
$\langle\Sigma\rangle\{\mathcal{T} \mid \mathcal{T} \notin \langle\Sigma\rangle\mathcal{SS}\}$
$= $ (definition of $\langle\Sigma\rangle$, PC, set theory)
$\{\mathcal{S} \mid \neg(\forall\, \mathcal{T}' \bullet \mathcal{S}\ \Sigma\ \mathcal{T}' \Rightarrow \mathcal{T}' \in \langle\Sigma\rangle\mathcal{SS})\}$
$= $ (definition of $[\Sigma]$, PC, set theory)
$\mathrm{com}_A([\Sigma]\langle\Sigma\rangle\mathcal{SS} \cup \mathcal{CMF}\ {}^{\mathcal{NS}}_{\mathrm{A}})$
$= $ (PC, set theory)
$\mathrm{com}_A([\Sigma]\langle\Sigma\rangle\mathcal{SS}) \cap \mathcal{CMF}\ {}^{\mathcal{S}}_{\mathrm{A}} = \mathrm{com}_A([\Sigma]\langle\Sigma\rangle\mathcal{SS})^{\bullet}$
$= $ ($\mathcal{SS}$ is rectified)
$\overline{\mathcal{SS}}^{\bullet}$ (6.6)

Point (5) is now obvious. We are done. ☺

# 7 Ambivalent Properties and their Transformers

One weakness of the simulation transformers studied in the previous section is that the simulation relationship $\mathcal{S}\ \Sigma\ \mathcal{T}$ completely fails to restrict the lead-ins and lead-outs and other non-simulable segments of both $\mathcal{S}$ and $\mathcal{T}$. As a consequence, any property defined by a set comprehension of the form $\{\mathcal{S} \mid \ldots\ \mathcal{S}\ \Sigma\ \mathcal{T}\}$, where $\mathcal{T}$ ranges over some understood property, can take in all possible ways of embellishing the simulating segments of a given $\mathcal{S}$ with such non-simulable portions, almost without reservation. Fortunately there are other mappings we can consider that can alleviate this to a degree.

We observe that discussions of systems (and of system property transformations in particular) invariably take place in an appropriate meta level context, and although we have been working exclusively at the semantic level, any system property that we discuss must have some syntactic description at the meta level in order to be discussed at all. It is the structure of this meta level syntactic description in its meta level context that we aim to exploit. Typical candidate meta level contexts include:

[S] The class of all systems. (7.1)

[R] A number of systems together with certain retrenchments between some pairs of them. (7.2)

[$R_1$] A specific retrenchment between two specific systems. (7.3)

Essentially, an ambivalent property (of a system with respect to a context) is a property that can be expressed in the meta language by a predicate which is well defined (and each of whose subexpressions is well defined) for every choice of system in the context. Thus the predicate that defines the property must refer only to attributes of the system that are possessed by all systems belonging to the context. We call such predicates ambivalent predicates. To make this clearer we give a more precise definition for the context [S], but quickly move on to discuss other possibilities more informally.

**Definition 7.1** An ambivalent predicate for the context [S] is a predicate $\phi$ that for a candidate multifragment $\mathcal{U}$ of a candidate system $U$ of [S], refers only to its structure as a sequence of sequences, and the property $\mathcal{U}\mathcal{U} = \mathsf{PR}_U(\{\mathcal{U} \mid \mathcal{U} \in \mathcal{CMF}_U \wedge \phi\})$ that it defines is called an ambivalent property.

Thus consider a multifragment $\mathcal{U}$ of $U$. It is a sequence of sequences $\mathcal{U}_k$, each element $\mathcal{U}_k[l]$ of which is a step $u$ -$(i, Op_U, o)$-› $u'$ (and these steps connect together in the usual way). Consequently, any set of multifragments that is unrestricted regarding the data items $u$, $i$, $o$, $u'$, for the various steps, can be described by a set comprehension of the form $\{\mathcal{U} \mid \mathcal{U} \in \mathcal{CMF}_U \wedge \phi\}$ where $\phi$ does not mention st($e$), in($e$), Op($e$), ou($e$), st′($e$), for any $e$ that can refer to any step $\mathcal{U}_k[l]$. Such a predicate $\phi$ is thus equally applicable to any other system $W$ in [S] and defines, via $\mathsf{PR}_W$, a corresponding property of $W$, namely $\mathsf{PR}_W\{\mathcal{W} \mid \mathcal{W} \in \mathcal{CMF}_W \wedge \phi\}$. Ambivalent properties (for the context [S]) are properties that can be described in this way.

To put it another way, ambivalent predicates and properties refer only to aspects that can be considered to be part of the common structure of the context of discourse.

**Example 7.2** The following phrases, when formalised to the extent needed, yield ambivalent predicates for the context [S].

$\mathcal{S}$ contains one fragment and its length is 5. (7.4)

$\mathrm{md}(\mathcal{S}) = 0 \wedge \mathrm{md}(\mathcal{S}_0) = 4$ (7.5)

No fragment in $\mathcal{S}$ has length greater than 15. (7.6)

$k \in \mathrm{dom}(\mathcal{S}) \Rightarrow \mathrm{md}(\mathcal{S}_k) \leq 14$ (7.7)

Above, (7.4) and (7.5) are intended to be equivalent as are (7.6) and (7.7). In both cases the latter is a more formal statement of the former. Note that nothing other than the structure of $\mathcal{S}$ as a sequence of sequences is refered to.

**Example 7.3** The following phrases, yield ambivalent predicates for other contexts.

$\mathcal{S}$ contains one fragment and its length is 5; the second and third operation names are both *Inc*. (Assuming $Inc \in (\mathsf{Ops}_A \cap \mathsf{Ops}_C)$.) (7.8)

$\mathrm{md}(\mathcal{S}) = 0 \wedge \mathrm{md}(\mathcal{S}_0) = 4 \wedge \mathrm{Op}(\mathcal{S}_0[1]) = Inc \wedge \mathrm{Op}(\mathcal{S}_0[2]) \in \{Inc\}$ (7.9)

$\mathcal{S}$ contains one fragment; the second operation names is not *Inc*. (Assuming $Inc \in (\mathsf{Ops}_A \cap \mathsf{Ops}_C)$.) (7.10)

$\mathrm{md}(\mathcal{S}) = 0 \wedge \mathrm{Op}(\mathcal{S}_0[1]) \neq Inc$ (7.11)

The first fragment in $\mathcal{S}$ starts in an initial state. (7.12)

Note that (7.8)-(7.11) depend implicitly on there being only one retrenchment in play in the current discourse, otherwise the term $(\mathsf{Ops}_A \cap \mathsf{Ops}_C)$ becomes ambiguous, so the predicates defined are suitable for example for the context $[R_1]$.

The last item (7.12), initiates the relaxation of the strict tenets of the contexts we have defined earlier, in that it refers to initial states. Certainly this is not permitted according to Definition 7.1, but observing that all the systems we consider do indeed have initial states, we are entitled to regard the possession of an initial state as a structural element of our class of systems, and accordingly, mention of initial states ought to be permitted inside ambivalent predicates since it is well defined for all systems in the

context. We do not formalise this more precisely here, and instead we spend some time indicating other viable relaxations of the rather narrow notions of ambivalence introduced above.

**Example 7.4** The following phrases provide further cases of notions that are not strictly permitted according to earlier remarks, but that could potentially be seen as structural, and thus as yielding ambivalent predicates for suitable contexts.

> The last fragment in $\mathcal{S}$ ends in a final state. (7.13)
>
> Each fragment in $\mathcal{S}$ passes through at least one bottleneck state. (7.14)
>
> The second fragment in $\mathcal{S}$ ends in a state outside the field of the retrieve relation. (7.15)
>
> The second fragment in $\mathcal{S}$ has three simulable steps. (7.16)
>
> The one and only fragment in $\mathcal{S}$ has three consecutive simulable steps whose operation names are respectively *Inc*, *Inc*, *Dec*, and all the other steps are non-simulable. (7.17)

Phrase (7.13) uses the notion of final state, one which is not present in the system concept of Section 2, and so phrase (7.13) is not only undefined in that sense, but also according to eg. Definition 7.1. But many useful classes of system do have a notion of final state, and the significance and utility of retrenchment do not depend on whether this aspect of a system is present or not, so finality of states ought to be admitted in ambivalent predicates if the context is one where finality is a common structural property of all systems in the context.

Phrase (7.14) goes further. What is a bottleneck state? There is no predetermined answer to the question, but there can arise situations in which we are dealing with a class of systems that each have one or more states that merit some special consideration of this kind. Regarding this class as a context, it then becomes reasonable to elevate those states to a structural feature, and to allow them to contribute to ambivalent predicates.

The more liberal examples so far can be characterised by saying that all occurences in the ambivalent predicate $\phi$ of $\mathrm{st}(e)$, $\mathrm{in}(e)$, $\mathrm{ou}(e)$, $\mathrm{st}'(e)$, for any $e$ that can refer to any step $\mathcal{S}_k[l]$, are constrained to evaluate to elements inside (or outside) of given constant sets whose names have a fixed interpretation in each system in the context.

However (7.15) is more contentious than that. As in (7.8)-(7.11), it assumes there is only one retrenchment in the current discourse, otherwise it is ambiguous as to which retrieve relation is being refered to. Beyond this lies a typing issue regarding the domain and range of a heterogeneous relation: we need to define the field of a heterogeneous relation as the disjoint union of its domain and range before (7.15) is well typed.

Phrase (7.16) is similar to (7.15) though more complicated, requiring reference to all the ingredients of a retrenchment, while (7.17) goes even further. (We will return to (7.17) shortly.)

Reviewing the preceding examples confirms that what can be regarded as acceptable in an ambivalent predicate is heavily dependent on the meta context. And it suggests that in a more rigorous formalisation, the aspects we seek to capture can be expressed

by introducing additional constants into the object language, each with a fixed (though not necessarily first order) interpretation for each system in the context. Given the focus on the semantic domain in this paper, we do not pursue such formalisations here, though there is obviously much scope for detailed exploration of the ambivalent predicate idea in various scenarios.

**Definition 7.5** Let $\mathcal{AA} = \mathsf{PR}_{\mathrm{A}}(\{\mathcal{S} \mid \mathcal{S} \in \mathcal{CMF}_{\mathrm{A}} \wedge \phi\})$ where $\phi$ is an ambivalent predicate, be an ambivalent property of the abstract system of a retrenchment. Let $\mathcal{BB} = \mathsf{PR}_{\mathrm{C}}(\{\mathcal{T} \mid \mathcal{T} \in \mathcal{CMF}_{\mathrm{C}} \wedge \phi\})$ be the corresponding concrete ambivalent property. The ambivalent transformer operator (A) is defined as follows.

$$(\mathrm{A})\mathcal{AA} = \mathcal{BB} \tag{7.18}$$

$$(\mathrm{A})\mathcal{BB} = \mathcal{AA} \tag{7.19}$$

For ambivalent properties there are now two ways of mapping them between levels of abstraction: firstly using $[\Sigma]$ and $\langle\Sigma\rangle$, and secondly using (A). Consider the abstract property $\mathcal{SS}$ defined by (7.17) again, which is a good example for comparing these mappings. We know by Proposition 6.10 that $[\Sigma]\mathcal{SS} \subseteq \langle\Sigma\rangle\mathcal{SS}$, and that both consist only of concrete multifragments (of length at most 3, by Definition 4.6.(1)), that are each in simulation with at least one element of $\mathcal{SS}$. The individual fragments may contain arbitrary lead-ins, lead-outs, and other non-simulable segments. On the other hand $(\mathrm{A})\mathcal{SS}$ consists of concrete multifragments of length 1, each of which contains a segment of three steps with operation names *Inc*, *Inc*, *Dec*, such that each of these three steps is individually simulable. But there is no requirement that the simulations of the concrete *Inc*, *Inc*, *Dec*, steps fit together to make a segment of length 3 in a single fragment. Thus if the retrieve and other relations of the retrenchment are sufficiently perverse, there may be elements of $\mathcal{SS}$ not in simulation with any element of $(\mathrm{A})\mathcal{SS}$, and elements of $(\mathrm{A})\mathcal{SS}$ not in simulation with any element of $\mathcal{SS}$. This provokes the thought of elevating *consecutive simulability* to a structural property and examining the ambivalent properties generated. If (7.17) were strengthened in this way, then we would find that $(\mathrm{A})\mathcal{SS} \subseteq \langle\Sigma\rangle\mathcal{SS}$. These remarks suffice to illustrate that there is no universally valid relationship between the $[\Sigma]$ and $\langle\Sigma\rangle$ transformers on the one hand, and the (A) transformers on the other.

The (A) transformer is obviously of limited use since it only applies to ambivalent properties. But since ambivalent predicates are capable of constraining the lead-ins, lead-outs, and other non-simulable segments of fragments, the interaction between ambivalent properties mapped between levels of abstraction by (A) and arbitrary properties mapped via $[\Sigma]$ and $\langle\Sigma\rangle$ is obviously of great interest.

**Proposition 7.6** Let a retrenchment be understood, and $\mathcal{AA}$ and $\mathcal{BB}$ be abstract and concrete ambivalent properties related via (7.18) and (7.19). Let $\mathcal{SS}$ be an arbitrary abstract property. Then:

(1) $\langle\Sigma\rangle(\mathcal{SS} \cap \mathcal{AA}) \cap \mathcal{BB} \subseteq \langle\Sigma\rangle(\mathcal{SS} \cap \mathcal{AA}) \subseteq \langle\Sigma\rangle\mathcal{SS}$

(2) $\langle\Sigma\rangle(\mathcal{SS} \cap \mathcal{AA}) \cap \mathcal{BB} \subseteq \langle\Sigma\rangle\mathcal{SS} \cap \mathcal{BB} \subseteq \langle\Sigma\rangle\mathcal{SS}$

(3) $[\Sigma](\mathcal{SS} \cap \mathcal{AA}) \cap \mathcal{BB} \subseteq [\Sigma](\mathcal{SS} \cap \mathcal{AA}) \subseteq [\Sigma]\mathcal{SS}$

(4) $[\Sigma](\mathcal{SS} \cap \mathcal{AA}) \cap \mathcal{BB} \subseteq [\Sigma]\mathcal{SS} \cap \mathcal{BB} \subseteq [\Sigma]\mathcal{SS}$

*Proof.* The inclusions (1)-(4) follow by monotonicity of $\langle\Sigma\rangle$ and $[\Sigma]$. ☺

That we cannot uncritically relate the middle terms in (1) and (2) or (3) and (4) in Proposition 7.6 for arbitrary ambivalent properties is vividly shown as follows.

**Counterexample 7.7** Let $\mathcal{AA}$ be all multifragments that begin in an abstract initial state, and $\mathcal{BB}$ be all multifragments that begin in a concrete initial state. Then the standard initialisation PO (2.1), cannot prove that all elements of $[\Sigma](\mathcal{SS} \cap \mathcal{AA})$ begin in a concrete initial state, which is true of $[\Sigma]\mathcal{SS} \cap \mathcal{BB}$. The $\langle\Sigma\rangle$ case is similar. And the corresponding facts for the concrete counterpart of Proposition 7.6, using an arbitrary concrete property $\mathcal{TT}$ instead of $\mathcal{SS}$, also fail, as (2.1) only says that for each concrete intial state there is *some* abstract initial state, which is eg. too weak to prove that all elements of $[\Sigma](\mathcal{TT} \cap \mathcal{BB})$ start in abstract initial states. Of course specific systems may enjoy stronger properties regarding initialisation, that enable the equality of $\langle\Sigma\rangle(\mathcal{SS} \cap \mathcal{AA})$ and $\langle\Sigma\rangle\mathcal{SS} \cap \mathcal{BB}$ or of $[\Sigma](\mathcal{SS} \cap \mathcal{AA})$ and $[\Sigma]\mathcal{SS} \cap \mathcal{BB}$ to be proved, but that is another matter.

# 8 Constrained Property Transformers

Proposition 7.6 and Counterexample 7.7 showed that the interaction between ambivalent properties and arbitrary properties was not particularly clean. While this might be viewed initially as a disappointment, on reflection it is not, as it generates greater expressivity in transforming properties between levels of abstraction; and given the somewhat unruly nature of the $[\Sigma]$ and $\langle\Sigma\rangle$ transformers as regards non-simulable segments within fragments, this can only be welcomed by developers when utilising retrenchment in practice.

We now consider in more detail property transformations that take an abstract property $\mathcal{SS}$ to $\langle\Sigma\rangle\mathcal{SS} \cap \mathcal{QQ}$ or to $[\Sigma]\mathcal{SS} \cap \mathcal{QQ}$. We focus on imposing a constraint on the result of a simulation transformer $\langle\Sigma\rangle$ or $[\Sigma]$, as we assume that any constraint needed for $\mathcal{SS}$ itself, would naturally be part of the definition of $\mathcal{SS}$ already. But now there is no reason to insist that $\mathcal{QQ}$ is a purely ambivalent property. There may be every reason to be interested in the restriction of $\langle\Sigma\rangle\mathcal{SS}$ or $[\Sigma]\mathcal{SS}$ to properties which are quite specific to the concrete system. Thus constrained property transformers will translate an abstract property $\mathcal{SS}$ to a concrete property of the form $\langle\Sigma\rangle\mathcal{SS} \cap \mathcal{BB} \cap \mathcal{YY}$, or to $[\Sigma]\mathcal{SS} \cap \mathcal{BB} \cap \mathcal{YY}$, where $\mathcal{BB}$ is an ambivalent property and $\mathcal{YY}$ is a property specific to the concrete system. We will continue to write these as $\langle\Sigma\rangle\mathcal{SS} \cap \mathcal{QQ}$ or $[\Sigma]\mathcal{SS} \cap \mathcal{QQ}$, with $\mathcal{QQ} = \mathcal{BB} \cap \mathcal{YY}$, since no benefit is gained at the mathematical level by maintaining the distinction between ambivalent and specific constraints, as Proposition 7.6 and Counterexample 7.7 indicated.

**Remark 8.1** Even if there is no mathematical benefit in separating $\mathcal{BB}$ and $\mathcal{YY}$, there may be a benefit at the software methodology level in maintaining the distinction. These different kinds of constraint may arise from different requirements, and thus developers may wish to manipulate them differently for that reason. However our concern now is with mathematics, so we merely note the point for future reference, and continue to use the abbreviated form.

**Definition 8.2** Let $\mathcal{SS}$ and $\mathcal{PP}$ be abstract properties and $\mathcal{TT}$ and $\mathcal{QQ}$ be concrete ones. The property transformers $[\Sigma_{\mathcal{QQ}}]$, $\langle\Sigma_{\mathcal{QQ}}\rangle$, $[\Sigma_{\mathcal{PP}}]$, $\langle\Sigma_{\mathcal{PP}}\rangle$, called constrained property tranformers (CPTs), are defined as follows.

$$[\Sigma_{QQ}]\mathcal{SS} \;=\; [\Sigma]\mathcal{SS} \cap QQ \tag{8.1}$$

$$\langle\Sigma_{QQ}\rangle\mathcal{SS} \;=\; \langle\Sigma\rangle\mathcal{SS} \cap QQ \tag{8.2}$$

$$[\Sigma_{\mathcal{PP}}]\mathcal{TT} \;=\; [\Sigma]\mathcal{TT} \cap \mathcal{PP} \tag{8.3}$$

$$\langle\Sigma_{\mathcal{PP}}\rangle\mathcal{TT} \;=\; \langle\Sigma\rangle\mathcal{TT} \cap \mathcal{PP} \tag{8.4}$$

We proceed to study $[\Sigma_{QQ}]$, $\langle\Sigma_{QQ}\rangle$, $[\Sigma_{\mathcal{PP}}]$, $\langle\Sigma_{\mathcal{PP}}\rangle$, along familiar lines. The following are immediate from the definitions.

**Proposition 8.3** The following hold for the CPTs $[\Sigma_{QQ}]$, $\langle\Sigma_{QQ}\rangle$.

(1) $\mathcal{SS}_1 \subseteq \mathcal{SS}_2 \;\Rightarrow\; \langle\Sigma_{QQ}\rangle\mathcal{SS}_1 \subseteq \langle\Sigma_{QQ}\rangle\mathcal{SS}_2 \wedge [\Sigma_{QQ}]\mathcal{SS}_1 \subseteq [\Sigma_{QQ}]\mathcal{SS}_2$

(2) $[\Sigma_{QQ}]\mathcal{SS} \;\subseteq\; \langle\Sigma_{QQ}\rangle\mathcal{SS}$

**Proposition 8.4** Let $\mathcal{SS}$ be an abstract property and $QQ$ a concrete one. Then:

(1) $[\Sigma_{QQ}]\mathcal{SS} \;\subseteq\; [\Sigma]\mathcal{SS}$

(2) $\langle\Sigma_{QQ}\rangle\mathcal{SS} \;\subseteq\; \langle\Sigma\rangle\mathcal{SS}$

**Proposition 8.5** Let $\mathcal{SS}$ and $\mathcal{PP}$ be abstract properties and $QQ$ be a concrete one.

(1) $\ldots (\langle\Sigma_{\mathcal{PP}}\rangle[\Sigma_{QQ}])^3\mathcal{SS} \;\subseteq\; (\langle\Sigma_{\mathcal{PP}}\rangle[\Sigma_{QQ}])^2\mathcal{SS} \;\subseteq\; \langle\Sigma_{\mathcal{PP}}\rangle[\Sigma_{QQ}]\mathcal{SS} \;\subseteq\; \mathcal{SS}^{\bullet} \;\subseteq\; \mathcal{SS}$

(2) $[\Sigma_{QQ}]\langle\Sigma_{\mathcal{PP}}\rangle[\Sigma_{QQ}]\langle\Sigma_{\mathcal{PP}}\rangle[\Sigma_{QQ}]\mathcal{SS} \;=\; [\Sigma_{QQ}]\langle\Sigma_{\mathcal{PP}}\rangle[\Sigma_{QQ}]\mathcal{SS}$

(3) $[\Sigma_{\mathcal{PP}}]\langle\Sigma_{QQ}\rangle[\Sigma_{\mathcal{PP}}]\langle\Sigma_{QQ}\rangle\mathcal{SS} \;=\; [\Sigma_{\mathcal{PP}}]\langle\Sigma_{QQ}\rangle\mathcal{SS}$

(4) $w[\Sigma_{QQ}]\langle\Sigma_{\mathcal{PP}}\rangle[\Sigma_{QQ}]\langle\Sigma_{\mathcal{PP}}\rangle[\Sigma_{QQ}]\mathcal{SS} \;=\; w[\Sigma_{QQ}]\langle\Sigma_{\mathcal{PP}}\rangle[\Sigma_{QQ}]\mathcal{SS}$, where $w$ is any well formed[6] word over $\{\langle\Sigma_{\mathcal{PP}}\rangle, [\Sigma_{QQ}], \langle\Sigma_{QQ}\rangle, [\Sigma_{\mathcal{PP}}]\}$

(5) $w[\Sigma_{\mathcal{PP}}]\langle\Sigma_{QQ}\rangle[\Sigma_{\mathcal{PP}}]\langle\Sigma_{QQ}\rangle\mathcal{SS} \;=\; w[\Sigma_{\mathcal{PP}}]\langle\Sigma_{QQ}\rangle\mathcal{SS}$, where $w$ is any well formed word over $\{\langle\Sigma_{\mathcal{PP}}\rangle, [\Sigma_{QQ}], \langle\Sigma_{QQ}\rangle, [\Sigma_{\mathcal{PP}}]\}$

*Proof.* For (1), the conclusion follows by Proposition 8.3 and Proposition 8.4. Note that there is no analogue of Proposition 6.12.(2) because of the conflict between Proposition 8.4.(2) and the desired ascending chain. For (2), we use $[\Sigma_{QQ}]\mathcal{SS}$ in place of $\mathcal{TT}$ in the concrete counterpart of (3).

For (3), let $\mathcal{S} \in [\Sigma_{\mathcal{PP}}]\langle\Sigma_{QQ}\rangle[\Sigma_{\mathcal{PP}}]\langle\Sigma_{QQ}\rangle\mathcal{SS}$. Then $\mathcal{S} \in \mathcal{PP}$, and $(\exists\, \mathcal{T} \bullet \mathcal{S}\ \Sigma\ \mathcal{T} \wedge \mathcal{T} \in \langle\Sigma_{QQ}\rangle[\Sigma_{\mathcal{PP}}]\langle\Sigma_{QQ}\rangle\mathcal{SS}) = (*_1)$, and $(\forall\, \mathcal{T} \bullet \mathcal{S}\ \Sigma\ \mathcal{T} \Rightarrow \mathcal{T} \in \langle\Sigma_{QQ}\rangle[\Sigma_{\mathcal{PP}}]\langle\Sigma_{QQ}\rangle\mathcal{SS}) = (*_2)$, all hold.

Let $\mathcal{T}$ witness $(*_1)$. Then $\mathcal{T} \in QQ$, and there is an $\mathcal{S}'$ such that $\mathcal{S}'\ \Sigma\ \mathcal{T} \wedge \mathcal{S}' \in [\Sigma_{\mathcal{PP}}]\langle\Sigma_{QQ}\rangle\mathcal{SS} = (*_3)$ holds. From $(*_3)$ we deduce $(\forall\, \mathcal{T}' \bullet \mathcal{S}'\ \Sigma\ \mathcal{T}' \Rightarrow \mathcal{T}' \in \langle\Sigma_{QQ}\rangle\mathcal{SS}) = (*_4)$ among other things. And since $(*_4)$ applies to $\mathcal{T}' = \mathcal{T}$ because of $(*_3)$, we deduce $\mathcal{T} \in \langle\Sigma_{QQ}\rangle\mathcal{SS}$; so we have $(\exists\, \mathcal{T} \bullet \mathcal{S}\ \Sigma\ \mathcal{T} \wedge \mathcal{T} \in QQ \wedge \mathcal{T} \in \langle\Sigma_{QQ}\rangle\mathcal{SS}) = (*_5)$.

From $(*_2)$ we know that if $\mathcal{S}\ \Sigma\ \mathcal{T}$ holds, then $\mathcal{T} \in QQ$, and there is an $\mathcal{S}'$ such that $(*_3)$ holds. By the reasoning just used we deduce $\mathcal{T} \in \langle\Sigma_{QQ}\rangle\mathcal{SS}$ again; so we have $(\forall\, \mathcal{T} \bullet \mathcal{S}\ \Sigma\ \mathcal{T} \Rightarrow \mathcal{T} \in QQ \wedge \mathcal{T} \in \langle\Sigma_{QQ}\rangle\mathcal{SS}) = (*_6)$. Now $\mathcal{S} \in \mathcal{PP} \wedge (*_5) \wedge (*_6)$ means that $\mathcal{S} \in [\Sigma_{\mathcal{PP}}]\langle\Sigma_{QQ}\rangle\mathcal{SS}$, so we have $[\Sigma_{\mathcal{PP}}]\langle\Sigma_{QQ}\rangle[\Sigma_{\mathcal{PP}}]\langle\Sigma_{QQ}\rangle\mathcal{SS} \subseteq [\Sigma_{\mathcal{PP}}]\langle\Sigma_{QQ}\rangle\mathcal{SS}$.

6. Well formed means that the $\mathcal{PP}$ and $QQ$ subscripts alternate appropriately.

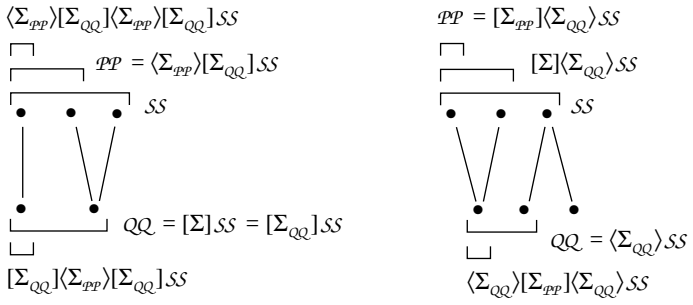



Fig. 5

To prove that $[\Sigma_{\mathcal{PP}}]\langle\Sigma_{\mathcal{QQ}}\rangle\mathcal{SS} \subseteq [\Sigma_{\mathcal{PP}}]\langle\Sigma_{\mathcal{QQ}}\rangle[\Sigma_{\mathcal{PP}}]\langle\Sigma_{\mathcal{QQ}}\rangle\mathcal{SS}$, suppose for a contradiction that $\mathcal{S} \in [\Sigma_{\mathcal{PP}}]\langle\Sigma_{\mathcal{QQ}}\rangle\mathcal{SS}$ but that $\mathcal{S} \notin [\Sigma_{\mathcal{PP}}]\langle\Sigma_{\mathcal{QQ}}\rangle[\Sigma_{\mathcal{PP}}]\langle\Sigma_{\mathcal{QQ}}\rangle\mathcal{SS}$. Then either $\mathcal{S} \notin \mathcal{PP} = (*_1)$ holds, or there is no $\mathcal{T}$ such that $\mathcal{S}\ \Sigma\ \mathcal{T} \wedge \mathcal{T} \in \langle\Sigma_{\mathcal{QQ}}\rangle[\Sigma_{\mathcal{PP}}]\langle\Sigma_{\mathcal{QQ}}\rangle\mathcal{SS} = (*_2)$ holds, or there is some $\mathcal{T}$ such that $\mathcal{S}\ \Sigma\ \mathcal{T} \wedge \mathcal{T} \notin \langle\Sigma_{\mathcal{QQ}}\rangle[\Sigma_{\mathcal{PP}}]\langle\Sigma_{\mathcal{QQ}}\rangle\mathcal{SS} = (*_3)$ holds.

Now $(*_1)$ is contradicted by the $\mathcal{PP}$ constraint in $\mathcal{S} \in [\Sigma_{\mathcal{PP}}]\langle\Sigma_{\mathcal{QQ}}\rangle\mathcal{SS}$.

Suppose next that there is no $\mathcal{T}$ such that $(*_2)$ holds. So for all $\mathcal{T}$, either $\neg(\mathcal{S}\ \Sigma\ \mathcal{T})$ or $\mathcal{T} \notin \langle\Sigma_{\mathcal{QQ}}\rangle[\Sigma_{\mathcal{PP}}]\langle\Sigma_{\mathcal{QQ}}\rangle\mathcal{SS}$ holds. Now $\mathcal{S} \in [\Sigma_{\mathcal{PP}}]\langle\Sigma_{\mathcal{QQ}}\rangle\mathcal{SS}$ implies $(\exists\ \mathcal{T} \bullet \mathcal{S}\ \Sigma\ \mathcal{T} \wedge \mathcal{T} \in \langle\Sigma_{\mathcal{QQ}}\rangle\mathcal{SS})$, so pick a $\mathcal{T}$ that witnesses this. Then for this $\mathcal{T}$, $\neg(\mathcal{S}\ \Sigma\ \mathcal{T})$ is contradicted, so $\mathcal{T} \notin \langle\Sigma_{\mathcal{QQ}}\rangle[\Sigma_{\mathcal{PP}}]\langle\Sigma_{\mathcal{QQ}}\rangle\mathcal{SS}$ is forced. But then this witness also satisfies $\mathcal{T} \in \mathcal{QQ}$, so $\mathcal{S} \in [\Sigma_{\mathcal{PP}}]\langle\Sigma_{\mathcal{QQ}}\rangle\mathcal{SS}$ and $\mathcal{S}\ \Sigma\ \mathcal{T}$ and $\mathcal{T} \in \mathcal{QQ}$, give $\mathcal{T} \in \langle\Sigma_{\mathcal{QQ}}\rangle[\Sigma_{\mathcal{PP}}]\langle\Sigma_{\mathcal{QQ}}\rangle\mathcal{SS}$, which is the contradiction that rules out $(*_2)$.

Finally suppose that there is a $\mathcal{T}$ such that $(*_3)$ holds. Then $\mathcal{S} \in [\Sigma_{\mathcal{PP}}]\langle\Sigma_{\mathcal{QQ}}\rangle\mathcal{SS}$ and $\mathcal{S}\ \Sigma\ \mathcal{T}$ and $\mathcal{T} \notin \langle\Sigma_{\mathcal{QQ}}\rangle[\Sigma_{\mathcal{PP}}]\langle\Sigma_{\mathcal{QQ}}\rangle\mathcal{SS}$ (the latter being equivalent to $\neg(\mathcal{T} \in \mathcal{QQ} \wedge \mathcal{T} \in \langle\Sigma\rangle[\Sigma_{\mathcal{PP}}]\langle\Sigma_{\mathcal{QQ}}\rangle\mathcal{SS})$) are all assumed, and these imply $\mathcal{T} \notin \mathcal{QQ}$. But $\mathcal{S} \in [\Sigma_{\mathcal{PP}}]\langle\Sigma_{\mathcal{QQ}}\rangle\mathcal{SS}$ implies $(\forall\ \mathcal{T}' \bullet \mathcal{S}\ \Sigma\ \mathcal{T}' \Rightarrow \mathcal{T}' \in \langle\Sigma_{\mathcal{QQ}}\rangle\mathcal{SS})$ and this applies in particular to $\mathcal{T}' = \mathcal{T}$, which means that $\mathcal{T} \in \mathcal{QQ}$ holds, giving the last contradiction we need. Therefore $(*_3)$ is impossible and (3) is proved.

Now (4) and (5) quickly follow. We are done. ☺

**Counterexample 8.6** With the conventions of Counterexample 6.15, Fig. 5.(a) and Fig. 5.(b) show that clauses (2) and (3) of Proposition 8.5 are respectively optimal. Thus Fig. 5.(a) shows that $\langle\Sigma_{\mathcal{PP}}\rangle[\Sigma_{\mathcal{QQ}}]\langle\Sigma_{\mathcal{PP}}\rangle[\Sigma_{\mathcal{QQ}}]\mathcal{SS} \neq \langle\Sigma_{\mathcal{PP}}\rangle[\Sigma_{\mathcal{QQ}}]\mathcal{SS}$, while Fig. 5.(b) shows that $\langle\Sigma_{\mathcal{QQ}}\rangle[\Sigma_{\mathcal{PP}}]\langle\Sigma_{\mathcal{QQ}}\rangle\mathcal{SS} \neq \langle\Sigma_{\mathcal{QQ}}\rangle\mathcal{SS}$.

The defaults for the above when either $\mathcal{PP}$ or $\mathcal{QQ}$ reduces to the trivial property are of interest. The following have obvious extensions by suitable words *w* as in Proposition 8.5.

**Corollary 8.7** Let $\mathcal{PP}$ be the trivial abstract property so that $[\Sigma_{\mathcal{PP}}] = [\Sigma]$ and $\langle\Sigma_{\mathcal{PP}}\rangle = \langle\Sigma\rangle$. Then:

(1) $[\Sigma_{\mathcal{QQ}}]\langle\Sigma\rangle[\Sigma_{\mathcal{QQ}}]\mathcal{SS} = [\Sigma_{\mathcal{QQ}}]\mathcal{SS}$

(2) $[\Sigma]\langle\Sigma_{\mathcal{QQ}}\rangle[\Sigma]\langle\Sigma_{\mathcal{QQ}}\rangle\mathcal{SS} = [\Sigma]\langle\Sigma_{\mathcal{QQ}}\rangle\mathcal{SS}$

*Proof.* For (1) we know that $\langle\Sigma\rangle[\Sigma_{\mathcal{QQ}}]\mathcal{SS} \subseteq \mathcal{SS}$ by Proposition 8.5.(1), so we have $[\Sigma_{\mathcal{QQ}}]\langle\Sigma\rangle[\Sigma_{\mathcal{QQ}}]\mathcal{SS} \subseteq [\Sigma_{\mathcal{QQ}}]\mathcal{SS}$ by monotonicity. Therefore we must show $[\Sigma_{\mathcal{QQ}}]\mathcal{SS} \subseteq [\Sigma_{\mathcal{QQ}}]\langle\Sigma\rangle[\Sigma_{\mathcal{QQ}}]\mathcal{SS}$. Suppose $\mathcal{T} \in [\Sigma_{\mathcal{QQ}}]\mathcal{SS}$, i.e. $\{\mathcal{T}\} \subseteq [\Sigma_{\mathcal{QQ}}]\mathcal{SS}$. Then $\mathcal{T} \in \mathcal{QQ}$ and $(\exists\, \mathcal{S} \bullet \mathcal{S} \in \mathcal{SS} \wedge \mathcal{S}\ \Sigma\ \mathcal{T})$ and $(\forall\, \mathcal{S} \bullet \mathcal{S}\ \Sigma\ \mathcal{T} \Rightarrow \mathcal{S} \in \mathcal{SS})$ all hold. Thus the set $\mathcal{SS}_{\mathcal{T}} = \{\mathcal{S} \mid \mathcal{S} \in \mathcal{SS} \wedge \mathcal{S}\ \Sigma\ \mathcal{T}\}$ which is equal to $\langle\Sigma\rangle\{\mathcal{T}\}$ by (6.4), satisfies $\mathcal{SS}_{\mathcal{T}} \subseteq \langle\Sigma\rangle[\Sigma_{\mathcal{QQ}}]\mathcal{SS}$. Since $\{\mathcal{T}\} \subseteq [\Sigma_{\mathcal{QQ}}]\mathcal{SS}_{\mathcal{T}}$, we get $\{\mathcal{T}\} \subseteq [\Sigma_{\mathcal{QQ}}]\langle\Sigma\rangle[\Sigma_{\mathcal{QQ}}]\mathcal{SS}$, and so, because $\mathcal{T}$ was an arbitrary element of $[\Sigma_{\mathcal{QQ}}]\mathcal{SS}$, we deduce $[\Sigma_{\mathcal{QQ}}]\mathcal{SS} \subseteq [\Sigma_{\mathcal{QQ}}]\langle\Sigma\rangle[\Sigma_{\mathcal{QQ}}]\mathcal{SS}$.

Note that (2) is the same as Proposition 8.5.(3), which cannot be improved as Fig. 5.(b) shows. ☺

**Corollary 8.8** Let $\mathcal{QQ}$ be the trivial concrete property so that $[\Sigma_{\mathcal{QQ}}] = [\Sigma]$ and $\langle\Sigma_{\mathcal{QQ}}\rangle = \langle\Sigma\rangle$. Then:

(1) $[\Sigma]\langle\Sigma_{\mathcal{PP}}\rangle[\Sigma]\langle\Sigma_{\mathcal{PP}}\rangle[\Sigma]\mathcal{SS} = [\Sigma]\langle\Sigma_{\mathcal{PP}}\rangle[\Sigma]\mathcal{SS}$

(2) $[\Sigma_{\mathcal{PP}}]\langle\Sigma\rangle[\Sigma_{\mathcal{PP}}]\langle\Sigma\rangle\mathcal{SS} = [\Sigma_{\mathcal{PP}}]\langle\Sigma\rangle\mathcal{SS}$

*Proof.* We note that (1) is as in Proposition 8.5.(2), which cannot be improved as Fig. 5.(a) shows. Also (2) is as in Proposition 8.5.(3); to see that this cannot be improved it is enough to let $\mathcal{QQ}$ include also the last concrete element in Fig. 5.(b) and to recalculate $\langle\Sigma\rangle\mathcal{SS}$, $[\Sigma_{\mathcal{PP}}]\langle\Sigma\rangle\mathcal{SS}$, and $\langle\Sigma\rangle[\Sigma_{\mathcal{PP}}]\langle\Sigma\rangle\mathcal{SS}$ in succession. ☺

**Definition 8.9** Let $\mathcal{SS}$ and $\mathcal{PP}$ be abstract properties and $\mathcal{QQ}$ be a concrete one. Then $\mathrm{core}_{(\mathcal{PP},\mathcal{QQ})}(\mathcal{SS}) = \langle\Sigma_{\mathcal{PP}}\rangle[\Sigma_{\mathcal{QQ}}]\langle\Sigma_{\mathcal{PP}}\rangle[\Sigma_{\mathcal{QQ}}]\mathcal{SS} \subseteq \mathcal{SS}$, is called the $(\mathcal{PP},\mathcal{QQ})$-core of $\mathcal{SS}$. If $\mathrm{core}_{(\mathcal{PP},\mathcal{QQ})}(\mathcal{SS}) = \mathcal{SS}$ then $\mathcal{SS}$ is a $(\mathcal{PP},\mathcal{QQ})$-core property. Similarly $\mathrm{rect}_{(\mathcal{PP},\mathcal{QQ})}(\mathcal{SS}) = [\Sigma_{\mathcal{PP}}]\langle\Sigma_{\mathcal{QQ}}\rangle\mathcal{SS}$, gives the $(\mathcal{PP},\mathcal{QQ})$-rectification of $\mathcal{SS}$. If $\mathrm{rect}_{(\mathcal{PP},\mathcal{QQ})}(\mathcal{SS}) = \mathcal{SS}$ then $\mathcal{SS}$ is a $(\mathcal{PP},\mathcal{QQ})$-rectified property. A property is $(\mathcal{PP},\mathcal{QQ})$-robust iff it satisfies $\mathcal{SS}^{\bullet} = \mathcal{SS}$, and $[\Sigma_{\mathcal{QQ}}]\mathcal{SS} = \langle\Sigma_{\mathcal{QQ}}\rangle\mathcal{SS}$, and $\langle\Sigma_{\mathcal{PP}}\rangle[\Sigma_{\mathcal{QQ}}]\mathcal{SS} = [\Sigma_{\mathcal{PP}}]\langle\Sigma_{\mathcal{QQ}}\rangle\mathcal{SS}$.

In both the case of the $(\mathcal{PP},\mathcal{QQ})$-core and the $(\mathcal{PP},\mathcal{QQ})$-rectification of $\mathcal{SS}$, these derived properties of $\mathcal{SS}$ feature the smallest number of iterations of $\langle\Sigma_{\mathcal{PP}}\rangle[\Sigma_{\mathcal{QQ}}]$ and $[\Sigma_{\mathcal{PP}}]\langle\Sigma_{\mathcal{QQ}}\rangle$ respectively, such that a further application guarantees no change. Also, compared with the situation in Section 6, there is now greater scope for defining various weakenings of $(\mathcal{PP},\mathcal{QQ})$-robust properties, but we do not do so.

**Proposition 8.10** Let $\mathcal{SS}$ and $\mathcal{PP}$ be abstract properties and $\mathcal{QQ}$ be a concrete one. Then if $\mathcal{SS}$ is a $(\mathcal{PP},\mathcal{QQ})$-core property then $\langle\Sigma_{\mathcal{PP}}\rangle[\Sigma_{\mathcal{QQ}}]\mathcal{SS} = \mathcal{SS}$.

*Proof.* Proposition 8.5.(1) says that $\langle\Sigma_{\mathcal{PP}}\rangle[\Sigma_{\mathcal{QQ}}]$ is a contraction mapping, therefore $\langle\Sigma_{\mathcal{PP}}\rangle[\Sigma_{\mathcal{QQ}}]\langle\Sigma_{\mathcal{PP}}\rangle[\Sigma_{\mathcal{QQ}}]\mathcal{SS} = \mathcal{SS}$ cannot hold unless $\langle\Sigma_{\mathcal{PP}}\rangle[\Sigma_{\mathcal{QQ}}]\mathcal{SS} = \mathcal{SS}$. ☺

**Proposition 8.11** Let $\mathcal{SS}$ and $\mathcal{PP}$ be abstract properties and $\mathcal{QQ}$ be a concrete one.

(1) $[\Sigma_{\mathcal{QQ}}]\langle\Sigma_{\mathcal{PP}}\rangle[\Sigma_{\mathcal{QQ}}]\mathcal{SS}$ is $(\mathcal{PP},\mathcal{QQ})$-rectified.

(2) $\langle\Sigma_{\mathcal{PP}}\rangle[\Sigma_{\mathcal{QQ}}]\langle\Sigma_{\mathcal{PP}}\rangle[\Sigma_{\mathcal{QQ}}]\mathcal{SS}$ is $(\mathcal{PP},\mathcal{QQ})$-core.

*Proof.* Proposition 8.5.(2) gives both (1) and (2), though only (1) is an 'efficient' result. ☺

Finally, when we come to reprise Proposition 6.16, the unconstrained interaction between $\mathcal{SS}$, $\mathcal{PP}$ and $\mathcal{QQ}$ generates more complexity than it profits us to fully explore. Thus $\mathcal{CMF}_{\mathrm{A}}$ partitions into five subsets all potentially non-empty, thus:

$$\mathcal{CMF}_{\mathrm{A}} = \mathcal{CMF}_{\mathrm{A}}^{\mathcal{NS}} \ \cup\!\!\!\cdot\ (\mathcal{SS}^{\bullet} \cap \mathcal{PP}) \ \cup\!\!\!\cdot\ (\mathcal{SS}^{\bullet} \cap \overline{\mathcal{PP}}) \ \cup\!\!\!\cdot\ (\overline{\mathcal{SS}}^{\bullet} \cap \mathcal{PP}) \ \cup\!\!\!\cdot\ (\overline{\mathcal{SS}}^{\bullet} \cap \overline{\mathcal{PP}}) \tag{8.5}$$

At the concrete level, this generates a partition of $\mathcal{CMF}_{\mathrm{C}}$ into $\mathcal{CMF}_{\mathrm{C}}^{\mathcal{NS}}$ and fifteen potentially non-empty subsets of $\mathcal{CMF}_{\mathrm{C}}^{\mathcal{S}}$ as follows. There are four subsets containing concrete multifragments whose $\Sigma$ images lie entirely in the four abstract subsets:

$$(\mathcal{SS}^{\bullet} \cap \mathcal{PP}),\ (\mathcal{SS}^{\bullet} \cap \overline{\mathcal{PP}}),\ (\overline{\mathcal{SS}}^{\bullet} \cap \mathcal{PP}),\ (\overline{\mathcal{SS}}^{\bullet} \cap \overline{\mathcal{PP}}) \tag{8.6}$$

i.e. they are:

$$[\Sigma](\mathcal{SS}^{\bullet} \cap \mathcal{PP}),\ [\Sigma](\mathcal{SS}^{\bullet} \cap \overline{\mathcal{PP}}),\ [\Sigma](\overline{\mathcal{SS}}^{\bullet} \cap \mathcal{PP}),\ [\Sigma](\overline{\mathcal{SS}}^{\bullet} \cap \overline{\mathcal{PP}}) \tag{8.7}$$

There are six subsets containing concrete multifragments $\mathcal{T}$ whose abstract images lie entirely in pairs drawn from (8.6). There are four subsets whose abstract images lie entirely in triples drawn from (8.6), and a final subset related to all four subsets in (8.6). None of this mentions $\mathcal{QQ}$. Each of the above concrete subsets splits into a part in $\mathcal{QQ}$ and a part outside $\mathcal{QQ}$, i.e.:

$$\begin{aligned}&[\Sigma_{\mathcal{QQ}}](\mathcal{SS}^{\bullet} \cap \mathcal{PP}),\ [\Sigma_{\overline{\mathcal{QQ}}}](\mathcal{SS}^{\bullet} \cap \mathcal{PP}),\ [\Sigma_{\mathcal{QQ}}](\mathcal{SS}^{\bullet} \cap \overline{\mathcal{PP}}),\ [\Sigma_{\overline{\mathcal{QQ}}}](\mathcal{SS}^{\bullet} \cap \overline{\mathcal{PP}}),\\ &[\Sigma_{\mathcal{QQ}}](\overline{\mathcal{SS}}^{\bullet} \cap \mathcal{PP}),\ [\Sigma_{\overline{\mathcal{QQ}}}](\overline{\mathcal{SS}}^{\bullet} \cap \mathcal{PP}),\ [\Sigma_{\mathcal{QQ}}](\overline{\mathcal{SS}}^{\bullet} \cap \overline{\mathcal{PP}}) \ \ldots \text{ etc.}\end{aligned} \tag{8.8}$$

So we get thirty potentially non-empty simulating subsets of $\mathcal{CMF}_{\mathrm{C}}$. This is without considering iterations of the $\langle\Sigma_{\mathcal{PP}}\rangle$, $[\Sigma_{\mathcal{QQ}}]$, $\langle\Sigma_{\mathcal{QQ}}\rangle$, $[\Sigma_{\mathcal{PP}}]$ operators, which create even finer subdivisions, as Counterexample 8.6 shows. We do not explore these more detailed partitions further here.

# 9 CPTs for Regular Simulation Relations

We recall that a relation $R : \mathsf{X} \leftrightarrow \mathsf{Y}$ is regular iff $R;R^{-1};R = R$, where ; is forward relational composition. Regular relations are also often called difunctional because any regular relation $R$ can be equivalently characterised by the property that there are two partial functions $f : \mathsf{X} \to \mathsf{T}$ and $g : \mathsf{Y} \to \mathsf{T}$ such that $f;g^{-1} = R$. This means in particular that functions and inverse functions are subsumed by regularity, which makes regularity widely applicable in practice, since many development steps feature functions or inverse functions in the passage from abstract to concrete; see [Banach (1995)].

As an easy consequence of difunctionality, a regular relation is one whose domain $\mathrm{dom}(R)$ and range $\mathrm{rng}(R)$ are partitioned into an equal number of equivalence classes, such that for any two classes $[x] \subseteq \mathrm{dom}(R)$ and $[y] \subseteq \mathrm{rng}(R)$, the restriction of $R$ is either empty from $[x]$ to $[y]$, or universal from $[x]$ to $[y]$, the universal cases corresponding to $f^{-1}(t)\times g^{-1}(t)$ when $t \in \mathsf{T}$ is in the range of both $f$ and $g$. Such elements of $\mathsf{T}$ consequently set up a bijection between the equivalence classes of the domain and those of the range.

**Proposition 9.1** Suppose the simulation relation $\Sigma$ arising from a retrenchment is regular. Then the abstract equivalence classes in $\mathcal{CMF}_{\mathrm{A}}$ of $\Sigma$ are robust properties.

*Proof.* Let $\mathcal{SS}$ be an abstract equivalence class in $\mathcal{CMF}_A$ of $\Sigma$. It is clearly a property. Evidently $\mathcal{SS}^\bullet = \mathcal{SS}$. Also there is a concrete equivalence class in $\mathcal{CMF}_C$ of $\Sigma$, $\mathcal{TT}$, such that $\mathcal{S} \in \mathcal{SS}$ and $\mathcal{S}\,\Sigma\,\mathcal{T}$ implies $(\mathcal{S}, \mathcal{T}) \in \mathcal{SS} \times \mathcal{TT}$. And $\mathcal{T} \in \mathcal{TT}$ and $\mathcal{S}\,\Sigma\,\mathcal{T}$ implies the same thing. From this it is easy to see that $\langle\Sigma\rangle\mathcal{SS} = \mathcal{TT} = [\Sigma]\mathcal{SS}$. ☺

**Corollary 9.2** Suppose the $\Sigma$ relation arising from a retrenchment is (partial) functional or (partial) inverse functional. Then the abstract equivalence classes of $\Sigma$ are robust properties.

**Counterexample 9.3** The converse of Proposition 9.1 (which would say that robust properties arose from regular $\Sigma$ relations) does not hold. This is shown by a variant of Fig. 4 in which the rightmost abstract and concrete multifragments are removed, resulting in a $\not|$ shape, a $\Sigma$ relation which is not regular. Now $[\Sigma]\mathcal{SS}$ becomes equal to $\langle\Sigma\rangle\mathcal{SS}$, and $\mathcal{SS}$ is robust.

**Proposition 9.4** Suppose the $\Sigma$ relation arising from a retrenchment is regular. Let $\mathcal{SS}$ be an abstract equivalence class of $\Sigma$ and $\mathcal{QQ}$ a concrete property. Then:

(1) $\langle\Sigma\rangle\mathcal{SS} \cap \mathcal{QQ} \neq \emptyset \;\Leftrightarrow\; \langle\Sigma\rangle[\Sigma_{QQ}]\mathcal{SS} = \mathcal{SS}$

(2) $\langle\Sigma\rangle\mathcal{SS} \subseteq \mathcal{QQ} \;\Leftrightarrow\; [\Sigma]\langle\Sigma_{QQ}\rangle\mathcal{SS} = \mathcal{SS}$

*Proof.* For (1), let $\mathcal{T} \in \langle\Sigma\rangle\mathcal{SS} \cap \mathcal{QQ}$. By Proposition 9.1, $\mathcal{T} \in [\Sigma]\mathcal{SS} \cap \mathcal{QQ}$, i.e. $\mathcal{T} \in [\Sigma_{QQ}]\mathcal{SS}$. By regularity, $\mathcal{S}\,\Sigma\,\mathcal{T}$ iff $\mathcal{S} \in \mathcal{SS}$, so that $\langle\Sigma\rangle[\Sigma_{QQ}]\mathcal{SS} = \mathcal{SS}$. Conversely, if $\langle\Sigma\rangle[\Sigma_{QQ}]\mathcal{SS} = \mathcal{SS}$, then $[\Sigma_{QQ}]\mathcal{SS} \neq \emptyset$, so there must be some $\mathcal{T} \in \langle\Sigma\rangle\mathcal{SS} \cap \mathcal{QQ}$.

For (2), if $\langle\Sigma\rangle\mathcal{SS} \subseteq \mathcal{QQ}$, then $\langle\Sigma_{QQ}\rangle\mathcal{SS} = \langle\Sigma\rangle\mathcal{SS}$, hence $[\Sigma]\langle\Sigma_{QQ}\rangle\mathcal{SS} = [\Sigma]\langle\Sigma\rangle\mathcal{SS}$. Since by Proposition 9.1 $\mathcal{SS}$ is robust, by Proposition 6.14.(3) $\mathcal{SS}$ is weakly robust. This implies $[\Sigma]\langle\Sigma\rangle\mathcal{SS} = \mathcal{SS}$. For the converse, suppose $[\Sigma]\langle\Sigma_{QQ}\rangle\mathcal{SS} = \mathcal{SS}$ holds, but that $\mathcal{T} \in \langle\Sigma\rangle\mathcal{SS} - \mathcal{QQ}$. Then for $\mathcal{T}$, since by regularity $\mathcal{S}\,\Sigma\,\mathcal{T}$ iff $\mathcal{S} \in \mathcal{SS}$, we have that for all $\mathcal{S} \in \mathcal{SS}$, $\mathcal{S} \notin [\Sigma]\langle\Sigma_{QQ}\rangle\mathcal{SS}$ since $\mathcal{T} \notin \mathcal{QQ}$. In fact $[\Sigma]\langle\Sigma_{QQ}\rangle\mathcal{SS}$ is empty. ☺

Proposition 9.4 describes a particularly clean interaction between the $[\Sigma]$ and $\langle\Sigma\rangle$ transformers on the one hand, and constraints arising from a property $\mathcal{QQ}$ on the other, due to the fact that $\Sigma$ breaks up into a number of disjoint universal relations. We now examine sufficient conditions for the simulation relation of a retrenchment to be regular.

Consider a retrenchment given by the data $G$, $P_{Op}$, $O_{Op}$, $C_{Op}$. When we say any of these relations (or any relations formed from these using the usual combinators) is regular, we mean that it is regular when regarded as a relation from the relevant cartesian product of abstract data spaces to the corresponding cartesian product of concrete spaces.

**Definition 9.5** We say that a retrenchment has regular data iff for all operations $Op \in \mathsf{Ops}_A \cap \mathsf{Ops}_C$, the relation given by $G(u, v) \wedge P_{Op}(i, j, u, v)$, the relation given by $G(u', v') \wedge O_{Op}(o, p; u', v', i, j, u, v)$, and the one given by $C_{Op}(u', v', o, p; i, j, u, v)$, are all regular in the sense just stated (where in the case of $G \wedge P_{Op}$ and of $G' \wedge O_{Op}$, we implicitly assume that $G$ and $G'$ are extended by appropriate universal relations on the other variables involved, in order that the overall relation has the correct signature). We say that the retrenchment is fully regular iff in addition, the relation given by $((G(u', v') \wedge O_{Op}(o, p; u', v', i, j, u, v)) \vee C_{Op}(u', v', o, p; i, j, u, v))$ is regular.

Since the intersection of regular relations is regular, regularity of say $G \wedge P_{Op}$, follows from the individual regularity of $G$ and $P_{Op}$ (though this is not a necessary condition). However, since the union of regular relations need not be regular, regular data for a retrenchment does not imply that the retrenchment is fully regular.

**Proposition 9.6** Suppose a retrenchment is fully regular. Then its one-step simulation relation $\Sigma^1$ is regular.

*Proof.* We must show that $\Sigma^1 ; (\Sigma^1)^{-1} ; \Sigma^1 \subseteq \Sigma^1$ since the converse is obvious. So suppose $Op_A$ and $Op_C$ are corresponding abstract and concrete operations and

$$\begin{array}{l}(u_1 \text{-}(i_1, Op_A, o_1)\text{-}\rightarrow u'_1)\ \Sigma^1\ (v_1 \text{-}(j_1, Op_C, p_1)\text{-}\rightarrow v'_1)\\ (u_2 \text{-}(i_2, Op_A, o_2)\text{-}\rightarrow u'_2)\ \Sigma^1\ (v_1 \text{-}(j_1, Op_C, p_1)\text{-}\rightarrow v'_1)\\ (u_2 \text{-}(i_2, Op_A, o_2)\text{-}\rightarrow u'_2)\ \Sigma^1\ (v_2 \text{-}(j_2, Op_C, p_2)\text{-}\rightarrow v'_2)\end{array} \quad (9.1)$$

all hold. Refering to (4.4), we see that $(G \wedge P_{Op})(u_1, v_1, i_1, j_1)$ and $(G \wedge P_{Op})(u_2, v_1, i_2, j_1)$ and $(G \wedge P_{Op})(u_2, v_2, i_2, j_2)$ all hold. By regularity $(G \wedge P_{Op})(u_1, v_2, i_1, j_2)$ is true. Similarly we derive $((G' \wedge O_{Op}) \vee C_{Op})(u'_1, v'_2, o_1, p_2; i_1, j_2, u_1, v_2)$ by full regularity. We know from (9.1) that $stp_{Op_C}(v_2, j_2, v'_2, p_2) \wedge stp_{Op_A}(u_1, i_1, u'_1, o_1)$ hold, so we have all the ingredients of $(u_1 \text{-}(i_1, Op_A, o_1)\text{-}\rightarrow u'_1)\ \Sigma^1\ (v_2 \text{-}(j_2, Op_C, p_2)\text{-}\rightarrow v'_2)$ which is what is needed. ☺

**Proposition 9.7** A default retrenchment has a regular $\Sigma^1$ relation iff its original $G(u, v) \wedge P_{Op}(i, j, u, v)$ relations are regular for each $Op$.

*Proof.* To fix terms, suppose we are given $G$, $P_{Op}$, $O_{Op}$, and from these we generate a default retrenchment for which the default within relation ${P^{\text{Def}}}_{Op}$ and default concedes relation ${C^{\text{Def}}}_{Op}$ are given via (3.1) and (3.2). Let us calculate its $\Sigma^1$ relation. Suppressing variable names, we obtain from (4.4):

$$\begin{array}{l}G \wedge {P^{\text{Def}}}_{Op} \wedge stp_{Op_C} \wedge stp_{Op_A} \wedge ((G' \wedge O_{Op}) \vee {C^{\text{Def}}}_{Op})\\ =\ \text{(definitions)}\\ G \wedge (G \wedge P_{Op} \wedge (\exists\, u', o, v', p \bullet stp_{Op_A} \wedge stp_{Op_C})) \wedge stp_{Op_C} \wedge stp_{Op_A} \wedge\\ ((G' \wedge O_{Op}) \vee (G \wedge P_{Op} \wedge stp_{Op_A} \wedge stp_{Op_C} \wedge \neg\, (G' \wedge O_{Op}))\\ =\\ G \wedge P_{Op} \wedge stp_{Op_C} \wedge stp_{Op_A} \wedge ((G' \wedge O_{Op}) \vee (G \wedge P_{Op} \wedge stp_{Op_A} \wedge stp_{Op_C}))\\ =\\ G \wedge P_{Op} \wedge stp_{Op_C} \wedge stp_{Op_A}\end{array} \quad (9.2)$$

Now let us check $\Sigma^1 ; (\Sigma^1)^{-1} ; \Sigma^1 \subseteq \Sigma^1$ with $\Sigma^1$ given by (9.2). Refering to (9.1), the left hand side implies $stp_{Op_C}(v_2, j_2, v'_2, p_2) \wedge stp_{Op_A}(u_1, i_1, u'_1, o_1)$ immediately, and also $((G \wedge P_{Op})(u_1, v_1, i_1, j_1) \wedge (G \wedge P_{Op})(u_2, v_1, i_2, j_1) \wedge (G \wedge P_{Op})(u_2, v_2, i_2, j_2)) = (*)$. If $(G \wedge P_{Op})$ is regular, then we infer $(G \wedge P_{Op})(u_1, v_2, i_1, j_2)$ and thence the regularity of the one-step simulation relation. However if $(G \wedge P_{Op})$ is not regular then there will be a counterexample that witnesses it, given by some values $u_1$, $v_1$, $i_1$, $j_1$, $u_2$, $v_2$, $i_2$, $j_2$ for which $(*)$ holds but $(G \wedge P_{Op})(u_1, v_2, i_1, j_2)$ doesn't, defeating the regularity of the $\Sigma^1$ relation also. ☺

Note that this result was obtained by a direct argument rather than by exploiting Proposition 9.6. An algebraic proof along the expected lines would not be possible since for a default retrenchment, the $\Sigma^1$ relation is defined using a complement of a relation

expected to be regular, i.e. $\neg\,(G' \wedge O_{Op})$. Unfortunately complements of regular relations are not normally regular.

**Proposition 9.8** A retrenchment has a regular $\Sigma^1$ relation iff it has a regular (multifragment) $\Sigma$ relation.

*Proof.* Assume the $\Sigma^1$ relation is regular. Let $\mathcal{S}$ and $\mathcal{T}$ be abstract and concrete multifragments. By Corollary 4.8, $\mathcal{S}\ \Sigma\ \mathcal{T}$ iff there is a totally ordered bijection between the simulable steps of $\mathcal{S}$ and those of $\mathcal{T}$, such that each pair in the bijection is a pair of steps in simulation. Now let $\mathcal{S}_1\ \Sigma\ \mathcal{T}_1$ and $\mathcal{S}_2\ \Sigma\ \mathcal{T}_1$ and $\mathcal{S}_2\ \Sigma\ \mathcal{T}_2$ all hold. Composing the relevant totally ordered bijections in the obvious way, gives a totally ordered bijection between the simulable steps of $\mathcal{S}_1$ and those of $\mathcal{T}_2$. For each pair of simulable steps in the bijection, say $u_1$ -($i_1$, $Op_A$, $o_1$)-› $u'_1$ and $v_2$ -($j_2$, $Op_C$, $p_2$)-› $v'_2$, we have (9.1), and so, since the one-step simulation relation is regular, $u_1$ -($i_1$, $Op_A$, $o_1$)-› $u'_1$ and $v_2$ -($j_2$, $Op_C$, $p_2$)-› $v'_2$ are in simulation. Repeating for all simulable steps in $\mathcal{S}_1$ and $\mathcal{T}_2$ yields $\mathcal{S}_1\ \Sigma\ \mathcal{T}_2$, and thus $\Sigma$ for multifragments is regular. For the converse we note that the $\Sigma^1$ relation is effectively a restriction of the $\Sigma$ relation to exception-free multifragments containing just one simulable step. ☺

The impact of Proposition 9.7 and Proposition 9.8 is profound. It says that 'almost any' development step done with retrenchment can enjoy a regular $\Sigma$ relation; with the appreciable simplification of property mappings that ensues from this. The reason is that arranging for the $G(u, v) \wedge P_{Op}(i, j, u, v)$ relations of the retrenchment to be regular is usually not too difficult, as they can frequently be restricted to straightforward mappings between the representations of states and inputs at the two levels of abstraction. Once this holds, the default retrenchment is always available, and this guarantees a regular $\Sigma$ relation via Proposition 9.8.

# 10 Examples

It is time to examine some examples of the preceding theory. To start with we revisit the two main examples developed in detail in [Banach et al. (2005)], on multisets and sequences, and on elementary control theory. Finally we look at a very simple model of noninterfering processes, in which the main question of interest refers to the interaction between concepts of ambivalence and simulation mappings of a property.

## 10.1 Multisets and Sequences

For this example the abstract model is a multiset of NATs equipped with operations $put_A$, $get_A$ which insert an element and remove an element from the multiset. The concrete world is more constrained, having as state, a sequence of NATs of maximum length 10, or one of two exceptional states *Uflow*, *Oflow*, and with $put_C$, $get_C$ as the analogues of $put_A$, $get_A$. The more constrained $put_C$, $get_C$ can get to *Uflow*, *Oflow*, when there is an attempt to breach the bounds, at which point a $reset_C$ is able to restore order. In detail the operations and state spaces are:

$$\{put_A, get_A\} = \mathsf{Ops}_A \subseteq \mathsf{Ops}_C = \{put_C, get_C, reset_C\} \tag{10.1}$$

$$\begin{aligned}
&\mathsf{U} = \mathcal{M}(\mathsf{NAT}),\ \mathsf{I}_{put_A} = \mathsf{NAT},\ \mathsf{O}_{put_A} = \emptyset,\ \mathsf{I}_{get_A} = \emptyset,\ \mathsf{O}_{get_A} = \mathsf{NAT}\\
&\mathsf{V} = \{ll \in \mathrm{seq}(\mathsf{NAT}) \mid \mathrm{length}(ll) \leq 10\} \cup \{Uflow, Oflow\},\\
&\mathsf{J}_{put_C} = \mathsf{NAT},\ \mathsf{P}_{put_C} = \{\mathrm{FULL}\},\ \mathsf{J}_{get_C} = \emptyset,\ \mathsf{P}_{get_C} = \mathsf{NAT} \cup \{\mathrm{EMPTY}\}\\
&\mathsf{J}_{reset_C} = \emptyset,\ \mathsf{P}_{reset_C} = \{\mathrm{OK}\}
\end{aligned} \tag{10.2}$$

The abstract transitions are:

$$mset \text{ -}(n,\ put_{\mathrm{A}},\ \varepsilon)\text{-}\!\rightarrow mset + \{n\} \ \ ;\ \ mset + \{n\} \text{ -}(get_{\mathrm{A}},\ n)\text{-}\!\rightarrow mset \tag{10.3}$$

and the concrete ones are:

$$\begin{array}{ll} mseq \text{ -}(n,\ put_{\mathrm{C}},\ \varepsilon)\text{-}\!\rightarrow mseq@[n]\ , & \text{if length}(mseq) \leq 9 \\ mseq \text{ -}(n,\ put_{\mathrm{C}},\ \mathrm{FULL})\text{-}\!\rightarrow Oflow\ , & \text{if length}(mseq) = 10 \end{array} \tag{10.4}$$

$$\begin{array}{ll} n::mseq \text{ -}(get_{\mathrm{C}},\ n)\text{-}\!\rightarrow mseq\ , & \text{if length}(mseq) \leq 9 \\ mseq \text{ -}(get_{\mathrm{C}},\ \mathrm{EMPTY})\text{-}\!\rightarrow Uflow\ , & \text{if length}(mseq) = 0 \end{array} \tag{10.5}$$

$$Oflow \text{ -}(reset_{\mathrm{C}},\ \mathrm{OK})\text{-}\!\rightarrow [\,]\ ,\ \ Uflow \text{ -}(reset_{\mathrm{C}},\ \mathrm{OK})\text{-}\!\rightarrow [\,] \tag{10.6}$$

The retrieve relation is the obvious:

$$G(mset, mseq) \ = \ (\mathrm{mrng}(mseq) = mset) \tag{10.7}$$

with the two values *Uflow*, *Oflow* being outside the range of $G$. The initialisations are:

$$Init_{\mathrm{A}}(\varnothing) \ \ ;\ \ Init_{\mathrm{C}}([\,]) \tag{10.8}$$

The concrete model is not a refinement of the abstract one, but it easily becomes a retrenchment of it. One possible set of within, output, and concedes relations for the *put* and *get* operations is the following:

$$P_{put}(i, j, mset, mseq) \ = \ (i = j \wedge mseq \notin \{Uflow, Oflow\}) \tag{10.9}$$

$$O_{put}(o, p; mset', mseq', i, j, mset, mseq) \ = \ (o = \varepsilon \wedge p = \varepsilon) \tag{10.10}$$

$$\begin{array}{l} C_{put}(mset', mseq', o, p; i, j, mset, mseq) \ = \\ \quad (p = \mathrm{FULL} \wedge mseq' = Oflow \wedge \mathrm{length}(mseq) = 10 \wedge mset' = mset + \{i\}) \end{array} \tag{10.11}$$

$$P_{get}(i, j, mset, mseq) \ = \ (mseq \notin \{Uflow, Oflow\} \wedge \mathrm{length}(mseq) \neq 0) \tag{10.12}$$

$$O_{get}(o, p; mset', mseq', i, j, mset, mseq) \ = \ (o = p) \tag{10.13}$$

$$C_{get}(mset', mseq', o, p; i, j, mset, mseq) \ = \ \mathsf{false} \tag{10.14}$$

The first thing to do with an example like this is to check the regularity conditions, since if the example proves to be regular, enormous simplifications follow as we saw in the previous section. Evidently the retrieve relation $G$ is regular; and extending $G$ by universal relations on other variables, as needed in forming well typed more complex expressions in the operation PO preserves this. Let us now look at the *get* operation. $P_{get}$ is regular since it is universal aside from restricting the value of *mseq*, and therefore $(G \wedge P_{get})$ is regular. $O_{get}$ is regular since it is an equality relation, and so $(G' \wedge O_{get})$ is regular. $C_{get}$ is empty, so $((G' \wedge O_{get}) \vee C_{get}) = (G' \wedge O_{get})$ is regular and so the *get* part of the $\Sigma^1$ relation is regular too.

We check the *put* operation. $P_{put}$ is a conjunction of an equality relation and a restriction on the value of *mseq*, both regular, so $P_{put}$ is regular; therefore $(G \wedge P_{put})$ is regular. $O_{put}$ is a pair of constraints on abstract and concrete outputs so is regular, hence $(G' \wedge O_{put})$ is regular. Turning to $C_{put}$, the first three clauses of (10.11) obviously define regular relations, so their conjunction is too. The last clause is an equality relation on three of the abstract variables, so reduces to a restriction on their values, so is regular; hence $C_{put}$ is regular.

Finally we check $((G' \wedge O_{put}) \vee C_{put})$. Consider abstract and concrete quadruples (before-state, input, after-state, output), where the components need not form a transition step. The abstract tuple A1 = ($\{1\}$, 2, $\{1 \ldots 11\}$, ε) is related by $(G' \wedge O_{put})$ to the concrete tuple C1 = ([291], 378, $[1 \ldots 11]$, ε). C1 in turn is related by $(G' \wedge O_{put})$ to the abstract tuple A2 = ($\{2 \ldots 11\}$, 1, $\{1 \ldots 11\}$, ε). A2 is related by $C_{put}$ to C2 = ([2912], 3789, *Oflow*, FULL). Now C2 cannot be related by $(G' \wedge O_{put})$ to anything since its after-state is *Oflow* and $(G' \wedge O_{put})$ is not defined for such values. Moreover C2 is not related to A1 by $C_{put}$ because $\{1\} + \{2\} \neq \{1 \ldots 11\}$ as would be demanded by $C_{put}$. So A1 $((G' \wedge O_{put}) \vee C_{put})$ C2 is false and $((G' \wedge O_{put}) \vee C_{put})$ is not regular.

Although we don't have a fully regular retrenchment, the general result about default retrenchments should not leave us despondent. All it means is that $C_{put}$ lacks some clauses that make it strong enough for full regularity to emerge. In fact it is useful to check the $\Sigma^1$ relation for *put* directly. There are two subcases. When $put_C$ does not overflow, there is clearly a partial function taking concrete (*mseq*, *j*, *mseq*@[*j*], ε) to abstract (mrng(*mseq*), *j*, mrng(*mseq*@[*j*]), ε) which is regular. When $put_C$ overflows, there is another partial function taking (*mseq*, *j*, *Oflow*, FULL) to (mrng(*mseq*), *j*, mrng(*mseq*@[*j*]), ε), also regular. The domains and ranges of these partial functions are disjoint, so their union is regular. Thus since the $\Sigma^1$ relations for *put* and *get* are regular, the whole of the $\Sigma^1$ relation (which is their disjoint union due to the tagging by the operation names themselves) is regular, and we have what was desired after all.

Let us now examine the (multi)fragments of the two systems. For the abstract system, a fragment starts with some $mset_0$ and can perform arbitrary sequences of $put_A$s and $get_A$s, provided that at each state visited, the number of $get_A$s does not exceed the number of $put_A$s by more than the cardinality of $mset_0$, i.e. $|get_A| - |put_A| \leq |mset_0|$. A multifragment is just a sequence of such fragments.

For the concrete system, a simulable segment of a fragment starts off at some $mseq_{sim}$ and can perform sequences of $put_C$s and $get_C$s, such that at each state visited: (1) the outcome of any subsequent $get_C$ is unique (unlike the abstract case), (2.a) the length of *mseq* either stays in bounds after a subsequent $put_C$ or $get_C$ (because $|get_C| - |put_C| \leq |mseq_{sim}|$ and $|put_C| - |get_C| \leq 10 - |mseq_{sim}|$), or (2.b) a subsequent $put_C$ takes the state to *Oflow*, or (2.c) a subsequent $get_C$ takes the state to *Uflow*. After *Oflow* or *Uflow*, an exception may follow. It must start with a $reset_C$, possibly followed by some number of alternating $get_C$ and $reset_C$ steps. And after a $reset_C$, there may be another simulable segment etc. If $mseq_{sim} = [\,]$, then the simulable segment may be preceded by some suffix of an exception as just described. A multifragment is a sequence of such fragments.

Since $\Sigma^1$ is regular, so is Σ by Proposition 9.8, and on the equivalence classes of Σ, $\langle\Sigma\rangle$ and $[\Sigma]$ coincide by Proposition 9.1. We illustrate these on the simple abstract property:

$$\{\varnothing \text{ -}(1, put_A, \varepsilon)\text{-}\rightarrow \{1\} \text{ -}(2, put_A, \varepsilon)\text{-}\rightarrow \{1, 2\} \text{ -}(get_A, 1)\text{-}\rightarrow \{2\} \text{ -}(get_A, 2)\text{-}\rightarrow \varnothing\,,$$
$$\varnothing \text{ -}(1, put_A, \varepsilon)\text{-}\rightarrow \{1\} \text{ -}(2, put_A, \varepsilon)\text{-}\rightarrow \{1, 2\} \text{ -}(get_A, 2)\text{-}\rightarrow \{1\} \text{ -}(get_A, 1)\text{-}\rightarrow \varnothing\} \tag{10.15}$$

In Fig. 6 we indicate how the elements of this property map under Σ. The first fragment is easily simulated by the concrete systems as the elements come out in the same order they went in. However the second fragment runs into a problem half way

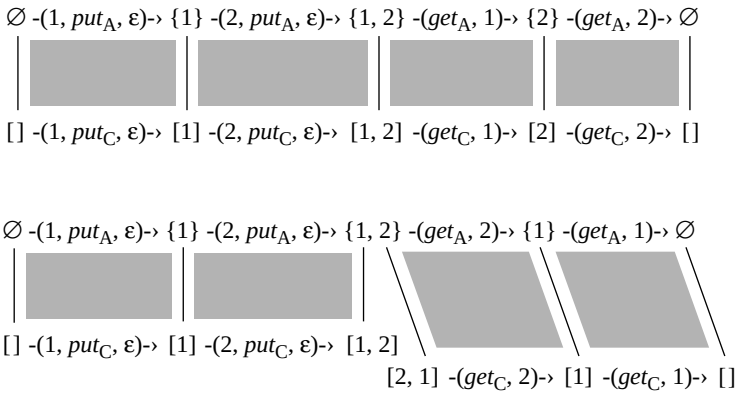



Fig. 6

through, as the desired element, 2, is not at the head of the queue. The formulation of the Σ relation in Section 4 enables us to express what happens via a break in the simulation. Of course the interpretation of this break must come form the context within which the example is being discussed. Moreover, each of the three concrete fragments in Fig. 6 may be embellished with lead-ins and lead-outs as described in the discussion of exceptions above, in many ways. We have suppressed these from the Figure, and their interpretation too would be contingent on the context.

## 10.2 Elementary Control Theory

This example considers a simple control redesign problem in which a simple continuous control situation is remodelled in the discrete domain. We summarise the main details, noting that the subscripts C/D here indicate continuous/discrete. In the state space formulation, the continuous system is described by the differential equation:

$$\dot{x}_C(t) = A_C x_C(t) + B_C r_C(t) \tag{10.16}$$

where $\dot{x}_C(t)$ is the time derivative of $x_C(t)$, $r_C(t)$ is the external input, and $A_C$ and $B_C$ are contants. The solution of (10.16) is standard, and for a period $T$ to the future of a starting point $t = kT$, is given by:

$$x_C((k+1)T) = e^{A_C T} x_C(kT) + \int_0^T e^{A_C(T-\tau)} B_C r_C(kT + \tau)\, d\tau \tag{10.17}$$

This solution can be interpreted as a transition system for an abstract operation $Op_A$ as follows:

$$(x_C(t), \dot{x}_C(t)) \text{ -}(r_C(t), Op_A)\text{-› } (x_C(t'), \dot{x}_C(t')) \tag{10.18}$$

where $t$ and $t'$ are nonnegative real valued and $t < t'$; and there is a requirement that there exists a (global) solution of (10.16), such that for any $0 < t < t'$ the solution agrees with the quantities appearing in (10.18) at $t$ and $t'$. (Standard theory says that there is enough information recorded in the state vector $(x_C(t), \dot{x}_C(t))$, to ensure that a unique analytic solution is determined by it and the input, on which (10.17) is based.)

To enable comparison with the discrete system to be introduced below, we expand (10.17) by Taylor's Theorem and integrate term by term, keeping only terms up to order O($T$), giving:

$$x_C((k+1)T) = (1 + TA_C)x_C(kT) + TB_C r_C(kT) + o(T) \tag{10.19}$$

Now on the discrete side we have:

$$\Delta^{+T}x_D(k) = A_D x_D(k) + B_D r_D(k) \tag{10.20}$$

where $x_D$ is the discrete system state, $\Delta^{+T}x_D(k)$ is its forward difference for sampling period $T$, given by:

$$\Delta^{+T}x_D(k) = \frac{x_D(k+1) - x_D(k)}{T} \tag{10.21}$$

$r_D(k)$ is the external (discrete) input, and $A_D$ and $B_D$ are constants, $r_D$ being the discrete external input signal. The solution of (10.20) for the next state is immediate:

$$x_D(k+1) = (1 + TA_D)x_D(k) + TB_D r_D(k) \tag{10.22}$$

which can be immediately cast as a transition system for a concrete operation $Op_C$ of the following kind:

$$x_D(k) \text{ -}(r_D(k), Op_C)\text{-}\rightarrow x_D(k+1) \tag{10.23}$$

where $k$ is natural valued and the quantities appearing in (10.23) must constitute a solution of (10.20). There is enough information recorded in the state $x_D(k)$ to ensure that a unique solution to (10.22) is determined by it and the input.

Combining (10.19) and (10.22) yields:

$$\begin{aligned} x_C((k+1)T) - x_D(k+1) = (x_C(kT) - x_D(k)) + T(A_C x_C(kT) - A_D x_D(k)) + \\ T(B_C r_C(kT) - B_D r_D(k)) + o(T) \end{aligned} \tag{10.24}$$

which can now easily relate the two systems by a retrenchment. As a retrieve relation we take:

$$G(x_C(kT), x_D(k)) \;=\; | x_C(kT) - x_D(k) | \leq \varepsilon \tag{10.25}$$

where $\varepsilon$ is adequately small. Now (10.24) lends itself to a retrenchment reinterpretation with within, output and concedes relations:

$$\begin{aligned} &P(r_C(kT), r_D(k), x_C(kT), x_D(k)) \;= \\ &\quad | A_C x_C(kT) - A_D x_D(k) + B_C r_C(kT) - B_D r_D(k) | \leq o(1) \end{aligned} \tag{10.26}$$

$$O(x_C((k+1)T), x_D(k+1), r_C(kT), r_D(k), x_C(kT), x_D(k)) \;=\; \text{true} \tag{10.27}$$

$$\begin{aligned} &C(x_C((k+1)T), x_D(k+1); r_C(kT), r_D(k), x_C(kT), x_D(k)) \;= \\ &\quad | x_C((k+1)T) - x_D(k+1) | \leq o(1) \end{aligned} \tag{10.28}$$

The retrenchment (10.25)-(10.28) deserves comment. The first thing to bear in mind is that it is not an exact result, due to the (relatively crude) approximations adopted in deriving it. So, as in many engineering situations, an ideal quantitative description of the situation at hand is not available.

Secondly, the crude analysis performed to arrive at (10.25)-(10.28) entails the inevitable decay of the *provable* precision in estimating $| x_C(kT) - x_D(k) |$ as $k$ increases, due to the estimate being in effect based on a Lipshitz condition, which is what the Taylor's Theorem analysis introduces — and it is well known that *all* estimates based purely on a Lipshitz condition decay rapidly in precision, regardless of the realities of the situation. To what extent the above analysis might be improved by using a different strategy, is an exercise in control theory beyond the scope of this paper.

Thirdly, there is no hope of this retrenchment yielding a regular $\Sigma^1$ relation. For consider $G(x_C(kT), x_D(k))$. Suppose $x^1{}_D(k) = x^1{}_C(kT) + \varepsilon/2$, $x^2{}_C(kT) = x^1{}_D(k) + \varepsilon/2$, $x^2{}_D(k) = x^2{}_C(kT) + \varepsilon/2$, all of which satisfy (10.25). Then $| x^1{}_C(kT) - x^2{}_D(k) | = 3\varepsilon/2$, flouting any prospective regularity of (10.25). This behaviour is typical of numerical approximations, because of the way that the width $\varepsilon$ windows can overlap. Considering the other relations needed for a regular $\Sigma^1$ relation would not improve matters.

Despite these shortcomings, the machinery we built earlier gives us a means to speak about what is desirable in this example. Suppose we have an abstract property $\mathcal{SS}$ that we wish to implement via the concrete system. Then what we are interested in is $[\Sigma]\mathcal{SS}$ (or perhaps $[\Sigma_{QQ}]\mathcal{SS}$ for a suitable $\mathcal{QQ}$), since that is the concrete property whose multifragments simulate the abstract ones in $\mathcal{SS}$ in such a way that the bounds on precision expressed in (10.25) with respect to multifragments in $\mathcal{SS}$ are never violated. Of interest is whether $[\Sigma]\mathcal{SS}$ is empty or not, and whether or not $\mathcal{SS} = \mathcal{SS}^{\bullet}$.

Of course the analysis we did in deriving the retrenchment above is not strong enough for us to be able to formally derive that some arbitrary $\mathcal{TT}$ that we might come up with is in fact $[\Sigma]\mathcal{SS}$ or $[\Sigma_{QQ}]\mathcal{SS}$, but this is no worse than what engineers have to tolerate in real life situations every day. The belief that some $\mathcal{TT}$ is adequate to serve as a $[\Sigma]\mathcal{SS}$ would have to be based on engineering experience (or on a different analysis), and would be as secure or insecure as these can turn out to be.

## 10.3 Noninterfering Processes

Our final example concerns noninterfering processes, and illustrates the utility of the interplay between the simulation property transformers $[\Sigma]$ and $\langle\Sigma\rangle$ on the one hand, and ambivalent notions on the other. Suppose we we make a model of two abstract processes, $X_A$ and $Y_A$, which run concurrently, but which have no influence on each other. We model the concurrency by interleaving semantics. Let the transition relation of $X_A$ be $stp_{X_A}(x, x')$ and that of $Y_A$ be $stp_{Y_A}(y, y')$. For simplicity we assume no I/O and no skip steps in either of $X_A$ or $Y_A$. A step of the combined system is then $(x, y)$ -$(stp_A)$-› $(x', y')$ where either $y = y'$ and $x$ -$(stp_{X_A})$-› $x'$ is a step of $X_A$, or $x = x'$ and $y$ -$(stp_{Y_A})$-› $y'$ is a step of $Y_A$. For any execution fragment of the combined system, projecting onto the $X_A$ states and execution steps in the obvious way clearly yields a valid execution fragment of $X_A$, and similarly for $Y_A$.

Noninterference means that in an execution fragment $\mathcal{S}$, an adjacent pair of steps of $X_A$ and $Y_A$:

$$(x, y) \text{ -}(stp_{X_A})\text{-› } (x', y) \text{ -}(stp_{Y_A})\text{-› } (x', y') \tag{10.29}$$

may be swapped giving:

$$(x, y) \text{-}(stp_{Y_A})\text{-}\rightarrow (x, y') \text{-}(stp_{X_A})\text{-}\rightarrow (x', y') \tag{10.30}$$

and vice versa. Let $swap_{i,k}$ be the operator that performs this swap on the $k$'th and $k+1$'th elements of the $i$'th execution fragment (of a multifragment) provided both steps exist and are performed by different processes. Iterated, such swaps generate all permutations of the steps of the multifragment, consistent with never interchanging the order of any two $X_A$ steps with each other, nor of ever interchanging the order of any two $Y_A$ steps with each other. Clearly, such swapping generates an equivalence relation on fragments, and the equivalence classes are essentially Mazurkiewicz traces for an independence relation which makes all $X_A$ steps independent of all $Y_A$ steps. (See eg. [Mazurkiewicz (1986), Mazurkiewicz (1988)].)

Let $\langle swap \rangle$ be the operator on properties, defined as follows:

$$\langle swap \rangle \mathcal{SS} \;=\; \cup_{i,k} \{swap_{i,k}\mathcal{S} \mid \mathcal{S} \in \mathcal{SS},\, swap_{i,k} \text{ is applicable to } \mathcal{S}_i\} \tag{10.31}$$

Then:

$$[\mathcal{SS}]_{sw} \;=\; \cup_l \langle swap \rangle^l \mathcal{SS} \tag{10.32}$$

yields the smallest fixed point of $\langle swap \rangle$ containing $\mathcal{SS}$, called the swap closure of $\mathcal{SS}$, where the index $l$ ranges over sufficiently high ordinals that the fixed point is reached. It is clear that the swap closure of $\mathcal{SS}$ consists of the Mazurkiewicz traces refered to above.

The idea of swapping steps of independent processes has a very 'ambivalent' feel about it. However we do not have an ambivalent property as in Section 7, since the fixed point of the swapping is not given in terms of a simple predicate on the underlying primitives that define a (multi)fragment. Instead we have a higher order ambivalent operator; i.e. given a property $\mathcal{SS}$, (10.32) produces an in general different property $[\mathcal{SS}]_{sw}$.

Clearly this goes quite a way beyond what we entertained for ambivalence earlier, and illustrates the danger of being premature in formalising ambivalent notions before considering the applications for which they will be used. In the present situation we are dealing with a class of systems for which the state space is a cartesian product, and such that each execution step affects only one component of the product, doing so in a manner indpendent of the value of the other component. Fixing this as a meta level context [SW], for any system $U$ in the class, the $\langle swap \rangle$ and $[\ldots]_{sw}$ operations are well defined, and so can be applied to any property of $U$.

Suppose now that the next development step creates concrete versions of the two processes $X_A$ and $Y_A$, doing so via a retrenchment whose simulation relation $\Sigma$ is regarded as known. Thereby we get processes $X_C$ and $Y_C$. Suppose furthermore that it is important to conserve noninterference through this development step (eg. it may contribute to a model of an important security property). We regard it as a fundamental feature of all systems in [SW], that for any property $\mathcal{SS}$ that is considered to be implementable, all elements of $[\mathcal{SS}]_{sw}$ must also be implementable. Consequently we would need to be able to prove something akin to the following:

*Security ProtoTheorem* Let $\mathcal{TT}$ be a concrete property and let $\mathcal{QQ}$ be a concrete property that captures the exceptional aspects of the multifragments in $\mathcal{TT}$. Then:

$$[\langle\Sigma\rangle\mathcal{TT}\,]_{sw} = \langle\Sigma\rangle[\mathcal{TT}\,]_{sw} = [\Sigma][\mathcal{TT}\,]_{sw} = [[\Sigma]\mathcal{TT}\,]_{sw} \tag{10.33}$$

$$[[\Sigma_{QQ}]\langle\Sigma\rangle\mathcal{TT}\,]_{sw} = [\Sigma_{QQ}][\langle\Sigma\rangle\mathcal{TT}\,]_{sw} = [\mathcal{TT}\,]_{sw} = [\langle\Sigma_{QQ}\rangle[\Sigma]\mathcal{TT}\,]_{sw} = \langle\Sigma_{QQ}\rangle[[\Sigma]\mathcal{TT}\,]_{sw} \tag{10.34}$$

The conditions expressed in (10.33) and (10.34) state that the swap closures of concrete properties are separated in a strong way from other properties. The simulation transformers do not enlarge the swap closures no matter how they are applied. In particular, the abstract images of the swap closures of concrete properties are themselves swap closed, and so concrete swap closures are simulation transformer images of abstract swap closed properties, the embodiment of the security properties of interest.

Note the role of $\mathcal{QQ}$ in constraining the exceptional aspects of $\mathcal{TT}$. Without it, the equality of $[\mathcal{TT}\,]_{sw}$ to the other terms in (10.34) would most likely not be provable, because of the ease with which exceptions can typically be inserted into multifragments. Of course the utility of this formalisation depends on the extent to which $\mathcal{QQ}$ captures the exceptional aspects of $\mathcal{TT}$ independently of $\mathcal{TT}$ itself, since setting $\mathcal{QQ} = \mathcal{TT}$ would yield a credible, if more trivial, statement.

# 11 Conclusions

The preceding sections set up a general theory, first of execution fragments and multifragments, and ultimately of property transformers of various kinds, applicable to pairs of systems related via a retrenchment. As more sensitivity to the context of the transformation got built into the theory, the theory became more complex. However when the criterion of regularity was introduced, considerable simplifications were obtained. Since functional and inverse functional relationships (both instances of regular relations) are so common in practical cases of formal development, the simplification that ensues is very welcome and can be viewed as widely applicable.

Three very different examples illustrated quite well what the theory can and cannot do. The discrete multiset/sequence example exhibited most of what one would expect in a cleanly defined discrete example. The continuous digital redesign control problem showed how different the continuous world is as regards the techniques of this paper; nevertheless the terminology developed enabled the desirable aspects of the redesign to be neatly expressed, even if the actual calculations of the retrenchment were only approximate and did not enable the mentioned desired aspects to actually be proved. Finally the noninterfering processes example led to an interesting higher order development of the ambivalence notion, and showed that system design objectives themselves may interact usefully with the concepts introduced in this paper.

The present theory can be considered as providing a useful framework for discussing properties and their mappings, when the retrenchment relationship is drafted in terms of individual execution steps at abstract and concrete levels, and properties are defined as sets of multifragments. While these provide the simplest approach, they are not the only possibilities. Both for the retrenchment relationship itself and for concepts of system properties, different granularities and greater sensitivity to the system's branching structure can be introduced into the theory, enabling more subtle and potentially more interesting relationships between systems to be brought into the remit of the theory. For example, if we generalised retrenchment to speak about relationships between sequences of steps at the abstract and concrete levels, the content

of the operation POs could acquire a more global perspective. Such a perspective might well beneficially inform continuous scenarios in particular.[7] However this remains work for the future.

## References


Banach R. (1995); On Regularity in Software Design. Sci. Comp. Prog. **24**, 221-248.

Banach R. (2003); General Retrenchment: Symmetric Propositional Theory. *work in progress*.

Banach R., Jeske C. (2002); Output Retrenchments, Defaults, Stronger Compositions, Feature Engineering. `http://www.cs.man.ac.uk/~banach/some.pubs/Retrench.Def.Out.ps.gz` *submitted*.

Banach R., Poppleton M. (1999); Retrenchment and Punctured Simulation. *in*: Proc. IFM-99, Araki, Gallway, Taguchi (eds.), 457-476, Springer.

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

Banach R., Poppleton M., Jeske C., Stepney S. (2005); Engineering and Theoretical Underpinnings of Retrenchment. `http://www.cs.man.ac.uk/~banach/some.pubs/Retrench.Underpin.ps.gz` *submitted*.

Mazurkiewicz A. (1986); Trace Theory. *in*: Advances in Petri Nets. Brauer, Reisig, Rozenberg (eds.), LNCS **255**, 279-324. Springer.

Mazurkiewicz A. (1988); Basic Theory of Traces. *in*: Proc. REX Workshop. de Bakker, de Roever, Rozenberg (eds.), LNCS **354**, 285-363. Springer.

Popkorn S. (1994); First Steps in Modal Logic. C.U.P.


7. Control theory for example is almost entirely concerned with global properties, rather than the local real-time perspective employed in this paper.