



A Calculus and Logic of Resources and Processes[♦]

David Pym, Chris Tofts
Trusted Systems Laboratory
HP Laboratories Bristol
HPL-2004-170(R.2)
October 19, 2006*

logic, concurrency,
resources,
processes,
modelling, parallel

Recent advances in logics for reasoning about resources provide a new approach to compositional reasoning in interacting systems. We present a calculus of resources and processes, based on a development of Milner's synchronous calculus of communication systems, SCCS, that uses an explicit model of resource. Our calculus models the co-evolution of resources and processes with synchronization constrained by the availability of resources. We provide a logical characterization, analogous to Hennessy-Milner logic's characterization of bisimulation in CCS, of bisimulation between resource processes which is compositional in the concurrent and local structure of systems.

This technical report has been updated to match exactly the version of the paper to be published under the same title in the journal *Formal Aspects of Computing*.

A CALCULUS AND LOGIC OF RESOURCES AND PROCESSES

DAVID PYM AND CHRIS TOFTS

ABSTRACT. Recent advances in logics for reasoning about resources provide a new approach to compositional reasoning in interacting systems. We present a calculus of resources and processes, based on a development of Milner's synchronous calculus of communication systems, SCCS, that uses an explicit model of resource. Our calculus models the co-evolution of resources and processes with synchronization constrained by the availability of resources. We provide a logical characterization, analogous to Hennessy-Milner logic's characterization of bisimulation in CCS, of bisimulation between resource processes which is compositional in the concurrent and local structure of systems.

This technical report has been updated to match exactly the version of the paper to be published under the same title in the journal *Formal Aspects of Computing*.

1. INTRODUCTION

The notion of *resource* is a basic one in many fields, including economics, engineering, and the humanities, but it is perhaps most clearly illuminated in the computing sciences. The location, ownership, access to, and consumption of resources are central concerns in the design of systems, such as networks, within which processors must access devices such as file servers, disks, and printers, and in the design of programs, which access memory and manipulate data structures, such as pointers.

Within mathematical models of computational systems, however, the rôle of resource is quite central. This observation is illustrated quite directly in modelling systems such as Demos [Bir79, Bir81, BT93, BT94, BT98, BT00a, BT00b], in which the central notions are *entities*, that are processes which execute trajectories within the model, and *resources*, which are required to enable, and are manipulated by, entities' actions. Systems such as Demos, however, implement a process-theoretic view of the world in which not only is the dynamics of systems represented as processes but so too are the essentially static resource components. We would argue that this situation is conceptually unsatisfactory. Moreover, pragmatically, the computational cost of modelling interactive systems is, typically, dominated by the handling of the resource components.

We present a calculus of resources and processes, based on a development of Milner's synchronous calculus of communication systems, SCCS, that uses an explicit model of resource. Our calculus models the co-evolution of resources and processes with synchronization constrained by the availability of resources. We provide a logical characterization, analogous to Hennessy-Milner logic's characterization of bisimulation in CCS, of bisimulation between resource processes which is compositional in the concurrent and local structure of systems. Thus we are able to address

the problem of providing a compositional framework for reasoning about concurrent programs. That is, expressed in terms of a Hoare-style logic [OG76], give a rule for concurrent composition of the form

$$\frac{\{\phi_1\} C_1 \{\psi_1\} \quad \{\phi_2\} C_2 \{\psi_2\}}{\{\phi_1 \text{ op}_1 \phi_2\} C_1 \text{ par } C_2 \{\psi_1 \text{ op}_2 \psi_2\}}$$

subject only to minimal side-conditions.

Our explicit model of resource is based on the semantics of the logic of bunched implications, **BI**, introduced in [OP99, Pym02, Pym04, POY04, GMP02]. As O’Hearn [O’H04] discusses, there is some history, dating from the 1960s and 1970s, of work on resource control in the context of work on multiprogramming [BH72], operating systems [BH73], and concurrent programming [Dij68, Dij71, Hoa72, Hoa74, Hoa85]. More recently, O’Hearn [O’H04] has considered a resource-based view of local reasoning in concurrent systems based on semantics in the Hoare logic-style of separation logic for Owicki and Gries’s calculus of resource declarations and concurrent commands [OG76]. Our analysis operates at the conceptually more general level of process calculus and, consequently, gives a direct analysis of the interaction between processes and resources.

In § 2, we give a conceptual overview of our views of resources and of processes. Our basic model of resource derives from that considered in the setting of the logic of bunched implications, **BI**, introduced by O’Hearn and Pym [OP99, Pym02, Pym04, POY04, GMP02], in which multiplicative (or linear) and additive (or intuitionistic/classical) connectives co-exist at the same level of abstraction. The semantics of **BI** is naturally motivated as a model of resource, later influenced by developments of sophisticated examples such as pointer logic [IO01] and separation logic [Rey02]. **BI**’s semantics is sketched in § 6. Our model of processes derives from Milner’s SCCS, which has the asynchronous calculus CCS as a sub-calculus, influenced by developments in the π -calculus [Mil99]. In § 3, we give a detailed definition of our calculus of resource-processes, **SCRP**. In § 5, we provide examples of the use of **SCRP** to specify some core aspects of concurrent systems, namely *mutual exclusion*, *resource transfer*, *handshaking*, *private channels*, and *asynchronous handover*. In § 6, we provide a brief review of **BI** and, in § 7, we give the definition of **MBI**, a Hennessy-Milner-style modal logic for **SCRP**, based on **BI**. Then, in § 8, we show how **MBI** may be used to specify properties of concurrent systems specified in **SCRP**, including examples of **MBI**’s specification of the properties of concurrent composition and of local resources. In § 9, we prove that bisimulation in **SCRP** is characterized by logical equivalence in **MBI**. In § 10, we provide a brief discussion of the analysis of concurrent imperative programs provided by our framework. We conclude, in § 11, with a brief discussion of some further work.

2. RESOURCES, PROCESSES, AND LOGICS

A mathematical account of a useful notion of resource can be given using logic. Our starting position is that the following properties are reasonable requirements for a simple model of resource [Pym02, Pym04, POY04, GMP02]:

- A set \mathbf{R} of resource elements;
- A (partial) combination, $\circ : \mathbf{R} \times \mathbf{R} \rightarrow \mathbf{R}$ of resource elements;
- A comparison, \sqsubseteq , of resource elements; and
- A zero resource element, e .

In the usual spirit and methodology of mathematically modelling, these conceptually evidently well-motivated properties correspond well to a wide a range of natural examples [POY04, Pym02, Pym04]. Mathematically, we obtain this structure as pre-ordered partial commutative monoid,

$$\mathcal{R} = (\mathbf{R}, \circ, e, \sqsubseteq),$$

subject to the condition that if $r \sqsubseteq s$ and $r' \sqsubseteq s'$, then $r \circ r' \sqsubseteq s \circ s'$, and, recalling the preordering of a Kripke structure [Kri63, Kri65], call it a *Kripke resource monoid*, or KRM, with worlds being resources. The ordering \sqsubseteq gives rise to an equality,

$$= := \supseteq \cup \sqsubseteq.$$

A simple example is provided by the natural numbers, here including 0,

$$\mathcal{N} = (\mathbb{N}, +, 0, \leq),$$

in which combination is given by addition, with unit 0, and comparison is given by less than or equals. This is an example of resource as *cost*.

A richer example is provided by Petri nets [POY04]. Formally, a net

$$\mathcal{N} = (P, T, pre, post)$$

consists of sets P and T of places and transitions and two functions

$$pre, post : T \rightarrow \mathcal{M},$$

from transitions to markings, where a marking is a finite multiset of places and \mathcal{M} denotes the set of all markings. A marking amounts to a function $M : P \rightarrow \mathbb{N}$ from places to natural numbers that is zero on all but finitely many places. Addition of markings is given by

$$(M + N)p = Mp + Np.$$

This formalization of Petri nets can be seen as a Kripke resource monoid in several ways. One way is to internalize the reachability relation on markings. If M and N are markings, then define

$$M \Rightarrow N \text{ iff there are } t, M' \text{ such that } M = pre(t) + M' \text{ and } N = post(t) + M'.$$

We can then define a preorder, \sqsubseteq , on markings by

$$M \sqsubseteq N \text{ iff there are } M_1, \dots, M_n \text{ such that } M = M_1 \Rightarrow \dots \Rightarrow M_n = N.$$

We let $[-]$, the unit of $+$, denote the empty marking. It follows that $(\mathcal{M}, +, [-], \sqsubseteq)$ is a preordered commutative monoid.

Of perhaps more direct relevance to our concerns is the ‘basic separation model’ [POY04, Rey02]. Suppose we are given an infinite set $Res = \{r_0, r_1, \dots\}$. We think of the elements of Res as primitive resources, or resource IDs, that can be allocated and deallocated. The partial monoid structure is given by taking a world to be a finite subset of Res , and \circ to be union of disjoint sets. In more detail, where \uparrow denotes undefinedness (and, later on, \downarrow definedness),

$$m \circ n = \begin{cases} m \cup n & \text{if } m \cap n = \emptyset \\ \uparrow & \text{otherwise.} \end{cases}$$

The unit of \circ is $\{e\}$, and we take \sqsubseteq to be equality. This example is the basis of Ishtiaq and O’Hearn’s pointer logic [IO01] and Reynolds’ separation logic [Rey02].

The composition and ordering structure lifts to sets of resource elements. Let $\wp(\mathbf{R})$ denote the powerset of \mathbf{R} and let $R, S \in \wp(\mathbf{R})$. Then define, for example,

$$R \circ S = \begin{cases} \{r \circ s \mid r \in R \text{ and } s \in S\} & \text{if each } r \circ s \downarrow \\ \uparrow & \text{otherwise,} \end{cases}$$

with unit $\{e\}$ and, for example,¹

$$R \sqsubseteq S \quad \text{iff} \quad \text{for all } r \in R \text{ there is } s \in S \text{ such that } r \sqsubseteq s.$$

Such sets of resources are a convenient level of abstraction for our present purposes, for which we shall require no further special properties. We might also require that $R \circ S$ be defined only if R and S are disjoint. We write R_1, R_2 for the union of R_1 and R_2 , and emphasize that composition is quite different from union. Our notational choices should be clear *in situ*. Other constructions, based on Kripke resource monoids, might also provide a basis for a calculus and logic. The space of choices is, however, quite large, so that a discussion of it is beyond our present scope.

More generally, we might take a more complex structure of resources. For example, we might take $\mathbf{R} = \mathbf{R}_1 \times \dots \times \mathbf{R}_m$, with a composition \circ_i and ordering \sqsubseteq_i on each \mathbf{R}_i .

Kripke resource monoids provide the basis for the semantics of **BI**, the logic of bunched implications [OP99, Pym02, Pym04]. The judgement $r \models \phi$, for $r \in \mathbf{R}$, is read as ‘resource element r is sufficient to support proposition ϕ ’. The ordering structure admits the usual Kripke semantics for the usual, additive, connectives (\top , \wedge , \perp , \vee , \rightarrow) of intuitionistic logic and, in the discrete case, classical logic.² The monoidal structure admits a semantics for a multiplicative conjunction, $*$, given by

$$r \models \phi_1 * \phi_2 \quad \text{iff} \quad \text{there are } s_1 \text{ and } s_2 \text{ such that } s_1 \circ s_2 \sqsubseteq r, \text{ and} \\ s_1 \models \phi_1 \text{ and } s_2 \models \phi_2.$$

The semantics of the multiplicative conjunction, $*$, is interpreted as follows: the resource r is sufficient to support $\phi_1 * \phi_2$ just in case it can be divided into resources s_1 and s_2 such that s_1 is sufficient to support ϕ_1 and s_2 is sufficient to support ϕ_2 . The assertions ϕ_1 and ϕ_2 — think of them as expressing properties of programs — *do not share* resources. In contrast, in the semantics of the additive conjunction, $r \models \phi_1 \wedge \phi_2$ iff $r \models \phi_1$ and $r \models \phi_2$, the assertions ϕ_1 and ϕ_2 may *share* the resource m .

Along with the multiplicative conjunction comes a multiplicative implication, $-*$, given by

$$r \models \phi -* \psi \quad \text{iff} \quad \text{for all } s \text{ such that } s \models \phi, \\ r \circ s \models \psi.$$

The semantics of the multiplicative implication, $-*$, may be interpreted as follows: the resource r is sufficient to support $\phi -* \psi$ just in case for any resource s which is sufficient to support ϕ the combination $r \circ s$ is sufficient to support ψ . We can think of the proposition $\phi -* \psi$ as (the type of) a function and the proposition ϕ as (the type of) its argument. The resources then describe the cost of applying

¹Note that the ordering on $\wp(\mathbf{R})$ given here is just one of many possible choices; see, for example, [Gun92].

²Our use of the terms ‘additive’ and ‘multiplicative’ derives from their use in linear logic [Gir87] and bunched logic [OP99, Pym02, Pym04].

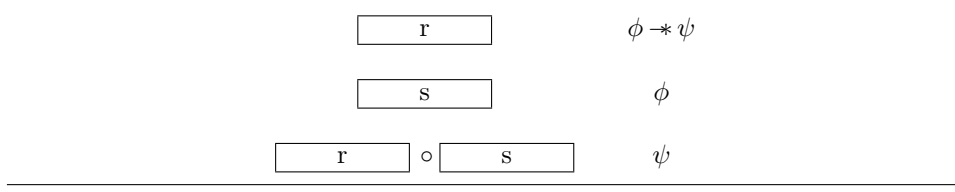
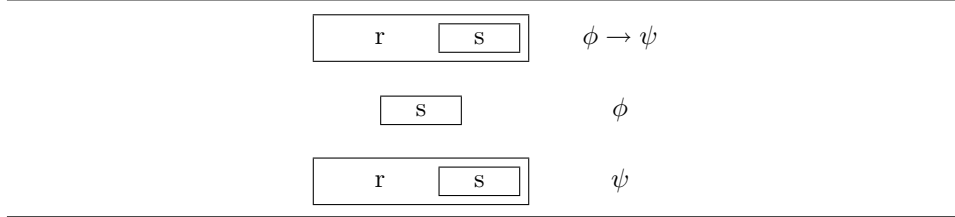


FIGURE 1. Multiplicative Resources

FIGURE 2. Additive Resources ($\phi \rightarrow \psi \stackrel{\text{def}}{=} (\neg\phi) \vee \psi$)

the function to its argument in order to obtain the result. The function and its argument *do not share* resources. This semantics can be pictured as in Figure 1.

In contrast, in the semantics of additive implication, the function and its argument may *share* the resource s . Intuitionistically, $r \models \phi \rightarrow \psi$ iff for all $r \sqsubseteq s$, $s \models \phi$ implies $s \models \psi$; classically, $r \models \phi \rightarrow \psi$ iff $r \models \phi$ implies $r \models \psi$. Figure 2, in which s may be equal to r , illustrates the classical case.³

We emphasize that **BI** and linear logic [Gir87] are very different. Logically, they are incomparable extensions of a basic system, sometimes called Lambek logic, of a (commutative) tensor product, with a unit, and an implication; their treatments of the additives and the structural laws of weakening and contraction are radically different. Moreover, linear logic's resource reading amounts to counting occurrences of propositions, whereas **BI**'s resource semantics incorporates a basic model of the notion of resource.

Returning to our earlier 'basic separation model', we note that taking the ordering to be given by equality gives a model of Boolean **BI**, that is, with classical additives. (A model of **BI** with intuitionistic additives is obtained by taking \sqsubseteq to be inclusion.) In this model, if $\phi * \psi$ holds for a given collection of resources, then ϕ and ψ hold for disjoint sub-collections.

Within a process algebra [Mil83, BK84, Hoa85, Mil89], the common representation of resource is as a separated process. For instance, a semaphore is represented as a two-state process, representing whether the token is currently available or not. There have been extensions [BGL97] that attempt to model resource explicitly but these approaches carry both the communication structures of the process algebras alongside the representation of resource. We take the view that resource *is* the fundamental organizing principle of the underlying calculus, an approach taken within process-oriented discrete event languages [Bir79, Bir81]. There has been a

³**BI** is so called because natural deduction systems for it manipulate sequents of the form $\Gamma \vdash \phi$, in which Γ is a *bunch* of formulæ [OP99, Pym02, Pym04]; that is, a tree-structured context built from two combining operations, one corresponding to $*$ and one to \wedge .

demonstration that Milner’s calculus SCCS can support a compositional view of resource directly [Tof03]. It is clear, however, that this approach still contains all of the fundamental action structures of SCCS. Our approach is to consider the co-evolution of resources, as discussed above, and processes, in the sense of SCCS, with synchronization being constrained by the availability of resources and with resources being modified by the occurrence of actions.

Process calculi such as SCCS and CCS come along with a modal logic, usually called Hennessy-Milner logic, with a semantic judgement of the form

$$E \models \phi,$$

read as ‘process E has property ϕ ’. The language of propositions typically consists of classical conjunction, disjunction, and negation, together with modalities $\langle a \rangle$ and $[a]$ for describing the properties of evolutions $E \xrightarrow{a} E'$. Notice that Hennessy-Milner logic provides no characterization of the (asynchronous) parallel composition $|$. One proposal for dealing with this deficiency, at least in CCS-like settings, and which is discussed in [Sti01], is to introduce a ‘slicing’ operator, $/$, on formulæ with the property that $E | F \models \phi$ iff $E \models \phi/F$. The definition of ϕ/F involves ‘distributing the process through the formula’. Such a construction breaks the distinction between specifications, given by the logic, and implementations, given by the process terms. Moreover, the logical meaning of ϕ/F seems unclear. Dam [Dam90] also considered this question in the context of models of relevant logics.

In our setting, with an explicit model of resources and a corresponding logic, we are able to work with a judgement

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

read as ‘relative to the available resources R , process E has property ϕ ’. In our richer logic, we obtain a finer analysis of this judgement than is available in Hennessy-Milner logic. Specifically, we obtain, essentially, the following characterization of parallel composition, denoted by \times , as in SCCS, where \sim_μ is the evident notion of bisimulation,

$$R, E \models \phi_1 * \phi_2 \quad \text{iff} \quad \begin{array}{l} \text{there are } R_1 \text{ and } R_2 \text{ such that } R_1 \circ R_2 = R \\ \text{and there are } E_1 \text{ and } E_2 \text{ such that } E_1 \times E_2 \sim_\mu E, \\ \text{such that } R_1, E_1 \models \phi_1 \text{ and } R_2, E_2 \models \phi_2. \end{array}$$

That is, as a direct consequence of our formulation, we are able to characterize the concurrent structure of the system, together with its resource-constrained synchronization. Finally, by working with **BI**’s multiplicative quantifiers, we are also able to characterize a notion of local resource, with a corresponding logical construct (see §§6, 7, and 9).

A similar objective is encountered in the work of Cardelli and Caires [CC03] in which a ‘spatial logic’, in many ways similar to **MBI** but lacking a notion of resource, is used to model the asynchronous π -calculus. A detailed exploration of possible relationships between this work and ours — perhaps via particular choices of resource monoid — is beyond our present scope.

Another approach to resources, in a synchronous setting, is that of Brémont-Grégoire and Lee’s ACSR [BGL97]. Our approach is more foundational, starting from a logically well-founded model of resource and developing a foundational, yet applicable, theory of the interaction between processes and resources. A similar point of view may be found in the work of Gastin and Mislove [GM04] in which,

perhaps deriving from its denotational semantics motivations, the association of resources and processes uses a global construction in the style of Mazurkiewicz traces [Maz87]. Neither [BGL97] nor [GM04], however, provides a logical characterization of the interaction between resources and processes. In contrast, our analysis provides not only a logical characterization but one which captures the concurrent structure of the system.

3. A SYNCHRONOUS CALCULUS OF RESOURCE PROCESSES

Our starting point for our calculus of resource process is Milner's synchronous calculus of communicating systems, SCCS [Mil83]. Note that the asynchronous calculus CCS is a *sub*-calculus of SCCS.

Our main development is to view the statement $E \xrightarrow{a} E'$ as meaning that by using resource required for the action a to be enabled, the process E evolves to E' , with a corresponding modification of the available resource.

The natural point at which the presence or absence of a resource should impact upon the definition of a processes activities is within the action rule. A simple extension to the standard action rule would be to add an *enabling* function associated with each action, which one might call $\rho(a)$. Consequently rendering the action rule thus:

$$\frac{}{R, a : E \xrightarrow{a} R, E} \quad \rho(a) \sqsubseteq R$$

However, this approach is limiting: resources will be fixed throughout the computation. In such an account, we should have to record, within the process, state information (such as the amount in a bank within a vending machine) which one might consider as more naturally recorded within the resource element. To that end one should then posit the existence of a (partial) *modification* function, which one might call μ , in which $\mu(a, R) = R'$ has the interpretation that the effect of a on a resource R is to modify it to R' . With this view the atomic action rule should then be written:

$$\frac{}{R, a : E \xrightarrow{a} \mu(a, R), E} \quad \rho(a) \sqsubseteq R$$

a natural question with such a rule would be what should be the relationship between $\rho(a)$ and $\mu(a, R)$. Clearly there will be a problem if $\rho(a) \sqsubseteq R$ whilst $\mu(a, R)$ is undefined. To accomodate this problem we could require that

$$\rho(a) = \min\{R \mid \mu(a, R) \downarrow\}$$

but with this requirement it is clear that ρ is redundant, since its definition is forced. Further, if we use an enabling function view then its integration with the modal logic, as discussed in § 7, seems to be awkward. For descriptive convenience, we sometimes retain the ρ presentation of process/resource constraints, but in all cases this derives from the implied definition of μ . Given the above we fix on the following definition of action within our calculus:

$$\frac{}{R, a : E \xrightarrow{a} \mu(a, R), E} \quad \text{provided } \mu(a, R) \text{ is defined.}$$

Synchronization is achieved by requiring that a parallel composition of actions, $a\#b$, be possible only if the resource environment can be decomposed to support

a and b separately. Thus our operational rule for parallel composition essentially takes the form

$$\frac{R_1, E_1 \xrightarrow{a_1} \mu(a_1, R_1), E'_1 \quad R_2, E_2 \xrightarrow{a_2} \mu(a_2, R_2), E'_2}{R, E_1 \times E_2 \xrightarrow{a_1 \# a_2} \mu(a_1 \# a_2, R), E'_1 \times E'_2} \quad \text{provided } \mu(a_1 \# a_2, R_1 \circ R_2) \text{ is defined;}$$

that is, it must be possible to decompose R into the resources R_1 and R_2 , the resources required to support a_1 and a_2 simultaneously, though we admit the possibility of an equality between R_1 and R_2 , so allowing sharing as required. Note that synchronization is regulated by resources, in contrast to ACSR [BGL97], in which instantaneous events provide the basic synchronization mechanism. Note also that, in contrast to our local conditions, Gastin and Mislove's [GM04] mechanism requires a global construction for synchronization.

One fundamental consequence of this approach is that we should wish to maintain all of the interactions that lead to the current resource use transition within a process. In some sense, we need to know how the current resource utilization can be decomposed. Consequently, we must abandon the elegant use of the free abelian group of actions within SCCS to describe actions, restricting ourselves to the more basic free abelian monoid [Mil83],

$$\mathcal{A} = (\text{Act}, \#, \mathbf{1}).$$

If we were to take an abelian group, then an action a might result from the composition $a \# b^{-1}$ and b thus, in some sense, making use of more resource. Taking resource as the basic organizing principle, this form of hiding makes decomposition difficult to track. Nevertheless, our formulation permits the formulation of compound atomic actions (see the definition of μ , below) which are able to emulate the difficult wait-until construct of discrete event simulation languages such as ECSL [Cle65] and Demos [Bir79].

SCCS, in common with CCS, uses a notion of *restriction*. In our setting, a more natural concept is that of a *local* action, in which a collection of resources is available only to the process to which it is bound. Informally, the operational rule should take the form

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

where $'(\nu S)a'$ denotes the action a without the components of it that are associated with the bound resource S . These components are 'hidden' in the subsequent evolution.

As it is important to have an acronym and name for the calculus we shall call it the *synchronous calculus of resource processes*, or **SCRP** (pronounced 'scrap'). Following the notation of SCCS and the π -calculus, we present the syntax of the process element of the calculus as follows:

- $\mathbf{1}$, the unit process;
- $a : E$ a process that performs the action a to become the process E ;
- $E + F$ a process that evolves as E or as F , unit $\mathbf{0}$;
- $E \times F$ a process that synchronously uses resources as E and F ;
- $C \stackrel{\text{def}}{=} E$ is the definition of a constant C , allowing recursive processes to be defined;
- $(\nu R)E$ a process with a local, or hidden, evolution relative to resource R .

Unlike for standard process algebras, we must, as we have seen, define the environment of resources wherein the process evolves. This is given as a set of permitted actions drawn from the monoid. Thus our operational judgement is essentially of the form

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

and is intended to be read as ‘process E evolves via action a relative to the set of resources R ’. R is assumed to be a set of elements of resource drawn from a Kripke resource monoid. But this alone is not sufficient. As we have seen, we must set up an association between the actions of the monoid and the resources in the system.

For now, we take a partial function

$$\mu : Act \times \wp(\mathbf{R}) \rightarrow \wp(\mathbf{R})$$

which should be understood as describing the modification to a set R of resources caused by the execution of the action a , satisfying the following conditions:

- $\mu(\mathbf{1}, R) = R$ and, if $\mu(a, R) \downarrow$ and $\mu(b, S) \downarrow$, then $\mu(a\#b, R \circ S) = \mu(a, R) \circ \mu(b, S)$;
- for all a and R , the identity $\text{id}(a, R) = R$.

Modification functions are partial because the effects of actions, just like the combination of resources, need not be everywhere defined. We omit, in our definition of **SCR**P’s operational semantics, explicit mention of the partial function μ wherever possible. We write $\mu^{-1}S$ for the set of actions a such that $\mu(a, S) \downarrow$. These ideas will be illustrated by examples, after our calculus and logic have been formally defined, in §§ 5 and 8. In our present context, our examples will require very simple functions. In general, however, we might admit more complex models of resource evolution.

Now we can state precisely the fully general form of our operational judgement:

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

where $R' = \mu(a, R)$. With this form set up, the operational rules of **SCR**P, which should be read from conclusion to premisses, are given in Table 1. In order to give the rule for hiding, we need an auxiliary definition of *deletion*:

- Let $A = \{a_1, \dots, a_m\} \subseteq Act$ be a finite set of actions. Then define

$$\Pi A = \Pi_{i=1}^m a_i = a_1 \# \dots \# a_m.$$

If b is the product over a subset of A , then we refer to ‘ b in ΠA ’.

- Let α s and β s denote atomic actions. Let

$$\begin{aligned} a &= \alpha_1^{k_1} \# \dots \# \alpha_m^{k_m} \\ b &= \beta_1^{l_1} \# \dots \# \beta_n^{l_n}. \end{aligned}$$

Then define

$$a / b = \Pi_{i=1}^m \{\alpha_i^{k_i} \mid \text{for all } 1 \leq j \leq n, \alpha_i \neq \beta_j\}.$$

Notice that, in the *Hide* rule, in which we write $\mu^{-1}S$ for the set of actions a such that $\mu(a, S) \downarrow$, the product $\Pi\mu^{-1}S$ may include irrelevant actions but that these actions are discarded by the deletion operation. Notice also the *separation condition* in the *Prod* rule given in Table 1: we ensure that the composite resource is defined. The non-interference of the components of the composition can be enforced by requiring also that $R \circ S$ be defined only if R and S are disjoint (cf. separation logic [Rey02]).

$$\begin{array}{l}
\text{Act} \quad \frac{}{R, a : E \xrightarrow{a} \mu(a, R), E} \quad \mu(a, R) \downarrow \\
\text{Prod} \quad \frac{R, E \xrightarrow{a} \mu(a, R), E' \quad S, F \xrightarrow{b} \mu(b, S), F'}{R \circ S, E \times F \xrightarrow{a\#b} \mu(a\#b, R \circ S), E' \times F'} \\
\text{Sum}_i \quad \frac{R, E_i \xrightarrow{a} \mu(a, R), E'_i}{R, E_1 + E_2 \xrightarrow{a} \mu(a, R), E'_i} \quad i = 1, 2 \\
\text{Hide} \quad \frac{R \circ S, E \xrightarrow{a} R' \circ S', E'}{R, (\nu S)E \xrightarrow{a / \Pi\mu^{-1}(S)} R', (\nu S')E'} \quad \mu(a / \Pi\mu^{-1}(S), R) = R' \\
\text{Con} \quad \frac{R, E \xrightarrow{a} \mu(a, R), E'}{R, C \xrightarrow{a} \mu(a, R), E'} \quad C \stackrel{\text{def}}{=} E
\end{array}$$

TABLE 1. Operational Semantics of **SCR**P

Notice that we do not take a relabelling rule. Such a rule would follow the usual pattern but rather we simply regard relabelling as a structural operation at the level of the language.

4. METATHEORY

Bisimulation for **SCR**P, $R, E \sim_\mu R, F$, is defined in the usual way. Note that, for now, we consider the processes E and F relative to the same resource environment.

Definition 1. *Bisimulation, \sim_μ , is the largest binary relation on resource–process pairs, R, E such that if $R, E \sim_\mu R, F$, then*

(i) $R, E \xrightarrow{a} \mu(a, R), E'$ implies, for some F' ,

$$R, F \xrightarrow{a} \mu(a, R), F' \text{ and } \mu(a, R), E' \sim_\mu \mu(a, R), F',$$

and

(ii) $R, F \xrightarrow{a} \mu(a, R), F'$ implies, for some E' ,

$$R, E \xrightarrow{a} \mu(a, R), E' \text{ and } \mu(a, R), E' \sim_\mu \mu(a, R), F'.$$

□

We shall need the familiar property that bisimulation is a congruence, that is, in our setting, that if $R, E \sim_\mu R, F$, then, for all evident terms a, G , and S , $R, a : E \sim_\mu R, a : F$, $R, E + G \sim_\mu R, F + G$, $R, E \times G \sim_\mu R, F \times G$, and $R, (\nu S)E \sim_\mu R, (\nu S)F$.

Proposition 4.1. *Bisimulation is a congruence.*

Proof. Straightforward arguments by induction on the structure of resource–process pairs establish that \sim_μ is both an equivalence relation and substitutive. □

At this point in the development of a process calculus it is conventional to present the equational theory that matches the notion of equivalence that we have employed. Whilst the structure of the equational theory is important as a demonstration of the naturality of the equivalence, it is by no means necessary to perform proofs within the calculus. Throughout our presentation we have an intention that the calculus will be used to represent *implementation* and that an extended resource logic will be used to represent *requirements*. As a consequence there is little need to reason directly within the process calculus exploiting an equational theory.

In our system, the usual basic equations of SCCS, such as commutativity, associativity, and distribution of \times over $+$, do indeed hold. For example, it is easy to see, from the operational semantics, that

$$R, E \times (F \times G) \sim_{\mu} R, (E \times F) \times G,$$

since the monoidal composition \circ on resources is associative.

This being said, one of the most important aspects of a process system is the interaction between the sequential part of the language and the concurrent parts, usually expressed via an expansion theorem [Mil89]. In our instance, it is clear that there is no useful form of the expansion theorem as an equivalence. The main reason for this is that when we consider the constituent parts of a parallel composition we will have a particular allocation of resources to each of those parts. When we form the parallel composition we naturally form a (typically larger) compound resource, it is clear that this could have been divided in many ways other than that with which we chose to do the original proofs of the behaviours of the sub-components. Whilst this observation does not matter when we are reasoning operationally and decomposing the structure, it is clearly important when we are reasoning equationally and forming terms by composition. The appropriate form in this instance is an *inequational* theory with the obvious extension of the standard expansion theorem for SCCS, with the caveat that, as a consequence of the potential ability to change the division of resource, the relationship will be one of *simulation* and not bisimulation. Given our observations above that this is of no handicap when taking a ‘two-language’ approach to specification and verification, we omit this development within the current presentation.

5. SOME ILLUSTRATIVE EXAMPLES

We present some examples to illustrate the commonly required interactions in a concurrent setting:

- mutual exclusion;
- resource transfer;
- handshaking;
- private channels;
- asynchronous handover.

These examples have been chosen to illustrate how we can specify core concurrent systems concepts in this simple setting.

The first four examples can be stated clearly enough without giving a detailed description of the Kripke resource monoids involved. For the last example — asynchronous handover — we take the following Kripke resource monoid: assume basic sets of resource elements R , writing R^n for $n \geq 0$ distinct copies of R , that is,

$\underbrace{R \circ \dots \circ R}_{n \text{ times}}$, with $R^0 = \{e\}$. If we take the ordering on resources to be $R^m \sqsubseteq R^n$ iff $m \leq n$, then bifunctionality follows immediately and, up to some requirements about the definedness of the modification function and the interpretation of predicates, which we discuss in § 7, we get an essentially intuitionistic model. If we were to take the discrete ordering, which is not appropriate for all examples, we should obtain an essentially classical model, a situation similar to that which obtains in separation logic [Rey02].

Although we have explained that taking ‘enabling’ functions as distinct from the modification functions is logically problematic, we nevertheless find it helpful to use the idea as a notational abbreviation. Specifically, following our discussion in § 3, we write

$$\rho(a) = \min \{R \mid \mu(a, R) \downarrow\}$$

where such a minimum exists. This abbreviation is particularly convenient in the context of the Kripke resource monoid, described above, used in the asynchronous handover example.

5.1. Mutual Exclusion. In this example, we have two (or more) processes and some points in the computation at which at most one of them may be active, often termed a critical region or section. To that end we define a process in the following manner:

$$\begin{aligned} E &\stackrel{\text{def}}{=} nc : E + \text{critical} : E_{\text{critical}} \\ E_{\text{critical}} &\stackrel{\text{def}}{=} \text{critical} : E_{\text{critical}} + \text{critical} : E \end{aligned}$$

To give the rest of the system we define an enabling map by $\rho(nc) = \{e\}$ (nc is ‘not critical’), and $\rho(\text{critical}) = \{R\}$. Now, the resource process

$$R, E \times E,$$

where $\mu(a, R) = R$, for all a , defines a system exhibiting mutual exclusion. The important point is that, in the application of the *Prod* rule, we have $R = R \circ \{e\}$. In other words, the resource is available as itself at the same time as an empty resource. Consequently, the evolutions of our system are as follows:

$$\begin{aligned} R, E \times E &\xrightarrow{nc\#\#nc} R, E \times E \\ R, E \times E &\xrightarrow{nc\#\#critical} R, E \times E_{\text{critical}} \\ R, E \times E_{\text{critical}} &\xrightarrow{nc\#\#critical} R, E \times E_{\text{critical}} \\ R, E \times E_{\text{critical}} &\xrightarrow{nc\#\#critical} R, E \times E. \end{aligned}$$

Notice that at no point is the action $\text{critical}\#\#critical$ performed nor do we see both processes in the state $E_{\text{critical}} \times E_{\text{critical}}$.

Note that in this example the resource we have used plays the rôle of a semaphore; this demonstrates the calculus’s ability to directly exploit resource. A more standard process algebra solution to this problem is to use communication via handshaking; we shall demonstrate how that is achieved within this calculus in our next example.

5.2. Resource Transfer. As an extension of the semaphores example, above, we may wish to establish a system in which only one of the parallel tasks is ‘active’ at any one time, but in which the tasks take turns. One way of achieving this is described below.

We take as resources sets R_1 and R_2 . Then we define the following modification functions:

$$\begin{aligned}\mu(\text{put}_1, R_1) &= R_2 & \mu(\text{get}_1, R_1) &= R_1 \\ \mu(\text{put}_2, R_2) &= R_1 & \mu(\text{get}_2, R_2) &= R_2.\end{aligned}$$

Here we take $\rho(\text{get}_1) = R_1$ and $\rho(\text{get}_2) = R_2$, and $\rho(\text{put}_1) = R_1$ and $\rho(\text{put}_2) = R_2$.

Ignoring the resources for the moment, we take the following process definitions:

$$\begin{aligned}E1 &\stackrel{\text{def}}{=} \text{get}_1 : E1_{\text{critical}} + \mathbf{1} : E1 \\ E1_{\text{critical}} &\stackrel{\text{def}}{=} \mathbf{1} : E1_{\text{critical}} + \text{put}_1 : E1 \\ E2 &\stackrel{\text{def}}{=} \text{get}_2 : E2_{\text{critical}} + \mathbf{1} : E2 \\ E2_{\text{critical}} &\stackrel{\text{def}}{=} \mathbf{1} : E2_{\text{critical}} + \text{put}_2 : E2,\end{aligned}$$

where, of course, we take $\rho(\mathbf{1}) = \{e\}$.

Then the system $R_1, E1 \times E2$ represents one in which the processes $E1$ and $E2$ exchange ownership of a resource in order that they may enter their respective critical sections:

$$R_1, E1 \times E2 \xrightarrow{\text{get}_1 \# \mathbf{1}} R_2, E1_{\text{critical}} \times E2.$$

There are two clear generalizations of this system: we can extend the number of resources whilst keeping the same pass-on property and so obtain a ‘round-robin’ scheduler [Mil89]; alternatively, we can extend the processes as follows:

$$E1 \stackrel{\text{def}}{=} \text{get}_1 : E1_{\text{critical}} + \text{swap}_1 : E1 + \mathbf{1} : E1,$$

with $\mu(\text{swap}_1, R_1) = R_2$ and $\rho(\text{swap}_1) = R_1$, and the obvious symmetric definitions. We then obtain a mutual exclusion system with a designated, or transferable, token.

It should be clear from these examples that the modification functions, of the form $\mu : \text{Act} \times \wp(\mathbf{R}) \rightarrow \wp(\mathbf{R})$, allow very flexible models of resource transfer between concurrent processes.

5.3. Handshaking. In this example, we desire that two processes should proceed only if they mutually agree on progress. In other words, there is a point in the computation that is preceded by a point of mutual agreement or a ‘join’. The following process definitions will illustrate the point:

$$\begin{aligned}E1 &\stackrel{\text{def}}{=} \text{wait}_{E1} : E1 + \text{go}_{E1} : E1' \\ E2 &\stackrel{\text{def}}{=} \text{wait}_{E2} : E2 + \text{go}_{E2} : E2'\end{aligned}$$

we take $\rho(\text{go}_{E1}) = R_1$ and $\rho(\text{go}_{E2}) = R_2$ but *importantly* $R = R_1 \circ R_2$, remembering that this is *not* the same as R_1, R_2 (recall that we use this list notation for the union of sets of resources) and can only be ‘split up’ (composition is *not*, in general, union) by a use of a *Prod* rule. We assume $\rho(\text{wait}_1) = \rho(\text{wait}_2) = \{e\}$.

The evolutions of $R, E1 \times E2$ are as follows:

$$R, E1 \times E2 \xrightarrow{\text{wait}_{E1} \# \text{wait}_{E2}} R, E1 \times E2$$

$$R, E_1 \times E_2 \xrightarrow{g \circ E_1 \# g \circ E_2} R, E'_1 \times E'_2.$$

Notice that E_1 and E_2 either wait or proceed together. Obviously in a larger system the states E_1 and E_2 need not be arrived at at the same time.

5.4. Privacy. In the foregoing example, we may wish to ensure that *only* E_1 and E_2 can interact by using the composed resource. So we would form the process

$$(\nu R_1 \circ R_2)(E_1 \times E_2),$$

with E_1 , E_2 , and ρ as above. Note that the requirement that that each $g \circ E_i$ be enabled by the available resource leads, essentially, to the association of each R_i and E_i .

5.5. Asynchronous Handover. The classic form of this example is the producer–consumer problem: that is, there is one process that can generate work and leave it for another process to handle later. Consider the following process definitions, using the ‘powers of R ’ monoid described at the beginning of this section:

$$\begin{aligned} \text{Prod} &\stackrel{\text{def}}{=} \text{nowork} : \text{Prod} + \text{work} : \text{Prod} \\ \text{Cons} &\stackrel{\text{def}}{=} \text{wait} : \text{Cons} + \text{cons} : \text{Cons}, \end{aligned}$$

with $\rho(\text{nowork}) = \{e\}$, $\rho(\text{wait}) = \{e\}$, $\rho(\text{work}) = \{e\}$, $\rho(\text{cons}) = R$ and, writing R^n for $n > 0$ distinct copies of R , i.e., $\underbrace{R \circ \dots \circ R}_{n \text{ times}}$,

$$\begin{aligned} \mu(\text{nowork}, \{e\}) &= \{e\} & \mu(\text{nowork}, R^n) &= R^n \\ \mu(\text{wait}, \{e\}) &= \{e\} & \mu(\text{wait}, R^n) &= R^n \\ \mu(\text{work}, \{e\}) &= \{R\} & \mu(\text{work}, R^n) &= R^{n+1} \\ & & \mu(\text{cons}, R^n) &= R^{n-1}. \end{aligned}$$

It follows that the system

$$\{e\}, \text{Prod} \times \text{Cons}$$

behaves as a producer–consumer system with a counter R . Given a generic state of the system, we have the following evolutions

$$\begin{aligned} \{e\}, \text{Prod} \times \text{Cons} &\xrightarrow{\text{nowork} \# \text{wait}} \{e\}, \text{Prod} \times \text{Cons} \\ \{e\}, \text{Prod} \times \text{Cons} &\xrightarrow{\text{work} \# \text{wait}} R, \text{Prod} \times \text{Cons} \\ R^n, \text{Prod} \times \text{Cons} &\xrightarrow{\text{nowork} \# \text{wait}} R^n, \text{Prod} \times \text{Cons} \\ R^n, \text{Prod} \times \text{Cons} &\xrightarrow{\text{nowork} \# \text{cons}} R^{n-1}, \text{Prod} \times \text{Cons} \\ R^n, \text{Prod} \times \text{Cons} &\xrightarrow{\text{work} \# \text{cons}} R^n, \text{Prod} \times \text{Cons} \\ R^n, \text{Prod} \times \text{Cons} &\xrightarrow{\text{work} \# \text{wait}} R^{n+1}, \text{Prod} \times \text{Cons}. \end{aligned}$$

Since our calculus is based on a synchronous view of computation, we have, as an immediate consequence, the ability to form compound yet atomic actions. Exploiting this construction and using non-determinism gives an account of the wait-until and cond type synchronizations of Demos2k [BT00a, BT00b] like languages, since the details are essentially identical to those of the corresponding construction within SCCS [Mil83], they are omitted in the interest of brevity.

6. BASIC BUNCHED LOGIC

We have argued that a basic and useful model of resource arises from a Kripke resource monoid, that is, a set equipped with a monoidal combination and a pre-order,

$$\mathcal{R} = (\mathbf{R}, \circ, e, \sqsubseteq),$$

subject to the bifunctionality condition. Such a structure provides the basis for the possible-worlds models of the logic of bunched implications, **BI** [OP99, Pym02, Pym04]. The axiomatization of such a structure provided by **BI** is given by a forcing relation \models between resources and propositions such that, for a given denotation of the propositional letters, $\llbracket p \rrbracket$,

$$\begin{aligned} R \models p & \text{ iff } R \in \llbracket p \rrbracket \\ R \models \top & \text{ always} \\ R \models \phi_1 \wedge \phi_2 & \text{ iff } R \models \phi_1 \text{ and } R \models \phi_2 \\ R \models \perp & \text{ never} \\ R \models \phi_1 \vee \phi_2 & \text{ iff } R \models \phi_1 \text{ or } R \models \phi_2 \\ R \models \phi \rightarrow \psi & \text{ iff for all } S \text{ such that } R \sqsubseteq S, \\ & S \models \phi \text{ implies } S \models \psi \\ R \models \phi_1 * \phi_2 & \text{ iff there are } S_1 \text{ and } S_2 \text{ such that } S_1 \circ S_2 \sqsubseteq R \text{ and} \\ & S_1 \models \phi_1 \text{ and } S_2 \models \phi_2 \\ R \models I & \text{ iff } R \sqsubseteq e \\ R \models \phi * \psi & \text{ iff for all } S \text{ such that } S \models \phi, \\ & R \circ S \models \psi \end{aligned}$$

Intuitionistic negation is then given by $\neg\phi \stackrel{\text{def}}{=} \phi \rightarrow \perp$. Boolean **BI**, in contrast, takes classical additive negation,

$$R \models \neg\phi \text{ iff } R \not\models \phi,$$

with classical implication defined by

$$\begin{aligned} R \models \phi \supset \psi & \text{ iff } R \models \phi \text{ implies } R \models \psi \\ & \text{ iff } R \models (\neg\phi) \vee \psi. \end{aligned}$$

In a slightly more complex formulation of **BI** and its models, we can also interpret both additive and multiplicative quantifiers. Here, we simply consider

$$\begin{aligned} R \models \exists x.\phi & \text{ iff for some term } t \text{ defined at } R, \\ & R \models \phi[t/x] \\ R \models \exists_\nu x.\phi & \text{ iff for some term } t \text{ defined at } S, \\ & R \circ S \models \phi[t/x]. \end{aligned}$$

The additive and multiplicative universal quantifiers are defined analogously.

The metatheory of **BI**, including a range of proof systems and a range of soundness and completeness theorems, is presented in [OP99, Pym02, Pym04, GMP02, POY04].

7. A LOGIC OF RESOURCES AND PROCESSES

Having seen that **BI** provides a logic of resources, with judgement $R \models \phi$, and that Hennessy-Milner provides a modal logic of processes, with judgement $E \models \phi$,

we are now in a position to introduce our modal logic of resources and processes, **MBI**, with judgement

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

where

- R is a set of resources, with composition and ordering lifted from the underlying Kripke resource monoid,

$$\mathcal{R} = (\mathbf{R}, \circ, e, \sqsubseteq),$$

as previously discussed,

- $\mu : Act \times \wp(\mathbf{R}) \rightarrow \wp(\mathbf{R})$ is a modification function.

The language of **MBI** is summarized below. The intended meanings of the less familiar connectives are discussed in the subsequent text; the formal semantics of all of the connectives is given in Table 2. We take Act as the domain of predication and quantification.⁴ Otherwise, our formulation is based on quite standard methods and so is presently concisely.

- *Atoms*: $p(a_1, \dots, a_m)$, predication is over actions $a_i \in Act$.
- *Basic Additives*: The classical propositional connectives, $\phi \wedge \psi$, \top , $\phi \vee \psi$, \perp , and $\neg\phi$.
- *Additive Modalities*: The usual Hennessy-Milner modalities, $[a]\phi$ and $\langle a \rangle\phi$, where $a \in Act$.
- *Additive Quantifiers*: The usual classical quantifiers, $\forall x.\phi$ and $\exists x.\phi$, where the domain of quantification is Act .
- *Basic Multiplicatives*: The usual propositional multiplicatives from the bunched logic **BI**, $\phi * \psi$, I , and $\phi \multimap \psi$.
- *Multiplicative Modalities*: Multiplicative forms of the usual Hennessy-Milner modalities, $[a]_\nu\phi$ and $\langle a \rangle_\nu\phi$, where $a \in Act$.
- *Multiplicative Quantifiers*: A simple form of the multiplicative quantifiers found in **BI**, $\forall_\nu x.\phi$ and $\exists_\nu x.\phi$, where the domain of quantification is Act , in which predication is additive [Pym02, Pym04].⁵

Let p be an m -ary predicate symbol. Then the interpretation of p in a Kripke resource monoid, $\mathcal{R} = (\mathbf{R}, \circ, e, \sqsubseteq)$,

$$\llbracket p \rrbracket : (\wp(\mathbf{R}))^m \rightarrow \mathbf{2},$$

is an m -ary relation on $\wp(\mathbf{R})$. Notice how, in the clause form atoms in Figure 2, the meaning of an action a corresponds, locally, to the resource for which its modification, $\mu(a, R)$, is defined. Other choices may be possible here. Note, however, that taking $\llbracket a \rrbracket = \rho(a)$ seems to be awkward. Such a choice would suggest taking a side-condition of the form $\rho(a) \sqsubseteq R$ in the Act rule of **SCRIP**'s operational semantics, with a corresponding condition in the definiens of the clause for atoms in **MBI**'s consequence relation. It seems that these choices would lead to an hereditary property which, as we discuss below, we prefer to avoid in general.

⁴It is clear that other, possibly richer, possibly application-specific, choices are possible.

⁵For binary connectives, such as $*$, one might require that each component of the formula be formed with respect to a different set of variables, with the composite formula requiring the multiplicative combination of the two sets of variables. Here we use the much simpler additive predication, with both components, and so the composite formula, being formed over the same set of variables.

R		E	$\phi \multimap \psi$
S		F	ϕ
R	\circ	S	
		$E \times F$	ψ

FIGURE 3. Multiplicative Resources

R	S		E	$\phi \rightarrow \psi$
S		E	ϕ	
R	S		E	ψ

FIGURE 4. Additive Resources ($\phi \rightarrow \psi \stackrel{\text{def}}{=} (-\phi) \vee \psi$)

Definition 2 (MBI model). An MBI model⁶ is a quadruple

$$\mathcal{M} = \langle \mathcal{R}, \mu, \llbracket - \rrbracket, \models_{\mathcal{M}}^{\mu} \rangle,$$

where $\mathcal{R} = (\mathbf{R}, \circ, e, \sqsubseteq)$ is a Kripke resource monoid, μ is a modification function, $\llbracket - \rrbracket$ is an interpretation of the predicate symbols in $\wp(\mathcal{R})$, and $\models_{\mathcal{M}}^{\mu}$ is a satisfaction relation such the conditions given in Table 2 hold.

□

Where no confusion can arise, we write just \models rather than $\models_{\mathcal{M}}^{\mu}$.

The clauses for the multiplicative conjunction, $*$, and implication, \multimap , establish the basic characterization of concurrent composition. The correspondence between the composition of resources, \circ , and concurrent composition, \times , is illustrated in Figure 3.

Here, we can imagine that the process E , characterized by a multiplicative implication, imports a module F , together with the *separate* resources required by F . The resulting concurrent process requires the composite resource.

In contrast, the (classical) additive implication does not characterize the formation of a concurrent composition. Rather, it expresses a disjunctive property of (possibly) *shared* resources for a fixed process. This is illustrated by Figure 4, in which S may be all of R .

It would be tempting to take the following Kripke monotonicity, or hereditary, condition:

$$\text{for all } S \text{ s.t. } R \sqsubseteq S, R, E \models \phi \text{ implies } S, E \models \phi.$$

⁶It should be noted that, whilst adequate for our present purposes, the rather simple definition of MBI model presented here seems, with the classical additives, to be too naïve for more delicate logical results, such as the completeness of a tableaux system (*cf.* [GMP02]), to be available. For such purposes, more sophisticated classes models, such as those based on ternary relations [Dun86, RM72, GMP02] seem to be necessary.

ATOMS

$$R, E \models_{\mathcal{M}}^{\mu} p(a_1, \dots, a_m) \quad \text{iff} \quad \text{for all } 1 \leq i \leq m, \mu(a_i, R) \downarrow, \text{ and } \llbracket p \rrbracket \underbrace{(R, \dots, R)}_{m \text{ times}}$$

BASIC ADDITIVES

$$\begin{aligned} R, E \models_{\mathcal{M}}^{\mu} \top & \quad \text{always} \\ R, E \models_{\mathcal{M}}^{\mu} \phi \wedge \psi & \quad \text{iff} \quad R, E \models_{\mathcal{M}}^{\mu} \phi \text{ and } R, E \models_{\mathcal{M}}^{\mu} \psi \\ R, E \models_{\mathcal{M}}^{\mu} \perp & \quad \text{never} \\ R, E \models_{\mathcal{M}}^{\mu} \phi \vee \psi & \quad \text{iff} \quad R, E \models_{\mathcal{M}}^{\mu} \phi \text{ or } R, E \models_{\mathcal{M}}^{\mu} \psi \\ R, E \models_{\mathcal{M}}^{\mu} \neg \phi & \quad \text{iff} \quad R, E \not\models_{\mathcal{M}}^{\mu} \phi \end{aligned}$$

ADDITIVE MODALITIES

$$\begin{aligned} R, E \models_{\mathcal{M}}^{\mu} [a]\phi & \quad \text{iff} \quad \text{for all } R, E \xrightarrow{a} \mu(a, R), E' \text{ s.t. } \mu(a, R) \downarrow, \\ & \quad \mu(a, R), E' \models_{\mathcal{M}}^{\mu} \phi \\ R, E \models_{\mathcal{M}}^{\mu} \langle a \rangle \phi & \quad \text{iff} \quad \text{for some } R, E \xrightarrow{a} \mu(a, R), E' \text{ s.t. } \mu(a, R) \downarrow, \\ & \quad \mu(a, R), E' \models_{\mathcal{M}}^{\mu} \phi \end{aligned}$$

ADDITIVE QUANTIFIERS

$$\begin{aligned} R, E \models_{\mathcal{M}}^{\mu} \exists x. \phi & \quad \text{iff} \quad \text{for some } a \text{ s.t. } \mu(a, R) \downarrow, R, E \models_{\mathcal{M}}^{\mu} \phi[a/x] \\ R, E \models_{\mathcal{M}}^{\mu} \forall x. \phi & \quad \text{iff} \quad \text{for all } a \text{ s.t. } \mu(a, R) \downarrow, R, E \models_{\mathcal{M}}^{\mu} \phi[a/x] \end{aligned}$$

BASIC MULTIPLICATIVES

$$\begin{aligned} R, E \models_{\mathcal{M}}^{\mu} I & \quad \text{iff} \quad R \sqsubseteq e, E \sim_{\mu} \mathbf{1} \\ R, E \models_{\mathcal{M}}^{\mu} \phi * \psi & \quad \text{iff} \quad \text{there exist } S, T \text{ s.t. } S \circ T \sqsubseteq R, \text{ and} \\ & \quad F, G \text{ s.t. } R, F \times G \sim_{\mu} R, E, \text{ and} \\ & \quad S, F \models_{\mathcal{M}}^{\mu} \phi \text{ and } T, G \models_{\mathcal{M}}^{\mu} \psi \\ R, E \models_{\mathcal{M}}^{\mu} \phi \multimap \psi & \quad \text{iff} \quad \text{for all } S, F \text{ s.t. } R \circ S \downarrow, S, F \models_{\mathcal{M}}^{\mu} \phi, \\ & \quad R \circ S, E \times F \models_{\mathcal{M}}^{\mu} \psi \end{aligned}$$

MULTIPLICATIVE MODALITIES

$$\begin{aligned} R, E \models_{\mathcal{M}}^{\mu} [a]_{\nu} \phi & \quad \text{iff} \quad \text{for all } R \circ S, E \xrightarrow{a} \mu(a, R \circ S), E' \text{ s.t.} \\ & \quad R \circ S \downarrow, \mu(a, R \circ S) \downarrow, \text{ and } \mu(a, R \circ S), E' \models_{\mathcal{M}}^{\mu} \phi \\ R, E \models_{\mathcal{M}}^{\mu} \langle a \rangle_{\nu} \phi & \quad \text{iff} \quad \text{for some } R \circ S, E \xrightarrow{a} \mu(a, R \circ S), E' \text{ s.t.} \\ & \quad R \circ S \downarrow, \mu(a, R \circ S) \downarrow, \text{ and } \mu(a, R \circ S), E' \models_{\mathcal{M}}^{\mu} \phi \end{aligned}$$

MULTIPLICATIVE QUANTIFIERS

$$\begin{aligned} R, E \models_{\mathcal{M}}^{\mu} \exists_{\nu} x. \phi & \quad \text{iff} \quad \text{for some } S, F \text{ s.t. } R, E \sim_{\mu} R, (\nu S)F, \\ & \quad R \circ S \downarrow, R \circ S, F \models_{\mathcal{M}}^{\mu} \phi[b/x], \\ & \quad \text{for some } b \text{ in } \Pi \mu^{-1} S \\ R, E \models_{\mathcal{M}}^{\mu} \forall_{\nu} x. \phi & \quad \text{iff} \quad \text{for all } S, F \text{ s.t. } R, E \sim_{\mu} R, (\nu S)F, \\ & \quad R \circ S \downarrow, R \circ S, F \models_{\mathcal{M}}^{\mu} \phi[b/x], \\ & \quad \text{for all } b \text{ in } \Pi \mu^{-1} S \end{aligned}$$

TABLE 2. Satisfaction for an **MBI** model, \mathcal{M}

That is, established properties of resource–process pairs would remain true if the available resource were increased. We do *not* take this condition in general. To see why, consider that we might assert that a process has insufficient resource available to evolve even though adding more resource would allow evolution. Such a condition might, however, be derived from the properties of a given monoid. If a monoid, such as the one taken in our illustrative examples, has the property that, for all $R, S, R \sqsubseteq R \circ S$, if the definedness of the modification function is preserved as resource increases, and if the truth of predicate symbols is preserved as resource increases, then, via the clause of Table 2 for atoms, Kripke monotonicity will hold.

In addition to the basic multiplicative connectives already discussed, **MBI** makes use of multiplicative quantifiers and multiplicative modalities. The basic idea of the multiplicative quantifiers has already been introduced, for example,

$$R \models \exists_\nu x. \phi \quad \text{iff} \quad \begin{array}{l} \text{for some term } t \text{ defined at } S, \\ R \circ S \models \phi[t/x], \end{array}$$

the point being that new resource is required to form the substituting term. In **MBI**, the additional resource corresponds to that which is hidden by the *Hide* rule, so that the semantic clause for \exists_ν characterizes hiding up to bisimulation.

The multiplicative modalities are similar. For example, we have that $R, E \models \langle a \rangle_\nu \phi$ just in case we have that for some action $R \circ S, E \xrightarrow{a} \mu(R \circ S), E'$ — that is, some a that is enabled by additional, separated, resource, $S \multimap R' \circ S', E' \models \phi$.

Our principal metatheorem concerning **MBI**, that logical truth in **MBI** models corresponds to bisimulation, is established in § 9. Before proceeding with our theoretical development, however, we revisit our **SCRIP** examples in the from the logical perspective provided by **MBI**.

8. THE ILLUSTRATIVE EXAMPLES REVISITED

We revisit the examples introduced in § 5 in order to illustrate the interaction between resource processes and the polymodal logic **MBI**.

The additive connectives correspond to those available in logics of the usual Hennessy-Milner type and are able to express the usual things [Mil89]. Accordingly, we concentrate here on examples of the use of the multiplicatives.

8.1. Mutual Exclusion. Recall that we define a process in the following manner:

$$\begin{array}{l} E \stackrel{\text{def}}{=} nc : E + \text{critical} : E_{\text{critical}} \\ E_{\text{critical}} \stackrel{\text{def}}{=} \text{critical} : E_{\text{critical}} + \text{critical} : E, \end{array}$$

with $\rho(nc) = \{e\}$ (recall nc is ‘not critical’), and $\rho(\text{critical}) = \{R\}$. Then the resource process

$$R, E \times E$$

defines a system exhibiting mutual exclusion.

Recall that, in this example, at no point is the action $\text{critical}\#\text{critical}$ performed and at no point do we see the state $E_{\text{critical}} \times E_{\text{critical}}$. Simple (true) assertions about the system include

$$R, E \times E \models [\text{critical}\#\text{critical}]_\perp.$$

8.2. Resource Transfer. As we have seen, the modification functions allow complex notions of resource transfer to be expressed quite naturally. Returning to our simple example of scheduling, again following on from the example above, with process definitions as given in § 5, we have

$$R, E1 \times E2 \models [get_1 \# get_2] \perp$$

and

$$R, E1_{critical} \times E2_{critical} \models [put_1 \# put_2] \perp.$$

Defining a rather unsubtle resource-ownership predicate in a system R, E by $owns_{R,E}(a)$ iff an evolution $R, E \xrightarrow{a} \mu(a, R), E'$ occurs, then we get

$$R_1, E_1 \times E_2 \models [get_1 \# \mathbf{1}] \langle \mathbf{1} \# get_2 \rangle owns_{R_2, E_2}(get_2).$$

8.3. Handshaking. Recall that the processes

$$\begin{aligned} E_1 &\stackrel{\text{def}}{=} wait_{E_1} : E_1 + go_{E_1} : E'_1 \\ E_2 &\stackrel{\text{def}}{=} wait_{E_2} : E_2 + go_{E_2} : E'_2 \end{aligned}$$

determine a system that can proceed only if they mutually agree on progress: that is, $E_1 \times E_2$ can evolve to $E'_1 \times E'_2$ only if $go_{E_1} \# go_{E_2}$ is enabled, that is, R can be decomposed into R_1 and R_2 . For $i = 1, 2$, let ϕ'_i be some assertion such that

$$R_i, E'_i \models \phi'_i.$$

Then we have that

$$R, E_1 \times E_2 \models \langle go_{E_1} \# go_{E_2} \rangle (\phi'_1 * \phi'_2)$$

provided $R_1 \circ R_2 = R$. This assertion, which demonstrates how a property of a concurrent system may be expressed as a conjunction of properties of its concurrent components, forms part of our next example. Note that if ϕ'_2 , say, is of the form $\phi'_1 \multimap \psi$, then we obtain

$$\langle go_{E_1} \# go_{E_2} \rangle \psi$$

as an ‘emergent property’ of the concurrent system.

8.4. Privacy. Recall again that

$$\begin{aligned} E_1 &\stackrel{\text{def}}{=} wait_{E_1} : E_1 + go_{E_1} : E'_1 \\ E_2 &\stackrel{\text{def}}{=} wait_{E_2} : E_2 + go_{E_2} : E'_2. \end{aligned}$$

For $i = 1, 2$, again let ϕ'_i be such that

$$R_i, E'_i \models \phi'_i.$$

Then we have that

$$\{e\}, (\nu R_1 \circ R_2)(E_1 \times E_2) \models \exists \nu x. \langle x \rangle (\phi'_1 * \phi'_2),$$

since unpacking \models , using the $\exists \nu$, clause gives

$$R_1 \circ R_2, E_1 \times E_2 \models \langle go_{E_1} \# go_{E_2} \rangle (\phi'_1 * \phi'_2),$$

then, using the $\langle - \rangle$ clause,

$$R_1 \circ R_2, E'_1 \times E'_2 \models \phi'_1 * \phi'_2,$$

and finally, using the $*$ clause,

$$R_1, E'_1 \models \phi'_1 \quad \text{and} \quad R_2, E'_2 \models \phi'_2.$$

This assertion expresses the property that there exists an action, namely $go_{E_1} \# go_{E_2}$, which is separated from the ambient resources, which allows $E_1 \times E_2$ to evolve locally, using R_1 and R_2 privately, and which leads to a state having the given properties, ϕ'_1 and ϕ'_2 .

Again, this assertion provides an example of the use of the multiplicative conjunction, $*$, in order to express a property of the concurrent system as a conjunction of properties of its component systems. It also provides an example of the use of the multiplicative existential quantifier, in order to describe a local binding of resources to a component of the system, as well as the more familiar diamond modality, $\langle - \rangle$. Notice that the separation condition, between the ambient resource, here $\{e\}$, and the local resource, $R_1 \circ R_2$, is satisfied trivially.

8.5. Asynchronous Handover. Recall the producer–consumer system,

$$\begin{aligned} Prod &\stackrel{\text{def}}{=} nowork : Prod + work : Prod \\ Cons &\stackrel{\text{def}}{=} wait : Cons + cons : Cons, \end{aligned}$$

with $\rho(nowork) = \{e\}$, $\rho(wait) = \{e\}$, $\rho(work) = \{e\}$, $\rho(cons) = R$ and, writing R^n for $n > 0$ distinct copies of R , i.e., $\underbrace{R \circ \dots \circ R}_{n \text{ times}}$,

$$\begin{aligned} \mu(nowork, R^n) &= R^n \\ \mu(wait, R^n) &= R^n \\ \mu(work, R^n) &= R^{n+1} \\ \mu(cons, R^n) &= R^{n-1}. \end{aligned}$$

Let ϕ_{Prod} and ϕ_{Cons} be properties of $Prod$ and $Cons$, respectively, relative to resource R . Then the system $\{e\}, Prod \times Cons$ has the property

$$\{e\}, Prod \times Cons \models \langle nowork \# cons \rangle_\nu (\phi_{Prod} * \phi_{Cons})$$

since, unpacking \models using $\langle - \rangle_\nu$, noting that

$$\{e\} \circ R, Prod \times Cons \xrightarrow{nowork \# cons} \mu(nowork \# cons, \{e\} \circ R), Prod \times Cons$$

gives

$$R, Prod \times Cons \models \phi_{Prod} * \phi_{Cons},$$

which follows using the case of \models for $*$.

This property says that the system $\{e\}, Prod \times Cons$ may perform the action $nowork \# cons$ provided the required resource be added.

Since our primary modalities, $\langle a \rangle$ and $[a]$, are defined over actions, we can exploit the action structure to define compound requirements which match the compound interaction forms of wait-until [BT00a]. For example, $\langle a \# b \rangle \phi$ requires that actions a and b are performed simultaneously, this could either be as a result of a single process insisting on the availability of both resources (a wait-until) or the result of a parallel system performing both actions as a synchronous parallel.

9. LOGICAL METATHEORY

The logical characterization of bisimulation provided by Hennessy-Milner logic for a process calculus such as CCS [Mil89] takes the form

$$E \sim F \quad \text{iff} \quad \text{for all } \phi, E \models \phi \text{ iff } F \models \phi.$$

Here we show that such a theorem is available for the finer analysis of process equivalence and logical equivalence provided by **SCRP** and **MBI**. More specifically, our result, expressed as Theorems 9.1 and 9.2, shows that **MBI** provides an explicit characterization of the concurrent and local structure of a system, via the definitions of \models for the connective $*$ and the multiplicative quantifiers, \forall_ν and \exists_ν , respectively.

Definition 3. *Let Γ be a set of **MBI** formulæ. Then the equivalence \equiv_Γ between **SCRP** processes is defined by*

$$R, E \equiv_\Gamma R, F \quad \text{iff} \quad \text{for all } \mathcal{M}, \quad \begin{aligned} & \{\phi \in \Gamma \mid R, E \models_{\mathcal{M}}^\mu \phi\} \\ &= \\ & \{\psi \in \Gamma \mid R, F \models_{\mathcal{M}}^\mu \psi\}. \end{aligned}$$

□

We have the usual derived definition:

$$R, E \equiv_{\mathbf{MBI}} R, F \quad \text{iff} \quad \text{for all } \Gamma, \text{ we have } R, E \equiv_\Gamma R, F.$$

Theorem 9.1. *If, for all R and μ , $R, E \sim_\mu R, F$, then, for all R , it follows that $R, E \equiv_{\mathbf{MBI}} R, F$.*

Proof. By induction on the structure of formulæ, ϕ .

p : Let ϕ be $p(a_1, \dots, a_m)$. By the definition of \models , we have that $R, E \models p(a_1, \dots, a_m)$ iff, for each $1 \leq i \leq m$, $\mu(a_i, R) \downarrow$ and $\llbracket p \rrbracket(\underbrace{R, \dots, R}_{m \text{ times}})$. But

these conditions are independent of E , so we are done.

\neg : Let ϕ be $\neg\psi$. By the induction hypothesis, we may assume that the result holds for ψ . By the definition of \models , we have that $R, E \models \neg\psi$ iff $R, E \not\models \psi$. Therefore, by the induction hypothesis, we have that $R, F \not\models \psi$, and so, by the definition of \models , $R, F \models \neg\psi$.

\top : Similar to the case for atoms, p .

\wedge : Let ϕ be $\psi_1 \wedge \psi_2$. By the induction hypothesis, we may assume that the result holds for ψ_1 and ψ_2 . By the definition of \models , $R, E \models \psi_1 \wedge \psi_2$ iff $R, E \models \psi_1$ and $R, E \models \psi_2$. Therefore, by the induction hypothesis, $R, F \models \psi_1$ and $R, F \models \psi_2$. Therefore, by the definition of \models , $R, F \models \psi_1 \wedge \psi_2$.

\perp : Similar to the case for atoms, p .

\forall : Similar to the case for \wedge .

$[a]$: Let ϕ be $[a]\psi$. It follows that, for any E' such that $R, E \xrightarrow{a} \mu(a, R), E'$, $\mu(a, R), E' \models \psi$. By the definition of bisimulation, we have that, for some E' such that $R, E \xrightarrow{a} \mu(a, R), E'$, there is an $R, F \xrightarrow{a} \mu(a, R), F'$ such that $\mu(a, R), E' \sim_\mu \mu(a, R), F'$. So, by the induction hypothesis, $\mu(a, R), F' \models \psi$, and so, by the definition of \models , $R, F \models [a]\psi$.

$\langle a \rangle$: Similar to the case for $[a]$.

\exists : Let ϕ be $\exists x.\psi$. By the induction hypothesis, we may assume that the result holds for each $\psi[a/x]$, where $a \in \text{Act}$. By the definition of \models , we have that $R, E \models \exists x.\psi$ iff, for some a such that $\mu(a, R) \downarrow$, $R, E \models \psi[a/x]$. Therefore, by the induction hypothesis, we have that $R, F \models \psi[a/x]$, and the result follows.

\forall : Similar to the case for \exists .

I : Let ϕ be I . By the definition of \models , we have that $R, E \models I$ iff $R \sqsubseteq e$ and $R, E \sim_\mu R, \mathbf{1}$. But if $R, E \sim_\mu R, F$, then $R, F \sim_\mu R, \mathbf{1}$, and the result follows.

$*$: Let ϕ be $\psi_1 * \psi_2$. By the induction hypothesis, we may assume that the result holds for ψ_1 and for ψ_2 . By the definition of \models , we have that $R, E \models \psi_1 * \psi_2$ iff, for some R_1 and R_2 such that $R_1 \circ R_2 \sqsubseteq R$ and some E_1 and E_2 such that

$$R_1 \circ R_2, E_1 \times E_2 \sim_\mu R, E,$$

$R_1, E_1 \models \psi_1$ and $R_2, E_2 \models \psi_2$. So, by the assumption of a bisimulation, $R_1 \circ R_2 = R$.

Now suppose that $R, E \sim_\mu R, F$. It follows immediately that $R, F \sim_\mu R_1 \circ R_2, E_1 \times E_2$, and so we are done.

\rightarrow : Let ϕ be $\psi_1 \rightarrow \psi_2$. By the induction hypothesis, we may assume that the result holds for ψ_1 and ψ_2 . By the definition of \models ,

$$\begin{aligned} R, E \models \psi_1 \rightarrow \psi_2 \quad \text{iff} \quad & \text{for all } R' \text{ and } E' \text{ such that } R \circ R' \downarrow \\ & \text{and } R', E' \models \psi_1, \\ & R \circ R', E \times E' \models \psi_2 \end{aligned}$$

Since $R, E \sim_\mu R, F$ and since \sim_μ is a congruence, we have

$$\begin{aligned} R, E \models \psi_1 \rightarrow \psi_2 \quad \text{iff} \quad & \text{for all } R' \text{ and } E' \text{ such that } R \circ R' \downarrow \\ & \text{and } R', E' \models \psi_1, \\ & R \circ R', F \times E' \models \psi_2 \\ \text{iff} \quad & R, F \models \psi_1 \rightarrow \psi_2 \end{aligned}$$

$[a]_\nu$: Let ϕ be $[a]_\nu \psi$. By the induction hypothesis, we may assume that the result holds for ψ . By the definition of \models , we have that $R, E \models [a]_\nu \psi$ iff, for all a and S such that $R \circ S, E \xrightarrow{a} \mu(a, R \circ S), E'$, subject to some conditions, $\mu(a, R \circ S), E' \models \psi$. Suppose that $R \circ S, F \xrightarrow{a} \mu(a, R \circ S), F'$. Then, by the definition of bisimulation, for some E' such that $R \circ S, E \xrightarrow{a} \mu(a, R \circ S), E'$, $\mu(a, R \circ S), E' \sim_\mu \mu(a, R \circ S), F'$. So, by the induction hypothesis, $\mu(a, R \circ S), F' \models \psi$, and so, by the definition of \models , $R, F \models [a]_\nu \psi$.

$\langle a \rangle_\nu$: Similar to the case for $[a]_\nu$.

\exists_ν : Let ϕ be $\exists_\nu x. \psi$. By the induction hypothesis we may assume that the result holds for any $\psi[a/x]$, where $a \in \text{Act}$. By the definition of \models , we have that $R, E \models \exists_\nu x. \psi$ iff, for some T some H such that $R, E \sim_\mu R, (\nu T)H$, and some b in $\Pi \mu^{-1}T$,

$$R \circ T, H \models \psi[b/x],$$

provided $R \cap T = \emptyset$. Now suppose that $R \circ T, H' \sim_\mu R \circ T, H$. Then, by the induction hypothesis, we have that

$$R \circ T, H' \models \psi[b/x] \text{ iff } R \circ T, H \models \psi[b/x].$$

Now suppose that $R, E \sim_\mu R, F$. It follows immediately that $R, F \sim_\mu R, (\nu T)H$, and so we are done.

\forall_ν : Similar to the case for \exists_ν .

□

We now turn to the converse, that logical equivalence implies bisimulation equivalence. Unfortunately, it does not seem to be possible to use the first-order quantifiers that are naturally present in our system to capture non-image-finite systems.

It seems that, just as for Hennessy-Milner logic for CCS, some form of infinitary conjunction would be necessary in order to handle the non-image-finite case.

It follows that, for image-finite processes, the argument of Stirling [Sti01] can be applied rather straightforwardly. This brings into focus the rôle of the multiplicatives in our setting. They provide a refinement of the analysis usual relationship between logical equivalence and bisimulation equivalence but their absence from the proof of Theorem 9.2 reveals the crudeness of the characterization provided by results of this form.

Theorem 9.2. *If, for all R and μ , R, E and R, F are image-finite and if, for all R , it is the case that $R, E \equiv_{\mathbf{MBI}} R, F$, then, for all R and μ , $R, E \sim_{\mu} R, F$.*

Proof. We adopt Stirling's technique [Sti01] and show that the relation

$$\{ (R, E, R, F) \mid \begin{array}{l} R, E \text{ and } R, F \text{ image-finite and} \\ R, E \equiv_{\mathbf{MBI}} R, F \end{array} \}$$

is a bisimulation.

Seeking a contradiction, we suppose not. Then, without loss of generality, for some R, G and R, H such that $R, G \equiv_{\mathbf{MBI}} R, H$, there are an a and a $\mu(a, R), G'$ such that $R, G \xrightarrow{a} \mu(a, R), G'$ but $\mu(a, R), G' \not\equiv_{\mathbf{MBI}} \mu(a, R), H'$, for all $\mu(a, R), H'$ such that $R, H \xrightarrow{a} \mu(a, R), H'$. Following Stirling's argument, we observe that

$$\mathcal{H} = \{ \mu(a, R), H' \mid R, H \xrightarrow{a} \mu(a, R), H' \}$$

is either empty or not.

Suppose that \mathcal{H} is empty. Then we have both that $R, G \models \langle a \rangle \top$ and that $R, H \not\models \langle a \rangle \top$, so contradicting $R, G \equiv_{\mathbf{MBI}} R, H$.

Otherwise \mathcal{H} is non-empty but finite, since we have assumed image-finiteness. So let it be $\{R, H_i \mid 1 \leq i \leq m\}$.

Suppose that, for each $1 \leq i \leq m$, $R, G' \not\equiv_{\mathbf{MBI}} R, H_i$. Then, for each i , there is some $\phi_i(a_i)$, for some $a_i \in Act$, such that $R, G' \models \phi_i(a_i)$ but $R, H_i \not\models \phi_i(a_i)$.

Now let ϕ be $\phi_1(a_1) \wedge \dots \wedge \phi_m(a_m)$. Then $R, G' \models \phi$ but $R, H_i \not\models \phi$. Therefore $R, G \models \langle a \rangle \phi$ and $R, H \not\models \langle a \rangle \phi$, and we have a contradiction. Therefore the relation

$$\{ (R, E, R, F) \mid \begin{array}{l} R, E \text{ and } R, F \text{ image-finite and} \\ R, E \equiv_{\mathbf{MBI}} R, F \end{array} \}$$

is a bisimulation. The result follows. \square

It remains to be considered whether results of this form are recoverable if the bisimulation is defined for processes with different resource environments.

10. ON IMPERATIVE PROGRAMS

A full presentation of the analysis of concurrent imperative programs provided by our framework is beyond our present scope. Nevertheless, we are able to explain, at least informally, how we are able to address the question, expressed in terms of a Hoare-style logic [OG76, O'H04], of giving a rule for concurrent composition of the form

$$\frac{\{\phi_1\} C_1 \{\psi_1\} \quad \{\phi_2\} C_2 \{\psi_2\}}{\{\phi_1 \text{ op}_1 \phi_2\} C_1 \text{ par } C_2 \{\psi_1 \text{ op}_2 \psi_2\}}$$

subject only to minimal side-conditions. Specifically, for commands C and D , expressible in SCCS terms, and recalling Milner's representation of Hoare logic in Hennessy-Milner logic [Mil89], we represent such a rule in the form

$$\frac{R, C \models \phi \quad S, D \models \psi}{R \circ S, C \times D \models \phi * \psi}.$$

Here we require just the constraint that the resources R and S , which in this set-up would handle (perhaps among other things) the representation of the program variables, be such that $R \circ S$ is defined and enforces sufficient separation.

To see the point, we introduce two small programs and consider, informally, their translation into **SCR**P. The program P_1 is

$$(X := X+1 ; Y := Y+1) \parallel (Y := Y+1 ; X := X+1)$$

The presence of races notwithstanding, this program is one that one might naturally wish to write. It cannot be treated within, for example, O'Hearn's Hoare-style concurrent separation logic [O'H04], because of the races, unless one wraps critical sections [OG76, O'H04] around the statements. This imposes a default level of granularity. When that is done, the program P_1 can be seen to increment both X and Y by 2.

The program P_2 , below, is very similar to P_1 but has no races. In fact, we'd like our semantics to treat them as equivalent.

$$(X := X+1 \parallel Y := Y+1) ; (Y := Y+1 \parallel X := X+1)$$

To see, at least informally, how our semantics treats these programs as being equivalent, we consider their actions on resources, when represented in **SCR**P. To this end, we assume a definition of a combinator Seq , for sequential composition, in **SCR**P, essentially as defined for CCS in [Mil89], such that, for actions c, d , we have essentially

$$\mu(c \text{ Seq } d, R) = \mu(c, \mu(d, R)).$$

Let R_X and R_Y be the resource components corresponding to the variables X and Y , respectively, and consider a suitable monoid of pairs (R_X, R_Y) . Let a and b be actions for the assignments $X := X + 1$ and $Y := Y + 1$, respectively. Then, abusing our notation somewhat, we can calculate the resource modification of P_1 , call it $\mu(P_1, (R_X, R_Y))$, as follows:

$$\begin{aligned} \mu(P_1, (R_X, R_Y)) &= \mu(a, \mu(b, (R_X, R_Y))) \circ \mu(b, \mu(a, (R_X, R_Y))) \\ &= \mu(a\#b, \mu(b, (R_X, R_Y))) \circ \mu(a, (R_X, R_Y)) && \text{definition of } \mu \\ &= \mu(a\#b, \mu(b\#a, ((R_X, R_Y) \circ (R_X, R_Y)))) && \text{definition of } \mu \end{aligned}$$

Notice that we have assumed that the composite

$$\mu(a, \mu(b, (R_X, R_Y))) \circ \mu(b, \mu(a, (R_X, R_Y))),$$

is defined; that is, we have a separation condition. Thus the program P_2 can also be seen to increment both X and Y by 2, since the modification corresponding to $a\#b$, incrementing each of R_X and R_Y , occurs twice.

Similarly, since we have, by the definition of μ ,

$$\mu(a, (R_X, R_Y)) \circ \mu(b, (R_X, R_Y)) = \mu(a\#b, (R_X, R_Y) \circ (R_X, R_Y))$$

we can calculate the resource modification of P_2 , $\mu(P_2, (R_X, R_Y))$, immediately as:

$$\mu(b\#a, \mu(a\#b, (R_X, R_Y) \circ (R_X, R_Y))).$$

But $a\#b = b\#a$, so we have $\mu(P_1, (R_X, R_Y)) = \mu(P_2, (R_X, R_Y))$. Again, notice that we have assumed that the composite $\mu(a, (R_X, R_Y)) \circ \mu(b, (R_X, R_Y))$ is defined; again, a separation condition.

So, in summary, we have demonstrated how our resource semantics will give the correct meaning to a concurrent imperative program, even one with races, provided it is, in a suitable sense, equivalent to a corresponding race-free program.

11. QUO VADIS

We have presented a calculus, **SCRIP**, of resources and processes, based on a development of Milner's synchronous calculus of communication systems, SCCS, that uses an explicit model of resource. Our calculus models the co-evolution of resources and processes with synchronization constrained by the availability of resources. We provide a logical characterization, analogous to Hennessy-Milner logic's characterization of bisimulation in CCS, of bisimulation between resource processes which is compositional in the concurrent and local structure of systems.

In many ways the power of the calculus **SCRIP** derives from the interaction between the decomposing behaviour of the parallel rule and the composition on resources. This has permitted us to avoid explicit notions of handshaking whilst retaining that computational power. The full calculus presented above appears to be very expressive, so for instance in the cases of the producer-consumer system we can present an infinite state system with a very small collection of syntax. One question is whether there are 'safe' (in the sense of Petri nets) sub-calculi that are interesting.

A possibly useful sub-calculus is one in which we omit the capability of actions to manipulate resource, so that the system will run with only the resources with which it starts. Whilst, superficially, it may seem that we force the system to have the same level of resource throughout its execution, that is not the case. To see this, consider first the process

$$Destroy \stackrel{\text{def}}{=} wait : Destroy + use : Destroy,$$

in the full system, with $\mu(wait, R^n) = R^n$, $\mu(use, R^n) = R^{n-1}$, using the Kripke resource monoid of our earlier examples.

Now consider the process

$$\begin{aligned} Hold &\stackrel{\text{def}}{=} hold : Hold \\ Destroy &\stackrel{\text{def}}{=} wait : Destroy + use : (Destroy \times Hold), \end{aligned}$$

where $\mu(hold, R) = R$, $\mu(wait, R^n) = R^n$, $\mu(use, R^n) = R^n$. Then the effect of this process is to deplete the resource R^n until there is no more free, since each instance of *Hold* will exploit one instance of R . This example illustrates that, in combination with recursion, modification functions of the form $\mu(a, R) = R$ can describe non-trivial systems.

In this example, we have a system that will eventually dissipate to a fix-point. The presence of such fix-points within systems is often a powerful lever in proving their properties and it is possible to envision proof approaches within this calculus where we can demonstrate that no matter what the initial resource levels then

eventually some ‘good’ outcome is reached and hence demonstrate that this property must hold in general. This may be of particular use within a probabilistic or queue-like setting.

As we noted in the producer–consumer example there, are certain ‘evident’ behaviours of the system that we cannot code within the current formalism. There has been some success in extending synchronous calculi with notions of both probability and priority [Tof94] and it would be interesting to see if such methods could be applied here; the major obvious point of difficulty is the non-uniqueness of the parallel rule applications. Given such extensions to the basic calculus, we may well have a tool that will attach the calculation of probabilistic properties for a large range of complex problems.

From the logical point of view, it will be necessary to investigate the evident extensions to include greatest and least fixed points [Sti01].

The analysis of concurrent imperative programs, sketched in § 10, remains to be developed in detail.

Acknowledgments. We are grateful to Jules Bean, Graham Birtwistle, Matthew Collinson, Jonathan Hayman, Brian Monahan, Peter O’Hearn, and two anonymous referees for their comments on drafts of this paper. Pym was partially supported by a Royal Society Industry Fellowship.

REFERENCES

- [BGL97] Patrice Brémont-Grégoire and Insup Lee. A process algebra of communicating shared resources with dense time and priorities. *Theoretical Computer Science*, 189(1–2):179–219, 1997.
- [BH72] P. Brinch Hansen. Structured multiprogramming. *Comm. ACM*, 15(7):574–78, 1972.
- [BH73] P. Brinch Hansen. *Operating System Principles*. Prentice Hall, 1973.
- [Bir79] G. Birtwistle. *Demos — discrete event modelling on Simula*. Macmillan, 1979.
- [Bir81] G. Birtwistle. *Demos implementation guide and reference manual*. Technical Report 81/70/22, University of Calgary, 1981.
- [BK84] J.A. Bergstra and J.W. Klop. The algebra of recursively defined processes and the algebra of regular processes. In *Proc 11th ICALP, LNCS 172*, 1984.
- [BT93] G. Birtwistle and C. Tofts. An operational semantics of process-orientated simulation languages: Part I π Demos. *Transactions of the Society for Computer Simulation*, 10(4):299–333, 1993.
- [BT94] G. Birtwistle and C. Tofts. An operational semantics of process-orientated simulation languages: Part II μ Demos. *Transactions of the Society for Computer Simulation*, 11(4):303–336, 1994.
- [BT98] G. Birtwistle and C. Tofts. A denotational semantics for a process-based simulation language. *ACM ToMaCS*, 8(3):281–305, 1998.
- [BT00a] G. Birtwistle and C. Tofts. Getting Demos Models Right — Part I Practice. *Simulation Practice and Theory*, 3:281–305, 2000.
- [BT00b] G. Birtwistle and C. Tofts. Getting Demos Models Right — Part II ... and Theory. *Simulation Practice and Theory*, 3:281–305, 2000.
- [CC03] L. Cardelli and L. Caires. A spatial logic of concurrency (part i). *Information and Computation*, 186(2):194–235, 2003.
- [Cle65] A.T. Clementson. Extended control and simulation language. *Computer Journal*, 9(3), 1965.
- [Dam90] M. F. Dam. *Relevance logic and concurrent computation*. Ph.D. thesis, University of Edinburgh, 1990.
- [Dij68] E.W. Dijkstra. Cooperating sequential processes. In *Programming Languages*, pages 43–112. Academic Press, 1968. Reprinted in [?].
- [Dij71] E.W. Dijkstra. Hierarchical ordering of sequential processes. *Acta Informatica*, 1(2):115–138, 1971. Reprinted in [?].

- [Dun86] J. M. Dunn. Relevant logic and entailment. In D. Gabbay and F. Guentner, editors, *Handbook of Philosophical Logic, vol. III: Alternatives to Classical Logic*, number 166 in Synthese Library, pages 117–224. D. Reidel, Dordrecht, Holland, 1986.
- [Gir87] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, pages 1–102, 1987.
- [GM04] P. Gastin and M. Mislove. A simple process algebra based on atomic actions with resources. *Mathematical Structures in Computer Science*, 14:1–55, 2004.
- [GMP02] D. Galmiche, D. Méry, and D. Pym. Resource Tableaux. In *Proc. CSL 2002*, volume 2471 of *LNCS*, pages 183–199, 2002.
- [Gun92] C. A. Gunter. *Semantics of Programming Languages: Structures and Techniques*. The MIT Press, Cambridge, Mass., and London, England, 1992.
- [Hoa72] C.A.R. Hoare. Towards a theory of parallel programming. In Hoare and Perrot, editors, *Operating Systems Techniques*. Academic Press, 1972. Reprinted in [?].
- [Hoa74] C.A.R. Hoare. Monitors: An operating system structuring concept. *Communications of the ACM*, 17(10):549–557, 1974.
- [Hoa85] C. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.
- [IO01] S.S. Ishtiaq and P. O’Hearn. **BI** as an assertion language for mutable data structures. In *28th ACM-SIGPLAN Symposium on Principles of Programming Languages, London*, pages 14–26. Association for Computing Machinery, 2001.
- [Kri63] S. A. Kripke. Semantical considerations on modal logic. *Acta Philosophica Fennica*, 16:83–94, 1963.
- [Kri65] S. A. Kripke. Semantical analysis of intuitionistic logic I. In J. N. Crossley and M. A. E. Dummett, editors, *Formal Systems and Recursive Functions*, pages 92–130. North-Holland, Amsterdam, 1965.
- [Maz87] A. Mazurkiewicz. Trace theory. In *Lecture Notes in Computer Science 255*, pages 279–324, 1987.
- [Mil83] R. Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science*, 25(3):267–310, 1983.
- [Mil89] R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.
- [Mil99] R. Milner. *Communication systems and the π -calculus*. Cambridge University Press, 1999.
- [OG76] S. Owicki and D. Gries. Verifying properties of parallel programs: An axiomatic approach. *Communications of the ACM*, 19(5):279–285, 1976.
- [O’H04] P. O’Hearn. Resources, concurrency, and local reasoning. In *Proc. Concur 04, London*, LNCS. Springer-Verlag, 2004.
- [OP99] P.W. O’Hearn and D.J. Pym. The logic of bunched implications. *Bulletin of Symbolic Logic*, 5(2):215–244, June 1999.
- [POY04] D.J. Pym, P.W. O’Hearn, and H. Yang. Possible worlds and resources: The semantics of **BI**. *Theoretical Computer Science*, 315(1):257–305, 2004. Erratum: p. 285, l. -12: “, for some $P', Q \equiv P; P'$ ” should be “ $P \vdash Q$ ”.
- [Pym02] D.J. Pym. *The Semantics and Proof Theory of the Logic of the Logic of Bunched Implications*, volume 26 of *Applied Logic Series*. Kluwer Academic Publishers, 2002. Errata and Remarks [Pym04] maintained at: <http://www.cs.bath.ac.uk/~pym/BI-monograph-errata.pdf>.
- [Pym04] D.J. Pym. Errata and Remarks for *The Semantics and Proof Theory of the Logic of Bunched Implications* [Pym02]. Maintained at: <http://www.cs.bath.ac.uk/~pym/BI-monograph-errata.pdf>, 2004.
- [Rey02] J.C. Reynolds. Separation Logic: A Logic for Shared Mutable Data Structures. In *Proc. LICS '02*, pages 55–74. IEEE Computer Society Press, 2002.
- [RM72] R. Routley and R.K. Meyer. The semantics of entailment, II-III. *Journal of Philosophical Logic*, 1:53–73 and 192–208, 1972.
- [Sti01] Colin Stirling. *Modal and Temporal Properties of Processes*. Springer Verlag, 2001.
- [Tof94] C. Tofts. Processes with probability, priority and time. *Formal Aspects of Computer Science*, 6(5):536–564, 1994.
- [Tof03] C. Tofts. Efficiently modelling resource in a process algebra. Technical Report HPL-2003-181, HP Laboratories, Bristol, 2003.

HP LABS, BRISTOL, UK
E-mail address: david.pym@hp.com

HP LABS, BRISTOL, UK
E-mail address: chris.tofts@hp.com