Algebra and Logic for Resource-based Systems Modelling

Matthew Collinson, David Pym

HP Laboratories
HPL-2009-21

Keyword(s):
process, resource, logic, algebra, system, modelling

Abstract:

Mathematical modelling is one of the fundamental tools of science and engineering. Very often, models are required to be executable, as a simulation, on a computer. In this paper, we present some contributions to the process-theoretic and logical foundations of discreteevent modelling with resources and processes. We present a process calculus with an explicit representation of resources in which processes and resources co-evolve. The calculus is closely connected to a logic that may be used as a specification language for properties of models. The logic is strong enough to allow requirements that a system has certain structure; for example, that it is a parallel composite of subsystems. This work consolidates, extends, and improves upon aspects of earlier work of ours in this area. An extended example, consisting of a semantics for a simple parallel programming language, indicates a connection with separating logics for concurrency.

External Posting Date: February 6, 2009 [Fulltext] Approved for External Publication
Internal Posting Date: February 6, 2009 [Fulltext]

Submitted to Mathematical Structures in Computer Science

Algebra and Logic for Resource-based Systems Modelling

Matthew Collinson and David Pym

Hewlett-Packard Laboratories, Bristol, UK

February 2, 2009

Abstract

Mathematical modelling is one of the fundamental tools of science and engineering. Very often, models are required to be executable, as a simulation, on a computer. In this paper, we present some contributions to the process-theoretic and logical foundations of discrete-event modelling with resources and processes. We present a process calculus with an explicit representation of resources in which processes and resources co-evolve. The calculus is closely connected to a logic that may be used as a specification language for properties of models. The logic is strong enough to allow requirements that a system has certain structure; for example, that it is a parallel composite of subsystems. This work consolidates, extends, and improves upon aspects of earlier work of ours in this area. An extended example, consisting of a semantics for a simple parallel programming language, indicates a connection with separating logics for concurrency.

1 Introduction

Mathematical modelling and simulation are fundamental tools of science and engineering. They are important in almost all fields, at many scales and at many levels of complexity. This paper deals with the mathematical and logical foundations of discrete-event modelling.

Modelling is the process of making a precise description, a *model*, of a system in order that its properties may be subjected to a rigorous analysis. The precise form of the model, the analysis it is subjected to and even the modelling process itself depend upon the object of study. Some general observations are, however, in order to put this paper in context. Any model should be *sound*, in the sense that all parts somehow represent aspects of the system being modelled. On the other-hand a model need not be *complete* in order to be useful, it does not have to represent every aspect of the system being described. Thus it is important to distinguish between the model and the underlying system which the model represents. Very often this introduces feedback into the modelling process, in which a hierarchy of successively refined models is created.

One kind of model that frequently arises is the *discrete-event* model. In such models, the (model of the) system evolves in discrete jumps. In traditional applied mathematics, such systems are often described by families of difference equations that describe how the system changes locally in time from one instant to the next. From these equations an evolution (or flow) operator is produced that completely describes the behaviour of the system. This calculational method is undoubtedly very powerful. In practice, however, it can be difficult to formulate the equations in a soluble form. This is often the case when the system is complex, for example, with many mutually dependent, heterogeneous components, evolving concurrently in different ways and on different time-scales. In such situations when calculation is difficult, infeasible or cannot be carried out within a given time, it can be particularly useful to produce a computational model of the system. In practice, this is done with a whole host of programming languages and tools. There are even languages specifically designed for such tasks. Perhaps the best-known simulation language is Simula (Dahl, Myhrhaug & Nygaard 1970). Most of the time the semantics of such languages are not well-understood. Recall, however, the soundness criterion on models. There is a need for

simulation languages that rest upon rigorous foundations, in order that no spurious trajectories are introduced inadvertently into simulations.

In this paper, we use Demos2k (Birtwistle & Tofts 2001*a*, Birtwistle & Tofts 2001*b*, *Demos2k* 2002) as an important example of such a language. Demos2k is a descendant of the original Demos tool, (Birtwistle 1979), which itself is a descendant of Simula. Henceforward, all specific references to the workings of Demos should be taken to refer to the later Demos2k. Demos is a discrete-event modelling tool used to describe the concurrent co-evolution of many *entities* together with the *resources* they use. It is closely related to languages for concurrent and distributed programming, see for example (Ben-Ari 1990). The soundness of Demos2k as a modelling tool is encapsulated in the statement that it is *semantically justified*: there is a precise mathematical description of the structure and evolution of every model written in the language. Alternative modelling tools that use languages influenced by the theory of semantics include the Concurrency Workbench (Cleaveland, Parrow & Steffen 1993), the Mobility Workbench (Victor & Moller 1994), and PRISM (Hinton, Kwiatkowska, Norman & Parker 2006) amongst others. In addition, there is a well-developed modelling paradigm, in which systems of interest are characterized in terms of their environment (typically represented as collections of stochastic events), the spatial or logical distribution of the system, the resources present in the system, and the processes that the system executes. Further discussion of this paradigm is beyond the scope of this paper, but the analysis presented herein directly supports its last two aspects. Demos conforms much more closely with this paradigm than do the alternative tools mentioned above.

The area of semantics most closely connected to this kind of work is known as *process algebra*. A process algebra can be thought of as a precise mathematical language for describing concurrently evolving entities called *processes*. Indeed, they can fruitfully be thought of as idealized simulation languages (of a certain kind). This point of view has been expounded by others in the past and a significant body of work has been built-up in pursuing it. The paper (Birtwistle, Pooley & Tofts 1993), for example, is a good introduction, whilst (Tofts 2006) provides an appraisal of the methodology, including an account of its scope and suggestions of areas and problems to which it may be expected to make further contributions.

The process calculi which we shall develop in this paper are strongly influenced by Milners' Synchronous Calculus of Communicating Processes SCCS (Milner 1983). This followed an earlier development, CCS, which was asynchronous (Milner 1980, Milner 1989). Other process calculi emerged around the same time from other researchers, and there have been many developments since, but for the most part these will not concern us here. A survey (Baeten 2005) of the history of process algebra has appeared.

A good process algebra usually has several distinguishing features. To begin with, processes should be constructed formally from a well-defined collection of atomic *actions* and a small number of process *constructors*. Models of systems are assembled from sub-models by means of the constructors. The meaning of every process should be given by a *structural operational semantics*, (Plotkin 2004), with constructors having a natural, intuitive reading. The operational semantics generates a *transition structure* (or system) that describes the flow of processes. An algebraic theory of process equality called *(bi)simulation* is used to identify processes with the same behaviour. In addition, it is usually necessary (for entirely practical reasons) to have a second language that makes logical assertions about properties of processes. Thus the process algebra presents not only a precise description of the evolution of processes but also a method for specifying and verifying process properties. Certain forms of process are known to give rise to terminating algorithms for constructing transition structures, and for model-checking processes against specifications. Automated tools are then often provided in support. When all of these features are present the process algebra may be seen to be an integrated environment for both the synthesis and analysis of models.

There are, however, some difficulties associated with using existing process algebras as a foundation for modelling and simulation of the kind we have described, although it can be done, (Birtwistle et al. 1993). These difficulties arise from there being no *direct* representation of resource, which must therefore be represented via an encoding. This has two consequences: firstly, there is a lack of clear conceptual analysis of the notion of resource, a significant burden for the modeller who must track important quantitative information conveyed by the resource; and, sec-

ondly, keeping track of the evolution of resource becomes a heavy burden when computing the evolution of a process.

The process algebra **SCRP**, Synchronous Calculus of Resource Processes, introduced in (Pym & Tofts 2006, Pym & Tofts 2007, Collinson, Pym & Tofts 2007), provides the beginnings of an approach to addressing these issues. In **SCRP**, resources are taken to be first-class citizens along with processes. That is, a model consists of a complex process together with some resources. The notion of resource used is closely related to the resource-semantics of the **BI**-family of logics, (O'Hearn & Pym 1999, Pym 1999, Pym, O'Hearn & Yang 2004, Pym 2002), that have enjoyed a good deal of attention in recent years, particularly in the guise of the program logic known as Separation Logic, (Ishtiaq & O'Hearn 2001, Reynolds 2002, O'Hearn 2007). Such logics have variants of standard logical connectives that are often useful for internalizing statements about resource usage. The modelling approach which **SCRP** is intended to support is reflected in, and begins with, its treatment of resource. Specifically, it is hypothesized that the following properties of resource are basic:

- A basic collection of resource elements, including a zero element;
- A notion of combination of resource elements; and
- A notion of comparison of resource elements.

These properties are captured mathematically as a preordered, (initially) commutative monoid, ($\mathbf{R}$, $\circ$, e, $\sqsubseteq$), satisfying various algebraic laws, including a functoriality condition for the product relative to the order.

The basic judgement in **SCRP** is the evolution of a process relative to a collection of resources,

$$R, E \xrightarrow{a} R', E',$$

where the resource $R' = \mu(a, R)$ is determined by a *modification function*, μ defined on pairs of actions and resources. The basic judgement, for action prefix, then has the form

$$\frac{}{R, a : E \xrightarrow{a} \mu(a, R), E}.$$

As well as straightforward non-deterministic sum, **SCRP** admits a synchronous concurrent product (recall the generality of the synchronous product (Milner 1983, Simone 1985)) which requires the monoidal product on resources; roughly

$$\frac{R, E \xrightarrow{a} R', E' \quad S, F \xrightarrow{b} S', F'}{R \circ S, E \times F \xrightarrow{ab} R' \circ S', E' \times F'}.$$

There is also a hiding operator; roughly,

$$\frac{R \circ S, E \xrightarrow{a} R' \circ S', E'}{R, \nu S.E \xrightarrow{a} R', \nu S'.E'},$$

in which part of the initial resource is bound locally to the process. This construct allows, among other things, **SCRP** to recover concepts that are expressible using the restriction combinator of SCCS.

SCRP's use of the same structure for resources as taken in bunched logics suggests the possibility of a logic, in the style of Hennessy-Milner logic, characterizing the combinatorial structure of the calculus. The semantic judgement of this logic, called **MBI**, takes the form

$$R, E \models \phi,$$

which is read as 'the property ϕ holds of the process E relative to resources R' or 'ϕ is true for the system R, E'. Such a logic can be thought of as providing a language **MBI** for specifying and

verifying systems expressed, using **SCRP**, as assemblies of resources and processes. One valuable and useful consequence of this set-up is that the **MBI**-language leads to a logical characterization of the synchronous product; roughly, we get

$$R, E \models \phi_1 * \phi_2$$

iff there exist R_1 and R_2 such that $R_1 \circ R_2 = R$ and E_1 and E_2 such that $E_1 \times E_2 \sim E$, where $\sim$ is an appropriate notion of bisimulation, such that

$$R_1, E_1 \models \phi_1 \quad \text{and} \quad R_2, E_2 \models \phi_2.$$

The hiding construct is characterized in a similar way using a quantifier (see Section 3.5). The presence of these decompositions emphasizes the value of the two-language (algebraic and logical) approach to process calculus and modelling.

The process algebra **SCRP** and its logic **MBI** were introduced in three papers (Pym & Tofts 2006, Pym & Tofts 2007, Collinson et al. 2007), the last of these correcting an error in the first two. The intention of these papers was to establish the core ideas of the calculi and to demonstrate their practical effectiveness as a foundation for systems modelling in the spirit of Demos. Those papers did not, however, address a number of important theoretical issues relating to the metatheories of the calculi and the structures of the spaces over which they are defined. In this paper, we give a thorough investigation of these issues and show that the calculi can be significantly improved by making a number of small changes to the set-up.

This paper presents further developments, and a consolidation, of the ideas introduced in (Pym & Tofts 2006, Pym & Tofts 2007, Collinson et al. 2007). We present a new calculus **SCRPr** and logic with several important technical refinements that lead to better theoretical properties. The first refinement is to pick out a new kind of (bi)simulation relation, written $\sim$, which is a congruence. Certain identities for $\sim$ are seen to depend critically upon structural properties of the operational semantics. The simulation leads to a logic of system properties and part of a suitable Hennessy-Milner theorem. The third contribution is to introduce a proof system for (the propositional fragment of) the modal logic **MBI**. This is particularly important as the general model-checking problem for **MBI** is hard. The combination of substructural connectives means that the soundness result for the logic is non-trivial: indeed, it depends upon the Hennessy-Milner theorem. We study the way in which additional structure on resources is connected to the fundamental design of such process and logical calculi. In particular, we set-up an intuitionistic version of the logic on an ordered state-space. We study asynchrony and value-passing in **SCRP**-like calculi. Finally, we sketch a semantics of a simple programming languages and heap-manipulating commands.

Section 2 of this paper sets up the process calculus, its simulation relations and develops the algebraic theory. Section 3 gives an account of the logical calculi, the associated Hennessy-Milner theorem and the soundness result. Section 4 investigates calculi in the presence of ordered structure on resource and an intuitionistic logic. Section 5 gives extensions of this calculus to treat asynchrony and value-passing, sketches the development of a parallel programming language with variable-assignment. Section 6 discusses work in progress on extending and applying the calculus.

Finally, the reader should be aware that certain words used in this paper will have more than one technical meaning: for example, the words model and simulation. This unavoidable clash comes about because we are drawing upon the traditions of mathematical logic and theoretical computer science as well as those of applied mathematics. We hope that the reader will not have any difficulty in understanding what is meant in each context.

2 A Synchronous Calculus of Resources and Processes

The process algebra **SCRP** was sketched, along with its associated logic (**MBI**) and various properties, in (Pym & Tofts 2006, Pym & Tofts 2007, Collinson et al. 2007). In this section, we give a mature presentation of a family of systems, collectively known as **SCRP**, along with their

key technical properties. We make (ad-hoc) naming distinctions between the **SCRP** variants as necessary. In particular, we present a calculus with theoretical properties that are significantly improved relative to those of the system sketched in (Pym & Tofts 2006, Pym & Tofts 2007, Collinson et al. 2007). We also make a more detailed study of (bi)simulation relations and their corresponding algebraic theories. Indeed, we draw attention to a notion of (bi)simulation that was not presented in the earlier work and which is essential for the logical work that follows.

2.1 The Process Calculus

We now present a process calculus **SCRPr** that is a better-behaved, although less general, variant of the calculus **SCRP** presented in (Pym & Tofts 2006, Pym & Tofts 2007, Collinson et al. 2007). The set-up of these calculi assumes the provision of certain additional data pertaining to some semantic structure $(\mathsf{Act}, \mathbf{R}, \mu, \nu)$ over which we are working. Thus we should properly refer to the calculus as $(\mathsf{Act}, \mathbf{R}, \mu, \nu) - \mathbf{SCRPr}$. In this paper, however, we suppress the prefix as, at every stage, we work with a fixed such structure.

We assume a commutative monoid, Act, of *actions*. Just as in standard process algebra, these actions correspond to the events of a system. We reserve the letters a, b, c for actions. Composition is written by juxtaposition and the unit action is written 1. We do not, for now, need to assume that this monoid is generated from a collection of atomic actions (usually called particles). Nor do we need any assumptions about the cardinality of Act.

In this paper, we shall often work with partial functions. We use the standard notations $R\downarrow$ and $R\uparrow$ to mean that an expression R is, respectively, defined or undefined. We also make use of Kleene-equality between expressions: the left-hand-side of an equality, $L \simeq R$, is defined if and only if the right-hand-side is defined, and when defined they are equal.

A *resource monoid* is a structure $\mathbf{R} = (\mathbf{R}, \circ, e, \sqsubseteq)$. We do not use a separate notation to distinguish the carrier set $\mathbf{R}$ from the structure. The structure has a preorder $\sqsubseteq$, a partial, binary operation $\circ$ and has a distinguished element e. The operation $\circ$ satisfies monoid associativity and commutativity axioms up to Kleene equality. The unit of $\circ$ is e. Composition with this unit is always defined. Therefore, the structure satisfies the unit axiom for a commutative monoid up to actual equality. Resource monoids are further required to satisfy the *bifunctoriality condition*:

$$R \sqsubseteq R' \quad \text{and} \quad S \sqsubseteq S' \quad \text{and} \quad R' \circ S'\downarrow \quad \text{implies} \quad R \circ S\downarrow \quad \text{and} \quad R \circ S \sqsubseteq R' \circ S' \quad (1)$$

for all R, R', S, S' in $\mathbf{R}$.

Some simple examples of resource monoid are:

1. The natural numbers with addition, zero, and their usual order
2. The real numbers with addition, zero, and their usual order.
3. The set $\{0, 1\}$ with an operation $+$ such that $0 + 0 = 0$, $0 + 1 = 1 = 1 + 0$, $1 + 1\uparrow$ and the discrete order (equality).
4. A powerset $\mathcal{P}(L)$ of some set L. The composition is non-overlapping union: for any subsets X and Y of L, the composite $X \circ Y$ is defined just when $X \cap Y = \emptyset$, and when defined $X \circ Y = X \cup Y$. The unit is the empty set. The order is the discrete order.

The first example above is closely related to the kind of basic resources found 2̣k. The third can be used as a kind of semaphore resource. The fourth is that which lies at the root of Separation Logic (Ishtiaq & O'Hearn 2001, Reynolds 2002).

Note that the order-dual (obtained by reversing the order) of a resource monoid is not necessarily a resource monoid. Instead, it satisfies the property:

$$R \sqsubseteq R' \quad \text{and} \quad S \sqsubseteq S' \quad \text{and} \quad R \circ S\downarrow \quad \text{implies} \quad R' \circ S'\downarrow \quad \text{and} \quad R \circ S \sqsubseteq R' \circ S' \quad (2)$$

for all R, R', S, S' in $\mathbf{R}$. Define a resource monoid to be *special* when its dual is also a resource monoid, that is, when both (1) and (2) are satisfied. For the purposes of the rest of this paper the order $\sqsubseteq$ should be taken to be equality, except for Subsection 3.1 and Section 4.

A binary relation, $\Subset$, between resources is important in the development of **SCRP**. Let R and S be resources. Say that S *piggybacks* on R, and write $S \Subset R$, if, for every resource T, if $R \circ T$ is defined then $R \circ S \circ T$ is defined. Intuitively, $S \Subset R$ if, whenever R is consistent with any T then so is $R \circ S$. This predicate is used to ensure a well-behaved hiding operation. Note that if $S \Subset R$ then $R \circ S \downarrow$, and also that the relation $\Subset$ is total if and only if the composition operation is total.

We now set-up a function describing how actions transform resources. A *modification* is a partial function $\mu : \mathsf{Act} \times \mathbf{R} \longrightarrow \mathbf{R}$ satisyfing two *coherence* conditions:

1. $\mu(1, R) = R$ for all $R \in \mathbf{R}$
2. if $\mu(a, R)$, $\mu(b, S)$ and $R \circ S$ are all defined then $\mu(ab, R \circ S)$ and $\mu(a, R) \circ \mu(b, S)$ are both defined and $\mu(ab, R \circ S) = \mu(a, R) \circ \mu(b, S)$ holds.

Consider the resource monoid consisting of the natural numbers discussed above. Suppose that the action monoid is freely generated from a single action i, so that every action can be represented in the form i^m for some unique integer $m \geq 0$. As a simple example of a modification function consider:

$$\mu(i^m, n) = m + n$$

for all natural numbers m and n. Then i is incrementation.

We assume a total operation called *hiding*, $\nu : \mathbf{R} \times \mathsf{Act} \longrightarrow \mathsf{Act}$, that takes any resource R and any action a and produces an action $\nu R.a$. The precise form of this operation is unimportant for most of the development that follows, and a number of possibilities exist. One such possibility is given in (Pym & Tofts 2006, Pym & Tofts 2007, Collinson et al. 2007) under the assumption that the action monoid is generated as a free monoid from a set of atomic actions. Reserve the letter α for atomic actions. Any action a may be written uniquely (up to re-ordering) as a product $a = \prod\{\alpha_i \mid i \in I\}$ for some finite set I. Then we may take

$$\nu S.a = \prod\{\alpha_i \mid i \in I \ \&\ \mu(\alpha_i, S) \uparrow\} \tag{3}$$

and recall that the product of an empty set of actions gives the identity action. The intutition behind this is that this resultant action $\nu S.a$ consists of precisely those atomic constituent actions α_i of a that play no role in the evolution of a process in the resource environment S, since $\mu(\alpha_i, S)$ is not defined, and so the actions that fire when given S are hidden. The hiding processes introduced below only evolve along such actions $\nu S.a$ and thus any of their atoms that are enabled by S may not be externally observed. This is further clarified by the operational semantics of hiding processes described below.

We assume a countable collection of *process variables*, for which the letter X is reserved. *Processes* are formed according to the grammar

$$E ::= \mathsf{0} \mid X \mid a : E \mid \sum_{j \in J} E_j \mid E \times E \mid \nu R.E \mid \mathit{fix}_i \mathsf{X}.\mathsf{E} \ ,$$

where $\mathsf{0}$ is a special process, X is a process variable, a is an action, J is an arbitrary index set, R is a resource, X is an n-tuple of process variables, E is an n-tuple of processes and $1 \leq i \leq n$. The letters E, F, G are reserved for processes, and the letters E, F, G for tuples of processes.

The *fix* operator binds occurrences of process variables within processes. It will occasionally be necessary to distinguish processes that contain no free variables (sometimes called *agents*) from the more general process expressions that exist in the language. Let Agents be the set of agents and Proc be the set of processes. Let X_i be the ith component of any tuple of process variables X, E_i be the ith component of any tuple of process expressions E of the same length, then $F[\mathsf{E}/\mathsf{X}]$ is the process formed by the (capture-avoiding) substitution of each of the n components of E for the corresponding variable of X that is free in F. Similarly, there is substitution $\mathsf{F}[\mathsf{E}/\mathsf{X}]$ for process variables in a tuple F. The expression $\mathit{fix}_i \mathsf{X}.\mathsf{E}$ means $(\mathit{fix}\mathsf{X}.\mathsf{E})_i$, the ith component of the tuple $\mathit{fix}\mathsf{X}.\mathsf{E}$, where E and X may be of different length. We use brackets, (), to disambiguate processes

Prefix	$\dfrac{}{R, a : E \xrightarrow{a} \mu(a,R), E}$	$(\mu(a,R)\downarrow)$
Sum	$\dfrac{R, E_j \xrightarrow{a} R', E'}{R, \sum_{j\in J} E_j \xrightarrow{a} R', E'}$	$(j \in J)$
Product	$\dfrac{R, E \xrightarrow{a} R', E' \qquad S, F \xrightarrow{b} S', F'}{R \circ S, E \times F \xrightarrow{ab} R' \circ S', E' \times F'}$	$(R \circ S \downarrow)$
Hide	$\dfrac{R \circ S, E \xrightarrow{a} R' \circ S', E'}{R, \nu S.E \xrightarrow{\nu S.a} R', \nu S'.E'}$	$(\mu(\nu S.a, R) = R'\downarrow \ \& \ S \circledcirc R)$
Fix	$\dfrac{R, E_i[\mathit{fix}\mathsf{X}.\mathsf{E}/\mathsf{X}] \xrightarrow{a} R', E'}{R, \mathit{fix}_i\mathsf{X}.\mathsf{E} \xrightarrow{a} R', E'}$	

Figure 1: **SCRPr** transitions

in the absence of their construction trees. The unit process $\mathsf{1}$ is defined to be $\mathit{fix}X.(1 : X)$. Given a sequence of the form $s = b_1/a_1, \ldots b_n/a_n$, the notation $E[s]$ stands for the process formed by the substitution of actions b_i for the actions a_i occurring in E.

These processes should appear familiar, with the exception of $\nu R.E$, to those who work on process algebra. The calculus is intended to be a close relative of SCCS. Thus $a : E$ is a process with an action *prefix*, $\sum_{j\in J} E_j$ is a *sum*, $E \times F$ is a *(synchronous) product*, and $\mathit{fix}_i\mathsf{X}.\mathsf{E}$ is the ith component of the tuple of processes $\mathit{fix}\mathsf{X}.\mathsf{E}$ defined as a *fixed point*. The term $\nu R.E$ is a *hiding* process and is a resource-based form of restriction operation. We often write binary sums using the infix notation $E + F$.

A *state* is a pair consisting of a resource and a process. Thus $\mathsf{States} = \mathbf{R} \times \mathsf{Proc}$ is the set of all states. Define the set $\mathsf{CStates}$ of *closed states* to consist of those states with an agent as the process component. If E is a process and R is any resource then we say that R, E is an *E-state*.

The operational behaviour of processes is defined by a labelled family of transition relations

$$\xrightarrow{a} \ \subseteq \ \mathsf{States} \times \mathsf{States}$$

indexed by $a \in \mathsf{Act}$. The family is defined recursively using the derivation rules of Figure 1. This describes how states evolve. Notice that the evolution of prefix processes with a given resource is completely determined by the modification μ. Product processes share out the globally available resource in such a way as to enable the components to evolve synchronously; the fact that the resulting composite $R' \circ S'$ appearing in the rule is well-defined follows as an immediate consequence of the definition of modifications and Lemma 2 below. Essentially, a state $R, \nu S.E$ featuring a hiding process evolves along $\nu S.a$ when the underlying process E evolves along a given the resource formed $R \circ S$ by unpacking the hidden resource S.

Nondeterministic behaviour of processes is introduced into processes through the presence of sums. In most process caclculi sums are the only source of nondeterminism. In contrast, in **SCRP**-calculi nondeterminism can also be introduced by instances of the product and hiding constructors. An arbitrary resource R can have many possible decompositions (R_1, R_2) such that $R = R_1 \circ R_2$. In such situations a state of the form $R, E_1 \times E_2$ may make transitions induced by transitions of pairs of states $((R_1, E_1), (R_2, E_2))$ for each decomposition (R_1, R_2). Nondeterminism is induced by hiding since, looking at the rule for hiding in Figure 1, there can be many possible resources S' such that the premise of the rule is true.

Define a *(state) derivative* of a state R, E to be a state R', E' that is reachable via a (possibly null) sequence of transitions. An *immediate* derivative is a state that can be reached using a single

instance of a transition. A *proper derivative* is a derivative arising from a non-empty sequence of transitions. A *derivative* of a process E is any E' such that there are some R and R' such that R', E' is a derivative of R, E.

The system **SCRPr** is a restriction of the more general calculus **SCRP** originally suggested in (Pym & Tofts 2006, Pym & Tofts 2007, Collinson et al. 2007). The differences between the two systems are as follows.

1. The definition of *modification* for **SCRP** is more liberal. The equality in the second clause in the definition of modification given above is replaced by a Kleene-equality.
2. The piggybacking condition is not used in **SCRP**. In particular, it does not appear in the side-condition of the operational rule for hiding processes.

Clearly, **SCRP** applies to a wider range of situations than **SCRPr**. However, **SCRPr** has better operational behaviour and a closer correspondence with the logic **BI**.

2.2 Structural Properties of SCRP Systems

The transition systems associated with **SCRPr** systems have a number of important properties. The following lemma is an immediate consequence of the operational semantics and is established by induction on derivations.

Lemma 1. *If R, E is a state and E is an agent then the process component of every state-derivative is an agent. In other words, the subspace* CStates *is closed under transitions.*

The evolution of the resources is entirely deterministic in the chosen action.

Lemma 2. *If $R, E \xrightarrow{a} R', E'$ then $R' = \mu(a, R)$.*

The proof of the lemma is an easy induction over derivations, making essential use of the coherence conditions on μ and the explicit and implicit side-conditions on derivation rules.

The coherence properties for modifications leads immediately to a result about the extensibility of resources via composition.

Lemma 3. *If μ is a modification then it satisfies the following property,*

$$\textit{if} \quad \mu(a, R)\downarrow \quad \textit{and} \quad R \circ S \downarrow \quad \textit{then} \quad \mu(a, R \circ S) = \mu(a, R) \circ S \downarrow$$

for all resources R, S and actions a. We call this the simple-extension *property for resources.*

The existence of any transition from a state is closed under composition with further resource. That is, there is a simple-extension property for transitions as well as modifications.

Proposition 4. *Let μ be a modification. The following property holds:*

$$\textit{if} \quad R, E \xrightarrow{a} \mu(a, R), E' \quad \textit{and} \quad R \circ S \downarrow \quad \textit{then} \quad R \circ S, E \xrightarrow{a} \mu(a, R) \circ S, E'$$

for all resources R, S, processes E, E' and actions a.

The proof of the above lemma is an easy induction over derivations of transitions. Lemma 3 establishes the base case (Prefix) and the piggybacking side-condition ensures that the induction passes across the Hide case.

By an analogy with situations that arise in Proof Theory, this can be seen as establishing an *admissible rule*,

$$\frac{R, E \xrightarrow{a} \mu(a, R), E'}{R \circ S, E \xrightarrow{a} \mu(a, R) \circ S, E'} \qquad (R \circ S \downarrow)$$

for all suitable R, S, E, E', a. This is rather like a weakening rule for the resource component of a state. Many other structural rules can, of course, be considered, and their admissibility is linked to the structure of the underlying resource monoid. When such rules are not admissible (for example, if we forget about piggybacking when hiding) then one may choose to include them explicitly in the calculus as the algebraic and logical properties of these kinds of calculi sometimes rely critically upon their presence.

2.3 Bisimulation

It is usual to have a notion of equality for process terms that treats processes with the same behaviour as equal. The standard notion is that of an equivalence relation called *bismulation*. For calculi in the **SCRP** family the situation is a little delicate. Firstly, there is a question about where the equivalence should live. On one-hand simulation is usually defined via transition structures, see for example (Milner 1983, Popkorn 1994). This suggests an equivalence between states, that is, between systems or models. On the other-hand an equivalence of processes is probably more useful than an equivalence of states, as this is the part of a system in which it is most natural to exercise control. Furthermore, the compositional nature of the systems we build resides primarily in the process part, and we frequently want to know that two processes behave in the same way in any given resource context. We explore these issues carefully for the calculus **SCRPr**, and with an eye on logical equivalence of processes as well as behavioural equivalence of processes and states.

Define the *local equivalence* relation, $\approx$, to be the largest binary relation on closed states such that the following condition holds: if $R, E \approx S, F$ then

1. if there is a transition $R, E \xrightarrow{a} \mu(a, R), E'$ for any E' then there is transition $R, F \xrightarrow{a} \mu(a, R), F'$ with $\mu(a, R), E' \approx \mu(a, R), F'$ for some F'
2. if there is a transition $R, F \xrightarrow{a} \mu(a, R), F'$ for any F' then there is a transition $R, E \xrightarrow{a} \mu(a, R), E'$ with $\mu(a, R), F' \approx \mu(a, R), E'$ for some E'
3. $R = S$.

The relation $\approx$ is al states of by substitution: for any states R, E and S, F we define $R, E \approx S, F$ iff $R, E[\mathsf{G}/\mathsf{X}] \approx S, F[\mathsf{G}/\mathsf{X}]$ for all m-tuples of agents G, where X is an m-tuple representing the set of free variables of E and F.

This relation $\approx$ is almost that considered in (Pym & Tofts 2006, Pym & Tofts 2007, Collinson et al. 2007). Note however, that we have defined the relation initially for agents rather than for arbitrary process expressions, and that this gives a slightly smaller relation.

Fundamentally, this relation starts from the view that agents should be considered equivalent whenever they have the same behaviour given the same resources. In (Pym & Tofts 2007) this relation was shown to be intimately connected to a denotational semantics of **SCRP** that uses synchronization trees. Indeed, the relation on closed states looks very much like the standard notion of bisimulation for transition structures, see (Popkorn 1994) for example. In view of Lemma 2 the main difference is the insistence that the resource components of the states under comparison are identical. Clearly, local equivalence on closed states is contained in the standard kind of bisimulation for transition systems on those states. Consequently, a modal logic of system properties may be expected. However, the local equivalence relation fails to be a congruence as it is not respected by the product constructor for processes — the references (Pym & Tofts 2006, Pym & Tofts 2007) contain an error on this point.

Example 5. Consider the resource monoid $(\mathbb{N}, +, 0)$ consisting of the natural numbers with addition. Let Act be the monoid of actions freely generated from a single action d. Every element of Act has the form d^i for some unique $i \in \mathbb{N}$. There is a modification defined by

$$\mu(d^i, n) = \begin{cases} n - i & \text{if } i \leq n \\ \uparrow & \text{otherwise} \end{cases}$$

for all $i, n \in \mathbb{N}$. Let E be the process $d : \mathsf{0}$, let F be $\mathsf{0}$, and let G be $1 : \mathsf{0}$.

Then $0, E \approx 0, F$ holds, since neither state makes any transitions. However, $1, E \times G \approx 1, F \times G$ does not hold, since $1, E \times G$ has a transition while $1, F \times G$ does not.

We write $E \approx F$ whenever $R, E \approx R, F$ for all resource R. This relation is a congruence. It fails, however, to be closed under transitions in the sense that we can have $E \approx F$ and $R, E \xrightarrow{a} \mu(a, R), E'$ for some E' and R, but no corresponding F' with $R, F \xrightarrow{a} \mu(a, R), F'$ and $E' \approx F'$. This means

that the relation $\approx$ does not interact well with the modal logic we introduce below (the more natural relation for which seems to be $\approx$ on states). We believe it would be worthwhile to explore a version of $\approx$ on states in which the resource components need not be identical. This topic, which we suspect may quite difficult, is suggested on the one hand by a desire to compare *systems* and on the other by the general notion of bisimulation in modal logic (Popkorn 1994).

There is a natural alternative relation which is a congruence but which is defined initially on agents rather than states. Define the *global equivalence* relation, $\sim$, to be the largest relation binary on agents such that, whenever $E \sim F$ holds:

1. if $R, E \xrightarrow{a} \mu(a, R), E'$ for any R, E' then there is some F' with $R, F \xrightarrow{a} \mu(a, R), F'$ and $E' \sim F'$
2. if $R, F \xrightarrow{a} \mu(a, R), F'$ for any R, F' then there is some E' with $R, E \xrightarrow{a} \mu(a, R), E'$ and $F' \sim E'$.

The relation $\sim$ is then extended to all tuples of processes by substitution: for any n-tuples of processes E and F we define $\mathsf{E} \sim \mathsf{F}$ iff $E_i[\mathsf{G}/\mathsf{X}] \sim F_i[\mathsf{G}/\mathsf{X}]$ for all $1 \leq i \leq n$ and all m-tuples of agents G, where X is any m-tuple containing the free variables of E and F with each listed exactly once. The global equivalence is lifted to states by taking $R, E \sim R, F$ to hold just when $E \sim F$, for all E, F and R.

The global equivalence is intimately related to the logical language **MBIc** based on resource semantics that we develop in Section 3. Notice that, for a local equivalence $E \approx F$, it is enough to compare derivatives of states R, E and R, F for all initial resources R. In contrast, for a global equivalence, one must also compare states of the form S, E' and S, F', that are resource-perturbations of derivatives of the form R', E' and R', F' of R, E and R, F for any R. That is to say, for global equivalence we cannot just compare derivatives, we must also compare states that arise by perturbing the resource component of such derivatives.

Proposition 6. *The relation $\sim$ on processes is a congruence. That is, it is an equivalence relation which is respected by the process constructors. In particular, if $E \sim F$ between processes and $\mathsf{E} \sim \mathsf{F}$ between n-tuples of processes then for any action a, process G, resource S, n-tuple X and index i:*

$$a : E \sim a : F$$

$$E + G \sim F + G \qquad E \times G \sim F \times G$$

$$\nu S.E \sim \nu S.F \qquad \mathit{fix}_i \mathsf{X}.\mathsf{E} \sim \mathit{fix}_i \mathsf{X}.\mathsf{F} \ .$$

Proof. The reflexivity and symmetry and transitivity of the relation are all straightforward to observe.

The proofs of the equalities stated above are also quite standard. For example, consider the set of pairs of agents $A = \{(E \times G, F \times G) \mid E \sim F\}$. Consider some pair $(E \times G, F \times G) \in A$ Consider any R and suppose that there is a transition $R, E \times G \xrightarrow{a} \mu(a, R), E_1$ for some E'. Then E_1 must be of the form $E' \times G'$ and there must be some actions b, c with $a = bc$ and some resources S, T with $R = S \circ T$ such that $S, E \xrightarrow{b} \mu(b, S), E'$ and $T, G \xrightarrow{c} \mu(c, T), G'$. Since $E \sim F$ there must be a transition $S, F \xrightarrow{b} \mu(b, S), F'$ for some F' with $E' \sim F'$. Hence there is a transition $R, F \times G \xrightarrow{a} \mu(a, R), F' \times G'$ and $(E' \times G', F' \times G') \in A$. The symmetry of the components of the elements of set A then shows that $E \times G \sim F \times G$. The result then lifts to processes immediately.

Just as in Proposition 4.6 of (Milner 1983) the congruence property for the fixed point relies upon the way that the relation has been lifted from agents to processes and uses the preceding equalities. □

We omit the proofs of the following two lemmas, these being routine verifications. The first of these is established by showing that $\sim$ on closed states is closed under the conditions for a local equivalence.

Lemma 7. *The global equivalence relation, $\sim$, on closed states is contained in the local equivalence relation, $\approx$, on such states.*

We shall see that this is important for the modal logic **MBIc**. The relation $\approx$ on states is not a congruence and so cannot be contained in the relation $\sim$ on states.

We use the notations $\lesssim$ and $\lessapprox$ in the standard way for the asymmetric variants of $\sim$ and $\approx$. Thus, for example, $R, E \lessapprox R, F$ just if, whenever the agent R, E makes some transition into some R', E', then the agent R, F makes a transition along the same action and into a state R', F' with $R', E' \lessapprox R', F'$. We now return to the question of algebraic identities.

Lemma 8. *Various simple equalities and inequalities hold for the relation $\sim$ on processes:*

$$E \times F \sim F \times E \qquad E \times (F \times G) \sim (E \times F) \times G$$

$$E + F \sim F + E \qquad E + (F + G) \sim (E + F) + G$$

$$E \times 0 \sim 0 \qquad E \times 1 \sim E$$

$$E + 0 \sim E \qquad E + E \sim E$$

$$E \times (F + G) \sim (E \times F) + (E \times G) \qquad \nu S.(E + F) \sim (\nu S.E) + (\nu S.F)$$

$$(a : E) + (a : F) \lesssim a : (E + F) \qquad (a : E) \times (b : F) \lesssim (ab) : (E \times F)$$

for all processes E, F and G.

Proof. Again, it suffices to show the result on agents. The rest is straightforward verification using the fact that $\sim$ on closed states is defined to be the largest relation closed under the given conditions. It is important to note that the property $E \times 1 \sim E$ needs the simple-extension property for transitions (Proposition 4). □

All of the simple algebraic identities from Lemma 8 hold with $\sim$ replaced by $\approx$ throughout: this follows immediately from the fact that $\sim$ is contained in $\approx$.

Further inequalities may well hold for specific choices of resource monoid, modification, hiding and action set. For example, well-behaved hiding on actions leads to better-behaved hiding processes.

Lemma 9. *If $\nu(S \circ T).a = \nu S.\nu T.a$ for all actions a and any resources S and T, then the relation $\nu S.\nu T.E \lesssim \nu(S \circ T).E$ holds for any process E.*

However, the following two properties *do not* hold in general:

1. $R, a : (E + F) \lessapprox R, a : E + a : F$
2. $R, (ab) : (E \times F) \lessapprox R, (a : E) \times (b : F)$.

Simple counterexamples exist to each. Of course, if $R, F \not\lessapprox R, E$, then $R, F \not\lesssim R, E$ for any R, E, F. To see that the first point does not hold consider any pair of states $R, a : E + a : F$ and $R, a : (E + F)$ with $a = 1$, $E = 1 : \mathsf{0}$ and $F = \mathsf{0}$. For the second consider the trivial resource monoid $\mathbb{N}$ with addition, action monoid generated from the set $\{a, b\}$ and modification satisfying $\mu(a^m b^n, p) = p + n - m$ if $m \leq p + n$ and that is undefined otherwise. Then $\mu(ab, 0)$ is defined but $\mu(a, 0)$ is undefined, so that there is an ab-transition of the prefix process $ab : (\mathsf{0} \times \mathsf{0})$ but no transition of the product $(a : \mathsf{0}) \times (b : \mathsf{0})$.

The following lemma gives an important representation of any state.

Lemma 10. *Let E be any process. For any resource R we may write*

$$R, E \approx R, \sum \{a : E' \mid R, E \xrightarrow{a} \mu(a, R), E'\}$$

and furthermore,

$$\sum\{a : E' \mid \forall R.\ R, E \xrightarrow{a} \mu(a, R), E'\} \lesssim E \lesssim \sum\{a : E' \mid \exists R.\ R, E \xrightarrow{a} \mu(a, R), E'\}$$

holds.

Proof. For the first part, the transitions and immediate derivatives on the left are precisely the same as those on the right. □

This result specializes in the case of a product of processes.

Lemma 11.

$$\begin{aligned} R, E_1 \times \ldots \times E_n \quad \approx \quad & R, \sum\{(a_1 \cdots a_n).(E'_1 \times \ldots \times E'_n) \mid \exists R_1, \ldots, R_n. \\ & R = R_1 \circ \ldots \circ R_n \ \&\ \forall 1 \leq i \leq n.\ R_i, E_i \xrightarrow{a_i} \mu(a_i, R_i), E'_i\ \} \end{aligned}$$

Proof. The coherence conditions on modifications guarantee that a transition on the left is a transition on the right into the same derivative. The indexing set on the right guarantees that there are no more transitions on the right. □

We may expand out hiding processes in a similar way.

Lemma 12.

$$R, \nu S.E \approx R, \sum\{(\nu S.a) : (\nu S'.E') \mid R \circ S, E \xrightarrow{a} R' \circ S', E' \ \&\ \mu(\nu S.a, R) = R' \ \&\ S \Subset R\ \}$$

In the special case of a prefix

$$R, \nu S.(a : E) \approx R, \sum\{(\nu S.a) : (\nu S'.E') \mid \mu(a, R \circ S) = \mu(\nu S.a, R) \circ S' \ \&\ S \Subset R\ \}\ .$$

Proof. For any given resource the derivatives of the left-hand-side coincide exactly with the derivatives of the right-hand-side. □

The preceding results can be combined to give the *local expansion theorem* for states in the synchronous calculus.

Theorem 13.

$$\begin{aligned} R, \nu S.(E_1 \times \ldots \times E_n) \quad \approx \quad & R, \sum\{\nu S.(a_1 \ldots a_n).\nu S'.(E'_1 \times \ldots \times E'_n) \mid S \Subset R\ \& \\ & \exists R_1, \ldots, R_n.\ R \circ S = R_1 \circ \ldots \circ R_n\ \& \\ & \mu(a_1 \ldots a_n, R \circ S) = \mu(\nu S.(a_1 \ldots a_n), R) \circ S'\ \& \\ & \forall 1 \leq i \leq n.\ R_i, E_i \xrightarrow{a_i} \mu(a_i, R_i), E'_i\} \end{aligned}$$

Proof. Notice that if $S \Subset R$ does not hold then the sum on the right is empty and this gives the process $\mathtt{0}$. Once again, a transition to a derivative on the left exists if and only if it exists on the right. The result generalizes easily to a form with multiple hidings (rather than just one) outermost in the process term. □

It is rather unsatisfactory to have these results stated only for $\approx$ given that it fails to be a congruence. One would like to have an expansion theorem for processes, preferably using $\sim$. However, this would seem not to be possible in the general case, even using $\approx$. The expansion relies critically on the resource at which the expansion is performed. In particular, the second part of Lemma 10 cannot be tightened to make the second of the inequalities an equality, because there are processes E, E' and resources R and S with $R, E \xrightarrow{a} \mu(a, R), E'$ and $\mu(a, S) \downarrow$ such that there is no transition $S, E \xrightarrow{a} \mu(a, S), E$. Most of the results above have analogues in **SCRP**.

The relation $\sim$ does not feature in the papers (Pym & Tofts 2006, Pym & Tofts 2007) as the relation $\approx$ was believed to be appropriate for all purposes. Subsequently, a small hole in the proof (contained in those papers) of the Hennessy-Milner theorem for the logic **MBI** was detected. A rather subtle counterexample was produced — this is a relative of Example 33 below. A careful analysis of the counterexample led to the identification of $\sim$. This relation is sufficiently conservative to recover enough of the HM-theorem for a suitably modified interpretation of **MBIc**. These changes were then published in the erratum (Collinson et al. 2007).

2.4 Specifying Modifications

General methods are needed for specifying modifications. It is not always feasible to specify the modification function individually at all actions and all resources. Furthermore, when such a specification is made, the function defined must be explicitly checked to satisfy the coherence conditions.

One method that can often be employed is to specify the modification on atomic actions. Under suitable conditions this gives rise to a unique coherent modification. The conditions we use involve the pre-order that arises from the composition. For the purposes of this section assume that we are working with an action monoid that is freely generated from some set of atomic actions.

We will often suppose that we are working with a resource monoid $\mathbf{R}$ with *cancellation*, that is, the partial function $S \circ (-) : \mathbf{R} \longrightarrow \mathbf{R}$ is injective for every S. In other words, for any R and S, if whenever $R = S \circ T$ for some T, that T is unique. We usually write T as $R - S$. Define a resource monoid to be *good* when it has cancellation and composition is total. Define the preorder $\sqsubseteq$ by

$$S \sqsubseteq R \qquad \Longleftrightarrow \qquad \exists T. \quad S \circ T = R$$

for all resources R and S.

Define a partial function $f : \mathbf{R} \longrightarrow \mathbf{R}$ to be *rooted* if:

1. there is a unique resource R_0, called the *root*, such that for all R, $f(R)$ is defined if and only if $R_0 \sqsubseteq R$.
2. if $f(R)$ is defined and $R \circ S$ is defined then $f(R \circ S) = f(R) \circ S$ is defined.

The following lemma is then immediate. Indeed, it characterizes rooted functions on good monoids.

Lemma 14. *For any rooted function on a good resource monoid*

$$f(R) = \begin{cases} f(R_0) \circ (R - R_0) & \text{if } R_0 \sqsubseteq R \\ \uparrow & \text{otherwise} \end{cases}$$

for all resources R.

Define an Act-indexed family of resources to be *consistent* if, for any two actions a and b,

$$R_{ab} = R_a \circ R_b$$

holds.

Lemma 15. *Every consistent family of resources on a good resource monoid satisfies:*

$$(R \circ S) - R_{ab} = (R - R_a) \circ (S - R_b)$$

for all a, b, R, S such that the right-hand-side is defined.

Proof. Consider the calculuation

$$\begin{array}{rcl} ((R - R_a) \circ (R - R_b)) \circ R_{ab} & = & (R - R_a) \circ (S - R_b) \circ R_a \circ R_b \\ & = & R \circ S \end{array}$$

for any given R, R, a, b. Then the uniqueness of $(R \circ S) - R_{ab}$ gives the result. □

Suppose that we have a family of resources R_α indexed by atomic actions. Extend this to a family indexed by all actions by taking

$$R_a = \bigcirc_{1 \leq i \leq n} R_{\alpha_i}$$

for all $a = \alpha_1 \ldots \alpha_n$, where $n = 0$ is the special case for the unit action, and $\bigcirc_{i \in I}$ is I-indexed resource composition for any finite set I. The proof of the following lemma is then a routine verification.

Lemma 16. *The* Act*-indexed family of resources generated (as above) from the family of resources indexed by atomic actions is consistent.*

We now return to the issue of functions specified at actions.

Proposition 17. *Suppose that we have a good resource monoid. Suppose that we have a family of rooted, partial functions* $\mu_\alpha : \mathbf{R} \longrightarrow \mathbf{R}$ *indexed by atomic actions* α*, and that the root of each* μ_α *is* R_α*. Then there is a unique modification* $\mu : \mathsf{Act} \times \mathbf{R} \longrightarrow \mathbf{R}$ *such that*

$$\mu(\alpha, R) \simeq \mu_\alpha(R) \tag{4}$$

for all atomic actions α *and all resources* R*. Note that the equality here is a Kleene-equality. Moreover, this satisfies*

$$\mu(a, R) = \begin{cases} \mu(a, R_a) \circ (R - R_a) & \text{if } R_a \sqsubseteq R \\ \uparrow & \text{otherwise} \end{cases} \tag{5}$$

for all actions a *and resources* R*, where the family* $(R_a \mid a \in \mathsf{Act})$ *is generated from the family of roots* R_α *indexed by atomic actions.*

Proof. For any atomic action α and resource R,

$$\mu_\alpha(R) = \begin{cases} \mu_\alpha(R_\alpha) \circ (R - R_\alpha) & \text{if } R_\alpha \sqsubseteq R \\ \uparrow & \text{otherwise .} \end{cases}$$

since μ_α is rooted. For every $n \geq 1$ take

$$\mu(\alpha_1 \ldots \alpha_n, R) = \begin{cases} \mu(\alpha_1, R_{\alpha_1}) \circ \ldots \mu(\alpha_n, R_{\alpha_n}) \circ (R - R_{\alpha_1 \ldots \alpha_n}) & \text{if } R_{\alpha_1 \ldots \alpha_n} \sqsubseteq R \\ \uparrow & \text{otherwise} \end{cases} \tag{6}$$

for every sequence of atoms $\alpha_1, \ldots, \alpha_n$ and resource R. It is straightforward, using Lemmas 15 and 16, to verify that this is coherent and satisfies equations (4) and (5). For uniqueness, observe that the coherence property requires equation (6). □

The proposition tells us how to construct a modification from a specification on atoms. In practical modelling, many resource monoids are indeed good and modifications are (implicitly) specified through a small family of rooted functions. This is the case, for example, in Demos. It is also connected to the enabling functions discussed in (Pym & Tofts 2006), which we shall return to in Section 4.

It can be useful to define a modification by specifying on atomic actions as above, but where each atom does not have a unique root, and when resource composition is not required to be total. We now show how to get a **SCRP**-modification in such a situation. This may not always be a **SCRPr**-modification; we only show that it satsisfies the Kleene-equality version of the second coherence condition.

For the remainder of this section assume that $\mathbf{R}$ is a resource monoid.

Lemma 18.

$$R - (S \circ T) \simeq (R - S) - T \simeq (R - T) - S$$

for all resources R, S, T*.*

Proof. Suppose $(R-(S\circ T))$ is defined. Then $(R-(S\circ T))\circ S\circ T = R$, so $(R-(S\circ T))\circ S = R-T$, and then $(R - (S \circ T)) \circ T = (R - T) - S$.

Supose $(R - T) - S$ is defined. Then $((R - T) - S) \circ S = R - T$ is defined, and so $((R - T) - S) \circ S \circ T = R$. Then by definition $((R - T) - S) = R - (S \circ T)$. □

Lemma 19. *If* $R \circ S$ *and* $R - T$ *are defined then* $(R \circ S) - T = (R - T) \circ S$ *is defined.*

Proof. The calculation $((R - T) \circ S) \circ T = ((R - T) \circ T) \circ S = R \circ S$ gives the result. □

Lemma 20. *If $(R - R_1)$, $(S - S_1)$ and $R \circ S$ are defined then*

$$(R - R_1) \circ (S - S_1) \simeq (R \circ S) - (R_1 \circ S_1) \ .$$

Proof. Suppose that $(R - R_1) \circ (S - S_1)$ is defined. Then $(R - R_1) \circ (S - S_1) \circ R_1 \circ S_1 = R \circ S$ is defined. So $(R - R_1) \circ (S - S_1) = (R \circ S) - (R_1 \circ S_1)$.

Suppose that $(R \circ S) - (R_1 \circ S_1)$ is defined. Then $((R \circ S) - (R_1 \circ S_1)) \circ (R_1 \circ S_1) = R \circ S$ is defined, and so $((R \circ S) - (R_1 \circ S_1)) \circ S_1 = (R \circ S) - R_1$ is defined. By Lemma 19, $((R \circ S) - (R_1 \circ S_1)) \circ S_1 = (R - R_1) \circ S$, so $(R \circ S) - (R_1 \circ S_1) = ((R - R_1) \circ S) - S_1$ is defined. Hence $(R \circ S) - (R_1 \circ S_1) = (R - R_1) \circ (S - S_1)$ by Lemma 19. □

A partial function $f : \mathbf{R} \longrightarrow \mathbf{R}$ is said to be *multi-rooted* if there is a set A , called the *set of roots* of f, such that:

1. $f(R_0) \downarrow$ for all $R_0 \in A$
2. $f(R) \downarrow$ iff there is some $R_0 \in A$ such that $f(R_0) \circ (R - R_0)$ is defined and $f(R) = f(R_0) \circ (R - R_0)$.
3. The set of roots is coherent:

$$f(R_1) \circ (R - R_1) = f(R_2) \circ (R - R_2)$$

for all roots $R_1, R_2 \in A$ such that $f(R_1) \circ (R - R_1)$ and $f(R_2) \circ (R - R_2)$ are defined.

Proposition 21. *Let* $\mathbf{R}$ *be a resource monoid with cancellation. Suppose that there is a family of multi-rooted partial functions μ_α indexed by atomic actions α. For each atomic α let* $\mathsf{Roots}(\alpha)$ *be the set of roots of μ_α. Then there is a* **SCRP***-modification on* $\mathbf{R}$ *as follows. At any resource R and action $a = \alpha_1 \cdots \alpha_n \neq 1$,*

$$\mu(\alpha_1 \cdots \alpha_n, R) = \mu_{\alpha_1}(R_1) \circ \ldots \circ \mu_{\alpha_n}(R_n) \circ (R - (R_1 \circ \ldots \circ R_n))$$

if there are $R_i \in \mathsf{Roots}(\alpha_i)$ for $1 \leq i \leq n$ such that the right-hand side is defined, otherwise we take $\mu(\alpha_1 \cdots \alpha_n, R)$ to be undefiend. If $a = 1$ then $n = 0$ and we take $\mu(1, R) = R$.

Proof. We first show that $\mu(a, -)$ is well-defined as a partial function by induction on the number n of actions in $a = \alpha_1 \cdots \alpha_n$.

Suppose that $n = 1$ and that R_1 and S_1 are both roots of μ_{α_1} such that $\mu_{\alpha_1}(R) \circ (R - R_1)$ and $\mu_{\alpha_1}(S_1) \circ (R - S_1)$ are both defined for some R. Then by coherence of the roots of μ_{α_1}, we have $\mu_{\alpha_1}(R) \circ (R - R_1) = \mu_{\alpha_1}(S_1) \circ (R - S_1)$, so $\mu(\alpha_1, R)$ is unambiguous.

Suppose that the result holds for $\alpha_1 \cdots \alpha_n$ and consider $a = \alpha_1 \cdots \alpha_{n+1}$. Suppose that there is a resource R and roots $R_i, S_i \in \mathsf{Roots}(\alpha_i)$ for $1 \leq i \leq n$ with $\mu_{\alpha_1}(R_1) \circ \ldots \circ \mu_{\alpha_n}(R_n) \circ \mu_{\alpha_{n+1}}(R_{n+1}) \circ (R - (R_1 \circ \ldots \circ R_n \circ R_{n+1}))$ and $\mu_{\alpha_1}(S_1) \circ \ldots \circ \mu_{\alpha_n}(S_n) \circ \mu_{\alpha_{n+1}}(S_{n+1}) \circ (R - (S_1 \circ \ldots \circ S_n \circ S_{n+1}))$ both defined. Then

$$\begin{array}{rl}
 & \mu_{\alpha_1}(R_1) \circ \ldots \circ \mu_{\alpha_n}(R_n) \circ \mu_{\alpha_{n+1}}(R_{n+1}) \circ (R - (R_1 \circ \ldots \circ R_n \circ R_{n+1})) \\
= & \mu_{\alpha_1}(R_1) \circ \ldots \circ \mu_{\alpha_n}(R_n) \circ \mu_{\alpha_{n+1}}(R_{n+1}) \circ ((R - (R_1 \circ \ldots \circ R_n)) - R_{n+1}) \\
= & \mu_{\alpha_1}(R_1) \circ \ldots \circ \mu_{\alpha_n}(R_n) \circ \mu_{\alpha_{n+1}}(S_{n+1}) \circ ((R - (R_1 \circ \ldots \circ R_n)) - S_{n+1}) \\
= & \mu_{\alpha_1}(R_1) \circ \ldots \circ \mu_{\alpha_n}(R_n) \circ ((R - S_{n+1}) - (R_1 \circ \ldots \circ R_n)) \circ \mu_{\alpha_{n+1}}(S_{n+1}) \\
= & \mu_{\alpha_1}(S_1) \circ \ldots \circ \mu_{\alpha_n}(S_n) \circ ((R - S_{n+1}) - (S_1 \circ \ldots \circ S_n)) \circ \mu_{\alpha_{n+1}}(S_{n+1}) \\
= & \mu_{\alpha_1}(S_1) \circ \ldots \circ \mu_{\alpha_n}(S_n) \circ \mu_{\alpha_{n+1}}(S_{n+1}) \circ (R - (S_1 \circ \ldots \circ S_n \circ S_{n+1}))
\end{array}$$

using Lemma 18 three times, the induction hypothesis and the facts that $\mu_{\alpha_{n+1}}$ is rooted and that $\mu_{\alpha_{n+1}}(R_{n+1}) \circ ((R - (S_1 \circ \ldots \circ S_n)) - R_{n+1}))$ and $\mu_{\alpha_{n+1}}(S_{n+1}) \circ ((R - (S_1 \circ \ldots \circ S_n)) - S_{n+1}))$ are both defined.

Suppose that $\mu(a, R)$, $\mu(b, S)$ and $R \circ S$ are all defined. If both of the actions are the unit then $\mu(ab, R \circ S) = \mu(1, R \circ S) = R \circ S = \mu(a, R) \circ \mu(b, S)$ are all defined. Consider the case where just

one of the actions is the unit — without loss of generality suppose that it is b. Then, for some atoms $\alpha_1, \ldots, \alpha_n$ and resources $R_1, \ldots, R_n$,

$$\begin{array}{ll} & \mu(a,R) \circ \mu(b,S) \\ \simeq & \mu(a,R) \circ S \\ \simeq & \mu_{\alpha_1}(R_1) \circ \ldots \circ \mu_{\alpha_m}(R_m) \circ (R - (R_1 \circ \ldots \circ R_m)) \circ S \\ \simeq & \mu_{\alpha_1}(R_1) \circ \ldots \circ \mu_{\alpha_m}(R_m) \circ ((R \circ S) - (R_1 \circ \ldots \circ R_m)) \\ \simeq & \mu(ab, R \circ S) \end{array}$$

using Lemma 19. Now suppose $a = \alpha_1 \cdots \alpha_m$ and $b = \beta_1 \cdots \beta_n$. Then

$$\begin{array}{ll} & \mu(a,R) \circ \mu(b,S) \\ \simeq & \mu_{\alpha_1}(R_1) \circ \ldots \circ \mu_{\alpha_m}(R_m) \circ (R - (R_1 \circ \ldots \circ R_m)) \circ \\ & \mu_{\beta_1}(S_1) \circ \ldots \circ \mu_{\beta_n}(S_n) \circ (S - (S_1 \circ \ldots \circ S_n)) \\ \simeq & \mu_{\alpha_1}(R_1) \circ \ldots \circ \mu_{\alpha_m}(R_m) \circ \mu_{\beta_1}(S_1) \circ \ldots \circ \mu_{\beta_n}(S_n) \circ \\ & ((R \circ S) - ((R_1 \circ \ldots \circ R_m) \circ (S_1 \circ \ldots \circ S_n))) \\ \simeq & \mu(ab, R \circ S) \end{array}$$

using Lemma 20, and roots $R_i \in \mathsf{Roots}(\alpha_i)$, $S_j \in \mathsf{Roots}(\beta_j)$ for all $1 \leq i \leq m$ and $1 \leq j \leq n$. Thus μ satisfies the conditions for a modification of **SCRP**. □

Note that this result allows atomic actions for which the modification is undefined at every resource and the set of roots is empty.

3 Bunched Modal Logic

The logic **MBI**, along with some basic properties, was sketched in (Pym & Tofts 2006, Pym & Tofts 2007, Collinson et al. 2007). **MBI** is a modal logic, resembling Hennessy-Milner logic (Hennessy & Milner 1985), based on bunched logic, (O'Hearn & Pym 1999, Pym 1999, Pym et al. 2004, Pym 2002). As such, it serves as a specification language for the process algebra **SCRP**.

The logic **MBI** has been shown to give a logical account of process constructs; in particular, synchronous product and hiding. It has also been shown, through a number of key examples, to give a useful account of resource-use by concurrent processes. In this section, we present a further developed account.

The logic as presented in (Pym & Tofts 2006, Pym & Tofts 2007, Collinson et al. 2007) is not equipped with a (proof-theoretic) deductive system. Here we give a proof system for (propositional) **MBI** that adds modal axioms to (propositional) **BI**'s natural deduction system (O'Hearn & Pym 1999, Pym 1999, Pym 2002). The logical calculus has a number of important properties that follow from the properties of the **SCRPr** calculus — these properties were not all present in the previous accounts.

3.1 Bunched Implications

The logical system we wish to consider is based on propositional **BI**, the logic of bunched implications. Here we present a brief review — more detailed accounts may be found in (O'Hearn & Pym 1999, Pym 1999, Pym et al. 2004, Pym 2002).

The logic **BI** combines a logic with structural rules of contraction and weakening (intuitionistic logic) with a substructural logic that lacks these rules (multiplicative linear logic). Furthermore, it does this in such a way that the two embedded logics have the same status (neither is definable from the other) and so that certain properties of those logics are retained. The composite logic provides two variants, additive and multiplicative, of several of the basic logical connectives. These have clear and distinct interpretations on resource monoids and this gives rise to many applications. An example of this is Separation Logic, (Ishtiaq & O'Hearn 2001, Reynolds 2002), a Hoare-style program logic with local reasoning regarding state.

Assume a set Prop_0 of basic propositions φ. Propositions are generated by the grammar

$$\phi ::= \varphi \mid \top \mid \bot \mid \phi \wedge \phi \mid \phi \to \phi \mid \phi \vee \phi \mid I \mid \phi * \phi \mid \phi \mathrel{-\!\!*} \phi$$

giving a set Prop of propositions. The connectives $\wedge$, $\to$, $\vee$, $\top$, $\bot$ stand respectively for *additive* conjunction, implication, disjunction, truth and falsity. The connectives $*$, $\mathrel{-\!\!*}$ and I are the *multiplicative* conjunction, implication and unit, respectively.

The development of bunched logic hinges on the use of contexts Γ for formulae that are structured in a particular way. *Bunches* of propositions are generated by

$$\Gamma ::= \emptyset \mid \emptyset_* \mid \phi \mid \Gamma ; \Gamma \mid \Gamma , \Gamma \ .$$

The constants $\emptyset$ and $\emptyset_*$ are the additive and multiplicative units, respectively. Notice that bunches are trees with leaves labelled by propositions or units and each internal node labelled by either the additive context former ';' or the multiplicative ','.

A *sub-bunch* of Γ is just a sub-tree such that all leaves are labelled by propositions. We write, for example, $\Gamma(\Delta)$ for a bunch containing a sub-bunch Δ. We may substitute bunches for sub-bunches. Given a bunch $\Gamma(\Delta)$ we write either $\Gamma[\Delta'/\Delta]$ or $\Gamma(\Delta')$ for the result of substituting Δ' for Δ in Γ.

We introduce a congruence relation $\equiv$ on bunches. This is generated by applying the commutative monoid axioms to each of the binary operations ';' and ',' at arbitrary depth in a bunch. The axioms ensure that the operation ';' with $\top$ defines a commutative monoid (up to $\equiv$) on the set of bunches, as does ',' with I. This relation is used to control the exchange rule for **BI**.

We will present our bunched logics in natural-deduction-style calculi. The rules for the basic system of intuitionistic **BI** are given in Figure 2. Write $\Gamma \vdash \phi$ and say that this is *derivable* when it occurs at the root of a derivation using the proof rules. The calculus has a number of important properties, including cut-elimination and the existence of known decision procedures.

Let $\mathbf{R}$ be a resource monoid. For the purposes of this subsection the preorder is not required to be discrete. Let $\mathcal{U}(\mathbf{R})$ be the collection of all upper sets of $\mathbf{R}$ (those that are upper closed with respect to the order). Write $\uparrow \mathcal{R}$ for the upwards closure of a subset $\mathcal{R}$ of $\mathbf{R}$. There is a binary operation $*$ on $\mathcal{U}(\mathbf{R})$ defined by

$$\mathcal{R} * \mathcal{S} = \uparrow \{R \circ S \mid R \in \mathcal{R} \ \& \ S \in \mathcal{S} \ \& \ R \circ S \text{ is defined}\}$$

for all $\mathcal{R}, \mathcal{S} \in \mathcal{U}(\mathbf{R})$.

The logical calculus can be given a forcing semantics on resource monoids. Suppose we have a valuation, $\mathcal{V} : \mathsf{Prop}_0 \longrightarrow \mathcal{U}(\mathbf{R})$, of atomic propositions. We define a satisfaction relation $\vDash \subseteq \mathbf{R} \times \mathsf{Prop}$ in Figure 3. Each valuation determines an interpretation function $[\![-]\!] : \mathsf{Prop} \longrightarrow \mathcal{U}(\mathbf{R})$ given by

$$R \in [\![\phi]\!] \quad \text{iff} \quad R \vDash \phi$$

for all $R \in \mathbf{R}$ and propositions ϕ. The intepretation of formulae extends to an interpretation of **BI**-sequents by taking

$$[\![\emptyset]\!] = [\![\top]\!] \qquad [\![\emptyset_*]\!] = [\![I]\!] \qquad [\![\phi]\!] = [\![\phi]\!] \qquad [\![\Gamma ; \Delta]\!] = [\![\Gamma]\!] \cap [\![\Delta]\!] \qquad [\![\Gamma , \Delta]\!] = [\![\Gamma]\!] * [\![\Delta]\!]$$

for all sequents Γ and Δ and formulae ϕ.

Proposition 22. *The axioms of* **BI** *are sound with respect to the forcing semantics. That is,*

$$\Gamma \vdash \phi \quad \textit{implies} \quad [\![\Gamma]\!] \subseteq [\![\phi]\!]$$

holds.

An algebraic re-formulation of soundness is useful. This says that the set $\mathcal{U}(\mathbf{R})$ has a natural **BI**-algebra structure (see (Pym et al. 2004, Pym 2002) for more on **BI**-algebras). In particular, this uses the operation $*$ above. This construction is a mild generalization of the construction of a

$$\text{(Axiom)}\quad \frac{}{\phi \vdash \phi} \qquad (\Gamma \equiv \Delta)\ \frac{\Gamma \vdash \phi}{\Delta \vdash \phi}\quad (E)$$

$$(W)\quad \frac{\Gamma(\Delta) \vdash \phi}{\Gamma(\Delta;\Delta') \vdash \phi} \qquad \frac{\Gamma(\Delta;\Delta) \vdash \phi}{\Gamma(\Delta) \vdash \phi}\quad (C)$$

$$(II)\quad \frac{}{\emptyset_* \vdash I} \qquad \frac{\Delta \vdash I \qquad \Gamma(\emptyset_*) \vdash \phi}{\Gamma(\Delta) \vdash \phi}\quad (IE)$$

$$(*I)\quad \frac{\Gamma \vdash \phi \quad \Delta \vdash \psi}{\Gamma,\Delta \vdash \phi * \psi} \qquad \frac{\Delta \vdash \phi * \psi \quad \Gamma(\phi,\psi) \vdash \theta}{\Gamma(\Delta) \vdash \theta}\quad (*E)$$

$$(-\!\!*I)\quad \frac{\Gamma,\phi \vdash \psi}{\Gamma \vdash \phi -\!\!* \psi} \qquad \frac{\Gamma \vdash \phi \quad \Delta \vdash \phi -\!\!* \psi}{\Gamma,\Delta \vdash \psi}\quad (-\!\!*E)$$

$$(\top I)\quad \frac{}{\Gamma \vdash \top} \qquad \frac{\Delta \vdash \top \quad \Gamma(\emptyset) \vdash \phi}{\Gamma(\Delta) \vdash \phi}\quad (\top E)$$

$$(\wedge I)\quad \frac{\Gamma \vdash \phi \quad \Delta \vdash \psi}{\Gamma;\Delta \vdash \phi \wedge \psi} \qquad \frac{\Delta \vdash \phi \wedge \psi \quad \Gamma(\phi;\psi) \vdash \theta}{\Gamma(\Delta) \vdash \theta}\quad (\wedge E)$$

$$(\to I)\quad \frac{\Gamma;\phi \vdash \psi}{\Gamma \vdash \phi \to \psi} \qquad \frac{\Gamma \vdash \phi \quad \Delta \vdash \phi \to \psi}{\Gamma;\Delta \vdash \psi}\quad (\to E)$$

$$(\vee I_i)\quad \frac{\Gamma \vdash \phi_i}{\Gamma \vdash \phi_1 \vee \phi_2}\ (i=1,2) \qquad \frac{\Delta \vdash \phi \vee \psi \quad \Gamma(\phi) \vdash \theta \quad \Gamma(\psi) \vdash \theta}{\Gamma(\Delta) \vdash \theta}\quad (\vee E)$$

$$\text{(Cut)}\quad \frac{\Delta \vdash \phi \qquad \Gamma(\phi) \vdash \psi}{\Gamma(\Delta) \vdash \psi} \qquad \frac{\Gamma \vdash \bot}{\Gamma \vdash \phi}\quad (\bot E)$$

Figure 2: Axioms for (intuitionistic) **BI**

$R \vDash \varphi$	iff $R \in \mathcal{V}(\varphi)$
$R \vDash \top$	always
$R \vDash \bot$	never
$R \vDash \phi \wedge \psi$	iff $R \vDash \phi$ and $R \vDash \psi$
$R \vDash \phi \vee \psi$	iff $R \vDash \phi$ or $R \vDash \psi$
$R \vDash \phi \to \psi$	iff $\forall S.\ R \sqsubseteq S$ and $S \vDash \phi$ implies $S \vDash \psi$
$R \vDash I$	iff $e \sqsubseteq R$
$R \vDash \phi_1 * \phi_2$	iff $\exists R_1, R_2.\ R_1 \circ R_2 \sqsubseteq R$ and $R_1 \vDash \phi_1$ and $R_2 \vDash \phi_2$
$R \vDash \phi -\!\!* \psi$	iff $\forall S.\ R \circ S \downarrow$ and $S \vDash \phi$ implies $R \circ S \vDash \psi$

Figure 3: Interpretation of **BI**

quantale from a special resource monoid (sometimes called a partially-ordered monoid). It is also an instance of Day's construction of (enriched) doubly-closed categories (Day 1970, Day 1973). The definition of interpretation can evidently be modified to give an interpretation on the lower sets of a dual resource monoid.

We take the system **BIc** of classical **BI** to consist of classical additive connectives and intuitionistic multiplicative connectives. This system as well as more intricate variants with classical multiplicatives are discussed in (O'Hearn & Pym 1999, Pym et al. 2004, Pym 1999, Pym 2002). We add the logical connective for negation by defining $\neg\phi$ to be $\phi \to \bot$ for all propositions ϕ. The system **BIc** is formed by adding the rule

$$\text{(RAA)} \qquad \frac{\Gamma \vdash \neg\neg\phi}{\Gamma \vdash \phi}$$

to **BI**.

In order to give a semantics we restrict to resource monoids, $\mathbf{R}$, with discrete order. That is $R \sqsubseteq S$ if and only if $R = S$ for all $R, S \in \mathbf{R}$. Notice that the bifunctoriality condition becomes vacuous in this situation. Valuations are defined as for **BI**. Note that now $\mathcal{U}(\mathbf{R}) = \mathcal{P}(\mathbf{R})$ so that atomic propositions are interpreted as arbitrary subsets of $\mathbf{R}$. It is then easily verified that the rule (RAA) is sound.

Proposition 23. *The axioms of* **BIc** *are sound with respect to the interpretation on resource monoids with discrete order.*

The algebraic formulation of this says that $\mathcal{P}(\mathbf{R})$ is a Boolean **BI**-algebra, that is, a **BI**-algebra such that $\neg$ (as complementation) makes it a Boolean algebra.

Define the system $\mathbf{BIc}^{-I}$ to be the same as **BIc** but with the unit I and all rules involving it excised. The propositional systems **BI** and **BIc** can be extended to include first-order predication and quantifiers as in (O'Hearn & Pym 1999, Pym 1999, Pym et al. 2004, Pym 2002). This provides additive and multiplicative variants of both the existential and the universal quantifier.

3.2 A Modal Logic

We now present an extended Hennessy-Milner logic for **SCRPr**. The logic we is a close relative of the logic **MBI** given in (Pym & Tofts 2006, Pym & Tofts 2007, Collinson et al. 2007). Here, we focus on the simplified language **MBIc**. The language **MBIc** is the same as **MBI** except that predication and quantifications over actions are omitted.

Assume sets Act of actions and Prop_0 of atomic propositions. Let φ range over such atomic propositions. The set Prop of propositions of **MBIc** is defined by the grammar

$$\phi ::= \varphi \mid \top \mid \bot \mid \phi \to \phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid I \mid \phi * \phi \mid \phi \mathrel{-\!\!*} \phi \mid [a]\phi \mid \langle a\rangle\phi \mid [a]_\nu\phi \mid \langle a\rangle_\nu\phi$$

where a is any action. Thus the language **MBIc** extends the language **BIc** with additive modalities $[a]$, $\langle a\rangle$ and multiplicative modalities $\langle a\rangle_\nu$, $[a]_\nu$ labelled by actions a. The language $\mathbf{MBIc}^{-I}$ omits the unit I. The additive modalities are the standard 'necessarily' and 'possibly' connectives familiar from modal logics, in particular Hennessy-Milner logics for process algebras. As such, they implicitly use meta-theoretic quantification to make statements about reachable states. The multiplicative variants are related to multiplicative quantifications, as described in (O'Hearn & Pym 1999, Pym 1999, Pym et al. 2004, Pym 2002), and make statements about reachable states in the presence of additional resource. The logic is classical for additives and so we may define $\neg\phi$ to be $\phi \to \bot$. We could have defined $[a]\phi$ to be $\neg\langle a\rangle\neg\phi$. We will see from the semantics that we could also have defined $[a]_\nu\phi$ to be $\neg\langle a\rangle_\nu\neg\phi$. Examples justifying the inclusion of multiplicatives modalities were included in (Pym & Tofts 2006, Pym & Tofts 2007).

For any bunch Γ of formulae let $[a]\Gamma$ be the bunch formed by putting $[a]\phi$ for each ϕ of Γ. Adopt a similar convention for $\langle a\rangle\Gamma$, $[a]_\nu\Gamma$ and $\langle a\rangle_\nu\Gamma$. The rules of **MBIc** consist of the rules of **BIc** together with the rules presented in Figures 4 and 5. Notice that there is a new introduction rule for each of the modalities. With the exception of these, all of the new modal rules may be presented as Hilbert-style tautologies.

$$([]I) \quad \frac{\Gamma \vdash \psi}{[a]\Gamma \vdash [a]\psi} \qquad \frac{\Gamma \vdash \psi}{\langle a\rangle\Gamma \vdash \langle a\rangle\psi} \quad (\langle\rangle I)$$

$$(\neg[]\neg 1) \quad \frac{\Gamma \vdash \langle a\rangle\phi}{\Gamma \vdash \neg[a]\neg\phi} \qquad \frac{\Gamma \vdash \neg[a]\neg\phi}{\Gamma \vdash \langle a\rangle\phi} \quad (\neg[]\neg 2)$$

$$([]\top) \quad \frac{\Gamma \vdash \top}{\Gamma \vdash [a]\top} \qquad \frac{\Gamma \vdash \langle a\rangle\bot}{\Gamma \vdash \bot} \quad (\langle\rangle\bot)$$

$$([] \wedge 1) \quad \frac{\Gamma \vdash [a]\phi \wedge [a]\psi}{\Gamma \vdash [a](\phi \wedge \psi)} \qquad \frac{\Gamma \vdash [a](\phi \wedge \psi)}{\Gamma \vdash [a]\phi \wedge [a]\psi} \quad ([] \wedge 2)$$

$$(\langle\rangle \vee 1) \quad \frac{\Gamma \vdash \langle a\rangle\phi \vee \langle a\rangle\psi}{\Gamma \vdash \langle a\rangle(\phi \vee \psi)} \qquad \frac{\Gamma \vdash \langle a\rangle(\phi \vee \psi)}{\Gamma \vdash \langle a\rangle\phi \vee \langle a\rangle\psi} \quad (\langle\rangle \vee 2)$$

$$([] \wedge \langle\rangle) \quad \frac{\Gamma \vdash [a]\phi \wedge \langle a\rangle\psi}{\Gamma \vdash \langle a\rangle(\phi \wedge \psi)} \qquad \frac{\Gamma \vdash \langle a\rangle(\phi \wedge \psi)}{\Gamma \vdash \langle a\rangle\phi \wedge \langle a\rangle\psi} \quad (\langle\rangle\wedge)$$

$$(\langle 1\rangle) \quad \frac{\Gamma \vdash \langle 1\rangle\phi}{\Gamma \vdash \phi} \qquad \frac{\Gamma \vdash \phi}{\Gamma \vdash [1]\phi} \quad ([1])$$

$$(\langle 1\rangle I) \quad \frac{\Gamma \vdash I}{\Gamma \vdash \langle 1\rangle I} \qquad \frac{\Gamma \vdash \phi}{\Gamma \vdash \langle 1\rangle\phi \vee [1]\bot} \quad (\langle 1\rangle[1])$$

$$(\langle\rangle *) \quad \frac{\Gamma \vdash \langle a_1\rangle\phi_1 * \langle a_2\rangle\phi_2}{\Gamma \vdash \langle a_1 a_2\rangle(\phi_1 * \phi_2)}$$

Figure 4: Axioms for additive modalities of **MBIc**

$$([]_\nu I) \quad \frac{\Gamma \vdash \psi}{[a]_\nu \Gamma \vdash [a]_\nu \psi} \qquad \frac{\Gamma \vdash \psi}{\langle a\rangle_\nu \Gamma \vdash \langle a\rangle_\nu \psi} \quad (\langle\rangle_\nu I)$$

$$(\neg[]_\nu \neg 1) \quad \frac{\Gamma \vdash \langle a\rangle_\nu \phi}{\Gamma \vdash \neg[a]_\nu \neg\phi} \qquad \frac{\Gamma \vdash \neg[a]_\nu \neg\phi}{\Gamma \vdash \langle a\rangle_\nu \phi} \quad (\neg[]_\nu \neg 2)$$

$$([-]_\nu \top) \quad \frac{\Gamma \vdash \top}{\Gamma \vdash [a]_\nu \top} \qquad \frac{\Gamma \vdash \langle a\rangle_\nu \bot}{\Gamma \vdash \bot} \quad (\langle -\rangle_\nu \bot)$$

$$([]_\nu \wedge 1) \quad \frac{\Gamma \vdash [a]_\nu \phi \wedge [a]_\nu \psi}{\Gamma \vdash [a]_\nu (\phi \wedge \psi)} \qquad \frac{\Gamma \vdash [a]_\nu (\phi \wedge \psi)}{\Gamma \vdash [a]_\nu \phi \wedge [a]_\nu \psi} \quad ([]_\nu \wedge 2)$$

$$([]_\nu \wedge \langle\rangle_\nu) \quad \frac{\Gamma \vdash [a]_\nu \phi \wedge \langle a\rangle_\nu \psi}{\Gamma \vdash \langle a\rangle_\nu (\phi \wedge \psi)} \qquad \frac{\Gamma \vdash \langle a\rangle_\nu (\phi \wedge \psi)}{\Gamma \vdash \langle a\rangle_\nu \phi \wedge \langle a\rangle_\nu \psi} \quad (\langle\rangle_\nu \wedge)$$

$$(\langle\rangle_\nu \vee 1) \quad \frac{\Gamma \vdash \langle a\rangle_\nu \phi \vee \langle a\rangle_\nu \psi}{\Gamma \vdash \langle a\rangle_\nu (\phi \vee \psi)} \qquad \frac{\Gamma \vdash \langle a\rangle_\nu (\phi \vee \psi)}{\Gamma \vdash \langle a\rangle_\nu \phi \vee \langle a\rangle_\nu \psi} \quad (\langle\rangle_\nu \vee 2)$$

$$([-][-]_\nu) \quad \frac{\Gamma \vdash [a]_\nu \phi}{\Gamma \vdash [a]\phi} \qquad \frac{\Gamma \vdash \langle a\rangle \phi}{\Gamma \vdash \langle a\rangle_\nu \phi} \quad (\langle -\rangle\langle -\rangle_\nu)$$

$$(\langle\rangle_\nu *) \quad \frac{\Gamma \vdash \langle a_1\rangle_\nu \phi * \langle a_2\rangle_\nu \phi_2}{\Gamma \vdash \langle a_1 a_2\rangle_\nu (\phi_1 * \phi_2)}$$

Figure 5: Axioms for multiplicative modalities of **MBIc**

3.3 Semantics

The mathematical structure on which we interpret **MBI** is the set States of states generated by resources and processes. Recall that each state generates a transition structure. We define the interpretation of a formula at a state to be the interpretation of that formula at the corresponding transition structure in the ambient set of states. For the purposes of this section assume that $(\mathsf{Act}, \mathbf{R}, \mu, \nu)$ is fixed.

Recall the global equivalence relation $\sim$. A set Σ of states is said to be $\sim$-*closed* if it satisfies the property

$$R, E \in \Sigma \quad \text{and} \quad E \sim F \quad \text{implies} \quad R, F \in \Sigma$$

for all states R, E and processes F. Let $\mathcal{P}_\sim(\mathsf{States})$ be the set of all $\sim$-closed sets of states. Another way to construct this is to lift $\sim$ up to the set of states via

$$R, E \sim S, F \qquad \text{iff} \qquad R = S \quad \text{and} \quad E \sim F$$

for all states R, E and S, F. This is evidently an equivalence relation. Furthermore, the $\sim$-closed subsets are seen to be in one-one correspondence with unions of families of equivalence classes of the relation $\sim$ on states. The set $\mathsf{CStates}$ does not, in general, have to be $\sim$-closed.

We now proceed to give an interpretation of the logical calculus on the set $\mathsf{CStates}$ of closed states. Consider the relation $\sim$ restricted to $\mathsf{CStates}$. Then we may consider the set $\mathcal{P}_\sim(\mathsf{CStates})$ of $\sim$-closed sets of closed states. A valuation is a function

$$\mathcal{V} : \mathsf{Prop}_0 \longrightarrow \mathcal{P}_\sim(\mathsf{CStates})$$

from the set of basic propositions to $\sim$-closed subsets of the set of all states. Every valuation extends in a canonical way to an interpretation for MBI-formulae, the satisfaction relation for which is indicated in Figure 6, and in which every process that appears is required to be an agent. A model for **MBIc** consists of the set of closed states together with this interpretation.

$R, E \vDash \varphi$ iff $R, E \in \mathcal{V}(\varphi)$

$R, E \vDash \bot$ never

$R, E \vDash \top$ always

$R, E \vDash \phi \wedge \psi$ iff $R, E \vDash \phi$ and $R, E \vDash \psi$

$R, E \vDash \phi \vee \psi$ iff $R, E \vDash \phi$ or $R, E \vDash \psi$

$R, E \vDash \phi \to \psi$ iff $R, E \vDash \phi$ implies $R, E \vDash \psi$

$R, E \vDash I$ iff $e = R$ and $E \sim \mathsf{1}$

$R, E \vDash \phi_1 * \phi_2$ iff $\exists R_1, R_2, E_1, E_2.\ R = R_1 \circ R_2$ and $E \sim E_1 \times E_2$ and $R_1, E_1 \vDash \phi_1$ and $R_2, E_2 \vDash \phi_2$

$R, E \vDash \phi \mathrel{-\!\!*} \psi$ iff $\forall S, F.\ R \circ S \downarrow$ & $S, F \vDash \phi$ implies $R \circ S, E \times F \vDash \psi$

$R, E \vDash [a]\phi$ iff $\forall R', E'.\ R, E \xrightarrow{a} R', E'$ implies $R', E' \vDash \phi$

$R, E \vDash \langle a \rangle \phi$ iff $\exists R', E'.\ R, E \xrightarrow{a} R', E'$ and $R', E' \vDash \phi$

$R, E \vDash [a]_\nu \phi$ iff $\forall T, R', E'.\ R \circ T, E \xrightarrow{a} R', E'$ implies $R', E' \vDash \phi$

$R, E \vDash \langle a \rangle_\nu \phi$ iff $\exists T, R', E'.\ R \circ T, E \xrightarrow{a} R', E'$ and $R', E' \vDash \phi$

Figure 6: Interpretation of **MBIc** on closed states

Example 24. One of the most interesting new axioms of **MBIc** is $(\langle\rangle*)$, which is equivalent to a tautology

$$(\langle a_1\rangle\phi_1 * \langle a_2\rangle\phi_2) \to \langle a_1 a_2\rangle(\phi_1 * \phi_2)$$

for all actions a_1, a_2 and propositions ϕ_1 and ϕ_2. This can be seen to describe an essential part of the operational behaviour of product processes as prescribed by the operational semantics: if resource can be split in such a way that enables actions for a pair of processes, then it enables the product process. Furthermore, if each of the two sub-processes are known to step into processes with known properties ϕ_1 and ϕ_2 then the product process can step into some (product) process satisfying $\phi_1 * \phi_2$.

A careful reading of the handshaking-process example from (Pym & Tofts 2006, Pym & Tofts 2007) reveals that this is precisely how the logical specification for the process is constructed. In this example, there are a pair of processes

$$\begin{array}{rcl} E_1 & = & 1 : E_1 + go_{E_1} : E_1' \\ E_2 & = & 1 : E_2 + go_{E_2} : E_2' \end{array}$$

that can evolve to a new state just when they agree on progress, and otherwise wait in the original state. The underlying resource monoid is assumed to be good and the modifications for all atomic actions are all rooted. Thus we have a modification as in Proposition 17. Let the root of $\mu(go_{E_i}, -)$ be $R_i \neq e$ for $i = 1, 2$ and suppose that $R_1 \neq R_2$. Let $R = R_1 \circ R_2$ and note that $R_1 \neq R \neq R_2$. The two processes either remain together in the initial state $R, E_1 \times E_2$ or progress to a new state via a transition $R, E_1 \times E_2 \overset{go_{E_1} go_{E_2}}{\to} R', E_1' \times E_2'$ where $R' = \mu(go_{E_1} go_{E_2}, R)$. If $\mu(a, R_i), E_i' \vDash \phi$ for $i = 1, 2$ then we see that we have

$$R, E_1 \times E_2 \vDash (\langle go_{E_1}\rangle\phi_1) * (\langle go_{E_2}\rangle\phi_2)$$

and so

$$R, E_1 \times E_2 \vDash \langle go_{E_1} go_{E_2}\rangle(\phi_1 * \phi_2) \ .$$

The additive version of the axiom $(\langle\rangle*)$ together with the **BI** rules entail:

$$\langle a_1\rangle(\phi \mathrel{-\!\!*} \psi) \vdash (\langle a_2\rangle\phi) \mathrel{-\!\!*} (\langle a_1 a_2\rangle\psi) \ .$$

Note that:

$$(\langle a_1\rangle\phi) * (\langle a_2\rangle\psi) \vdash \langle a_1 a_2\rangle(\phi * \phi) \text{ and } (\phi \mathrel{-\!\!*} \psi) * \phi \vdash \psi$$

are derivable. We can therefore make the derivation:

$$\dfrac{\dfrac{\dfrac{\vdots}{\langle a_1\rangle(\phi \mathrel{-\!\!*} \psi) * (\langle a_2\rangle\phi) \vdash \langle a_1 a_2\rangle((\phi \mathrel{-\!\!*} \psi) * \phi)} \quad \dfrac{\dfrac{\vdots}{(\phi \mathrel{-\!\!*} \psi) * \phi \vdash \psi}}{\langle a_1 a_2\rangle((\phi \mathrel{-\!\!*} \psi) * \phi) \vdash \langle a_1 a_2\rangle\psi}}{\langle a_1\rangle(\phi \mathrel{-\!\!*} \psi), (\langle a_2\rangle\phi) \vdash \langle a_1 a_2\rangle\psi}}{\langle a_1\rangle(\phi \mathrel{-\!\!*} \psi) \vdash (\langle a_2\rangle\phi) \mathrel{-\!\!*} (\langle a_1 a_2\rangle\psi)}$$

for any ϕ, ψ, a_1 and a_2, using the cut rule.

In the handshaking example, if we take $a_i = go_{E_i}$ for $i = 1, 2$, and ϕ_1 to be $\phi_2 \mathrel{-\!\!*} \psi$, then whenever we combine $R_1, E_1 \vDash \langle go_{E_1}\rangle\phi_1$ with $R_2, E_2 \vDash \langle go_{E_2}\rangle\phi_2$ we get $R, E_1 \times E_2 \vDash \langle go_{E_1} go_{E_2}\rangle\psi$.

Note that the satisfaction of certain formulae at a given state makes use of states that lie outside the transition structure generated by that state. This is a critical difference between **MBI** and most process logics. This means that the model-checking problem for **MBIc** can be very hard, indeed often only semi-decidable, depending on the properties of the underlying quadruple $(\mathbf{R}, \mathsf{Act}, \mu, \nu)$. The development of a logic to accompany the language is therefore an important step.

Define a binary relation on closed states by

$$R, E \overset{\mathbf{MBIc}}{\equiv} S, F \qquad \text{iff} \qquad \forall\phi.\ R, E \vDash \phi \iff S, F \vDash \phi$$

for all R, E, S, F. For any E, F write

$$E \overset{\mathbf{MBIc}}{\equiv} F \qquad \text{iff} \qquad \forall R.\ R, E \overset{\mathbf{MBIc}}{\equiv} R, F$$

holds.

The following result, which is related to the Hennessy-Milner theorem (Hennessy & Milner 1985), shows that there is a close relationship between the algebraic equivalence $\sim$ and the logical equivalence $\overset{\mathbf{MBIc}}{\equiv}$:

Theorem 25. *Let E and F be closed states. If $E \sim F$ then $E \overset{\mathbf{MBIc}}{\equiv} F$ also holds.*

Proof. . The proof is by induction on the structure of formulae. We show that for every formula ϕ, if we take any E, F, R with $E \sim F$ and $R, E \vDash \phi$ then $R, F \vDash \phi$. Since $\sim$ is symmetric the fact that $R, F \vDash \phi$ implies $R, E \vDash \phi$ follows.

The base cases, where ϕ is one of φ, $\top$, $\bot$, are all immediate. In particular, the case for φ goes through because atomic propositions are valued as $\sim$-closed sets.

The step cases use the following induction hypothesis: for all sub-formulae of ϕ, if for any E, F and R we have $E \sim F$ then $R, E \vDash \psi$ if and only if $R, F \vDash \psi$.

The cases for the connectives $\wedge$, $\vee$, $\rightarrow$ are all unsurprising. We now consider the other cases. We omit the $[a]$ and $[a]_\nu$ cases as they are dual to the $\langle a \rangle$ and $\langle a \rangle_\nu$ cases, respectively.

($\langle a \rangle$) Suppose $R, E \vDash \langle a \rangle \phi$ holds. Then there is some E' such that $R, E \xrightarrow{a} R', E' \vDash \phi$. Since $E \sim F$ it follows that there is an F' with $R, F \xrightarrow{a} R', F'$ with $E' \sim F'$. By the induction hypothesis we have that $R', F' \vDash \phi$. Therefore $R, F \vDash \langle a \rangle \phi$.

(I) Suppose $R, E \vDash I$. Then $R = e$ and $E \sim 1$. Therefore $F \sim 1$, and so $R, F \vDash 1$.

($*$) Suppose $R, E \vDash \phi_1 * \phi_2$. Then there are R_1, R_2, E_1, E_2 with $R = R_1 \circ R_2$, $E \sim E_1 \times E_2$, $R_1, E_1 \vDash \phi_1$ and $R_2, E_2 \vDash \phi_2$. Then $F \sim E_1 \times E_2$ and so $R, F \vDash \phi_1 * \phi_2$.

($-\!\!*I$) Suppose $R, E \vDash \phi -\!\!* \psi$. Consider any S, G such that $R \circ S$ is defined and $S, G \vDash \phi$ holds. Then $R \circ S, E \times G \vDash \psi$ holds. By Proposition 6 we have $E \times G \sim F \times G$ and so $R \circ S, F \times G \vDash \psi$ by the induction hypothesis.

($\langle a \rangle_\nu$) Suppose that $R, E \vDash \langle a \rangle_\nu \phi$. There are some T, R', E' such that $R \circ T$ is defined and $R \circ T, E \xrightarrow{a} R', E' \vDash \phi$. Since $E \sim F$ it follows that there is some F' with $R \circ T, F \xrightarrow{a} R', F'$ and $E' \sim F'$. By the induction hypothesis we have that $R', F' \vDash \phi$. Therefore $R, F \vDash \langle a \rangle_\nu \phi$. □

Note that neither of the I or $*$ cases requires the induction hypothesis. The fact that $\sim$ is a congruence is only required for the $-\!\!*$ case. There were errors in the original proof in (Pym & Tofts 2006, Pym & Tofts 2007) which used the relation $\approx$ instead of $\sim$. This was corrected in (Collinson et al. 2007). However, the following is true:

Proposition 26. *Consider the $\{\top, \bot, \wedge, \vee, \rightarrow, \langle - \rangle, [-], I, *\}$-fragment of* **MBIc**. *Assume that all atomic propositions are valued as sets of closed states that are closed under $\approx$. Alter the $\{I, *\}$-clause of the interpretation so that:*

$$R, E \vDash I \quad \textit{iff} \quad R, E \approx e, 1$$

$$R, E \vDash \phi_1 * \phi_2 \quad \textit{iff} \quad \exists R_1, R_2, E_1, E_2.\ R = R_1 \circ R_2 \ \textit{ and } \ R, E \approx R, E_1 \times E_2$$

The following version of Theorem 25 then holds: if $R, E \approx R, F$ then $R, E \overset{\mathbf{MBIc}}{\equiv} R, F$, for all resources R and processes E, F.

Proof. The proof is essentially as before. We suppose that $R, E \approx R, F$ and show, by induction on the structure of ϕ, that if $R, E \vDash \phi$ then $R, F \vDash \phi$. The I and $*$-cases only require the fact that $\approx$ is an equivalence relation. The other cases then hold for the standard reasons for the usual interpretation of a classical modal logic. □

Theorem 25 remains true for $\sim$ and with atomic predicates, additive and multiplicative quantifiers added to **MBIc**, as in (Pym & Tofts 2006, Pym & Tofts 2007, Collinson et al. 2007). See also Section 3.5 below.

Theorem 25 shows that the set of closed states satisfying any formula is $\sim$-closed.

Corollary 27. *Every interpretation yields a unique function*

$$[\![-]\!] : \mathsf{Prop} \longrightarrow \mathcal{P}_{\sim}(\mathsf{CStates})$$

with

$$R, E \in [\![\phi]\!] \qquad \textit{iff} \qquad R, E \vDash \phi$$

for all closed states R, E and **MBIc***-propositions ϕ.*

The sets States and $\mathsf{CStates}$ have monoidal structure that the interpretation is critically dependent upon. This is easily shown using the algebraic properties of $\sim$ that we have already determined.

Proposition 28. *The set* States *is a resource monoid with the equality given by $\sim$. The composition is defined by*

$$(R, E) \times (S, F) \simeq (R \circ S, E \times F)$$

for all (R, E) and (S, F). Note that this expression is defined just when $R \circ S$ is defined. The unit is $(e, 1)$. The set $\mathsf{CStates}$ *is a resource monoid with the same structure.*

We extend the monoid on $\mathsf{CStates}$ to a monoid $*$ with unit I on $\sim$-closed sets of $\mathsf{CStates}$ by taking

$$\begin{aligned} I &= \{(e, E) \mid E \sim 1\} \\ \Sigma_1 * \Sigma_2 &= \{(R, E) \mid \exists R_1, R_2, E_1, E_2.\ R = R_1 \circ R_2 \ \text{ and } \ E \sim E_1 \times E_2 \ \text{ and } \\ &\qquad\qquad R_1, E_1 \in \Sigma_1 \ \text{ and } \ R_2, E_2 \in \Sigma_2\} \end{aligned}$$

for any two sets of closed states Σ_1 and Σ_2. The interpretation extends to an interpretation of bunches and judgements following the pattern used for the semantics of **BIc**, but now using $\sim$-closed subsets of $\mathsf{CStates}$ in place of sets of resources.

Lemma 29. *If $[\![\Delta]\!] \subseteq [\![\Delta']\!]$ then $[\![\Gamma(\Delta)]\!] \subseteq [\![\Gamma(\Delta')]\!]$ for any Γ.*

The proof of this monotonicity property is by induction on the structure of Γ and uses the observation that both the intersection and $*$ operations are monotonic. In fact, they both satisfy the bifunctoriality condition (1).

Lemma 30. *We have the following simple results:*

$$[\![\phi]\!] = [\![\phi]\!] * [\![I]\!] \qquad \textit{and} \qquad [\![\phi]\!] * [\![\phi \mathrel{-\!\!*} \psi]\!] \subseteq [\![\psi]\!]$$

for all ϕ and ψ.

Proof. The proof is by unfolding the definitions and applying Theorem 25 and Lemma 8. The proof of the first rests upon the simple-extension property for transitions since it requires the result $E \sim E \times 1$ for all agents E. □

Theorem 31. *The calculus* **MBIc** *is sound on the model above. That is*

$$\Gamma \vdash \psi \quad \textit{implies} \quad [\![\Gamma]\!] \subseteq [\![\psi]\!]$$

holds, for all Γ, ψ.

Proof. The proof is an induction on the derivation of the judgement $\Gamma \vdash \psi$. This amounts to a case-analysis on the final rule of the derivation. We omit the cases for the introduction and elimination rules of $\top$, $\bot$, $\wedge$, $\vee$, $\rightarrow$ as they are rather standard. The cases $[-]$ and $[-]_\nu$ are omitted as they are dual to the $\langle - \rangle$ and $\langle - \rangle_\nu$ cases, respectively.

(II) This case is trivial since $[\![\emptyset_*]\!] = I$.

(IE) The induction hypothesis means we have $[\![\Delta]\!] \subseteq [\![I]\!]$ and $[\![\Gamma(\emptyset_*)]\!] \subseteq [\![\phi]\!]$. Now $[\![I]\!] = [\![\emptyset_*]\!]$, so by Lemma 29, $[\![\Gamma(\Delta)]\!] \subseteq [\![\Gamma(\emptyset_*)]\!]$ and hence $[\![\Gamma(\Delta)]\!] \subseteq [\![\phi]\!]$.

$(*I)$ We have $[\![\Gamma]\!] \subseteq [\![\phi]\!]$ and $[\![\Delta]\!] \subseteq [\![\psi]\!]$. Then $[\![\Gamma, \Delta]\!] = [\![\Gamma]\!] * [\![\Delta]\!] \subseteq [\![\phi]\!] * [\![\psi]\!] = [\![\phi * \psi]\!]$ using the monotonicity properties of $*$.

$(*E)$ We have $[\![\Delta]\!] \subseteq [\![\phi * \psi]\!] = [\![\phi, \psi]\!]$ and $[\![\Gamma(\phi, \psi)]\!] \subseteq [\![\theta]\!]$. Then $[\![\Gamma(\Delta)]\!] \subseteq [\![\theta]\!]$ by Lemma 29.

$(-\!\!*I)$ We have that $[\![\Gamma, \phi]\!] \subseteq [\![\psi]\!]$. Now suppose $R, E \in [\![\Gamma]\!]$. Consider any closed state S, F such that $R \circ S \downarrow$ and $S, F \vDash \phi$. By the definitions of satisfaction and interpretation we have $R \circ S, E \times F \in [\![\Gamma, \phi]\!]$. Therefore $R \circ S, E \times F \in [\![\psi]\!]$ and so $R \circ S, E \times F \vDash \psi$. It follows that $R, E \in [\![\phi -\!\!* \psi]\!]$. Therefore $[\![\Gamma]\!] \subseteq [\![\phi -\!\!* \psi]\!]$ holds.

$(-\!\!*E)$ The induction hypothesis gives $[\![\Gamma]\!] \subseteq [\![\phi]\!]$ and $[\![\Delta]\!] \subseteq [\![\phi -\!\!* \psi]\!]$. Since $*$ is bifunctorial we have that $[\![\Gamma, \Delta]\!] \subseteq [\![\phi]\!] * [\![\phi -\!\!* \psi]\!]$. Applying Lemma 30 we get $[\![\Gamma, \Delta]\!] \subseteq [\![\psi]\!]$.

$(\langle\rangle *)$ Suppose $R, E \vDash \langle a_1 \rangle \phi_1 * \langle a_2 \rangle \phi_2$. Then there are R_1, R_2, E_1, E_2 such that $R = R_1 \circ R_2$, $E \sim E_1 \times E_2$, $R_1, E_1 \vDash \langle a_1 \rangle \phi_1$ and $R_2, E_2 \vDash \langle a_2 \rangle \phi_2$ hold. Then there are R_1' R_2', E_1', E_2' such that $R_1, E_1 \stackrel{a_1}{\rightarrow} R_1', E_1'$, $R_2, E_2 \stackrel{a_2}{\rightarrow} R_2', E_2'$ and $R_1', E_1' \vDash \phi_1$ and $R_2', E_2' \vDash \phi_2$. We can then derive $R, E_1 \times E_2 \stackrel{a_1 a_2}{\rightarrow} R', E_1' \times E_2'$, where $R' = R_1' \circ R_2'$. Clearly, $R', E_1' \times E_2' \vDash \phi_1 * \phi_2$ holds, and therefore so does $R, E_1 \times E_2 \vDash \langle a_1 a_2 \rangle (\phi_1 * \phi_2)$. Since $E \sim E_1 \times E_2$ we have that $R, E \vDash \langle a_1 a_2 \rangle (\phi_1 * \phi_2)$, using Theorem 25. Thus, if $[\![\Gamma]\!] \subseteq [\![\langle a_1 \rangle \phi_1 * \langle a_2 \rangle \phi_2]\!]$ then $[\![\Gamma]\!] \subseteq [\![\langle a_1 a_2 \rangle (\phi_1 * \phi_2)]\!]$.

(E) The structural rule of equivalence makes use of the relation $\equiv$ between bunches. The soundness of this makes essential use of the first result in Lemma 30.

We omit the proofs of the other rules. They are all quite straightforward consequences of the definitions and results we have developed above. □

Notice that this soundness relies upon Theorem 25, the algebraic properties of Lemma 8 and the simple-extension property for transitions. For the process calculus **SCRP** (rather than **SCRPr**) the logic $\mathbf{MBIc}^{-I}$ is sound (but **MBIc** is not).

Corollary 32. *If $\emptyset \vdash \phi$ then $R, E \vDash \phi$ for all closed states R, E.*

Since $R, E \in [\![\emptyset_*]\!]$ entails $R = e$ and $E \sim \mathbf{1}$ we see that we are more interested in the additive theorems (those of the form $\emptyset \vdash \phi$) than in the multiplicative theorems ($\emptyset_* \vdash \phi$).

An algebraic restatement of the soundness result may be made, namely, that the set $\mathcal{P}_\sim(\mathsf{CStates})$ is naturally equipped with the structure of a **BIc**-algebra together with operators for additive and multiplicative modalities. This all follows from the fact that $\mathsf{CStates}$ is a resource monoid and the fact that $\sim$ is contained in the usual bisimulation relation on states familiar from modal logic (Popkorn 1994).

3.4 Characterization of the Logical Equivalence

We have shown above that the global simulation $\sim$ on states is contained in both the logical (semantic) equivalence $\stackrel{\mathbf{MBIc}}{\equiv}$ and the local simulation $\approx$. It is not always the case that the relation $\approx$ on processes is contained in the relation $\stackrel{\mathbf{MBIc}}{\equiv}$ on processes. The following counterexample, which is adapted from (Collinson et al. 2007), also shows that the relation $\approx$ on states is not contained in $\stackrel{\mathbf{MBIc}}{\equiv}$ on states:

Example 33. Consider the resource monoid $(\mathbb{N}, +, 0)$ and the action monoid $Act = \{i^p z^q \mid p, q \in \mathbb{N}\}$ generated freely from two primitive actions z and i. Take

$$\mu(i^p z^q, n) = \begin{cases} n + p & \text{if } p \neq 0 \\ 0 & \text{if } n = p = 0 \\ \uparrow & \text{if } n \neq p = 0 \end{cases}$$

for all $n \in \mathbb{N}$. In particular, write $1 = i^0 z^0$. This defines a modification. The action i is the incrementation and z is 'if zero then tick'.

Consider the processes E and F defined by

$$\begin{array}{lcllcl} E & = & i : E' & \qquad F & = & i : F' \\ E' & = & i : E + z : F & \qquad F' & = & F \end{array}$$

using auxilliary processes E' and F'.

These processes generate transtions of the form

$$n, E \xrightarrow{i} n+1, E' \xrightarrow{i} n+2, E \xrightarrow{i} \ldots$$

$$n, F \xrightarrow{i} n+1, F' \xrightarrow{i} n+2, F \xrightarrow{i} \ldots$$

starting from any n.

Now n, E and n, F have identical operational behaviour for any n, that is, $n, E \approx n, F$. Therefore $E \approx F$ holds. Furthermore $n+1, E' \approx n+1, F'$ holds for all n. However, $0, E'$ and $0, F'$ have distinct operational behaviour since $0, E' \xrightarrow{z} 0, E$, but there is no transition starting from $0, F'$. Therefore $0, E' \not\approx 0, F'$.

Using the interpretation we have $1, F' \not\models I \mathrel{-\!\!*} \langle z \rangle \top$ since $0, \mathsf{1} \models I$ and $1, F' \times \mathsf{1}$ makes no z-transition.

On the other-hand $1, E' \models I \mathrel{-\!\!*} \langle z \rangle \top$ since if $n, G \models I$ then $n = 0$ and $G \sim 1$, and then

$$\frac{0, E' \xrightarrow{z} 0, E \qquad 1, \mathsf{1} \xrightarrow{1} 1, \mathsf{1}}{1, E' \times \mathsf{1} \xrightarrow{z} 1, E \times \mathsf{1}}$$

and $1, E \times \mathsf{1} \models \top$.

We therefore have that

$$0, E \models \langle i \rangle (I \mathrel{-\!\!*} \langle z \rangle \top) \qquad \text{and} \qquad 0, F \not\models \langle i \rangle (I \mathrel{-\!\!*} \langle z \rangle \top)$$

hold.

For standard process algebras like SCCS there is a partial converse to Theorem 25 which says that, under certain conditions, any two logically equivalent processes are also algebraically equivalent.

Define a state R, E to be *image-finite* if it has finitely many immediate derivatives. Define an agent E to be image-finite if R, E is finitely branching for all R. Define a process E with all free variables amongst the n-tuple X to be image-finite if $R, E[\mathsf{G}/\mathsf{X}]$ is finitely branching for all n-tuples of agents G. The following result was shown for **SCRP** in (Pym & Tofts 2006):

Theorem 34. *For any image-finite processes E and F and any resource R, if $R, E \overset{\mathbf{MBIc}}{\equiv} R, F$ then $R, E \approx R, F$ holds. Consequently, if $E \overset{\mathbf{MBIc}}{\equiv} F$ then $E \approx F$ holds.*

Proof. Note that if the the theorem is true then $\overset{\mathbf{MBIc}}{\equiv}$ is contained in $\approx$ on states, and so $\overset{\mathbf{MBIc}}{\equiv}$ must satisfy the closure conditions defining $\approx$.

Suppose for a contradiction that the theorem is false. Then there must be some states R, E and R, F with $R, E \overset{\mathbf{MBIc}}{\equiv} R, F$ and, without loss of generality, some transition $R, E \xrightarrow{a} \mu(a, R), E'$ for some E' and some a, such that there is no F' with both $R, F \xrightarrow{a} \mu(a, R), F'$ and $R, E' \overset{\mathbf{MBIc}}{\equiv} R, F'$.

Let $\mathcal{F} = \{F' \mid R, F \xrightarrow{a} \mu(a, R), F'\}$. If $\mathcal{F}$ is empty then $R, E \models \langle a \rangle \top$ and $R, F \not\models \langle a \rangle \top$, which contradicts $R, E \overset{\mathbf{MBIc}}{\equiv} R, F$. Therefore $\mathcal{F}$ must be non-empty. Since F is image-finite we may enumerate the elements of $\mathcal{F}$ as $F_1, \ldots, F_n$ for some $n \geq 1$. Since $R, E \overset{\mathbf{MBIc}}{\not\equiv} R, F_i$ for every $F_i \in \mathcal{F}$ and **MBIc** has classical negation there is some ϕ_i such that $R, E' \not\models \phi_i$ and $R, F_i \not\models \phi_i$ for $1 \leq i \leq n$. But then $R, E \models \langle a \rangle (\phi_1 \wedge \ldots \wedge \phi_n)$ and $R, F \not\models \langle a \rangle (\phi_1 \wedge \ldots \wedge \phi_n)$. This contradicts $R, E \overset{\mathbf{MBIc}}{\equiv} R, F$ and so $\mathcal{F}$ cannot be non-enpty either. □

The main work in the proof is done by the presence of the additive diamond modality, which allows us to distinguish processes that make different transitions. This is a general fact of modal logic, see (Popkorn 1994) for a detailed explanation. Indeed, the proof will work for any fragment of **MBIc** including $\langle\rangle$, $\wedge$ and $\top$.

It does not seem that an analogous result can be produced for $\sim$, even with the multiplicative connectives. In particular, an equivalence $E \sim F$ makes comparisons of states S, E' where E' is the process component of a derivative R', E' of some state R, E, with $S \neq R'$. However, the multiplicative diamond only gives access to states with resource components that are formed as composites of some resource with the resource component of derivatives. In general, not all resources can be realized as composites. The connectives $-\!\!*$ and $*$ also do not seem to be of any help here.

We find ourselves in the situation having Theorem 25 stated using the relation $\sim$ but no converse. This is somewhat unsatisfactory. Ideally, one would wish to have a single bisimulation relation that matched perfectly with the logical equivalence.

The relation $\approx$ on states seems like the natural way to compare operation behaviour of states and is also intimately related to the soundness of the classical modal connectives. We have seen, however, that it is not always a congruence. This means that it cannot be used to give a logical interpretation of **MBIc** (along the lines of Proposition 26) that supports the connective $-\!\!*$ (it also does not support $\langle\rangle_\nu$).

The failure of congruence for $\approx$ on states is a consequence of the form of the operational rule for synchronous product and the fact that resource composition is not injective. The relation $\approx$ on processes does lead congruence, but is not usually be closed under transitions on states: we can have $E \approx F$ and $R, E \xrightarrow{a} R', E'$ but no F' with $R, F \xrightarrow{a} R', F'$ and $E' \approx F'$. Hence, this relation does not give a version of Theroem 25 featuring the additive or multiplicative modalities.

The relation $\sim$ is a conservative solution to the failure of both notions of $\approx$ above. The use of universal quantification across resource and the fact that it is closed under transitions guarantees that it is a congruence and that Theorem 25 and Theorem 31 hold.

The logic **MBIc** speaks about composition and decomposition of states and resources as well as operational behaviour. Thus one should perhaps expect that a bisimulation relation that matches $\overset{\mathbf{MBIc}}{\equiv}$ should compare more than just the operational behaviour of states — it should also compare composition and decomposition of states and resources.

The line of work (Sewell 1998, Leifer & Milner 2000, Sassone & Sobociński 2003) has developed methods for designing labelled transition systems for process calculi that satisfy general forms of bisimulation. We do not know if such methods can be used to redesign the labelled transition system of **SCRP** in such a way as to make $\approx$ better behaved, or to find a suitable alternative to $\sim$.

3.5 Quantification

The above system is purely propositional. In contrast, in (Pym & Tofts 2006, Pym & Tofts 2007, Collinson et al. 2007) quantification was shown to be extremely useful, in particular for describing the resource-hiding restriction mechanism of **SCRP**. In this section we consider an extension, **MBIq**, of our previous logical system, **MBIc**, with such quantification.

It turns out that what we need for the discussion of hiding is quantification over an action. Thus we assume a countable set, ActVar of *action variables*, ranged over by x, and a constant symbol a for each action a of **SCRPr**. Let $\mathsf{A} = \mathsf{ActVar} \cup \mathsf{Act}$ and let a range over this set. We assume a given set of function symbols on actions, each with some chosen arity. The *terms* t of the language are then formed in the standard way (variables and constants are terms, functions applied to terms are terms). We assume a given set of *relations* on the set of actions, each with a given arity. We assume the equality relation $=$ between terms to be included in this set. Atomic formulae φ consist of all instances of relations, that is, if p is a relation symbol of arity n and $t_1, \ldots, t_n$ are terms, then $p(t_1, \ldots, t_n)$ is an atomic formula. The formulae of the language **MBIq**

$R, E \vDash p(t_1, \ldots, t_n)\eta$ iff $(t_1\eta, \ldots, t_n\eta, (R, E)) \in \mathcal{V}(p)$

$R, E \vDash (\exists x.\phi)\eta$ iff $\exists a \in \mathsf{Act}.\ R, E \vDash \phi[a/x]\eta$

$R, E \vDash (\forall x.\phi)\eta$ iff $\forall a \in \mathsf{Act}.\ R, E \vDash \phi[a/x]\eta$

$R, E \vDash (\exists_\nu x.\phi)\eta$ iff $\exists (S, F) \in \mathsf{CStates}.\ \exists a \in \mathsf{Act}.$
$R, E \sim R, \nu S.F$ and $R \circ S \downarrow$ and $\mu(a, S) \downarrow$ and $R \circ S, F \vDash \phi[a/x]\eta$

$R, E \vDash (\forall_\nu x.\phi)\eta$ iff $\forall (S, F) \in \mathsf{CStates}. \forall a \in \mathsf{Act}.$
$R, E \sim R, \nu S.F$ and $R \circ S \downarrow$ and $\mu(a, S) \downarrow$ implies $R \circ S, F \vDash \phi[a/x]\eta$

Figure 7: Interpretation of **MBIq**

are then as follows

$$\phi ::= \quad \varphi \mid \top \mid \bot \mid \phi \to \phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid I \mid \phi * \phi \mid \phi \mathrel{-\!\!*} \phi$$

$$\mid [\mathsf{a}]\phi \mid \langle \mathsf{a} \rangle \phi \mid [\mathsf{a}]_\nu \phi \mid \langle \mathsf{a} \rangle_\nu \phi \mid \exists x.\phi \mid \forall x.\phi \mid \exists_\nu x.\phi \mid \forall_\nu x.\phi\ .$$

Notice that we now have modalities $\langle x \rangle$, $[x]$, $\langle x \rangle_\nu$ and $[x]_\nu$ labelled by variables (in this case x) as well as action constants. The additive quantifiers $\exists$, $\forall$ and the multiplicative quantifiers $\exists_\nu$, $\forall_\nu$ bind free action variables (in the usual way). The *sentences* are just the formulae without free variables. For any formula ϕ, let $\phi[t_1/x_n, \ldots, t_n/x_n]$ be the formula formed by replacing each occurrence of each variable x_i by the term t_i.

A valuation $\mathcal{V}$ for the language above is fixed by choosing a relation $\mathcal{V}(p) \subseteq \mathsf{Act}^n \times \mathsf{CStates}$ for each relation symbol p of arity n and an n-ary function on Act for each n-ary function symbol. In particular, $\mathcal{V}(=) = \{(a, a, (R, E)) \mid a \in \mathsf{Act}, (R, E) \in \mathsf{CStates}\}$. Each set $\mathcal{V}(p)$ must be closed under the relation $\sim$. An *assignment*, η, is a function from ActVar to Act. For any η, let $\eta[a/x]$ be the asssignment that is identical to η, except that $\eta(x) = a$. Constants are interpreted as themselves at any assignment. A variable x is interpreted as $\eta(x)$ at any assignment η. Compound terms are interpreted at an assignment by applying the interpretation of the outermost function symbol to the interpretation of sub-terms: thus all terms denote actions. A valuation is then extended to an interpretation of formulae as in Figure 7 with the understanding that the interpretation of all other formulae follows the pattern in Figure 6. In particular, the interpretations of $\langle x \rangle$, $[x]$, $\langle x \rangle_\nu$, $[x]_\nu$ follow from those of $\langle a \rangle$, $[a]$, $\langle a \rangle_\nu$, $[a]_\nu$, respectively, by replacing all occurrences of a by x.

There are a number of special cases of the above set-up that are particularly important. The first of these is the case where there are no function symbols, so that the only terms are the variables and (constant) actions. A second recovers atomic propositions that are independent of action (as in **MBIc**) by including relations of arity 0 (note that these are distinct from action constants). Notice that in this case, quantification is only over actions that occur as labels of modalities.

The situation in (Pym & Tofts 2006, Pym & Tofts 2007, Collinson et al. 2007) follows the first of the special cases (no function symbols). In addition, only a special form of relation is allowed. These are defined (with a slight change of notation) to be those susceptible to an interpretation

$$R, E \vDash p(a_1, \ldots, a_n) \ \text{ iff } \ \forall 1 \leq i \leq n.\ \mu(a_i, R) \downarrow \ \text{ and } \ [\![p]\!](R, \ldots, R)$$

where $[\![p]\!] \subseteq \mathbf{R}^n$ is an n-ary relation on resources associated with the relation symbol p. For a relationship $R, E \vDash \phi$ for a formula ϕ of the form $\exists x.\langle x \rangle \phi$ or $\exists x.p(x)$, any witness a for x will clearly satisfy $\mu(a, R) \downarrow$. Similarly, for formulae of the form $\forall x.\langle x \rangle \phi$ we only need to verify $\phi[a/x]$ whenever $\mu(a, R)$ is defined. A slighly different formulation of the additive quantifiers was presented in (Pym & Tofts 2006, Pym & Tofts 2007, Collinson et al. 2007) in which these

conditions $\mu(a, R) \downarrow$ were explicitly included. We have preferred the slightly more general version here. This is likely to have cleaner proof rules.

The multiplicative quantifiers $\exists_\nu$ is intended to characterize hiding in the same sense that $*$ characterizes synchronous product. Examples of the use of multiplicative quantification are given in (Pym & Tofts 2006, Pym & Tofts 2007).

Example 35. The privacy example contained in (Pym & Tofts 2007) can, in fact, be illustrated through the combination of the two special cases above. This example is a continuation of the Example 24 above. The resources used by the evolving state $R_1 \circ R_2, E_1 \times E_2$ may be hidden to form some new state

$$\{e\}, \nu(R_1 \circ R_2).(E_1 \times E_2)$$

in which the resources are no longer externally visible. If this state evolves (because it meets the side-conditions on the transition rule for hiding processes) then it does so in the same way as before, but with two differences. First, the evolution of $R_1 \circ R_2$ is no longer externally invisible. Second, the state now evolves along the action $\nu(R_1 \circ R_2).a$ so that a itself may not be externally observable. In the case of the choice of ν from Equation 3 above only the atoms of a that do not use $R_1 \circ R_2$ are externally visible — it may be helpful to think of $\nu(R_1 \circ R_2).a$ as being an encrypted version of a.

Suppose that the state $\{e\}, \nu(R_1 \circ R_2).(E_1 \times E_2)$ does evolve. Then we have the satisfaction relation

$$\{e\}, \nu(R_1 \circ R_2).(E_1 \times E_2) \vDash \exists_\nu x.\langle x \rangle(\phi_1 * \phi_2)$$

since $R_1 \circ R_2, E_1 \times E_2 \vDash \langle go_{E_1} go_{E_2} \rangle(\phi_1 * \phi_2)$ holds. Furthermore, the fact that $\exists_\nu$ is used, forces the process to be a hide (resource restriction), at least up to global bisimulation, and the action x (instantiated by $go_{E_1} go_{E_2}$) to be such that the hidden resource enables it to act.

Write,

$$R, E \overset{\mathbf{MBIq}}{\equiv} R, F \text{ iff } \forall \phi \,.R, E \vDash \phi \iff R, F \vDash \phi$$

for any closed states R, E and R, F. Write $E \overset{\mathbf{MBIq}}{\equiv} F$ whenever $R, E \overset{\mathbf{MBIq}}{\equiv} R, F$ for all R. The appropriate extension of Theorem 25 holds.

Theorem 36. *Let E and F be closed states. If $E \sim F$ then $E \overset{\mathbf{MBIq}}{\equiv} F$ also holds.*

Proof. The proof is an extension of the induction given for Theorem 25. The new clauses in the proof are almost trivial given the interpretation above. The valuations of atoms are assumed to be closed under $\sim$. The induction steps at the additive quantifiers are straightforward. The steps for the multiplicative quantifiers only use the fact that $\sim$ is an equivalence relation. □

We have not considered any proof rules for the quantifiers of **MBIq**. It seems that the additives should satisfy the standard rules for first-order quantification and that these should be susceptible to the usual semantics using indexed categories following (Lawvere 1969). How to do this for the multiplicatives and how to produce an appropriate extension of Theorem 31 is completely open.

It may well be the case that further enrichments of the calculus with, for example, a resource sort, function symbols for partial functions and equality would prove fruitful. Clearly, they would approximate more closely the level of detail involved in the satisfaction relation $\vDash$, in particular allowing us to deal directly with properties of modifications within the logic.

4 Ordered SCRP and Intuitionistic MBI

None of the members of the **SCRP** or **MBI** families of calculi have so far made any use of the order on resource monoids. In this section we will develop calculi that are sensitive to the order. The process calculus **OSCRP** deals with actions that are performed just when they have sufficient resources. A special version of such a calculus was already considered in (Pym & Tofts 2006), using an *enabling function*, $\rho : \mathsf{Act} \longrightarrow \mathbf{R}$ to specify the minimum resources required for an action

to fire. This kind of calculus was suitable for most of the modelling situations under consideration but was difficult to reconcile with a Hennessy-Milner logic with classical modalities. We now show that the appropriate logic for reasoning about **OSCRP** is a modal logic with an intuitionistic proof system and semantics. Again, we consider only the propositional part, and call the new logic **MBIi**.

The use of order in both the process calculus and the logic has considerable practical advantages. For example, in a system that automatically generates the transition system associated with a resource-process state the computationally-hard part can be determining the evolution of resource under an arbitrary modification. The use of an order can substantially simplify these calculations. In a similar way, a model-checker that attempts to automatically verify assertions of **MBI** against states must, in general, deal with unbounded searches across infinite resource spaces. Suitable structuring using the order and modification can bound this search and therefore yield a better terminating model-checker.

We begin with a special resource monoid $\mathbf{R} = (\mathbf{R}, \circ, e, \sqsubseteq)$ and an action monoid Act. A modification, μ, is a partial function satisfying the coherence conditions (for **SCRPr**) previously stated, but we suppose that it also satisfies both of the monotonicity conditions:

1. if $\mu(a, R) \downarrow$ and $R \sqsubseteq S$ then $\mu(a, R) \sqsubseteq \mu(a, S)$
2. if $\mu(a, R \circ S) = \mu(\nu S.a, R) \circ S'$ and $R \sqsubseteq T$ then $\mu(a, T \circ S) = \mu(\nu S.a, T) \circ S'$

for all actions a and resources R, R', S, S'.

It is easily verified that Lemma 3 concerning the extension of resources and modifications holds in this new setting. An additional observation about the piggybacking relation and the order is needed: the proof is immediate.

Lemma 37. *If the resource monoid is special, $S \Subset R$ and $R \sqsubseteq T$ then $S \Subset T$ holds.*

The process terms are precisely the same as for **SCRPr**, as are the operational rules, with the exception of the product, which becomes

$$\frac{R, E \xrightarrow{a} R', E' \qquad S, F \xrightarrow{b} S', F'}{T, E \times F \xrightarrow{ab} \mu(ab, T), E' \times F'} \qquad (R \circ S \sqsubseteq T)$$

for all appropriate R, S, T, a, b, E, E', F, F'.

As before, the form of the rules means that the only resource that appears in the target of a transition is the modification of the resource in the source of that transition. That is, Lemma 2 holds in this setting. The rule for prefix and the monotonicity conditions on μ means that if the state $R, a : E$ makes a transition and $R \sqsubseteq S$ then so does $S, a : E$. In fact, we have the following *order-extension property* for transitions:

Lemma 38. *If $R, E \xrightarrow{a} \mu(a, R), E'$ and $R \sqsubseteq R'$ then $R', E \xrightarrow{a} \mu(a, R'), E'$ is also derivable.*

Proof. The proof is much as one would expect, that is, by induction on the derivation of the process term E. The prefix case holds for the reasons noted above. The sum case is straightforward. The product case is taken care of through by the order in the side-condition. The hiding case follows because of the new condition on modifications regarding hiding actions and the monotonicity condition for the predicate $\Subset$ noted in Lemma 37. □

The simple-extension property for transitions, Lemma 4, holds in this new setting, and the proof is much as before. The product case now makes use of the bifunctoriality condition on the resource monoid and the hiding case makes use of the predicate $\Subset$.

The global bisimulation relation $\sim$ is defined precisely as before. That is, it takes no account of the order. On the other-hand, if we wished to work with the local variant then we would have to place some compatibility constraints between it and the order. The global bisimulation is shown to be an equivalence relation and a congruence as before (Lemma 6) with only minor alterations

$R, E \vDash \varphi$ iff $R, E \in \mathcal{V}(\varphi)$

$R, E \vDash \phi \to \psi$ iff $\forall S.\ R \sqsubseteq S$ and $S, E \vDash \phi$ implies $S, E \vDash \psi$

$R, E \vDash I$ iff $e \sqsubseteq R$ and $1 \sim E$

$R, E \vDash \phi_1 * \phi_2$ iff $\exists R_1, R_2, E_1, E_2.\ R_1 \circ R_2 \sqsubseteq R$ and $E \sim E_1 \times E_2$
and $R_1, E_1 \vDash \phi_1$ and $R_2, E_2 \vDash \phi_2$

$R, E \vDash [a]\phi$ iff $\forall S, S', E'.\ R \sqsubseteq S$ & $S, E \xrightarrow{a} S', E'$ implies $S', E' \vDash \phi$

$R, E \vDash \langle a \rangle \phi$ iff $\forall S.\ R \sqsubseteq S$ implies $\exists S', E'.\ S, E \xrightarrow{a} S', E'$ and $S', E' \vDash \phi$

$R, E \vDash [a]_\nu \phi$ iff $\forall S, T, S', E'.\ R \sqsubseteq S$ & $S \circ T, E \xrightarrow{a} S', E'$ implies $S', E' \vDash \phi$

$R, E \vDash \langle a \rangle_\nu \phi$ iff $\forall S.\ R \sqsubseteq S$ implies $\exists T, S', E'.\ S \circ T, E \xrightarrow{a} S', E'$ and $S', E' \vDash \phi$

Figure 8: Interpretation of MBIi

to the product clause. The simple algebraic properties of Lemma 8 continue to hold: in particular, the clause that says 1 is a unit for $\times$ makes use of Proposition 4 and Lemma 38; the associativity of $\times$ makes use of the new monotonicity condition on composition.

We introduce a logic **MBIi** of intuitionistic modal propositional **MBI**. This has the same connectives as **MBIc** and all of the same rules except for (RAA), $(\neg[]\neg 2)$, $(\langle\rangle \vee 2)$, $(\langle 1\rangle[1])$, $(\langle\rangle_\nu I)$, $(\neg[]_\nu\neg 2)$, $(\langle\rangle_\nu \vee 2)$, $(\langle\rangle\langle\rangle_\nu)$, which are omitted.

4.1 Interpretation

Define a preorder on states by

$$R, E \sqsubseteq S, F \qquad \text{iff} \qquad R \sqsubseteq S \quad \text{and} \quad E = F$$

for all E, F, R, S. Let the set of all upper sets amongst the states be $\Upsilon(\mathsf{States})$ and the set of all $\sim$-closed upper sets be $\Upsilon_\sim(\mathsf{States})$.

A valuation of atomic propositions is taken to be a map

$$\mathcal{V} : \mathsf{Prop}_0 \longrightarrow \Upsilon_\sim(\mathsf{CStates})$$

from the set of atomic propositions, Prop_0 to the $\sim$-closed upper sets of closed states. For any given valuation, $\mathcal{V}$, of the atomic propositions, the language **MBIi** is given an interpretation on closed states that makes use of the order on the resource monoid. We omit the interpretation of the $\wedge$, $\vee$ and $-\!\!*$ connectives as they remain unchanged from Figure 6. The recursive definition of the interpretation has been designed so as to maintain an important invariant.

Lemma 39. *Every proposition ϕ has an interpretation $[\![\phi]\!] = \{R, E \in \mathsf{CStates} \mid R, E \vDash \phi\}$ which is an upper set with respect to the order $\sqsubseteq$ on states.*

Write $R, E \overset{\mathbf{MBIi}}{\equiv} S, F$ whenever given states R, E and S, F satisfy exactly the same **MBIi**-formulae. Write $E \equiv F$ whenever the processes E and F satisfy the same formulae at all R.

Theorem 40. *If $E \sim F$ then $E \overset{\mathbf{MBIi}}{\equiv} F$.*

The proof is essentially as in the classical case, with a few easy modifications because of the new interpretation. This shows that the set $[\![\phi]\!]$ is $\sim$-closed for each ϕ. Thus, an interpretation is a function

$$[\![-]\!] : \mathsf{Prop} \longrightarrow \Upsilon_\sim(\mathsf{States})$$

given any valuation.

Lemma 29, which says that interpretation of substitution in a context is monotonic, holds in this setting. Lemma 30 also holds, by a proof that makes use of Theorem 40. The proof of soundness is also a trivial modification of the discrete version (Proposition 31). In particular, we have retained the important systems rule ($\langle - \rangle *$) through our set-up.

Proposition 41. *The calculus* **MBIi** *has a sound interpretation on* States.

Quantifiers may easily be included in the system **MBIi**: the interpretation of quantifiers in **MBIc** are modified in the obvious way so that they become upper sets of states. All of the results above then continue to hold. Just as in the discrete case, it is the extension properties that make the unit axioms for the logic work. Here, however, we also need the order-extension property to get the associativity of $\times$. Thus, in situations where it is not appropriate to use the piggybacking condition we must have a variant calculus in which this extension property is explicitly included as a structural transition rule. Note that when the order $\sqsubseteq$ on resources is taken to be discrete **OSCRP** reduces to **SCRPr**, and the interpretation of formulae of **MBIi** are identical to their interpretation in **MBIc**. Thus the discrete versions are special cases of the ordered versions. Furthermore, if the resource monoid is special then we may give an interpretation using downwards closed sets of states.

Example 42. The fact that all propositions are upper sets can be very useful for model-checking, since, for example, sometimes it suffices to verify that the given state has sufficient resource. Consider, for example, a language with just increment, i, and decrement, d, operations over the resource monoid consisting of the natural numbers with their usual ordering. If we take

$$\mu(d^m i^n, p) = \begin{cases} p + n - m & \text{if } m \leq p \\ \uparrow & \text{if } m > p \end{cases}$$

then μ is a modification in the sense of this section. Owing to the particular properties of this resource monoid, we do not get any more transitions than we did for the unordered calculus. However, we may take typical atomic propositions from the logical language to make assertions like ϕ_n, where this says 'the resource component of the given state is greater than n'. Propositions can then be checked at states by combinations of computable order-assertions. A proposition of the form $\langle ab \rangle (\phi_m * \phi_n)$ can be checked by finding a witness for $\langle a \rangle \phi_m * \langle b \rangle \phi_n$ at a state with a sufficiently large resource component. For example, suppose we are given a state $3, (d : E) \times (i : F)$. Then we find that this state satisfies $\langle d \rangle \phi_2 * \langle i \rangle \phi_1$ and $\langle di \rangle (\phi_2 * \phi_1)$. Moreover, any state $p, (d : E) \times (i : F)$ with $p \geq 3$ will also satisfy these propositions.

5 Definable Extensions

The calculus SCCS is so powerful that it can be used to capture many other frameworks for concurrent modelling and computation. Indeed, a remarkable functional completeness result exists (Simone 1985) which shows that all concurrent behaviour that can be described by calculi with operational rules of a certain form are already captured by SCCS. Nevertheless, specialized calculi remain very interesting in applications, and it is an extremely pleasing aspect of (S)CCS that it may be used to give an unambiguous semantics to these calculi, see (Milner 1980, Milner 1983, Milner 1989). In this section we show how similar definability results are possible for our resource-based process calculi.

For the purposes of this section we use a synchronous calculus **SCRP**ν which is a slightly modified version of **SCRP**. In Section 5.2, however, we employ a refinement **SCRP**ν called **SCRPr**ν, which extends **SCRPr** with the Hide-id rule of **SCRP**ν. This calculus is required in order to develop the theory of equivalence of asynchronous processes. The remaining sections revert to **SCRP**ν because the programming languages considered there do not naturally translate into a system with strong modfication, such as **SCRPr**ν.

The first additional requirement we make is that the hiding function on actions satisfies

$$\nu S.1 = 1$$

for all resources S. The example defined in Equation (3) above has this property.

This calculus **SCRP**ν has the same grammar as **SCRP** but one additional operational rule

$$\text{Hide-id} \qquad \frac{}{R, \nu S.E \xrightarrow{1} R, \nu S.E}$$

is added to the operational semantics of Figure 1. Thus any hiding process may always tick given any resource: in particular this is the case for $R, \nu S.E$ even when $R \circ S$ is undefined. Of course, if E was a process that could tick and composition is total then there is no new transition of $\nu S.E$ given by the rule Hide-id.

The Hide-id rule will be essential for the encoding of asynchrony below. The inclusion of this rule can be compared with the restriction operator of SCCS where in any restriction $E \restriction A$ the set A must contain the tick action 1.

We use the name **SCRPr**ν to refer to **SCRP**ν-calculi with a modification function μ which satisfies the same coherence conditions as **SCRPr** (rather than just the weaker conditions for **SCRP**).

It is straightforward to show that Lemma 1, Propositions 4, 6 and Lemmas 7, 8 hold in **SCRPr**ν. The proofs are by minor modifications of those for **SCRPr**.

5.1 Idleness and Delay

The key fact that lies behind the fact that the synchronous formalism of processes encodes the asynchronous formalism is the definability of processes that may tick (perform the identity action) for arbitrary finite time before performing any other action.

We begin by defining the extremely important *delay operator*, δ, that takes a process and produces another process that may wait arbitrarily long before making any non-identity action. For any E, we take

$$\delta(E) = fix X.(1 : X + E)$$

to be the delayed process. Note that it may still perform an E-action immediately and that if E is an agent then so is $\delta(E)$. The delay operator satisfies two derived rules

$$\frac{}{R, \delta(E) \xrightarrow{1} R, \delta(E)} \qquad \frac{R, E \xrightarrow{a} \mu(a, R), E'}{R, \delta(E) \xrightarrow{a} \mu(a, R), E'}$$

for all actions a, resources R and processes E. The following lemma is proved by routine verifications using the definition of $\sim$ and the transition rules:

Proposition 43. *The delay operator satisfies the following equalities and inequalities:*

$$\delta(E) \sim \delta(\delta(E)) \sim E + 1 : \delta(E) \sim E + \delta(E)$$

$$\delta(E) \times \delta(F) \sim \delta((E \times \delta(F)) + (\delta(E) \times F))$$

$$\delta(\nu S.E) \lesssim \nu S.\delta(E)$$

for all processes E, F and resources S. If the partial function $R \circ - : \mathbf{R} \longrightarrow \mathbf{R}$ is injective for every $R \in \mathbf{R}$ then

$$\nu S.\delta(E) \sim \delta(\nu S.E)$$

holds for all resources S and processes E.

We omit the proof (which uses the standard technique) but note that the property $\nu S.1 = 1$ is required for the relations involving hiding processes.

A state R, E is said to be *idle* if $R, E \approx R, \delta(E)$ A process E is said to be idle if R, E is idle for every resource R.

Proposition 44. *For any process E:*

1. *A state R, E is idle iff $R, E \xrightarrow{1} R, E$;*
2. *A process E is idle iff $E \approx \delta(E)$;*
3. *A process E is idle iff $E \sim \delta(E)$.*

Proof. The first property is immediate by definition of δ. The second is immediate by definition of $\approx$ on processes. Consider the third point. If $E \sim \delta(E)$ then $E \approx \delta(E)$ since Lemma 7 holds here, so we may apply the second point and conclude that E is idle. Now suppose that E is idle. If $R, E \xrightarrow{a} R, F$ then $R, \delta(E) \xrightarrow{a} R, F$ and $F \sim F$. If $R, \delta(E) \xrightarrow{a} R, F$ then either $R, E \xrightarrow{a} R, F$ or $a = 1$ and $F = \delta(E)$. In the first case there is nothing to show. In the second we have $R, E \xrightarrow{1} R, E$. Hence $E \sim \delta(E)$. □

Lemma 45. *For all processes E and F:*

1. *$\delta(E)$ is idle;*
2. *$\nu S.E$ is idle;*
3. *If E and F are idle then $E \times F$ is idle.*

Proof. All three points follow from Lemma 44. Point (1) is immediate by reflexivity of $\approx$. Point (2) is immediate by the Hide-id rule. Point (3) follows by the operational rule for product processes since any resource R decomposes into a pair (R, e) with $R \circ e = R$. □

Recall that in (Milner 1983) a process is said to be asynchronous if every proper derivative is idle. We make an appropriate change to this for **SCRP**. We define a state to be *asynchronous* if every proper derivative of that state is idle. A process E is *asynchronous* when all E-states are asynchronous.

5.2 Asynchronous Prefix

For the purposes of this section we assume that the action monoid Act contains countably many tick actions (actions b such that $\mu(b, R) = R$ for all resources R). This is mostly harmless in the light of the following proposition:

Proposition 46. *Let $\mathbf{R}$ be a resource monoid, Act be an action monoid, μ be a modification, and ν be a hiding function. Let B be any set disjoint from Act. The action set may be freely extended with B giving a new action monoid Act', a new modification $\mu' : \mathsf{Act}' \times \mathbf{R} \longrightarrow \mathbf{R}$ and a new hiding $\nu' : \mathbf{R} \times \mathsf{Act}' \longrightarrow \mathsf{Act}'$ such that:*

- *mu' and ν' agree with mu and nu, respectively on Act;*
- *B contains only tick actions.*

Proof. Let Act' be the free monoid of words over the set $\mathsf{Act} \cup B$. Words are written in the form $\langle x_1, \ldots, x_n \rangle$, the unit word is $\langle\rangle$, and multiplication is concatenation of words. For any word w let $\circledast w = \prod(x_i \mid w \equiv \langle x_1, \ldots, x_n \rangle \;\&\; 1 \leq i \leq n \;\&\; x_i \notin B)$ be the action of Act formed by forgetting all letters from B and replacing formal products by products of Act. In particular $\circledast\langle\rangle = 1$, the unit of Act. Define

$$\begin{aligned} \mu'(w, R) &\simeq \mu(\circledast w, R) \\ \nu' S.w &\simeq \nu S.(\circledast w) \end{aligned}$$

for all words w, and resources R and S.

For any S, $\nu' S.\langle\rangle = \nu S.1 = 1$, so ν' is a hiding (in the sense of this section).

Suppose that $\mu'(v, R)$, $\mu'(w, S)$ and $R \circ S$ are all defined. Then $\mu(\divideontimes v, R)$, $\mu(\divideontimes w, S)$ and $\mu((\divideontimes v)(\divideontimes w), R \circ S)) = \mu(\divideontimes v, R) \circ \mu(\divideontimes w, S)$ are defined. Now $\divideontimes(vw) = (\divideontimes v)(\divideontimes w)$ so

$$\begin{array}{rcl} \mu'(vw, R \circ S) & = & \mu(\divideontimes(vw), R \circ S) \\ & = & \mu((\divideontimes v)(\divideontimes w), R \circ S) \\ & = & \mu(\divideontimes v, R) \circ \mu(\divideontimes w, S) \\ & = & \mu'(v, R) \circ \mu'(w, S) \end{array}$$

and μ' is a modification.

Further, $\mu'(\langle b\rangle, R) = \mu(1, R) = R$, for every $b \in B$ and $R \in \mathbf{R}$. □

We now define the asynchrous calculus **ASCRP**. This calculus is the analogue for **SCRP** of the calculus ASCCS presented in (Milner 1983).

The grammar of the new language is

$$E ::= X \mid a.E \mid \delta(E) \mid \sum_{i \in I} E_i \mid E \times E \mid \nu R.E \mid \mathit{fix}_i \mathsf{X}.\mathsf{E}$$

where the notation for variables and processes is as before, and this calculus is to be considered relative to a fixed $(\mathsf{Act}, \mathbf{R}, \mu, \nu)$. The operator δ could have been omitted since it is definable. We take $\mathsf{0}$ to be the sum of an empty set of processes and $\mathsf{1}$ to be $\delta(\mathsf{0})$. This will turn out to be equivalent to $\mathit{fix} X.(1.X)$.

We give a semantics to this calculus by translating into **SCRP**ν over the same $(\mathsf{Act}, \mathbf{R}, \mu, \nu)$. Define this translation $\mathcal{T}_{as}$: **ASCRP** $\longrightarrow$ **SCRP**ν recursively, by the clause

$$\mathcal{T}_{as}(a.E) \quad = \quad a : \delta(\mathcal{T}_{as}(E))$$

together with clauses such that $\mathcal{T}_{as}$ passes through all the other process combinators: $\mathcal{T}_{as}(X) = X$, $\mathcal{T}_{as}(\delta(E)) = \delta(\mathcal{T}_{as}(E))$, $\mathcal{T}_{as}(\sum E_i) = \sum \mathcal{T}_{as}(E_i)$, $\mathcal{T}_{as}(E \times F) = \mathcal{T}_{as}(E) \times \mathcal{T}_{as}(F)$, $\mathcal{T}_{as}(\nu R.E) = \nu R.\mathcal{T}_{as}(E)$, $\mathcal{T}_{as}(\mathit{fix}_i \mathsf{X}.\mathsf{E}) = \mathit{fix}_i \mathsf{X}.\mathcal{T}_{as}(\mathsf{E})$.

The semantics of **ASCRP** is defined by the operational rules inherited from **SCRP**ν. To be precise the transitions of an **ASCRP** state R, E are induced from **SCRP**ν-transitions of the form

$$R, \mathcal{T}_{as}(E) \xrightarrow{a} R', E'$$

where R', E' is a **SCRP**ν-state. We then have the following proposition to show that the calculus **ASCRP** contains only asynchronous processes and is suitably closed under transitions:

Proposition 47. *If E is an agent of* **ASCRP** *and $R, \mathcal{T}_{as}(E) \xrightarrow{a} R', E'$ for some R, R' then E' is the translation of an agent of* **ASCRP** *and E' is idle. In particular R', E' is idle.*

Proof. The proof is by induction on the structure of the agent E of **ASCRP**. The prefixing case itself is a simple consequence of the properties of δ. The Hide-id rule ensures that the derivative of any hiding is idle. The other details are all routine. □

As an example of the above translations at work, observe that an **ASCRP**-state of the form $R, a.b.1$ gives a sequence of **SCRP**ν-transitions

$$R, a : \delta(b : \delta(1)) \xrightarrow{a} \mu(a, R), \delta(b : \delta(1)) \xrightarrow{b} \mu(b, \mu(a, R)), \delta(1)$$

when $\mu(b, \mu(a, R))$ is defined, making use of the Prefix rule in **SCRP** and one of the derived rules for δ.

It is often inappropriate to use either of the relations $\sim$ and $\approx$ to compare agents in definable asynchronous calculi since agents that delay by differing times should not necessarily be distinguished. In particular, the unit action 1 should be invisible from the point of view of the

equivalence. The remainder of the work in this subsection is concerned with the study of equivalence relations for asynchrounous processes. For this we suppose that the translation is, in fact, into a member of **SCRPr**ν.

For any set X let X^* be the set of all finite words on X. Let $u = \langle a_1, \ldots, a_i, a_{i+1}, \ldots, a_n \rangle \in \mathsf{Act}^*$. For any **SCRPr**$\nu$-states R, E and R', E' write

$$R, E \stackrel{u}{\Longrightarrow} R', E'$$

just when there is some sequence of transitions

$$R, E(\stackrel{1}{\rightarrow})^* \stackrel{a_1}{\rightarrow} (\stackrel{1}{\rightarrow})^* \ldots (\stackrel{1}{\rightarrow})^* \stackrel{a_n}{\rightarrow} (\stackrel{1}{\rightarrow})^* R', E'$$

or in other words, if there is a sequence of transitions along a_1 to a_n in order, but interspersed by arbitrarily many tick actions. In particular $R, E \stackrel{\langle\rangle}{\Longrightarrow} R, E$ where $\langle\rangle$ is the empty word. We write $R, E \stackrel{a}{\Longrightarrow} R', E'$ when $R, E \stackrel{u}{\Longrightarrow} R', E'$ with $u = \langle a \rangle$. We write $R, E \stackrel{a}{\rightarrow\!\!\Longrightarrow} R', E'$ for a sequence of transitions that begins with an a-transition and ends with a sequence of transitions from u (interspersed by ticks).

Define the relation $\sim_{\mathrm{a}}$ to be the largest binary relation on **SCRPr**ν agents such that whenever $R, R' \in \mathbf{R}$, $u \in (\mathsf{Act} \setminus \{1\})^*$ and $E \sim_{\mathrm{a}} F$:

- if there is some E' with $R, E \stackrel{u}{\Longrightarrow} R', E'$ then there is some F' with $R, F \stackrel{u}{\Longrightarrow} R', F'$ and $E' \sim_{\mathrm{a}} F'$;
- if there is some F' with $R, F \stackrel{u}{\Longrightarrow} R', F'$ then there is some E' with $R, E \stackrel{u}{\Longrightarrow} R', E'$ and $E' \sim_{\mathrm{a}} F'$.

The relation is then extended to all processes in the standard way (by substitution). Any relation contained in $\sim_{\mathrm{a}}$ is said to be a *weak global bisimulation* or WG-bisimulation, for short.

This relation is an equivalence. Furthermore any pair of processes which differ only by a number of ticks inserted as prefixes will be equivalent under this relation. However, such pairs of processes can be distinguished by the use of the synchronous parallel composition $\times$. The sequence of results that follow study a congruence formed from $\sim_{\mathrm{a}}$ by closing under substitution in all asynchronous contexts. These results are analogues for **ASCRP** of those in the theory of ASCCS in (Milner 1983), Section 8, pp. 296–301. The proof of the first proposition below is just an unwinding of the definition of WG-bisimulation. The second, third and fourth are implied by the first.

Proposition 48. *A relation $\sim'$ is contained in $\sim_{\mathrm{a}}$ iff whenever $R \in \mathbf{R}$ and $E \sim' F$, then*

- *if $R, E \stackrel{a}{\rightarrow} \mu(a, R), E'$ for some E' then either $a = 1$ and $E' \sim' F$ or there is some F' with $R, F \stackrel{a}{\Longrightarrow} \mu(a, R), F'$ and $E' \sim' F'$,*
- *if $R, F \stackrel{a}{\rightarrow} \mu(a, R), F'$ for some F' then either $a = 1$ and $E \sim' F'$ or there is some E' with $R, E \stackrel{a}{\Longrightarrow} \mu(a, R), E'$ and $E' \sim' F'$.*

Proposition 49. *The relation $\sim$ is contained in $\sim_a$.*

Proposition 50. *Let E be an agent of* **ASCRP**. *Then*

$$\mathcal{T}_{as}(E) \sim_{\mathrm{a}} 1 : \mathcal{T}_{as}(E) \sim_{\mathrm{a}} \mathcal{T}_{as}(1.E) \sim_{\mathrm{a}} \delta(\mathcal{T}_{as}(E)) \ .$$

Proposition 51. *If $\mathcal{T}_{as}(E) \sim_{\mathrm{a}} \mathcal{T}_{as}F$ then*

$$\begin{array}{rcl} \mathcal{T}_{as}(a.E) & \sim_{\mathrm{a}} & \mathcal{T}_{as}(a.F) \\ \mathcal{T}_{as}(\delta(E)) & \sim_{\mathrm{a}} & \mathcal{T}_{as}(\delta(F)) \\ \mathcal{T}_{as}(\nu S.E) & \sim_{\mathrm{a}} & \mathcal{T}_{as}(\nu S.F) \ . \end{array}$$

The simple-extension property of Proposition 4 together with Proposition 48 gives the following result (which also follows from Proposition 49):

Lemma 52. *The relation*

$$E \sim_{\mathrm{a}} E \times 1$$

holds for all **SCRPr**ν*-processes* E*.*

Lemma 53. *Suppose that* E*,* F *and* G *are idle processes of* **SCRPr**ν*. If* $E \sim_{\mathrm{a}} F$ *then* $E \times G \sim_{\mathrm{a}} F \times G$*.*

Proof. The characterization of Proposition 48 is used for the relation $\sim'$ defined by

$$E \times G \sim' F \times G \iff E \sim_{\mathrm{a}} F \text{ and E,F,G idle}$$

and only for such processes. If some transition $R, E \times G \xrightarrow{a} \mu(a, R), E' \times G'$ takes place then this splits into a transition for E and a transition for G using the operational rule for products. Since $E \sim_{\mathrm{a}} F$ one can apply Proposition 48 to find a sequence of transitions from F into some F' with $E' \sim_{\mathrm{a}} F'$, or else we have that $E' \sim_{\mathrm{a}} F$ and the E-state ticks, with $a = 1$. In the first case the fact that G is idle gives a sequence of matching length from G and hence $R, F \times G \stackrel{a}{\Longrightarrow} \mu(a, R), F' \times G'$ with $E' \times G' \sim' F' \times G'$. In the second case one finds $R, F \times G \xrightarrow{a} \mu(a, R), F \times G'$ and $E' \times G' \sim' F \times G$ since E', F, G' are idle. The verification of the other condition (when $R, F \times G$ make some transition) is similar. □

Lemma 54. *The following are equivalent:*

1. *For all agents* G *of* **ASCRP***, the relation* $\mathcal{T}_{as}(E \times G) \sim_{\mathrm{a}} \mathcal{T}_{as}(F \times G)$ *holds.*
2. *For all* $a \in \mathsf{Act}$*, all* $R \in \mathbf{R}$*:*
   - *If* $R, \mathcal{T}_{as}(E) \xrightarrow{a} \mu(a, R), E'$ *for some* E' *then* $R, \mathcal{T}_{as}(F) \xrightarrow{a} \stackrel{\langle\rangle}{\Longrightarrow} \mu(a, R), F'$ *for some* F' *with* $E' \sim_{\mathrm{a}} F'$
   - *If* $R, \mathcal{T}_{as}(F) \xrightarrow{a} \mu(a, R), F'$ *for some* F' *then* $R, \mathcal{T}_{as}(E) \xrightarrow{a} \stackrel{\langle\rangle}{\Longrightarrow} \mu(a, R), E'$ *for some* E' *with* $E' \sim_{\mathrm{a}} F'$*.*

Proof. Suppose point (2) is true and that there is some transition of an $\mathcal{T}_{as}(E \times G)$-state. Such a transition must be of the form

$$\frac{R, \mathcal{T}_{as}(E) \xrightarrow{a} \mu(a, R), E' \qquad S, \mathcal{T}_{as}(G) \xrightarrow{b} \mu(b, S), G'}{R \circ S, \mathcal{T}_{as}(E \times G) \xrightarrow{ab} \mu(ab, R \circ S), E' \times G'}$$

for some a, b, R, S, E', G'. By (2) there is some F' with $R, \mathcal{T}_{as}(F) \xrightarrow{a} \stackrel{\langle\rangle}{\Longrightarrow} \mu(a, R), F'$ for some F' with $E' \sim_{\mathrm{a}} F'$. So $R, \mathcal{T}_{as}(F) \xrightarrow{a} (\stackrel{1}{\rightarrow})^n \mu(a, R), F'$ for some n. Now G' is idle by Proposition 47 so $S, \mathcal{T}_{as}(G) \xrightarrow{b} (\stackrel{1}{\rightarrow})^n \mu(b, S), G'$. Therefore $R \circ S, \mathcal{T}_{as}(F \times G) \xrightarrow{ab} \stackrel{\langle\rangle}{\Longrightarrow} \mu(ab, R \circ S), F' \times G'$ and $E' \times G' \sim_{\mathrm{a}} F' \times G'$ by Lemma 53, since E' and F' are idle by Proposition 47. A similar argument can be made when given any transition of any $\mathcal{T}_{as}(F \times G)$-state, and so point (1) holds by Proposition 48.

Now suppose that point (2) is false. Without loss of generality, suppose that it is the first of the two conditions that fails. Since we have infinitely many tick actions there must be some such tick b not present in F. Take $G = b.0$. Then $R, \mathcal{T}_{as}(E \times G) \xrightarrow{ab} \mu(ab, R), E' \times 1$ for some E'. If $R, \mathcal{T}_{as}(F \times G) \stackrel{\langle ab \rangle}{\Longrightarrow} \mu(ab, R), F' \times 1$ for some F' then we must have $R, \mathcal{T}_{as}(F) \xrightarrow{a} \stackrel{\langle\rangle}{\Longrightarrow} \mu(a, R), F'$. Since (2) is false we have $E' \not\sim_{\mathrm{a}} F'$. Then $E' \times 1 \not\sim_{\mathrm{a}} F' \times 1$ by Lemma 52 and so $\mathcal{T}_{as}(E \times G) \not\sim_{\mathrm{a}} \mathcal{T}_{as}(F \times G)$. Hence (1) does not hold. □

Let E, F be processes of **ASCRP**. Define $E \sim_{\mathrm{a}}^{\times} F$ if for every agent G of **ASCRP** the relation $\mathcal{T}_{as}(E \times G) \sim_{\mathrm{a}} \mathcal{T}_{as}(F \times G)$ holds. The first lemma below is immediate. The second lemma below holds by taking $G = 1$ and applying Lemma 52.

Lemma 55. *Let* X *contain precisely the free variables of* E *and* F*. Then* $E \sim_{\mathrm{a}}^{\times} F$ *if and only if* $E[\mathsf{H}/\mathsf{X}] \sim_{\mathrm{a}}^{\times} F[\mathsf{H}/\mathsf{X}]$ *for all tuples of* **ASCRP** *agents with length matching* X*.*

Lemma 56. *If* $E \sim_{\mathrm{a}}^{\times} F$ *then* $\mathcal{T}_{as}(E) \sim_{\mathrm{a}} \mathcal{T}_{as}(F)$*.*

Lemma 57. *The relation* $\sim_{\mathrm{a}}^{\times}$ *is a congruence on* **ASCRP***-processes.*

1. *If* $E \sim_{\mathrm{a}}^{\times} F$ *then* $a.E \sim_{\mathrm{a}}^{\times} a.F$*,* $\delta(E) \sim_{\mathrm{a}}^{\times} \delta(F)$*,* $E \times G \sim_{\mathrm{a}}^{\times} F \times G$*,* $\nu S.E \sim_{\mathrm{a}}^{\times} \nu S.F$ *hold.*
2. *If* $E_i \sim_{\mathrm{a}}^{\times} F_i$ *for all* $i \in I$ *then* $\sum_{i\in I} E_i \sim_{\mathrm{a}}^{\times} \sum_{i\in I} F_i$ *holds.*
3. *If* $\mathsf{E} \sim_{\mathrm{a}}^{\times} \mathsf{F}$ *then* $fix_i \mathsf{X}.\mathsf{E} \sim_{\mathrm{a}}^{\times} fix_i.\mathsf{X}.\mathsf{F}$*.*

Proof. Suppose $E \sim_{\mathrm{a}}^{\times} F$.

A simple proof that $a.E \sim_{\mathrm{a}}^{\times} a.F$ uses the characterization of Lemma 54 and the fact that $\mathcal{T}_{as}(\delta(E)) \sim_{\mathrm{a}} \mathcal{T}_{as}(\delta(F))$ by Proposition 51.

In order to show that $\delta(E) \sim_{\mathrm{a}}^{\times} \delta(F)$ one shows that $\delta(\mathcal{T}_{as}(E)) \times \mathcal{T}_{as}(H) \sim_{\mathrm{a}} \delta(\mathcal{T}_{as}(F)) \times \mathcal{T}_{as}(H)$ for an arbitrary agent H of **ASCRP**. Supposing a transition of $\delta(\mathcal{T}_{as}(E)) \times \mathcal{T}_{as}(H) \xrightarrow{ab} \mu(ab, R), K$ there must be a decomposition as an a-transition of a $\delta(\mathcal{T}_{as}(E))$-state and b-transition of a $\mathcal{T}_{as}(H)$-state. Now $\mathcal{T}_{as}(\delta(E)) \sim_{\mathrm{a}} \mathcal{T}_{as}(\delta(F))$ by Proposition 51 so the characterization of Proposition 48 may then be applied. In either case of that Proposition 48 one finds $\delta(\mathcal{T}_{as}(F)) \times \mathcal{T}_{as}(H) \xrightarrow{ab}\!\!\Longrightarrow \mu(ab, R), K$ (using Lemma 53 in the idle case $a = 1$). The symmetrical argument for when $\delta(\mathcal{T}_{as}(F)) \times \mathcal{T}_{as}(H)$ makes a transition and an application of Proposition 48 once more gives the desired conclusion.

That $\mathcal{T}_{as}((E \times G) \times H) \sim_{\mathrm{a}} \mathcal{T}_{as}((F \times G) \times H)$ for an arbtirary agnet H of **ASCRP** follows from the fact that $(\mathcal{T}_{as}(E) \times \mathcal{T}_{as}(G)) \times \mathcal{T}_{as}(H) \sim_{\mathrm{a}} \mathcal{T}_{as}(E) \times (\mathcal{T}_{as}(G) \times \mathcal{T}_{as}(H))$ and $\mathcal{T}_{as}(F) \times (\mathcal{T}_{as}(G) \times \mathcal{T}_{as}(H)) \sim_{\mathrm{a}} (\mathcal{T}_{as}(F) \times \mathcal{T}_{as}(G)) \times \mathcal{T}_{as}(H)$ since $\sim$ is contained in $\sim_{\mathrm{a}}$, and the definition of $\sim_{\mathrm{a}}^{\times}$.

Proposition 51 gives that $\nu S.\mathcal{T}_{as}(E) \sim_{\mathrm{a}} \nu S.\mathcal{T}_{as}(F)$. Proposition 47, Proposition 48 and Lemma 53 can then be used to show that $\nu S.\mathcal{T}_{as}(E) \times \mathcal{T}_{as}(H) \sim_{\mathrm{a}} \nu S.\mathcal{T}_{as}(F) \times \mathcal{T}_{as}(H)$ for any agent H of **ASCRP**. Hence $\nu S.E \sim_{\mathrm{a}}^{\times} \nu S.F$.

Suppose now that $E_i \sim_{\mathrm{a}}^{\times} F_i$ for all $i \in I$. To show $\sum_{i\in I} E_i \sim_{\mathrm{a}}^{\times} \sum_{i\in I} F_i$ holds it suffices to show that $\mathcal{T}_{as}(\sum_{i\in I} E_i) \times \mathcal{T}_{as}(H) \sim_{\mathrm{a}}^{\times} \mathcal{T}_{as}(\sum_{i\in I} F_i) \times \mathcal{T}_{as}(H)$ holds for any agent H of **ASCRP**. But now any transition of the component $R, \mathcal{T}_{as}(\sum_{i\in I} E_i) \xrightarrow{a} \mu(a, R), E'$ with some given resource R comes from some $R, \mathcal{T}_{as}(E_i) \xrightarrow{a} \mu(a, R), E'$. By Lemma 54 this is matched by $R, \mathcal{T}_{as}(F_i) \xrightarrow{a}\!\!\Longrightarrow \mu(a, R), F'$ for some $F' \sim_{\mathrm{a}} E'$. But then $R, \mathcal{T}_{as}(\sum_{i\in I} F_i) \xrightarrow{a} \mu(a, R), F'$ and hence $\mathcal{T}_{as}(\sum_{i\in I} E_i) \times \mathcal{T}_{as}(H) \sim_{\mathrm{a}}^{\times} \mathcal{T}_{as}(\sum_{i\in I} F_i) \times \mathcal{T}_{as}(H)$ by Lemma 54.

Now suppose that $\mathsf{E} \sim_{\mathrm{a}}^{\times} \mathsf{F}$. Consider the relation $\sim'$ on **SCRPr**ν processes defined by pairs of the form

$$\mathcal{T}_{as}(G[fix\mathsf{X}.\mathsf{E}/\mathsf{Y}]) \sim' \mathcal{T}_{as}(G[fix\mathsf{X}.\mathsf{F}/\mathsf{Y}])$$

such that G is any **ASCRP** term with all free variables in Y. Define a further relation $\sim'_{\mathrm{a}}$

$$E \sim'_{\mathrm{a}} H \qquad \Longleftrightarrow \qquad \exists F, G.\quad E \sim_{\mathrm{a}} F \sim' G \sim_{\mathrm{a}} H$$

for all **SCRPr**ν-processes E, F, G, H.

Below we show by induction that the property

$$\begin{array}{l}\text{if } R, \mathcal{T}_{as}(G[fix\mathsf{X}.\mathsf{E}/\mathsf{Y}]) \xrightarrow{a} \mu(a, R), E' \text{ then for some idle } F' \text{ and } G' \text{ we have}\\ R, \mathcal{T}_{as}(G[fix\mathsf{X}.\mathsf{F}/\mathsf{Y}]) \xrightarrow{a}\!\!\Longrightarrow \mu(a, R), G' \text{ and } E' \sim' F' \sim_{\mathrm{a}} G'\ .\end{array} \tag{7}$$

holds.

It follows from Property (7) that $\sim'_{\mathrm{a}}$ is a WG-bisimulation, that is $\sim'_{\mathrm{a}} \subseteq \sim_{\mathrm{a}}$. Taking $G = Y_i$ then gives the special case:

$$\begin{array}{l}\text{if } R, \mathcal{T}_{as}(fix_i\mathsf{X}.\mathsf{E}) \xrightarrow{a} \mu(a, R), E' \text{ then for some idle } F' \text{ we have}\\ R, \mathcal{T}_{as}(fix_i\mathsf{X}.\mathsf{F}) \xrightarrow{a}\!\!\Longrightarrow \mu(a, R), F' \text{ and } E' \sim_a F'\ .\end{array} \tag{8}$$

The symmetric property to this and the characterization of Lemma 54 then shows that $\mathit{fix}_i\mathsf{X}.\mathsf{E} \sim_{\mathrm{a}}^{\times} \mathit{fix}_i\mathsf{X}.\mathsf{F}$, as required.

We now give the induction to prove property (7). The induction is on the inference of $R, \mathcal{T}_{as}(G[\mathit{fix}_i\mathsf{X}.\mathsf{E}/\mathsf{Y}]) \xrightarrow{a} \mu(a,R), E'$.

If $G = Y_i$ then we have $R, \mathcal{T}_{as}(\mathit{fix}_i\mathsf{X}.\mathsf{E}) \xrightarrow{a} \mu(a,R), E'$ and so $R, \mathcal{T}_{as}(E_i)[\mathit{fix}\mathsf{X}.\mathcal{T}_{as}(\mathsf{E})/\mathsf{X}] \xrightarrow{a} \mu(a,R), E'$ by a shorter inference. Then $R, \mathcal{T}_{as}(F_i)[\mathit{fix}\mathsf{X}.\mathcal{T}_{as}(\mathsf{F})/\mathsf{X}] \xrightarrow{a}\!\!\!\!\Longrightarrow \mu(a,R), F'$ with $E' \sim' H' \sim_{\mathrm{a}} F'$ for some idle H', F', by the induction hypothesis. But then $R, \mathcal{T}_{as}(\mathit{fix}_i\mathsf{X}.\mathsf{F}) \xrightarrow{a} \mu(a,R), F'$.

We omit the case $G = \delta(G_0)$ since δ is definable.

If $G = a.G_0$ then we have $R, \mathcal{T}_{as}(G_0[\mathit{fix}\mathsf{X}.\mathsf{E}/\mathsf{Y}]) \xrightarrow{a} \mu(a,R), E'$. This is the same as $R, a : \delta(\mathcal{T}_{as}(G_0[\mathit{fix}\mathsf{X}.\mathsf{E}/\mathsf{Y}])) \xrightarrow{a} \mu(a,R), \delta(\mathcal{T}_{as}(G_0[\mathit{fix}\mathsf{X}.\mathsf{E}/\mathsf{Y}]))$ with $\mu(a,R)$ defined. Then there is a transition $R, a : \delta(\mathcal{T}_{as}(G_0[\mathit{fix}\mathsf{X}.\mathsf{F}/\mathsf{Y}])) \xrightarrow{a} \mu(a,R), \delta(\mathcal{T}_{as}(G_0[\mathit{fix}\mathsf{X}.\mathsf{F}/\mathsf{Y}]))$ and

$$\begin{array}{rcl} \delta(\mathcal{T}_{as}(G_0[\mathit{fix}\mathsf{X}.\mathsf{E}/\mathsf{Y}])) & = & \mathcal{T}_{as}(\delta(G_0[\mathit{fix}\mathsf{X}.\mathsf{E}/\mathsf{Y}])) \\ & \sim' & \mathcal{T}_{as}(\delta(G_0[\mathit{fix}\mathsf{X}.\mathsf{F}/\mathsf{Y}])) \\ & = & \delta(\mathcal{T}_{as}(G_0[\mathit{fix}\mathsf{X}.\mathsf{F}/\mathsf{Y}])) \end{array}$$

as required.

The product case $G = G_0 \times G_1$ uses the Product rule to split product transitions into pairs of transitions, the induction hypothesis on those pairs, the easy fact that $\sim'$ is preserved by products and the fact that $\sim_{\mathrm{a}}^{\times}$ is a congruence for $\times$ (the third part of the first point of this Lemma).

The case $G = \sum_{i\in I} G_i$ follows by a straightforward application of the induction hypothesis.

Let $G = \nu S.G_0$. Suppose $R, \mathcal{T}_{as}(\nu S.G_0[\mathit{fix}\mathsf{X}.\mathsf{E}/\mathsf{Y}) \xrightarrow{a} \mu(a,R), E'$. This is the same as the transition $R, \nu S.\mathcal{T}_{as}(G_0[\mathit{fix}\mathsf{X}.\mathsf{E}/\mathsf{Y}) \xrightarrow{a} \mu(a,R), E'$. There are two rules from which we may derive such a transition. If we use the Hide-id rule with $a = 1$ and $E' = \nu S.\mathcal{T}_{as}(G_0[\mathit{fix}\mathsf{X}.\mathsf{E}/\mathsf{Y}])$ then evidently $R, \nu S.\mathcal{T}_{as}(G_0[\mathit{fix}\mathsf{X}.\mathsf{F}/\mathsf{Y}) \xrightarrow{1} \mu(a,R), F'$ and $E' \sim' F' \sim_{\mathrm{a}} F'$ with $F' = \nu S.\mathcal{T}_{as}(G_0[\mathit{fix}\mathsf{X}.\mathsf{F}/\mathsf{Y}])$. On the other-hand if the Hide rule was used to derive the given transition then there must be some b, S', E'_0 such that $R \circ S, \mathcal{T}_{as}(G_0[\mathit{fix}\mathsf{X}.\mathsf{E}/\mathsf{Y}]) \xrightarrow{b} \mu(a, R \circ S), E'$, where $a = \nu S.b$, $E' = \nu S'.E'_0$ and the side-conditions for the rule hold. By the induction hypothesis we have $R \circ S, \mathcal{T}_{as}(G_0[\mathit{fix}\mathsf{X}.\mathsf{F}/\mathsf{Y}]) \xrightarrow{b} \mu(a, R \circ S), F'_0$ for some F'_0, G'_0 with $E'_0 \sim' F'_0 \sim_{\mathrm{a}} G'_0$. Then $R, \nu S.\mathcal{T}_{as}(G_0[\mathit{fix}\mathsf{X}.\mathsf{F}/\mathsf{Y}]) \xrightarrow{a}\!\!\!\!\Longrightarrow \mu(a,R), \nu S'.F'_0$ by the Hide rule (since the side-conditions are exactly the same as the previous side-conditions). So $R, \mathcal{T}_{as}(\nu S.G_0[\mathit{fix}\mathsf{X}.\mathsf{F}/\mathsf{Y}]) \xrightarrow{a}\!\!\!\!\Longrightarrow \mu(a,R), F'$ with $F' = \nu S'.F'_0$. That $E' \sim F'$ follows easily from $E'_0 \sim' F'_0$, and $F' \sim_{\mathrm{a}} G'$ follows from the fact that $\sim_{\mathrm{a}}$ is a congruence for ν (the fourth part of the first point of this Lemma).

Consider $G = \mathit{fix}_j\mathsf{Z}.\mathsf{H}$ where X, Y and Z have no variables in common. Suppose that there is a transition $R, \mathcal{T}_{as}((\mathit{fix}_j\mathsf{Z}.\mathsf{H})[\mathit{fix}\mathsf{X}.\mathsf{E}/\mathsf{Y}])) \xrightarrow{a} \mu(a,R), E'$. This is identical to the transition $R, \mathit{fix}_j\mathsf{Z}.\mathcal{T}_{as}(\mathsf{H}[\mathit{fix}\mathsf{X}.\mathsf{E}/\mathsf{Y}]) \xrightarrow{a} \mu(a,R), E'$. Then there is a shorter inference of the transition $R, (\mathcal{T}_{as}(H_j[\mathit{fix}\mathsf{X}.\mathsf{E}/\mathsf{Y}]))[(\mathit{fix}\mathsf{Z}.\mathcal{T}_{as}(\mathsf{H}[\mathit{fix}\mathsf{X}.\mathsf{E}/\mathsf{Y}]))/\mathsf{Z}] \xrightarrow{a} \mu(a,R), E'$. Letting $\mathsf{G} = \mathit{fix}\mathsf{Z}.\mathsf{H}$ this is $R, \mathcal{T}_{as}(H_j[\mathsf{G}/\mathsf{Z}][\mathit{fix}_i\mathsf{X}.\mathsf{E}/Y]) \xrightarrow{a} \mu(a,R), E'$. Then $R, \mathcal{T}_{as}(H_j[\mathsf{G}/\mathsf{Z}][\mathit{fix}_i\mathsf{X}.\mathsf{F}/Y]) \xrightarrow{a} \mu(a,R), G'$ with $E' \sim' F' \sim_{\mathrm{a}} G'$ for some idle F', G' by the induction hypothesis. Then there is a transition $R, \mathcal{T}_{as}((\mathit{fix}_j\mathsf{Z}.\mathsf{H})[\mathit{fix}\mathsf{X}.\mathsf{F}/\mathsf{Y}]) \xrightarrow{a} \mu(a,R), G'$ as required. □

Define the *contexts*, $\mathcal{C}[-]$, of **ASCRP** in the standard way: a context is a process term of **ASCRP** but with multiple occurrences of a hole, $[-]$, that may be plugged with any process term. We define a binary relation by

$$E \sim_{\mathrm{a}}^{\mathrm{c}} F \qquad \text{iff} \qquad \text{for all contexts } \mathcal{C}[-],\ \ \mathcal{T}_{as}(\mathcal{C}[E]) \sim_{\mathrm{a}} \mathcal{T}_{as}(\mathcal{C}[F])$$

for all **ASCRP**-processes E and F.

Proposition 58.

$$E \sim_{\mathrm{a}}^{\times} F \qquad \textit{iff} \qquad E \sim_{\mathrm{a}}^{\mathrm{c}} F$$

for all **ASCRP***-processes E and F.*

Proof. If $E \sim_{\mathrm{a}}^{\times} F$ then by Lemma 57 we have $\mathcal{C}[E] \sim_{\mathrm{a}}^{\times} \mathcal{C}[F]$ for every context $\mathcal{C}[-]$ of **ASCRP**. By Lemma 56, $\mathcal{C}[E] \sim_{\mathrm{a}} \mathcal{C}[F]$ for every such context, and so $E \sim_{\mathrm{a}}^{\mathrm{c}} F$.

If $E \sim_{\mathrm{a}}^{\mathrm{c}} F$ then taking the context $[-] \times G$ for any agent G gives $E \times G \sim_{\mathrm{a}} F \times G$, and so $E \sim_{\mathrm{a}}^{\times} F$. □

Write $E \sim_{\mathrm{a}}^{\mathrm{c}} F$ for any **ASCRP**-processes E and F with $\mathcal{T}_{as}(E) \sim_{\mathrm{a}}^{\mathrm{c}} \mathcal{T}_{as}(F)$.

Theorem 59. *The following are equivalent for processes E and F of* **ASCRP***:*

1. $E \sim_{\mathrm{a}}^{\mathrm{c}} F$
2. $E \times G \sim_{\mathrm{a}} F \times G$ *for all processes G of* **ASCRP**
3. *For all $a \in \mathsf{Act}$, all $R \in \mathbf{R}$:*
   - *If $R, \mathcal{T}_{as}(E) \xrightarrow{a} \mu(a,R), E'$ for some E' in* **SCRP**ν *then $R, \mathcal{T}_{as}(F) \xrightarrow{a}\!\!\!\!\Longrightarrow^{\langle\rangle} \mu(a,R), F'$ for some F' in* **SCRPr**ν *with $E' \sim_{\mathrm{a}} F'$*
   - *If $R, \mathcal{T}_{as}(F) \xrightarrow{a} \mu(a,R), F'$ for some F' in* **SCRP**ν *then $R, \mathcal{T}_{as}(E) \xrightarrow{a}\!\!\!\!\Longrightarrow^{\langle\rangle} \mu(a,R), E'$ for some E' in* **SCRPr**ν *with $E' \sim_{\mathrm{a}} F'$.*

Moreover $\sim_{\mathrm{a}}^{\mathrm{c}}$ is a congruence, and if X contains all free variables of E and F, then $E \sim_{\mathrm{a}}^{\mathrm{c}} F$ iff $E[\mathsf{H}/\mathsf{X}] \sim_{\mathrm{a}}^{\mathrm{c}} F[\mathsf{H}/\mathsf{X}]$ for all tuples H of agents of **ASCRP** *with the same length as X.*

Proof. The first two points are equivalent by Proposition 58. The second and third points are equivalent by Lemma 54. The relation $\sim_{\mathrm{a}}^{\mathrm{c}}$ is then a congruence by Lemma 57. The final property holds by Lemma 55. □

Corollary 60. *If $E \sim_{\mathrm{a}} F$ then $a.E \sim_{\mathrm{a}}^{\mathrm{c}} a.F$.*

5.3 Asynchronous Parallel Composition

Although the calculus **ASCRP** has captured a class of agents that are asynchronous in the sense in the sense that all transitions are followed by idle states, it still contains a process constructor for synchronous, rather than asynchronous, parallel composition. For some purposes **ASCRP** is still too powerful: if one takes a product of prefix processes then the resulting agent must perform both of the prefixed actions simultaneously. For example, at most an ab-transition is possible for a state $R, \mathcal{T}_{as}((a.1) \times (b.1))$, given any resource R. We now introduce a further calculus **APSCRP** in which interleavings of such actions are allowed (but not forced) by a parallel composition. This is done by replacing the synchronous product $\times$ of **ASCRP** with an asynchronous parallel composition $|$.

The grammar of **APSCRP** is

$$E ::= X \mid a.E \mid \delta(E) \mid \sum_{i \in I} E_i \mid (E \,|\, E) \mid \nu R.E \mid \mathit{fix}_i \mathsf{X}.\mathsf{E}$$

where, the notation for variables and processes is as before, and this calculus is to be considered relative to a fixed $(\mathsf{Act}, \mathbf{R}, \mu, \nu)$. We then give a semantics to this calculus by translating into **SCRP**ν over the same $(\mathsf{Act}, \mathbf{R}, \mu, \nu)$. Define a translation $\mathcal{T}_{aps}^0$: **APSCRP** $\longrightarrow$ **ASCRP** recursively, by the clause

$$\mathcal{T}_{aps}^0(E \,|\, F) \quad = \quad (\mathcal{T}_{aps}^0(E) \times \delta(\mathcal{T}_{aps}^0(F))) + (\delta(\mathcal{T}_{aps}^0(E)) \times \mathcal{T}_{aps}^0(F))$$

together with clauses such that $\mathcal{T}_{aps}^0$ passes through the other process combinators in the evident way. Then define a translation $\mathcal{T}_{aps}$: **APSCRP** $\longrightarrow$ **SCRP**ν as the composite map

$$\mathcal{T}_{aps} = \mathcal{T}_{as} \circ \mathcal{T}_{aps}^0 \ .$$

The operational semantics is inherited from **SCRP**ν in a similar way to **ASCRP**, using **SCRP**ν-transitions of the form

$$R, \mathcal{T}_{aps}(E) \xrightarrow{a} \mu(a,R), E' \ .$$

Lemma 61.

$$\mathcal{T}_{aps}(\delta(E\,|\,F)) \sim \mathcal{T}_{aps}(\delta(E)\,|\,F) \sim \mathcal{T}_{aps}(E\,|\,\delta(F)) \sim \mathcal{T}_{aps}(\delta(E)) \times \mathcal{T}_{aps}(\delta(F))$$

The proof of the above lemma is straightforward using Proposition 43.

Proposition 62. *If* $R, \mathcal{T}_{aps}(E) \xrightarrow{a} \mu(a, R), F$ *then there is some agent* E' *of* **APSCRP** *such that* $F \sim \delta(\mathcal{T}_{aps}(E'))$.

Proof. The proof is by induction on inference of the given transition. Most cases are straightforward. The asynchronous product case uses Lemma 61. □

Note that in contrast to CCS, simultaneous actions are possible in **APSCRP** which are not pure synchronizations: compound actions are allowed. This is similar to the situation for the encoding of CCS in SCCS. In that situation, however, one may sequentialize compound actions and so prove an appropriate (asynchronous) simulation between CCS processes and their encodings. It remains an open problem to do this for a **SCRP**-like calculus. The resource based form of hiding we have adopted does not, in general, allow us to replace composite actions by a sequence of atomic actions corresponding to that composite.

5.4 Value-passing

In many process algebras, the ability to pass values into and out-of processes is an important modelling mechanism — the first chapter of (Milner 1989) contains an extended example. In this section we extend the asynchronous product language above with value-passing actions to give a calculus **APVSCRP**. This can be regarded as a collection of abbreviations (from **SCRP**ν) for efficiently expressing models of asynchronous situations with value-passing. We allow atomic actions that simultaneously input and output values in contrast to the standard treatment in CCS and SCCS (Milner 1980, Milner 1983).

We assume a collection $\mathcal{V}$ of *values*, ranged over by v, which contains a set $\mathcal{B}$ of boolean values, b, including special values true and false. We assume a collection of *(value) variables* x, not to be confused with the process variables. We assume a set $\mathcal{F}$ of *function symbols* f, each with a specified natural number, called the *arity*. We generate a set $\mathcal{E}$ of *(value) expressions* ε by

$$\varepsilon ::= v \mid x \mid f(\varepsilon_1, \ldots \varepsilon_n)$$

where f is any function symbol of arity n, for any n. Assume that all expressions evaluate to a unique value. For any expression ε containing no free variables let v_ε be the value to which it evaluates. Write a vector of value variables as $\overline{x}$ and of expressions as $\overline{\epsilon}$.

The actions of the language are assumed to include atoms of the form

$$\alpha^{x_1,\ldots,x_m}_{\epsilon_1,\ldots,\epsilon_n}$$

for any sequences $\epsilon_1, \ldots, \epsilon_n$ and $x_1, \ldots, x_m$ of expressions and variables such that none of the x_i occurs free in any of the ϵ_j. The variables written as superscripts are said to be *inputs* to the action and the expressions written as subscripts are said to be *outputs*. We suppose the action monoid ActV to contain such actions.

We omit the superscripts when actions have no inputs, and the subscripts when actions have no outputs. In some situations only input actions of the form $\alpha^{x_1,\ldots,x_m}$ and $\alpha_{\epsilon_1,\ldots,\epsilon_n}$ are required. We say that the former are *input actions* and the latter *output actions*, and in such situations we have essentially the standard form of value-passing (as for CCS). Actions without parameters can be treated as special cases of outputs.

The grammar of **APVSCRP** is as follows:

$$E ::= a.E \mid \alpha^{\overline{x}}_{\overline{\epsilon}}.E \mid \sum_{i\in I} E_i \mid (E\,|\,E) \mid \nu R.E \mid A(x_1, \ldots, x_n) \mid \mathsf{if\ b\ then}\ E\ .$$

APVSCRP	**APSCRP**
$\alpha^{x_1,\ldots,x_m}_{\epsilon_1,\ldots,\epsilon_n}.E \quad (\text{s.t. } 1 \leq m)$	$\sum_{(w_1,\ldots,w_m)\in\mathcal{V}^m} \alpha^{w_1,\ldots,w_n}_{v_{\epsilon_1},\ldots,v_{\epsilon_n}}.\mathcal{T}^0_{apv}(E)[w_1/x_1,\ldots,w_n/x_n]$
$a.E$	$a.\mathcal{T}^0_{apv}(E)$
$\sum_{i\in I} E_i$	$\sum_{i\in I}(\mathcal{T}^0_{apv}(E_i))$
$E_1 \mid E_2$	$\mathcal{T}^0_{apv}(E_1) \mid \mathcal{T}^0_{apv}(E_2)$
$A(\epsilon_1,\ldots\epsilon_n)$	$A_{\epsilon_1,\ldots,\epsilon_n}$
$\mathsf{if\ b\ then}\ E$	$\begin{cases} \mathcal{T}^0_{apv}(E) & \text{if } v_{\mathsf{b}} = \mathsf{true} \\ \mathsf{0} & \text{otherwise} \end{cases}$

Figure 9: Translation $\mathcal{T}^0_{apv}$

where $a \in \mathsf{Act}$ contains no free value variables. The process $\mathsf{if\ b\ then}\ E$ is a *conditional*. The process $A(x_1,\ldots,x_n)$ is a process *constant* with distinct value variables $x_1,\ldots,x_n$. We have chosen to use a process constant in order to be closer to the value-passing language of (Milner 1989). Process constants are an alternative way of presenting fixed-points, see (Pym & Tofts 2006, Pym & Tofts 2007) for a discussion of this in **SCRP**. Such constants are declared in families

$$\begin{aligned} A_1(x_1,\ldots,x_n) &= E_1 \\ &\vdots \\ A_m(x_1,\ldots,x_n) &= E_m \end{aligned}$$

where the E_i are process terms of **APVSCRP** containing no free value variables except $x_1,\ldots,x_n$, no free agent variables and no constants other than those of this family.

In a process $\alpha^{\overline{x}}_{\overline{\epsilon}}.E$ the input variables $\overline{x}$ are bound with scope E. In contrast, the output variables $\overline{\epsilon}$ are not bound by this occurrence.

We write $E[a/\alpha]$ for the (capture-avoiding) result of replacing every occurrence of the free variable α in E by a. We write $E[\beta/\alpha]$ when α is of the form $\alpha_{\epsilon_1,\ldots,\epsilon_n}$ and we substitute $\beta_{\epsilon_1,\ldots,\epsilon_n}$ for α in E.

A conditional of the form $\mathsf{if\ b\ then}\ E_1\ \mathsf{else}\ E_2$ is definable by $(\mathsf{if\ b\ then}\ E_1) + (\mathsf{if}\ \neg\mathsf{b\ then}\ E_2)$ where $\neg$ is Boolean negation.

We now show how to translate this value-passing calculus over ActV into an asynchronous fragment of **APSCRP** over another action monoid Act. We assume that Act contains every member of the family $(\alpha^{w_1,\ldots,w_m}_{v_1,\ldots,v_n} \mid v_i, w_j \in \mathcal{V})$ for each $\alpha^{x_1,\ldots,x_m}_{\epsilon_1,\ldots,\epsilon_n}$ in ActV.

The translation $\mathcal{T}^0_{apv}$ takes expressions of **APVSCRP** containing no free value variables to processes in **APSCRP**. The clauses that recursively define this translation are given in Figure 9. The constant term $A_{\epsilon_1,\ldots,\epsilon_n}$ of **APSCRP** is defined to be the fixed point expression $fix_i\mathsf{X}.\mathcal{T}^0_{apv}(\mathsf{E})$ for the appropriate i, where E is the tuple of declarations in which $A_{\epsilon_1,\ldots,\epsilon_n} = E_i$ was defined. Define the translation $\mathcal{T}_{apv}$ from **APVSCRP** to **SCRP**ν to be the composite map $\mathcal{T}_{aps} \circ \mathcal{T}^0_{apv}$.

As an example of the above translation $\mathcal{T}^0_{apv}$, consider the term

$$\alpha^x.\beta^y.\gamma_{x+y}.\mathsf{1}$$

$$\varepsilon ::= \quad v \mid Y \mid F(\varepsilon, \ldots, \varepsilon)$$

$$C ::= \quad \texttt{skip} \mid Y := \varepsilon \mid C; C' \mid \texttt{if } \varepsilon \texttt{ then } C \texttt{ else } C' \mid \texttt{while } \varepsilon \texttt{ do } C \mid$$
$$\mid C \texttt{ par } C' \mid \texttt{input } Y \mid \texttt{output } \varepsilon$$

Figure 10: Syntax of **P**

of **APVSCRP**. This translates as:

$$\begin{aligned}
& \mathcal{T}^0_{apv}(\alpha^x.\beta^y.\gamma_{x+y}.1) \\
= \quad & \textstyle\sum_{v\in\mathcal{V}} \alpha^v.\mathcal{T}^0_{apv}(\beta^y.\gamma_{v+y}.1) \\
= \quad & \textstyle\sum_{v\in\mathcal{V}} \alpha^v.\sum_{w\in\mathcal{V}} \beta^w.\mathcal{T}^0_{apv}(\gamma_{v+w}.1) \\
= \quad & \textstyle\sum_{v\in\mathcal{V}} \alpha^v.\sum_{w\in\mathcal{V}} \beta^w.\gamma_{v+w}.\mathcal{T}^0_{apv}(1) \\
= \quad & \textstyle\sum_{v\in\mathcal{V}} \alpha^v.\sum_{w\in\mathcal{V}} \beta^w.\gamma_{v+w}.1
\end{aligned}$$

in **APSCRP**, which illustrates the difference between the inputs and outputs.

Once again, the calculus is closed under transitions up to $\sim$. The proof is essentially the same as that for Proposition 62, except where prefixes of atomic actions now translate as sums, so a transition of the translation of a value-passing prefix is one of the component transtions of the outermost sum.

Proposition 63. *If E is an agent of* **APVSCRP** *and $R, \mathcal{T}_{apv}(E) \xrightarrow{a} \mu(a, R), E'$ for some R, then $E' \sim \delta(\mathcal{T}_{apv}(E''))$ for some agent E''.*

It seems that in most situations mixed input-output actions are not required. Even where such mixed actions appear necessary they can often be eliminated by extending the resource monoid to give a kind temporary storage locations and replacing each mixed action with a pair of input and output actions, which access the same cell in the temporary storage.

5.5 A Programming Language

It is possible to define parallel programming languages in the **SCRP**-family of languages by an appropriate modification of Milner's technique for CCS. The idea is to translate the phrases of some programming language **P** into phrases of the process algebra. The language **P** then inherits a precise operational semantics from the process algebra. The translation of any program should capture the intended operational behaviour. Axioms regarding such intended behaviour, for example the associativity of parallel composition, should be validated by the appropriate asynchronous congruence of the process algebra. Thoroughly worked examples of this for CCS can be found in (Milner 1980, Milner 1989). The example in (Milner 1989) is a simple imperative programming language extended with parallelism and recursive, concurrent procedures. Thus the language allows a rather complicated form of shared-variable concurrency. In this sketch we do not treat all of the language from (Milner 1989), for example we omit the concurrent, recursive procedures.

The grammar of **P** is shown in Figure 10. The language includes value expressions ε, program variable declarations D and imperative commands C, including a parallel command `par` . The value expressions are formed from program variables Y, atomic values v (including integers) from as set *Vals*, and by a collection of function symbols F of given arity.

The main task in providing a translation for a programming language into a member of the **SCRP**-family is to define, within the process algebra, appropriate structures and mechanisms for

handling the flow of control and transfer of values from expressions to procedures. We use the the notation from the value-passing process calculus **APVSCRP** in conjunction with semaphore-like resources to make such definitions. For the purposes this part of the work we revert to the calculus **SCRP**ν with the more general notion of modification and without the side-condition regarding piggybacking on the Hide rule. Thus the translation that defines the operational behaviour of terms written in the notation of **APVSCRP** takes terms into this version of **SCRP**ν. Note that here we are using **APVSCRP** terms merely as a convenient macro notation for **SCRP**ν terms rather than as a calculus in its own right.

The resource monoid $\mathbf{R}$ consists of triples (R^{st}, R^{sm}, R^{bu}) where the *store*, R^{st}, is a partial function from program variables to values, R^{sm} is a partial function from a countable set, *Sem*, to the set $\{0,1\}$, and R^{bu} is a partial function from a set *Buff* to values. The components R^{sm} and R^{bu} represent the current states of sets of semaphores and value-buffers. A resource R can of course be regarded as a partial map from the disjoint union of the sets of program variables, semaphores and buffers.

Write the empty partial map (from any source to any target) as $\emptyset$. For any partial function f let $dom(f)$ be the set of arguments of f at which f is defined. For any sets X, Y, partial function $f : X \longrightarrow Y$, $x \in X$, $y \in Y$ write $f \oplus \{f(x) = y\}$ for the partial function $g : X \longrightarrow Y$ such that $g(x) = y$, $g(x) = f(x)$ if $f(x) \downarrow$ and $x \neq y$, and $g(x) \uparrow$ if $f(x) \uparrow$ and $x \neq y$.

The set of stores is made into a resource monoid as follows. For each R^{st} and $R^{st\prime}$, the composite $R^{st} \circ R^{st\prime}$ is defined just when $dom(R^{st})$ is disjoint from $dom(R^{st\prime})$: the graph of $R^{st} \circ R^{st\prime}$ is then the union of the graphs of R^{st} and $R^{st\prime}$. The unit is $\emptyset$. The semaphores and buffers are made into resource monoids in a similar way. The resource monoid of interest is the product of the store, semaphore and buffer monoids: resource composition is given by $(R^{st}, R^{sm}, R^{bu}) \circ (R^{st\prime}, R^{sm\prime}, R^{bu\prime}) \simeq (R^{st} \circ R^{st\prime}, R^{sm} \circ R^{sm\prime}, R^{bu} \circ R^{bu\prime})$ for all resources (R^{st}, R^{sm}, R^{bu}) and $(R^{st\prime}, R^{sm\prime}, R^{bu\prime})$. The unit resource is then $e = (\emptyset, \emptyset, \emptyset)$.

The action monoid for the value-passing calculus is freely generated from the actions

$$\mathsf{in}^x \qquad \mathsf{pb}(i)^x \qquad \mathsf{get}(Y)^x$$

and

$$\mathsf{out}_\varepsilon \qquad \mathsf{ps}(i) \qquad \mathsf{qs}(i) \qquad \mathsf{qb}(i)_\varepsilon \qquad \mathsf{put}(Y)_\varepsilon \qquad \mathsf{res}_\varepsilon \qquad \mathsf{done}$$

indexed by value variables x, value expressions $\varepsilon \in \mathcal{E}$, program varables Y, and $i \in \mathbb{N}$. Notice that only input and output actions are required (no mixed input-output actions).

There is a corresponding modification which ensures that: $\mathsf{qs}(i)$ increments and $\mathsf{ps}(i)$ decrements the ith semaphore; $\mathsf{qb}(i)_\epsilon$ increments the ith buffer to hold the value corresponding to ϵ provided it currently holds 0; $\mathsf{pb}(i)^x$ sets the value in the ith buffer to 0, but note that because of the way values are passed, the value initially held in the ith buffer is bound to x in the process term; $\mathsf{put}(Y)_\varepsilon$ alters the store so that the memory allocated to the program variable Y holds the value corresponding to the expression ε; the action $\mathsf{get}(Y)^x$ retrieves the value stored at Y and binds it to x. The definition of the modification on the corresponding atomic actions in Act for **SCRP**ν is shown in Figure 11.

A partial function $\emptyset_{x,y} : X \longrightarrow Y$ is defined for any sets X, Y and $x \in X$ by taking $\emptyset_{x,y}(x) = y$ and $\emptyset_{x,y}(x') \uparrow$ if $x' \neq x$.

$$\mu(\mathsf{in}^v, R) = \mu(\mathsf{out}_\varepsilon, R) = \mu(\mathsf{res}_\varepsilon, R) = \mu(\mathsf{done}, R) = R$$

$$\mu(\mathsf{ps}(i), R) = \begin{cases} R \oplus \{R^{sm}(i) = 0\} & \text{if } R^{sm}(i) = 1 \\ \uparrow & \text{otherwise} \end{cases}$$

$$\mu(\mathsf{pb}(i)^v, R) = \begin{cases} R \oplus \{R^{sm}(i) = 0\} & \text{if } R^{bu}(i) = v \\ \uparrow & \text{otherwise} \end{cases}$$

$$\mu(\mathsf{qs}(i), R)) = \begin{cases} R \oplus \{R^{sm}(i) = 1\} & \text{if } R^{sm}(i) \downarrow \\ \uparrow & \text{otherwise} \end{cases}$$

$$\mu(\mathsf{qb}(i)_v, R) = \begin{cases} R \oplus \{R^{bu}(i) = v\} & \text{if } R^{bu}(i) \downarrow \\ \uparrow & \text{otherwise} \end{cases}$$

$$\mu(\mathsf{get}(Y)^v, R) = \begin{cases} R & \text{if } R^{st}(Y) = v \\ \uparrow & \text{otherwise} \end{cases}$$

$$\mu(\mathsf{put}(Y)_v, R) = \begin{cases} R \oplus \{R^{st}(Y) = v\} & \text{if } R^{st}(Y) \downarrow \\ \uparrow & \text{otherwise} \end{cases}$$

Figure 11: Modification function on atoms used for translation of $\mathbf{P}$

The atomic actions have families of roots as follows:

$$\mathsf{Roots}(\mathsf{in}^v) = \mathsf{Roots}(\mathsf{out}_v) = \mathsf{Roots}(\mathsf{res}_v) = \mathsf{Roots}(\mathsf{done}) = \{e\}$$

$$\mathsf{Roots}(\mathsf{get}(Y)^v) = \{(\emptyset_{Y,v}, \emptyset, \emptyset)\}$$

$$\mathsf{Roots}(\mathsf{put}(Y)_v) = \{(\emptyset_{Y,w}, \emptyset, \emptyset) \mid w \in \mathbb{N}\}$$

$$\mathsf{Roots}(\mathsf{ps}(i)) = \{(\emptyset, \emptyset_{i,1}, \emptyset)\}$$

$$\mathsf{Roots}(\mathsf{qs}(i)) = \{(\emptyset, \emptyset_{i,0}, \emptyset)\}, (\emptyset, \emptyset_{i,1}, \emptyset)\}$$

$$\mathsf{Roots}(\mathsf{pb}(i)^v) = \{(\emptyset, \emptyset, \emptyset_{i,v})\}$$

$$\mathsf{Roots}(\mathsf{qb}(i)_v) = \{(\emptyset, \emptyset, \emptyset_{i,w}) \mid w \in \mathbb{N}\} \ .$$

Proposition 21 shows that this is indeed a modification: the family is coherent because composition works on all components as disjoint union.

The atomic value expressions are given a semantics as processes in the asynchrounous value-passing calculus in such a way that every translation issues an action res_v for a unique value v immediately before it terminates, if it terminates, and no earlier. For atomic expressions this is done as follows:

$$[\![v]\!] = \mathsf{res}_v.0 \qquad\qquad [\![Y]\!] = \mathsf{get}(Y)^x.\mathsf{res}_x.0$$

where v is not a program variable. Complex expressions have the form $F(\varepsilon_1, \ldots, \varepsilon_n)$ where each ε_i is an expression and F is an n-ary function symbol of the programming language. Each such function symbol is assumed to be tracked by a given n-ary function f on the set of values. We then define

$$[\![F]\!] = \mathsf{pb}(1)^{x_1}.\cdots.\mathsf{pb}(n)^{x_n}.\mathsf{res}_{f(x_1,\ldots,x_n)}.0$$

for function symbols. The application is defined by

$$[\![F(\varepsilon_1, \ldots, \varepsilon_n)]\!] = \nu(R_1 \circ \ldots \circ R_n).([\![\varepsilon_1]\!][\mathsf{qb}(1)/\mathsf{res}] \mid \ldots \mid [\![\varepsilon_n]\!][\mathsf{qb}(n)/\mathsf{res}] \mid [\![F]\!])$$

$[\![Y := \varepsilon]\!] = [\![\varepsilon]\!]\ \mathit{Into}(x)\ \mathsf{put}(Y)_x.\ \mathit{Done}$

$[\![C; C']\!] = [\![C]\!]\ \mathit{Before}\ [\![C']\!]$

$[\![\texttt{if}\ \varepsilon\ \texttt{then}\ C\ \texttt{else}\ C']\!] = [\![\varepsilon]\!]\ \mathit{Into}(x)\ (\mathsf{if}\ x\ \mathsf{then}\ [\![C]\!]\ \mathsf{else}\ [\![C']\!])$

$[\![\texttt{while}\ \varepsilon\ \texttt{do}\ C]\!] = W \quad ,\text{where} \quad W = [\![\varepsilon]\!]\ \mathit{Into}(x)\ (\mathsf{if}\ x\ \mathsf{then}\ ([\![C]\!]; W)\ \mathsf{else}\ \ \mathit{Done})$

$[\![C\ \texttt{par}\ C']\!] = [\![C]\!]\ \mathit{Par}\ [\![C']\!]$

$[\![\texttt{input}\ Y]\!] = \mathsf{in}^x.\mathsf{put}(Y)_x.\ \mathit{Done}$

$[\![\texttt{output}\ \varepsilon]\!] = [\![\varepsilon]\!]\ \mathit{Into}(x)\ (\mathsf{out}_x.\ \mathit{Done})$

$[\![\texttt{skip}]\!] =\ \mathit{Done}$

Figure 12: Semantics of commands of **P**

where each resource R_i mentioned has $R_i = (\emptyset, \emptyset, R_i^{bu})$ and the R_i^{bu}, $\mathsf{qb}(i)$, $\mathsf{pb}(i)$ are distinct from all other buffers used within the term. In particular, $R_i^{bu}(j)$ is defined for a unique j, say j_i, and $R_i^{bu}(j_i) = 0$, and $R_k^{bu}(j_i) \uparrow$ for all $1 \leq i, k \leq n$ and $k \neq i$. This definition is not compatible with the piggybacking restriction on hiding from **SCRPr**ν.

We now turn to the denotation of commands. Processes used to denote commands are constructed in such a way that they send out an action done before terminating, if they terminate, and these actions occur only immediately prior to termination.

Define auxilliary control operators as follows:

$$\begin{aligned}
\mathit{Done} &= \mathsf{done}.0 \\
E\ \mathit{Before}\ F &= \nu R_i.(E[\mathsf{qs}(i)/\mathsf{done}] \,|\, \mathsf{ps}(i).F) \\
E\ \mathit{Par}\ F &= \nu(R_i \circ R_j).\ (E[\mathsf{qs}(i)/\mathsf{done}] \mid F[\mathsf{qs}(j)/\mathsf{done}] \mid \\
&\quad (\mathsf{ps}(i).\mathsf{ps}(j).\ \mathit{Done} + \mathsf{ps}(j).\mathsf{ps}(i).\ \mathit{Done} + \mathsf{ps}(i)\mathsf{ps}(j).\ \mathit{Done})) \\
E\ \mathit{Into}(x)\ F &= \nu R_k.(E[\mathsf{qb}(k)/\mathsf{res}] \mid \mathsf{pb}(k)^x.F)
\end{aligned}$$

where R_i and R_j each represent fresh semaphores (unused by actions in E and F) initialized to 0. In particular, $R_i = (\emptyset, {R^{sm}}_i, \emptyset)$ with ${R^{sm}}_i(n)$ defined for a unique $n = n_i$, and with ${R^{sm}}_i(n_i) = 0$. The process *Done* is a simple terminating process. The combinator *Before* is used to sequence processes. This is done by ensuring that E increments the semaphore R_i (using $\mathsf{qs}(i)$) immediately before termination, if that happens, and that the semaphore must be decremented (using $\mathsf{ps}(i)$) before F begins. The combinator *Par* is used to parallel compose processes. The term $\mathsf{ps}(i)\mathsf{ps}(j).\ \mathit{Done}$ is necessary in *Par* because the parallel process combinator | allows for the possibility that E and F complete in synchrony. The combinator $\mathit{Into}(x)$ takes a process E that outputs a value and transfers this value into a second process F through the private, mediating buffer R_k. The resource R_k represents a fresh buffer with $R_k = (\emptyset, \emptyset, {R^{bu}}_k)$, with ${R^{bu}}_k(n)$ defined for a unique $n = n_k$, and with ${R^{bu}}_i(n_k) = v$, where v is the value signalled by the output res_v of E.

The semantics of commands is given in Figure 12.

Let $R_0 = ({R^{st}}_0, \emptyset, \emptyset)$ be the resource with ${R^{st}}_0(Y) = 0$ for all program variables Y. A program of **P** is taken to be a command C. The behaviour of C is then determined by the operational behaviour of the translation of $[\![C]\!]$ into **SCRP**ν, considered in the environment R_0. Thus if $\mathcal{T}_{apv}$ is the translation from the asynchonous value-passing notation into **SCRP**ν then the semantics

$$\begin{array}{lll} \varepsilon ::= & \ldots \mid \varepsilon.i \\ \\ C ::= & \ldots \mid Y := \varepsilon.i \mid \varepsilon.i := \varepsilon' \mid Y := \mathtt{cons}(\varepsilon, \varepsilon') \\ & \mid \mathtt{dispose}(\varepsilon) \mid \mathtt{with}\ i\ \mathtt{when}\ \varepsilon\ \mathtt{do}\ C\ \mathtt{endwith} \end{array}$$

Figure 13: Syntax of commands of **PH**

of the program is determined by transitions of $R_0, \mathcal{T}_{apv}([\![C]\!])$.

The method above can be extended to allow block commands of the form

$$\mathtt{begin}\ D\ ;\ C\ \mathtt{end}$$

containing local variable declarations of the form $D ::= \mathtt{Var}\ Y$. A semantics can then be given by first extending the domain of the store component of each resource R with a disjoint countable set $TVar$ of temporary variables, giving a partial function $R^{st} : GVar \cup TVar \longrightarrow Vals$, where $GVar$ is the set of global variables. One then extends the definition of μ so that the Y in $\mathtt{put}(Y)_v$ and $\mathtt{get}(Y)^v$ ranges over both global and temporary variables. We then take

$$[\![\mathtt{Var}\ Y; C]\!] = \nu R_{Y'}.[\![C]\!][Y'/Y]$$

where Y'' is a fresh temporary variable which we substitute for Y, and $R_{Y'} = ({R^{st}}_{Y'}, \emptyset, \emptyset)$, ${R^{st}}_{Y'}(X) = 0$ if $X = Y'$ and ${R^{st}}_{Y'}$ is undefined for all other global and temporary variables X. Localization of variables is then realized as an instance of hiding. We have not attempted to extend the method to allow for procedure declarations in D.

5.6 Heap Manipulating Commands

We form a language **PH** by adding commands for manipulating pointers to **P**. These commands are similar to those found in languages used to illustrate Separation Logic for concurrency (O'Hearn 2007). In particular, the language has commands for assignment from a heap cell, update of a heap cell, pointer creation, pointer disposal, and conditional critical regions that are protected by semaphores.

The set of values is extended to include a set *Loc* of locations ranged over by l. The extensions to the syntax are shown in Figure 13. Note that locations are expressions. The expression $\varepsilon.i$ extracts the ith component of a location defined by ε, and is undefined if ε does not evaluate to a location. The command $Y := \varepsilon.i$ stores the contents of $\varepsilon.i$ at the global variable Y. The command $\varepsilon.i := \varepsilon'$ sets the contents of $\varepsilon.i$ to the value of ε'. The command $Y := \mathtt{cons}(\varepsilon, \varepsilon')$ inserts the values ε and ε' at some fresh location l and stores the value l at the global variable Y. The command $\mathtt{dispose}(\varepsilon)$ removes the location ε from the heap. The command $\mathtt{with}\ i\ \mathtt{when}\ \varepsilon\ \mathtt{do}\ C\ \mathtt{endwith}$ executes the command C when, both ε evaluates to $\mathtt{true}$ and the ith semaphore is free. The command C then owns the ith semaphore whilst executing: the disjointness condition on semaphores in resource composition means that no other command which requires the ith semaphore can make progress whilst C does.

Resources R now take the form $(R^{st}, R^{sm}, R^{bu}, R^{hp})$ where the store, sempahore and buffer components are as before and the *heap*, R^{hp}, is a partial function from a given countable set *Loc* of locations to pairs of values (conventionally said to be contained in cells). The heap R^{hp} is assumed to be finite in the sense that it is defined at only finitely many locations. We write $R^{hp}(l)(j)$ for the jth component of the pair at location l. Note that pointers to be stored at program variables and in cells because the set of values now contains locations. The composite of two resources is defined just when the stores, semaphores, buffers and heaps defined at one resource are disjoint from the stores, semaphores, buffers and heaps defined at the other resource: under such circumstances it is defined by the union of graphs.

$$
\begin{array}{rcl}
\mu(\mathsf{rem}_v, R) & = & \begin{cases} (R^{st}, R^{sm}, R^{bu}, R^{hp} \ominus v) & \text{if } R^{hp}(v) \downarrow \\ \uparrow & \text{otherwise} \end{cases} \\[2ex]
\mu(\mathsf{alloc}_v, R) & = & \begin{cases} R \oplus \{R^{hp}(v) = (0,0)\} & \text{if } R^{hp}(v) \uparrow \text{ and } v \in \mathit{Loc} \\ \uparrow & \text{otherwise} \end{cases} \\[2ex]
\mu(\mathsf{puth}(j)_{v,w}, R) & = & \begin{cases} R \oplus \{R^{hp}(v)(j) = w\} & \text{if } R^{hp}(v) \downarrow \\ \uparrow & \text{otherwise} \end{cases} \\[2ex]
\mu(\mathsf{geth}(j)^w_v, R) & = & \begin{cases} R & \text{if } R^{hp}(v)(j) = w \text{ and } v \in \mathit{Loc} \\ \uparrow & \text{otherwise} \end{cases}
\end{array}
$$

Figure 14: Modification function on atoms for heap manipulation

To the atomic actions above we add,

$$\mathsf{rem}_\varepsilon \qquad \mathsf{alloc}_\varepsilon \qquad \mathsf{puth}(i)_{\varepsilon,\varepsilon'} \qquad \mathsf{geth}(i)^x_\varepsilon \ ,$$

where $i = 1, 2$, to the set of atomic actions and generate a new action monoid; A mixed input-output action $\mathsf{geth}()$ is now present.

For any partial function $f : X \longrightarrow Y$ and $x \in X$ define $f \ominus x : X \longrightarrow Y$ by $f \ominus x(x') \simeq f(x')$ if $x' \neq x$ and $f \ominus x(x) \uparrow$.

The modification μ is chosen so that the action $\mathsf{puth}(i)_{\varepsilon,\varepsilon'}$ puts the value ε' at the ith component of the cell at location ε; the action $\mathsf{geth}(i)^x_\varepsilon$ retrieves the value stored at the ith component of the cell at location ε and binds it to x; the action $\mathsf{alloc}_\varepsilon$ is used in the allocation of a location ε, and is undefined if the location is not fresh; the action rem_ε removes the heap cells at location $[\![\varepsilon]\!]$. In each of these, the expression ε must evaluate to some location, otherwise the modification is undefined at every resource. The extensions of the modification to the new atomic actions is shown in Figure 14. It is worth noting that the action of the modification at alloc_v is incompatible with the simple-extension property, and so we use **SCRP**ν rather than **SCRPr**.

The roots of the atomic actions are as follows:

$$
\begin{array}{rcl}
\mathsf{Roots}(\mathsf{rem}_v) & = & \{(\emptyset, \emptyset, \emptyset, \emptyset_{v,w}) \mid w \in \mathit{Vals} \times \mathit{Vals}\} \\[1ex]
\mathsf{Roots}(\mathsf{alloc}_v) & = & \{e\} \\[1ex]
\mathsf{Roots}(\mathsf{puth}(j)_{v,w}) & = & \{(\emptyset, \emptyset, \emptyset, \emptyset_{v,v'}) \mid v' \in \mathit{Vals} \times \mathit{Vals}\} \\[1ex]
\mathsf{Roots}(\mathsf{geth}(j)^w_v) & = & \{(\emptyset, \emptyset, \emptyset, \emptyset_{v,v'}) \mid v' \in \mathit{Vals} \times \mathit{Vals} \text{ and } v'(j) = w\} \ .
\end{array}
$$

The modification that follows from the data in Figure 14 is then defined as in Proposition 21.

We extend the semantics of expressions above to translate the components of expressions that evaluate to locations by taking

$$[\![\varepsilon.i]\!] = [\![\varepsilon]\!] \ \mathit{Into}(z) \ \mathsf{geth}(i)^x_z.\mathsf{res}_x.\mathsf{0} \ .$$

for $i = 1, 2$. The extension of the semantics of commands is in Figure 15. The resource $(\emptyset, \emptyset_{i,1}, \emptyset, \emptyset)$ represents the ith semaphore (set to 1).

Milner evaluated his translation of a programming language into CCS by proving that the intended algebraic and logical structure of programs holds in the translation. It remains an open problem to define a calculus in the **SCRP**-family for which a well-behaved asynchronous equivalence exists, in which one may prove expected program equivalences using a semantics of **P** or **PH** in the style above. We anticipate a close connection between the Hennessy-Milner logic **MBIc** in this context and the Separation Logic for concurrency of (O'Hearn 2007). Concurrent

$[\![Y := \varepsilon.i]\!] = [\![\varepsilon.i]\!]\ \mathit{Into}(x)\ (\mathsf{put}(Y)_x.\ \mathit{Done})$

$[\![\varepsilon.i := \varepsilon']\!] = [\![\varepsilon]\!]\ \mathit{Into}(z)\ ([\![\varepsilon']\!]\ \mathit{Into}(x)\ (\mathsf{puth}(i)_{z,x}\ .\ \mathit{Done}))$

$[\![\mathtt{dispose}(\varepsilon)]\!] = \mathsf{rem}_{[\![\varepsilon]\!]}.\ \mathit{Done}$

$[\![\mathtt{with}\ i\ \mathtt{when}\ \varepsilon\ \mathtt{do}\ C\ \mathtt{endwith}]\!] = \nu(\emptyset, \emptyset_{i,1}, \emptyset, \emptyset).\ \mathsf{if}\ [\![\varepsilon]\!]\ \mathsf{then}\ [\![C]\!]$

$[\![Y := \mathtt{cons}(\varepsilon, \varepsilon')]\!] = [\![\varepsilon'[y/x]]\!]\ \mathit{Into}(y)\ ([\![\varepsilon]\!]\ \mathit{Into}(x)\ A(x,y))$, where

$A(x,y) = \sum_{l \in Loc} \mathsf{alloc}_l.\mathsf{put}(Y)_l.\mathsf{puth}(1)_{l,x}.\mathsf{puth}(2)_{l,y}.\ \mathit{Done}$

Figure 15: Semantics of commands of **PH**

separation logic is basically a Floyd-Hoare logic (with a **BI**-fragment as assertion language). Given a translation of the programming language into a **SCRP**-calculus, any triple $\{\phi\}C\{\psi\}$ corresponds to an assertion of the form $\phi \longrightarrow \bigwedge_{a_1 \ldots a_n \in \mathcal{A}([\![C]\!])}([a_1] \ldots [a_n]\psi)$, where $\mathcal{A}([\![C]\!])$ is the (finite) set of all finite sequences of actions which the translated process $[\![C]\!]$ can make. It would therefore be interesting to see whether the Concurrent Separation Logic rules can be derived from our logic.

6 Directions

There are a great many open problems and possible avenues in this area. In this section, we outline a few of these.

The process constuctors of **SCRP** seem natural, and appear to be sufficiently powerful for the description of many (if not most) concurrent situations. We have not, however, yet proved any kind of completeness result. This could, perhaps, be done by either proving a result along the lines of (Simone 1985), or by finding a suitable resource monoid such that there is a well-behaved translation of SCCS translated into **SCRP**. Equally, it would be very interesting to see if the process constructors we have defined emerge naturally as universal constructions in a fibred category following the methodology described in (Winskel & Nielsen 1995). It may be that a presheaf semantics for SCCS (Hildebrandt 1999) may also have useful connections — sheaves and presheaves can be used to interpret multiplicative conjunction (Pym 2002). Such approaches would also be helpful in assessing the value of further rules.

The availability of good model-checking procedures has been fundamental to the successful application of process algebras and modal-logics. Preliminary work has been done on implementing prototype systems for the automatic construction of **SCRP**-transitions and for model-checking formulae of **MBI**. Recall that the general model-checking problem is to decide whether a given state satisfies a given formula. We note that the model-checking problem for the whole of **SCRP** and **MBIc** is much harder than traditional model-checking for process algebras. The essential difference is that multiplicative formulae demand that subformulae must be checked against states that are outside the transition structure generated by the given state. In particular, this involves unbounded searches across infinite sets of states. Strategies for bounding such searches, such as the use of the underlying order and properties of the modification function will be critical to establishing better algorithms.

The model-checking of the multiplicative conjunction raises another issue. In order to check that some relation $R, E \vDash \phi_1 * \phi_2$ is statisfied, it is necessary to find R_1, R_2, E_1 and E_2 with $R = R_1 \circ R_2$, $E \sim E_1 \times E_2$, $R_1, E_1 \vDash \phi_1$ and $R_2, E_2 \vDash \phi_2$. In general, the global bisimulation will not be decidable. An alternative is to use the structural congruence $\equiv$ between processes. This is the congruence generated from associativety, commutativity and unit axioms for the sum and product constructors. These axioms are all true for global bisimulation when the frame rule is admissible,

so the global bisimulation must contain the structural congruence. The structural congruence is decidable. Hence, a better alternative for the interpretation of multiplicative conjunction might be to use: $R, E \vDash \phi_1 * \phi_2$ iff $\exists R_1, R_2, E_1, E_2.\ R = R_1 \circ R_2$ and $E \equiv E_1 \times E_2$ and $R_1, E_1 \vDash \phi_1$ and $R_2, E_2 \vDash \phi_2$ for all R, E, ϕ_1, ϕ_2. Suppose that we also change the interpretation of the multiplicative unit to $R, E \vDash I$ iff $R = e$ and $E \equiv \mathsf{1}$, for all R, E. Then:

- Theorem 25 holds with $\equiv$ in place of $\sim$;
- the axioms of **BI** are all soundly interpreted.

We intend to investigate this promising avenue further, in particular for model-checking. We note that (Cardelli & Gordon 1998) also contains a version of multiplicative conjunction, written $|$, which is interpreted by splitting processes using a structural congruence.

It appears that boundedness and convergence properties of modifications will be of help in dealing with some of these model-checking problems. They may also be the kind of properties that we wish to check. To this end we have constructed a version of **SCRP** that works over a topological, rather than ordered, resource space and have shown how to interpret **MBIi** as open sets of a space induced on states. However, this requires placing certain continuity conditions on **SCRP**-constructors and it remains to evaluate the practicality of the approach. This work dovetails rather well with our goal of studying the relationship between process algebras and other mathematical models of dynamical systems. In connection with this, we have also produced results regarding preservation results of dynamical and logical properties under transformation of the resource base. We imagine that this will have applications to the study of data refinement.

Open problems exist in establishing precise correspondences between the algebraic notions of equivalence (simulations) and logical equivalence. Is there a logic that characterizes the relation $\sim$, at least on some reasonable class of resource monoids? Alternatively, is there a simulation relation bigger than $\sim$, but smaller than $\approx$, such that suitable versions of Theorem 25 and Theorem 34 both hold for that simulation and for the whole language **MBI**?

The questions of completeness of the intepretation of **MBIc**, **MBIq** and of the equational theory of $\sim$ all remain open.

A good question is what the relation $\sim$ is for from an operational (rather than logical) point of view, given that $\approx$ characterizes observational and denotational equivalence (Pym & Tofts 2007, Collinson et al. 2007). In many modelling situations, resources are real-valued quantities. In such situations evolutions may be dependent upon functions that have rather wild (chaotic) behaviour. In such circumstances, it is important to understand, and be able to make guarantees about, the stability properties of such functions under small perturbations. There is a great deal of literature on this topic, but, for example, the recent paper (Hoyrup 2007) studies various notions of stability and computability, with a view to understanding those systems that admit reliable computer simulations. A careful reading of Example 33 shows that the logical connectives $\langle a \rangle$ and $-\!\!*$ make assertions about perturbations of the resource component during the evolution of the state. The relation $\sim$ can be seen to identify processes that have equivalent behaviour under all such perturbations of states that occur during evolution. It remains to evaluate to what extent **SCRP**, **MBI** and $\sim$ (or topological variants thereof) are practical modelling tools for problems in this area.

We have presented proof systems for variants of **MBI**. We have not considered the completeness of these proof systems with respect to their transition structure semantics. Indeed, since we are concerned only with one particular underlying structure, namely that generated by **SCRP**-transitions for each choice of $(\mathbf{R}, \mathsf{Act}, \mu, \nu)$, a complete axiomatization may be difficult. We have so far paid little attention to proof-theoretic aspects of **MBI**, for example, normalization/cut-elimination. The problem of finding decision procedures for this logic is particularly important because of the difficulties involved in model-checking. A full analysis of the logical rules and models for multiplicative quantification remains to be worked out. We expect that the models should be instances of (pre)sheaf models of predicate modal logic and that our previous work on multiplicative quantification, (Collinson, Pym & Robinson 2005, O'Hearn & Pym 1999, Pym 1999, Pym 2002), should provide some guidelines.

Further developments of the process calculus should also prove valuable. The addition of weights on actions (for priority and probabilistic distribution) along the lines of (Tofts 1994) is essential if the calculus is to be developed into a mature and sufficiently expressive modelling tool. We have begun a further refinement of the calculus to one in which states carry a component signifying location. This location is intended as an additional guard against action, but will evolve in a different way (movement between connected locations, rather than consumption) to resource in some of the modelling situations we have in mind. In particular, practical modelling work suggests that such an explicit notion of location would be of use in describing certain security properties of large-scale distributed systems.

Acknowledgements

We are grateful to Chris Tofts, who declined to be named as an author, for his input to this work, to Matthew Hennessy and the anonymous referees for raising questions which led to significant corrections. We are also grateful to Brian Monahan and Mike Yearworth for their support.

References

Baeten, J. (2005), 'A brief history of process algebra', *Theoretical Computer Science* **335**(2–3), 131–146.

Ben-Ari, M. (1990), *Principles of Concurrent and Distributed Programming*, Prentice Hall.

Biri, N. & Galmiche, D. (2007), 'Models and separation logics for resource trees', *Journal of Logic and Computation* **17**(4), 687–726.

Birtwistle, G. (1979), *Demos — discrete event modelling on Simula*, Macmillan.

Birtwistle, G. & Tofts, C. (2001*a*), 'Getting demos models right. (i). practice', *Simulation Practice and Theory* **8**(6–7), 377–393.

Birtwistle, G. & Tofts, C. (2001*b*), 'Getting demos models right. (ii) ... and theory', *Simulation Practice and Theory* **8**(6–7), 395–414.

Birtwistle, G., Pooley, R. & Tofts, C. (1993), 'Characterising the structure of simulations using CCS', *Transactions of the Simulation Society* **10**(3), 205–236.

Calcagno, C., Gardner, P. & Zarfaty, U. (2005), Context logic and tree update, *in* 'Proceedings of the 32nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages 2005 (POPL)', pp. 271–282.

Cardelli, L. & Gordon, A. (1998), Mobile ambients, *in* M. Nivat, ed., 'Foundations of Software Science and Computational Structures', Vol. 1378 of *Lecture Notes in Computer Science*, Springer, pp. 40–155.

Cleaveland, R., Parrow, J. & Steffen, B. (1993), The concurrency workbench: A semantics based tool for the verification of concurrent systems, *in* 'ACM Transactions on Programming Languages and Systems', Vol. 15, pp. 36–72.

Collinson, M., Pym, D. & Robinson, E. (2005), On Bunched Polymorphism (Extended Abstract), *in* 'Computer Science Logic', Vol. 3634 of *LNCS*, Springer.

Collinson, M., Pym, D. & Tofts, C. (2007), 'Errata for Formal Aspects of Computing (2006) 18:495–517 and their consequences', *Formal Aspects of Computing* **19**(4), 551–554.

Conforti, G., Macedonio, D. & Sassone, V. (2007), 'Static bilog: a unifying language for spatial structures', *Fundamenta Informaticae* **80**, 1–20.

Dahl, O.-J., Myhrhaug, B. & Nygaard, K. (1970), Simula 67 Common Base Language, NCC Publication S-52, Norwegian Computing Center, Oslo.

Day, B. (1970), On closed categories of functors, *in* 'Proceedings of the Midwest Category Seminar', Vol. 137 of *LNM*, Springer.

Day, B. (1973), An embedding theorem for closed categories, *in* 'Proceedings of the Sydney Category Seminar 1972/73', Vol. 420 of *LNM*, Springer.

Demos2k (2002). `http://www.demos2k.org`.

Hennessy, M. & Milner, R. (1985), 'Algebraic laws for nondeterminism and concurrency', *Journal of the ACM* **32**(1), 137–161.

Hildebrandt, T. (1999), A Fully Abstract Presheaf Semantics of SCCS with Finite Delay, *in* 'Proceedings of CTCS'99', Electronic Notes in Theoretical Computer Science, Elsevier.

Hinton, A., Kwiatkowska, M., Norman, G. & Parker, D. (2006), Prism: A tool for automatic verification of probabilistic systems, *in* H. Hermanns & J. Palsberg, eds, 'Proc. 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'06)', Vol. 3920, Springer, pp. 441–444.

Hoyrup, M. (2007), 'Dynamical systems: stability and simulability', *Mathematical Structures in Computer Science* **17**(2), 247–259.

Ishtiaq, S. & O'Hearn, P. (2001), BI as anassertion language for mutable data structures, *in* 'Proceedings of the 28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages 2001 (POPL), London', pp. 14–26.

Joyal, A., Nielsen, M. & Winskel, G. (1996), 'Bisimulation from open maps', *Information and Computation* **127**, 164–185.

Kripke, S. (1963), 'Semantical analysis of modal logic I', *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik* **9**, 67–96.

Kripke, S. (1965), Semantical analysis of intutionistic logic I, *in* J. Crossley & M. Dummett, eds, 'Formal Systems and Recursive Functions', Studies in Logic and the Foundations of Mathematics, North-Holland Publ. Co., pp. 92–130.

Lawvere, F. (1969), 'Adjointness in foundations', *Dialectica* **23**, 281–296.

Leifer, J. J. & Milner, R. (2000), Deriving bisimulation congruences for reactive systems, *in* 'Proc. CONCUR 2000', Vol. 1877 of *Lecture Notes in Computer Science*, Springer-Verlag.

Milner, R. (1980), *A Calculus of Communicating Systems*, Vol. 92 of *LNCS*, Springer Verlag.

Milner, R. (1983), 'Calculi for synchrony and asynchrony', *Theoretical Computer Science* **25**, 267–310.

Milner, R. (1989), *Communication and Concurrency*, Prentice-Hall.

Milner, R. (1999), *Communicating systems and the π-calculus*, Cambridge University Press.

O'Hearn, P. (2007), 'Resources, concurrency and local reasoning', *Theoretical Computer Science* **375**(1–3), 271–307.

O'Hearn, P. & Pym, D. (1999), 'The logic of bunched implications', *Bulletin of Symbolic Logic* **5**(2), 215–244.

Plotkin, G. (2004), 'Structural operational semantics', *Journal of Logic and Algebraic Programming* **60**, 17–139. Original unpublished manuscript 1981.

Popkorn, S. (1994), *First Steps in Modal Logic*, Cambridge University Press.

Pym, D. (1999), On bunched predicate logic, *in* 'Proceedings. 14th Symposium on Logic in Computer Science (LICS99), Trento, Italy', IEEE Computer Society Press, pp. 183–192.

Pym, D. (2002), *The Semantics and Proof Theory of the Logic of Bunched Implications*, Vol. 26 of *Applied Logic Series*, Kluwer Academic Publishers. Errata at: `http://www.cs.bath.ac.uk/~pym/BI-monograph-errata.pdf`.

Pym, D. & Tofts, C. (2006), 'A calculus and logic of resources and processes', *Formal Aspects of Computing* **18**(4), 495–517. Errata available: `http://www.cs.bath.ac.uk/~pym/pym-tofts-fac-errata.pdf`.

Pym, D. & Tofts, C. (2007), Systems Modelling via Resources and Processes: Philosphy, Calculus, Semantics, and Logic, *in* L. Cardelli, M. Fiore & G. Winskel, eds, 'Computation, Meaninig and Logic: articles dedicated to Gordon Plotkin', Vol. 172 of *Electronic Notes in Theoretical Computer Science*, Elsevier, pp. 545–587. Errata at: `http://www.cs.bath.ac.uk/~pym/pym-tofts-fac-errata.pdf`.

Pym, D., O'Hearn, P. & Yang, H. (2004), 'Possible worlds and resources: The semantics of **BI**', *Theoretical Computer Science* **315**(1), 257–305.

Reynolds, J. (2002), Separation logic: a logic for shared mutable data structures, *in* 'Proceedings. 17th Annual IEEE Symposium on Logic in Computer Science (LICS02), Copenhagen, Denmark', IEEE Press, pp. 55–74.

Sassone, V. & Sobociński, P. (2003), 'Deriving bisimulation congruences using 2-categories', *Nordic Journal of Computing* **10**, 163–183.

Sewell, P. (1998), 'From rewrite rules to bisimulation congruences', *Lecture Notes in Computer Science* **1466**, 269–284.

Simone, R. d. (1985), 'Higher-level synchronising devices in Meije-SCCS', *Theoretical Computer Science* **37**, 245–267.

Stirling, C. (2001), *Modal and temporal properties of processes*, Springer.

Tofte, M. & Talpin, J.-P. (1997), 'Region-based memory management', *Information and Computation* **132**, 109–176.

Tofts, C. (1994), 'Processes with probabilities, priority and time', *Formal Aspects of Computing* **6**, 536–564.

Tofts, C. (2006), 'Process algebra as modelling', *Electronic Notes in Theoretical Computer Science* **162**, 323–326. Proceedings of the Workshop "Essays on Algebraic Process Calculi" (APC25).

Victor, B. & Moller, F. (1994), The Mobility Workbench — a tool for the π-calculus, *in* D. Dill, ed., 'CAV'94: Computer Aided Verification', Vol. 818 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 428–440.

Winskel, G. (1993), *The formal semantics of programming languages, an introduction*, MIT Press.

Winskel, G. & Nielsen, M. (1995), Models for concurrency, *in* 'Handbook of Logic in Computer Science', Vol. 4, Oxford University Press, pp. 1–148.