Theoretical Computer Science 614 (2016) 63–96

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

A calculus and logic of bunched resources and processes

Gabrielle Anderson, David Pym [*]

University College London, United Kingdom

ARTICLE INFO

Article history:
Received 7 July 2015
Received in revised form 3 November 2015
Accepted 20 November 2015
Available online 27 November 2015
Communicated by A. Avron

Keywords:
Modal logic
Process algebra
Bunched logic
Resource semantics

ABSTRACT

Mathematical modelling and simulation modelling are fundamental tools of engineering, science, and social sciences such as economics, and provide decision-support tools in management. Mathematical models are essentially deployed at all scales, all levels of complexity, and all levels of abstraction. Models are often required to be executable, as a simulation, on a computer. We present some contributions to the process-theoretic and logical foundations of discrete-event modelling with resources and processes. Building on previous work in resource semantics, process calculus, and modal logic, we describe a process calculus with an explicit representation of resources in which processes and resources co-evolve. The calculus is closely connected to a substructural modal logic that may be used as a specification language for properties of models. In contrast to earlier work, we formulate the resource semantics, and its relationship with process calculus, in such a way that we obtain soundness and completeness of bisimulation with respect to logical equivalence for the naturally full range of logical connectives and modalities. We give a range of examples of the use of the process combinators and logical structure to describe system structure and behaviour.

1. Introduction

Mathematical modelling and simulation modelling are fundamental tools of engineering, science, and social sciences such as economics, and provide decision-support tools in management. Mathematical models are essentially deployed at all scales, all levels of complexity, and all levels of abstraction.

This paper contributes to the logical and mathematical foundations of discrete-event modelling of distributed systems. The classical theory of distributed systems (as described, for example, in [15]) provides a rigorous conceptual basis for this work, which can be conveniently abstracted to describe systems in terms of collections of interconnected locations, at which are situated resources, relative to which processes execute — consuming, creating, moving, and otherwise manipulating resources as they evolve — and so deliver a system's services. Distributed systems, as described here, exist not in isolation but within environments with which they interact. A system's environment is both a source of events, that are incident upon the system, and the recipient of events caused by the execution of the system's processes.

Modelling is a process of making a precise description — that is, a model — of a system in order to explore rigorously its properties. The process of constructing a model is described as a cycle of observation, model definition, analysis of properties, and exploration of implied consequences for the system, leading to further observation, and so on. This cycle of construction embodies the process by which a model is judged to be a sound, or valid, representation of the system,

[*] Corresponding author.
E-mail addresses: gabrielle.anderson@ucl.ac.uk (G. Anderson), d.pym@ucl.ac.uk (D. Pym).

http://dx.doi.org/10.1016/j.tcs.2015.11.035
0304-3975/© 2015 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

capturing just those aspects of the system's structure and dynamics that are determined to be relevant to the questions that model is required to address.

In discrete-event models, the (model of the) system evolves in discrete steps. In traditional applied mathematics, these models of dynamical systems are typically described by families of difference equations that describe the system's local evolution from one step to the next. An evolution (or flow) operator is derived that completely describes the behaviour of the system. For large and/or complex systems, models of this kind are rarely susceptible to exact solution and, in such situations, computational models, through techniques such as Monte Carlo simulations, provide alternative methods of analysis. These stochastic aspects of system modelling, though an essential part of the modelling approach, are not the subject of this paper. Rather, we are concerned here with some theoretical properties of the structural theory of models.

Mathematically, the components of distributed systems are modelled using various algebraic structures for the structural components (location, resource, and process) and probability distributions to represent stochastic interactions with the environment. For the remainder of this paper, we are concerned just with the structural aspects.

In [11,13,14], the work upon which this paper builds, the following approach is taken:

- Location is modelled using structures that satisfy some basic requirements of there being sets of connected places having some substitution properties. Leading examples are directed graphs and topological constructions [11,14];
- Resource is modelled by structures that satisfy the requirements that it should be possible both to combine and to compare resource elements. These requirements are captured by preordered commutative monoids, subject to a functoriality condition that relates the monoidal composition and the preorder — called resource monoids [11,12,14] — and leading examples are given by the monoid of natural numbers with addition (with unit 0) ordered by less-than-or-equals, computer memory (as in separation logic), and Petri nets [11,12,14];
- Process is modelled using an algebra of processes that is closely related to Milner's SCCS [27]. The key point in the formulation in [11,12,14] is that resources and processes co-evolve.

Mathematically, this set-up leads to an operational judgement of the form

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

which is read as 'the action a, with resources R at location L, evolves the process E to be the process E', able to evolve with resources R' at location L''.

The judgement is defined by a structural operational semantics [31] for action prefix, sum, (synchronous) product, and so on. For example,

$$\frac{}{L, R, a : E \xrightarrow{a} L', R', E} \; \textsc{Act},$$

where the action a — with access to resources R at location L — occurs, evolving the process $a : E$ to E, which is then further able to evolve starting at location L' with resources R',

$$\frac{R, E_i \xrightarrow{a} R', E_i'}{R, E_1 + E_2 \xrightarrow{a} R', E_i'} \; i \in \{1, 2\} \quad \textsc{Sum},$$

and

$$\frac{L, R_1, E_1 \xrightarrow{a_1} L', R_1', E_1' \quad L, R_2, E_2 \xrightarrow{a_2} L', R_2', E_2'}{L, R_1 \circ R_2, E_1 \times E_2 \xrightarrow{a_1 a_2} L', R_1' \circ R_2', E_1' \times E_2'} \; \textsc{Prod},$$

where $\circ$ is the monoidal composition of resources, $\times$ is synchronous concurrent product of processes, and $a_1 a_2$ denotes a monoidal product of actions.

More generally, each of the premisses in the Prod rule may be located separately — starting at L_1 and L_2, say, respectively — with the conclusion starting at a product of L_1 and L_2. Note our use, as in SCCS [27], of a synchronous product. From the modelling perspective, synchrony is preferred over asynchrony for its greater generality [27,35].

In subsequent sections of this paper, we neglect location: whilst it is conceptually significant and convenient in modelling, mathematically, we can essentially code it within resource and neglecting it simplifies the subsequent technical development. The calculus sketched above is known as SCRP (for synchronous calculus of resources and processes) and is developed fully in [11,12,14].

SCRP provides the semantic framework underpinning the modelling language Gnosis [13,14,18]. Gnosis and closely related tools (all owing much to Birtwistle's Demos [7]) have been employed in a range of industrial-strength applied modelling projects undertaken by Hewlett–Packard and others (e.g., [18,22,2–4,9]). The modelling methodology has been developed further in, for example, [10].

This mathematical formulation supports a modal logic of actions for assertions about the state of the system (model), with the judgement of the form

$$L, R, E \vDash \phi,$$

read as 'the state L, R, E has property ϕ'. The connection between the logic and the operational semantics derives from the action modalities, $\langle a\rangle$ and $[a]$, with the satisfaction clause

$$\begin{aligned} L, R, E \vDash \langle a\rangle\phi \text{ iff } & \text{there exist } L', R', E' \text{ such that} \\ & L, R, E \xrightarrow{a} L', R', E' \text{ and} \\ & L', R', E' \vDash \phi, \end{aligned}$$

for $\langle a\rangle$ and a similar one for $[a]$.

For CCS and SCCS, for example, this logic is called Hennessy–Milner logic [20,21,28] and, just as in [11,12,14], we adopt the name for the logic developed here. The key characterizing result here, which we can usefully describe as the Hennessy–Milner completeness theorem, relates logical equivalence and process bisimulation. Informally, one obtains ideally (as for, say, CCS) a theorem of the following form:

For all location-resource-processes, L_1, R_1, E_1 is bisimilar to L_2, R_2, E_2 if and only if, for all logical formulae ϕ, $L_1, R_1, E_1 \vDash \phi$ if and only if $L_2, R_2, E_2 \vDash \phi$.

For the resource–process calculus and associated modal logic presented in [11,12,14], the strength of the Hennessy–Milner completeness theorem is limited.

The reverse direction — that the logical equivalence of states implies their bisimulation equivalence — is obtained in full generality. But the forward direction — that bisimulation equivalence of resource–processes implies their logical equivalence — holds only for fragments of the logic that exclude certain multiplicative components. One such multiplicative component is the multiplicative modality $\langle a\rangle_\nu$, with satisfaction definition

$$\begin{aligned} L, R, E \vDash \langle a\rangle_\nu\phi \text{ iff } & \text{there exist } L', R', E' \text{ and } S, S' \text{ such that} \\ & L, R \circ S, E \xrightarrow{a} L', R' \circ S', E' \text{ and} \\ & L', R' \circ S', E' \vDash \phi, \end{aligned}$$

where $\circ$ is the monoidal resource composition. The multiplicative components that must be excluded from the logic for the forward direction to hold are the multiplicative modalities $\langle a\rangle_\nu$ and $[a]_\nu$, and the multiplicative implication, $-\!\!*$. This situation is both theoretically unsatisfactory and a limitation in practical reasoning about system models.

In this paper, we develop fully a formulation of a calculus of resources and processes, together with its associated modal logic, that solves this problem; that is, we obtain the result that

For all resource–process pairs, $R_1, E_1 \sim R_2, E_2$ (that is, R_1, E_1 is bisimilar to R_2, E_2) if and only if $R_1, E_1 \equiv_{MBI} R_2, E_2$ (that is, for all logical formulae ϕ, $R_1, E_1 \vDash \phi$ if and only if $R_2, E_2 \vDash \phi$).

In Section 2, we set up a calculus of resources and processes. We introduce the notion of bunched resources, which represents a conceptual departure from the simply motivated resource semantics employed in [11] and elsewhere, sketched above. Bunching of resources, just as with contexts in BI [30,32,17], employs two conjunctive combinators, which we denote $\oplus$ and $\otimes$, giving sharing and separating combinations of resources, respectively. The operational semantics of the calculus is (in part) determined by the way that actions modify resources. To maintain various properties concerning this modification relationship, we also provide a different structure on actions. In particular, one important property is that the 1 action 'modifies' all resources to themselves (it acts as a unit). Note that we do not, however, work with the monoid equations on actions. Technically, the improved theoretical relationship between the process calculus and the logic derives from the combinatorial match between the structure of processes and the structure of resources, which we use to establish the Hennessy–Milner completeness theorem. Were we to make use of the monoid equations on actions, we would not be able to maintain this close link between the structure of the resources and the processes. Conceptually, it would seem to be suggested that the capturing of sharing and separation within the logic by co-existing additive and multiplicative connectives should be reflected in the underlying resource semantics. We would conjecture, perhaps rather obviously, that an ordering on resources may need to be reintroduced in order to set up intuitionistic variants of the logical theory (here we work with a classical logic). We explore various examples, including several classic examples surrounding concurrent interaction, to demonstrate how our resource semantics works. With that grounding, we define a notion of bisimulation, and prove that various operators, including concurrent composition, are a congruence with respect to the bisimulation relation.

In Section 3, we provide an embedding of previous work [11,12,14]. We describe in more detail the relevant calculi. We provide an example embedding, and prove that our work can simulate any transition structure definable in the previous work.

In Section 4, we describe how to establish the standard algebraic properties of process calculi in our formalization. In many process calculi, the monoid equations on actions are used to establish these properties, for the standard notion of behavioural equivalence (bisimulation). As we cannot use the monoid equations, we take a different approach. To regain the algebraic properties we modify the notion of bisimulation via the introduction of an equivalence over actions, which functions in a similar manner to the monoid equations on actions. Note that the fact that we impose the use of the equivalence relation in the notion of bisimulation does not mean we must impose its use in the notion of actions modifying resources, and that two bisimilar resource–process pairs do not necessarily perform *exactly* the same actions, but that they perform *equivalent* actions. Hence, bisimilar pairs perform distinct (but equivalent) transitions that lead to bisimilar states. This is possible as, unlike in [11,12,14], the definition of bisimulation permits bisimilar resource–process pairs to have *different* resource components. Hence the distinct (but equivalent) actions can modify the resource components of the two pairs differently, and have the resulting pairs remain in the bisimulation relation.

In Section 5, we introduce a modal logic MBI — borrowing the name from the logic with the same formulæ taken in [11,12,14] — of resources and processes that provides an assertion language for the properties of resource–process states. This logic, presented in here in its classical form — that is, as in Boolean BI [25], in which the additives are classical — is defined in relation to resource–process states just as Hennessy–Milner logic is defined in relation to CCS states, and the previously established modal logic of [11,14] is defined in relation to SCRP states. We establish the Hennessy–Milner completeness theorem for MBI in full generality.

The key tract of related work is based around O'Hearn's concurrent separation logic [29]. These ideas have been developed in a range of directions, including the provision of a semantics for Hoare's Communicating Sequential Processes (CSP) [24] — a calculus of processes that shares some combinatorial properties with our calculus and its relatives — using the structures that support concurrent separation logic. We discuss, in Section 6, how work along these lines might inform an analysis of the relationship between concurrent separation logic and the resource–process semantics considered here. A summary of this work has been presented in [1].

2. A calculus of bunched resources and processes

We present a family of systems, known collectively as CBRP, along with their key technical properties.

Compared to the set-up employed in [11,14], we introduce additional structure into resources, which permits us to consider concurrent composition of, and non-deterministic choice between, resources and their transitions. We describe how actions modify resources, and how such modifications must behave with respect to the structure of the resources. We define a notion of hiding resources and actions that generalizes the approach used in [11,14] (which itself generalizes the notion of restriction).

We explore various examples, including several classic examples surrounding concurrent interaction [14]. In one example, we describe how to encode weak memory consistency [36] through the use of non-determinism within the resource component, an example which cannot be encoded within existing resource–process calculi.

We define processes and an operational semantics for resource–process pairs. We introduce the standard notion of bisimulation for such a transition system, and prove various properties of the bisimulation relation; in particular, that the key operators of the resource–process pairs are a congruence with respect to the bisimulation relation.

We conclude by establishing some expansion theorem results that are used in an embedding of SCRP calculi [11,14] into our calculus (Section 3).

The set-up of these calculi assumes the provision of certain additional data pertaining to some semantic structure $(\mathbf{Act}, \mathbf{Res}, \mathbf{R}, \mu, \Delta, \mathbf{H})$ over which we work and which we define in the development below (the structure **Act** pertains to the actions used in the calculus, the structures **Res**, **R**, and μ pertain to the resource semantics used in the calculus, Δ pertains to the how resources are handled by sequential compositions of processes, and **H** pertains to how action hiding is handled in the calculus). Thus we should properly refer to the calculus as $(\mathbf{Act}, \mathbf{Res}, \mathbf{R}, \mu, \Delta, \mathbf{H})$-CBRP. In this paper, however, we suppress the prefix as, at every stage, we work with a fixed such structure.

Just as in the systems described in [11,14], summarized briefly in the introduction, the operational semantics of a CBRP system defines a transition system in which resources and process co-evolve. The primary judgement in our set-up will be of the form

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

which is read as 'the process E, with available resources R, evolves by the action a to become the process E' with available resources R''. Alternatively, 'the resource–process R, E evolves by the action a to become the resource–process R', E''. As such evolutions occur, not only does the structure of the process components evolve, but also resources are manipulated: consumed, created, and distributed around the system. 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.

In [11,14], the manner in which processes are distributed around a system is explicitly represented — through the concurrent product, non-deterministic choice, and so on, of processes — but the manner in which the resources used by the processes are distributed is not. There, the distribution of resources to processes is performed non-deterministically, in the operational semantics rule for concurrent product of processes. This leads the operator for concurrent product to not be a congruence over the standard notion of bisimulation, in [11,14]. In this paper, we represent the distribution of resources around a system through the use of a 'bunching' structure on resources (described formally below).

We use sets of atomic resources, **Res**; these are similar to the atomic resource elements considered in [11,14], where they are required to form monoidal structures, and correspond to the basic resource infrastructure of a system; they denote, for example, CPU time, memory, vehicles, or money. The elements of a set **Res** of atomic resource are denoted, r, s, etc. There is a distinguished element, e, denoting the 'empty' resource. Note that the set of simple resources is one of the parameters to the calculus.

Following [33,30,11,12,14], and other works in the relevant logic tradition, bunches are trees with leaves labelled by simple resources, and internal nodes labelled by either $\oplus$ or $\otimes$. Let R, S, etc. denote bunches of resources. A node labelled by $\oplus$ denotes a (portion of a) system where one or the other of the sub-bunches of resources can be used by a process, but not both. For example, an agent leaving an underground station can choose to exit using either the stairs or the escalator, but cannot do both (at the same time). One can notice that $\oplus$ behaves sort of like an exclusive or, in that it makes use of one or the other of the sub-components. A node labelled by $\otimes$ denotes a (portion of a) system where the sub-branches of resources are allocated to the subprocesses of a concurrent product process. For an example of this set-up, consider two separate piles of objects that can be sorted by two separate agents, in parallel.

We now define bunched resources.

Definition 1 *(Bunched resources).* Let **Res** be a set of atomic resources, with a distinguished resource e (known as the 'empty resource'), and let $r \in \mathbf{Res}$. Then the set $\mathbf{\Omega}$ of bunched resources (over atomic resources **Res**) is formed according to the following grammar:

$$R ::= r \mid R \oplus R \mid R \otimes R. \quad \square$$

We use $=$ to denote syntactic equality of resource bunches. Note that a set of atomic resources is one of the parameters to the calculus.

Definition 2 *(Resource models).* Let $\mathbf{\Omega}$ be the set of bunched resources formed over a set of atomic resources **Res**, with 'empty' resource e. Then a resource model **R** is a set $\mathbf{R} \subseteq \mathbf{\Omega}$ that includes the atomic resources (i.e., $\mathbf{Res} \subseteq \mathbf{R}$) and is closed under the following, for all bunched resources $R, R_1, R_2, R_3 \in \mathbf{R}$:

- If the bunch $R_1 \otimes R_2$ is in **R**, then the bunches R_1 and R_2 are in **R**;
- If the bunch $R_1 \oplus R_2$ is in **R**, then the bunches R_1 and R_2 are in **R**;
- The bunch $R_1 \oplus R_2$ is in **R** if and only if the bunch $R_2 \oplus R_1$ is in **R**;
- The bunch $R_1 \otimes R_2$ is in **R** if and only if the bunch $R_2 \otimes R_1$ is in **R**;
- The bunch $R_1 \otimes (R_2 \otimes R_3)$ is in **R** if and only if the bunch $(R_1 \otimes R_2) \otimes R_3$ is in **R**;
- The bunch $R_1 \oplus (R_2 \oplus R_3)$ is in **R** if and only if the bunch $(R_1 \oplus R_2) \oplus R_3$ is in **R**;
- The bunch $R_1 \otimes (R_2 \oplus R_3)$ is in **R** if and only if the bunches $R_1 \otimes R_2$ and $R_1 \otimes R_3$ are in **R**;
- The bunches $R \oplus R$, $R \oplus e$, and $R \otimes e$ are in **R**. $\square$

In the sequel, for brevity, when we write $R \otimes S$, we assume that it is defined in the resource model that defines R and S. Note that a resource model is one of the parameters to the calculus.

Example 3 *(Semaphore resource model).* Consider a contested resource: a semaphore. Only one process should be able to (concurrently) access the semaphore at any given time. We model this scenario as follows. Let the set of atomic resources $\mathbf{Res} = \{s, e\}$ consist of the element s, which denotes the semaphore, and the element e, which denotes the empty resource. Let the resource model **R** be the least set such that Definition 2 holds. Then, we have that $s, s \otimes e, s \oplus s \in \mathbf{R}$, but do not have that $s \otimes s \in \mathbf{R}$. $\square$

Actions correspond to the events of a system. In process algebra, the set of actions is typically assumed to be a semi-group, or a commutative monoid, or a similar algebraic structure [27]. In resource–process algebra as set up in [11,14], however, actions are used to determine how resources evolve. This necessitates a relationship between the monoidal product of actions and the monoidal structure of resources. In order to obtain an analogous relationship in our setting (formally stated in Definition 5), given that the structure that we use for resources is bunching, we must weaken the structure on actions.

Definition 4 *(Actions).* Let **Act** be a set of atomic actions and let $\alpha \in \mathbf{Act}$. Then the set **A** of actions (over atomic actions **Act**) is formed according to the following grammar:

$$a ::= 1 \mid \alpha \mid a \cdot a. \quad \square$$

Note that we do not require that 1 be a unit for $\cdot$, so that **A** is not a monoid. We use $=$ to denote syntactic equality of actions. Note also that the set of atomic actions **Act** is one of the parameters to the calculus.

In many process algebras, such as SCCS and SCRP, the commutative monoid structure of actions is used to prove various algebraic properties of states. Here, the actions do not form a (commutative) monoid. In this paper, we first develop a notion of bisimulation for which resource–process concurrent composition can be proved to be a congruence with respect to bisimulation, but for which the standard algebraic properties do not hold. We do this to demonstrate how the bunched resource structure is used in the proof of congruence. Then, we extend the notion of bisimulation in a way that does not interfere with our ability to prove congruence, but enables us to regain the algebraic properties of states (Section 4).

Definition 5 *(Modification functions).* A partial function $\mu : \mathbf{A} \times \mathbf{R} \rightharpoonup \mathbf{R}$ is a modification function if, for all bunched resources $R, S \in \mathbf{R}$ and actions $a, b \in \mathbf{Act}$:

- If $\mu(a, R)$, $\mu(b, S)$, $R \otimes S \in \mathbf{R}$, then $\mu(a, R) \otimes \mu(b, S)$, $\mu(a \cdot b, R \otimes S) \in \mathbf{R}$ and $\mu(a \cdot b, R \otimes S) = \mu(a, R) \otimes \mu(b, S)$;
- If $\mu(a, R) = R'$, $\mu(b, S) = S'$, and $R' \otimes S' \in \mathbf{R}$, then $R \otimes S \in \mathbf{R}$;
- $\mu(1, R) = R$. □

Note that the action 1 is a unit for μ's action on resources. Note also that a modification function is one of the parameters to the calculus.

Example 6 *(Concurrent counters).* We model a series of counters, each of which can be incremented independently. Let the set of atomic resources $\mathbf{Res} = \mathbb{N}$ be the set of natural numbers, and let 0 be the empty resource. Let the resource model $\mathbf{R}$ be the least set such that, for all natural numbers $m, n \in \mathbb{N}$, $m \otimes n \in \mathbf{R}$, and Definition 2 holds. Let the set of atomic actions $\mathbf{Act} = \{i\}$ consist of a single action i, which denotes incrementation. Let the modification function $\mu : \mathbf{A} \times \mathbf{R} \rightharpoonup \mathbf{R}$ be the least function (under set inclusion of the domain) such that $\mu(i, n) = n + 1$, and Definition 5 holds. Consider a 'tuple' of counters, $2 \otimes 4$. The following properties of the modification function hold. The action $i \cdot 1$ increments the left counter: $\mu(i \cdot 1, 2 \otimes 4) = 3 \otimes 4$, as $\mu(i, 2) = 3$ and $\mu(1, 4) = 4$. The action $1 \cdot i$ increments the right counter: $\mu(1 \cdot i, 2 \otimes 4) = 2 \otimes 5$, as $\mu(1, 2) = 2$ and $\mu(i, 4) = 5$. Note that the modification function is undefined for action i and resource bunch $2 \otimes 4$. This is as we only define the modification function for action i on atomic resources $n \in \mathbb{N}$, and Definition 5 only extends this to bunched resources when the $\cdot$ structure of the actions can be matched with the $\otimes$ structure of the bunched resources. □

A related approach has been explored by Hennessy in [19]. There, resources are allocated to specific processes, and actions modify them functionally. These resources are in some cost domain, which has a minimum element and notions of addition and subtraction. There is no notion of choice between resources, or of richer structures than n resources in parallel (for n processes).

Modification functions are homomorphisms with respect to the concurrent product structure of resource bunches. As a result, we cannot use the modification function to 'move' resources from one side of a concurrent product to another (such a move corresponds to changing the process to which the resources are allocated, for example, passing an object from producer to consumer). Using the modification function, we can only add or remove resources to each side of a product independently of what is on the other side of the concurrent product.

As we cannot use the modification function for redistribution of resources, instead, we make use of *redistribution functions*, which are defined in terms of *redistribution operators*.

In Fig. 1 (page 71), the rules for the operational semantics of sequential composition are

$$\frac{R, E \xrightarrow{a} R', E' \quad \delta \in \Delta}{R, E :_\delta F \xrightarrow{a} R', E' :_\delta F}\ \textsc{PrefixOne} \qquad \frac{R, E \not\rightarrow \quad \delta(R), F \xrightarrow{a} R', F' \quad \delta \in \Delta}{R, E :_\delta F \xrightarrow{a} R', F'}\ \textsc{PrefixTwo}.$$

The resource–process pair $R, E :_\delta F$ consists of a resource bunch and a sequential composition. The sequential composition consists of two processes, E and F, and a redistribution function δ. If the prefix E can evolve with the resources R, then the sequential composition evolves similarly (the PrefixOne rule). If the prefix E cannot evolve with the resources R, then the redistribution function is applied to the resources R, and the pair that consists of the resulting resources and the suffix, $\delta(R), F$, is evolved (the PrefixTwo rule). The redistribution function is applied to the resources so that the structure of the resulting resources will match the structure of the suffix process.

Definition 7 *(Redistribution operators).* The following functions $\delta : \mathbf{R} \to \mathbf{R}$ are redistribution operators:

(1) Commutative operators: $\delta : R \otimes S \mapsto S \otimes R$ and $\delta : R \oplus S \mapsto S \oplus R$;
(2) Associative operators: $\delta : R \otimes (S \otimes T) \mapsto (R \otimes S) \otimes T$, $\delta : (R \otimes S) \otimes T \mapsto R \otimes (S \otimes T)$, $\delta : R \oplus (S \oplus T) \mapsto (R \oplus S) \oplus T$, and $\delta : (R \oplus S) \oplus T \mapsto R \oplus (S \oplus T)$;
(3) Distributive operators: $\delta : R \otimes (S \oplus T) \mapsto (R \otimes S) \oplus (R \otimes T)$;
(4) Operators that add resources: $\delta : R \mapsto (R \otimes S)$ and $\delta : R \mapsto (R \oplus S)$;
(5) Operators that delete resources: $\delta : R \otimes S \mapsto R$ and $\delta : R \oplus S \mapsto R$. □

Definition 8 *(Redistribution functions).* A redistribution function is a (partial) function $\delta : \mathbf{R} \rightharpoonup \mathbf{R}$ generated by the composition of a (subset of the) functions in Definition 7, together with the identity. □

Let δ, δ', etc. denote redistribution functions on a given resource model $\mathbf{R}$, and let Δ denote a set of redistribution functions, which is one of the parameters to the calculus.

Example 9. Consider the resource model in Example 3. Let the function $\delta : \mathbf{R} \rightharpoonup \mathbf{R}$ map $s \otimes e \mapsto (s \otimes e) \oplus (e \otimes s)$. This function can be defined using the subset of redistribution operators consisting only of the operators that add resources. □

From a modelling perspective, we argue that the use of redistribution functions encourages good discipline with respect to making decisions about how resources are allocated to processes within a system. In previous work [11,12,14], all possible allocations were possible, and a system could non-deterministically choose between them. In our work, whenever resources are to be re-allocated (i.e., following each reduction step, within a sequential composition), a conscious modelling decision is required as to where the resources should be allocated. This permits us to model complex behaviours, such as in Example 21, below, concerning weak memory consistency. There, after certain steps, resources are copied around; and, after others, some resources are removed. Under an alternative semantics for weak memory consistency, we could require that a process reads its own most recent write (rather than just any of its writes, as in the example; this is known as read-your-writes consistency [36]). This would be straightforward to model through a redistribution function that deletes old local writes, but distributes them to other processes. This demonstrates how intensional choices about the allocation of resources can be used to model important properties of complex systems.

In classical process calculi, restriction is used to ensure that certain behaviour is only visible, or accessible, in certain parts of a system. A similar feature can be incorporated into resource–process modelling [11]. If a resource–process pair is allocated additional resources, it may be able to perform additional behaviour. The *hiding* operator on processes associates additional resources with the process to which it is applied.

In Fig. 1 (page 71), the rule for the operational semantics of hiding is

$$\frac{h(R), E \xrightarrow{a} h(R'), E' \quad h \in \mathbf{H}}{R, \nu h.E \xrightarrow{\nu h.a} R', \nu h.E'} \ \textsc{Hide}.$$

The resource–process pair $R, \nu h.E$ consists of a resource bunch and a hiding process. The hiding process consists of a hiding function h and of a process E. The additional resources allocated to the process E are acquired by the application of the hiding function h to the resources R. The operational semantics of the resource–process pair $R, \nu h.E$ is then defined in terms of the operational semantics of the resource–process pair $h(R), E$. Note that the action $\nu h.a$ performed by the pair $R, \nu h.E$ is not the same as the action a that is performed by the pair $h(R), E$, though it is defined in terms of a. This is addressed further below.

Definition 10 *(Hiding functions on resources).* A function $h : \mathbf{R} \to \mathbf{R}$ on a resource model is a hiding function if: it is a bijection; it maps $\otimes$ nodes (respectively, $\oplus$ nodes) to $\otimes$ nodes (respectively, $\oplus$ nodes); and atomic resources are mapped to atomic resources. □

Let h, h', etc. denote hiding functions on a given resource model $\mathbf{R}$, and let $\mathbf{H}$ denote a set of hiding functions, which is the final parameter to the calculus.

Example 11. Let $\mathbf{Res} = \{e, s\}$, the empty resource be e,

$$\begin{array}{ll} R = (e \oplus s) \otimes e & R' = (s \oplus s) \otimes e \\ S = (e \otimes e) \oplus (s \otimes e) & S' = (s \otimes e) \oplus (s \otimes e) \\ T = e \otimes e & T' = s \otimes e, \end{array}$$

and let $\mathbf{R}$ be the least set such that $R, R', S, S', R, T' \in \mathbf{R}$ and the constraints in Definition 2 hold. We define a hiding function.

$$\begin{array}{lll} h(R) = R' & h(S) = S' & h(T) = T' \\ h(R') = R & h(S') = S & h(T') = T \end{array}$$

$$h(z) = z \quad \text{otherwise}$$

Note that where the function does not map elements of $\mathbf{R}$ to themselves, at the leaves, atomic resources are mapped to atomic resources. For example, h maps $(e \oplus s) \otimes e$ to $(s \oplus s) \otimes e$. Both bunches have the same bunching structure, $(\_ \oplus \_) \otimes \_$. At the first leaf from the left, the atomic resource e is mapped to the atomic resource s, and, at the other leaves, the atomic resources are mapped to themselves. □

If a resource–process pair is allocated additional resources, it may be able to perform additional actions. This behaviour must then be restricted, however; only actions that could be performed without the additional resources must be visible beyond the process where the hidden resources are available. An example determination of such an action is in Example 20.

Actions are generated by the composition of atomic actions (denoted α, β, etc.), using $\cdot$. An action a 'contains' an action b exactly when they have the same structure, and at each leaf, b has either the same atomic action as the former, or the action 1; for example, $1 \cdot (\beta \cdot 1) \leq \alpha \cdot (\beta \cdot 1) \leq \alpha \cdot (\beta \cdot \gamma)$. We define a partial order based on the notion of containment.

Definition 12 *(Action-containment order).* The least relation under reflexivity and transitivity such that the following hold:

$$\frac{}{1 \leq \alpha}\ (1) \qquad \frac{a \leq a' \quad b \leq b'}{a \cdot b \leq b' \cdot a'}\ (2). \qquad \square$$

The first rule defines that the action 1 is contained in any atomic action. The second rule defines how containment relates to the structure of actions. If $a \leq b$, then a is said to be a sub-action of b.

We can now formalize our definition of hiding function on actions, as described above.

Definition 13 *(Hiding functions on actions).* Let $a \in \mathbf{A}$ be an action, $\mu : \mathbf{A} \times \mathbf{R} \rightharpoonup \mathbf{R}$ be a modification function, $h : \mathbf{R} \to \mathbf{R}$ be a hiding function on resources, and

$$A_{a,h} = \{b \leq a \mid \text{for all } R, S \in \mathbf{R}, \mu(a, h(R)) = h(S) \text{ implies } \mu(b, R) = S\}$$

be an auxiliary set definition. Let $\sup : \mathcal{P}(\mathbf{A}) \rightharpoonup \mathbf{A}$ be the partial function from a set of actions to its supremum constituent element, with respect to the action containment order. This function is defined if and only if a unique supremum exists. Then, a hiding function on actions $\nu : (\mathbf{R} \to \mathbf{R}) \to \mathbf{A} \to \mathbf{A}$ is defined as

$$\nu h.a = \begin{cases} \sup(A_{a,h}) & \text{if} \sup(A_{a,h}) \text{ is defined and unique} \\ 1 & \text{otherwise.} \end{cases} \qquad \square$$

Definition 14 *(Processes).* Processes are formed according to the following grammar:

$$E ::= \mathbf{0} \mid X \mid a \mid E + E \mid E \times E \mid E :_\delta E \mid \nu h.E \mid \text{fix } X.E. \quad \square$$

Here, $\mathbf{0}$ is the zero process, X is a process variable, a is an action, $\delta \in \Delta$ is a redistribution function, and $h \in \mathbf{H}$ is a hiding function. Let **Proc** be the set of all processes, and E, F etc. denote processes. The process $\mathbf{1}$, which performs the action 1 infinitely, is denoted as $\mu X.1 :_{id} X$.

The process structure follows that of ACP [5], with the exception of hiding $\nu h.E$, which is based on [11,14], and the annotation δ on sequential composition $E :_\delta F$. Thus $E + F$ is a *sum*, $E \times F$ is a *synchronous product*, and $\text{fix } X.E$ is a *fixed point*. The term $\nu h.E$ is a *hiding* process. The term $E :_\delta F$ is an *annotated sequential composition*; the semantics of the annotation is explained below.

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 all agents. The process $F[E/X]$ is the process formed by the (capture-avoiding) substitution of E for the corresponding variable X that is free in F. We use brackets, (), to disambiguate processes in the absence of their construction trees.

A *state* is a pair consisting of a resource and a process. Let $\mathbf{State} = \mathbf{R} \times \mathbf{Proc}$ be the set of all states. A *closed state* is a pair consisting of a resource and an agent. Let $\mathbf{CState} = \mathbf{R} \times \mathbf{Agents}$ be the set of all closed states.

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

$$\xrightarrow{a} \quad \subseteq \quad \mathbf{State} \times \mathbf{State},$$

indexed by actions $a \in \mathbf{A}$. The family is defined recursively using the derivation rules in Fig. 1.

An action process reduces according to the modification function μ. Nondeterminism is introduced solely through the presence of sums. There, a choice must be made both in the process component and the resource component. Product processes distribute the resources according to the multiplicative structure in the resources.

Sequential composition mostly behaves intuitively. If the prefix can be reduced, with the accompanying resources, then the sequential composition follows similarly. If the prefix process cannot be reduced, then the suffix process is reduced. The suffix process is, however, accompanied by the resources that result from the application of the annotated redistribution function to the existing resources. The redistribution function is used to redistribute the resources between the process components, following a reduction that moves to the second part of a sequential composition. It should be noted that the use of process prefixing, rather than action prefixing, is a deliberate design decision, made so that models can more intuitively reflect the structure of the system they abstract.

$$\frac{}{R, a \xrightarrow{a} \mu(a,R), \mathbf{0}}\ \textsc{Act} \qquad \frac{R_i, E_i \xrightarrow{a} R_i', E_i'}{R_1 \oplus R_2, E_1 + E_2 \xrightarrow{a} R_i', E_i'}\ \textsc{Sum}_i\text{, for } i \in \{1,2\}$$

$$\frac{R_1, E_1 \xrightarrow{a_1} R_1', E_1' \quad R_2, E_2 \xrightarrow{a_2} R_2', E_2'}{R_1 \otimes R_2, E_1 \times E_2 \xrightarrow{a_1 \cdot a_2} R_1' \otimes R_2', E_1' \times E_2'}\ \textsc{Prod}$$

$$\frac{R, E \xrightarrow{a} R', E' \quad \delta \in \Delta}{R, E :_\delta F \xrightarrow{a} R', E' :_\delta F}\ \textsc{PrefixOne} \qquad \frac{R, E \not\rightarrow \quad \delta(R), F \xrightarrow{a} R', F' \quad \delta \in \Delta}{R, E :_\delta F \xrightarrow{a} R', F'}\ \textsc{PrefixTwo}$$

$$\frac{h(R), E \xrightarrow{a} h(R'), E' \quad h \in \mathbf{H}}{R, \nu h.E \xrightarrow{\nu h.a} R', \nu h.E'}\ \textsc{Hide} \qquad \frac{R, E[\text{fix } X.E/X] \xrightarrow{a} R', E'}{R, \text{fix } X.E \xrightarrow{a} R'.E'}\ FV(E) \subseteq \{X\}\ \textsc{Rec}$$

Fig. 1. Operational semantics.

Example 15 *(Semaphore).* A simple example of the use of $\oplus$-bunched resources and redistribution functions is a system where two processes repeatedly compete for the use of a semaphore. We use two atomic actions, a and b, both of which require access to a semaphore, s, in order to be performed. We differentiate between the a and b actions to help make clear which process is accessing the semaphore at any given point. Let $\mathbf{Act} = \{a, b\}$, $\mathbf{Res} = \{e, s\}$, with empty resource e,

$$R = (s \oplus s) \otimes (e \oplus e) \qquad S = (e \oplus e) \otimes (s \oplus s) \qquad T = R \oplus S,$$

and let $\mathbf{R}$ be the least set such that $R, S, T \in \mathbf{R}$ and the constraints in Definition 2 hold. Let the modification function be the least such that

$$\mu(a, s) = s \qquad \mu(b, s) = s$$

and the constraints in Definition 5 hold. The resource R denotes the scenario where the semaphore is allocated to the first process, and S where it is allocated to the second process. The resource T then denotes the scenario where the semaphore may be allocated to either of the processes, but not to both. The process $E = (1 + a) \times (1 + b)$ denotes a system where two subprocesses each attempt to access the semaphore (through actions a and b respectively). When the process $E + E$ is combined with the resource T, the state can either evolve through use of the resource R (with process E), or through the use of the resource S (with process E). In the first case, the first process can access the semaphore, but the second process can only tick (the action 1). Derivations should be read bottom–up. Rule names are included where space permits.

$$\cfrac{\cfrac{\cfrac{\cfrac{\mu(a,s) = s}{s, a \xrightarrow{a} s, \mathbf{0}}\ \textsc{Act}}{(s \oplus s), (1+a) \xrightarrow{a} s, \mathbf{0}}\ \textsc{Sum}_2 \qquad \cfrac{\cfrac{\mu(1,e) = e}{e, 1 \xrightarrow{1} e, \mathbf{0}}\ \textsc{Act}}{(e \oplus e), (1+b) \xrightarrow{1} e, \mathbf{0}}\ \textsc{Sum}_1}{(s \oplus s) \otimes (e \oplus e), (1+a) \times (1+b) \xrightarrow{a \cdot 1} s \otimes e, \mathbf{0} \times \mathbf{0}}\ \textsc{Prod}}{R \oplus S, E + E \xrightarrow{a \cdot 1} s \otimes e, \mathbf{0} \times \mathbf{0}}\ \textsc{Sum}_1.$$

In the second case, then the converse is true:

$$\cfrac{\cfrac{\cfrac{\cfrac{\mu(1,e) = e}{e, 1 \xrightarrow{1} e, \mathbf{0}}\ \textsc{Act}}{(e \oplus e), (1+a) \xrightarrow{1} e, \mathbf{0}}\ \textsc{Sum}_1 \qquad \cfrac{\cfrac{\mu(b,s) = s}{s, b \xrightarrow{b} s, \mathbf{0}}\ \textsc{Act}}{(s \oplus s), (1+b) \xrightarrow{b} s, \mathbf{0}}\ \textsc{Sum}_2}{(e \oplus e) \otimes (s \oplus s), (1+a) \times (1+b) \xrightarrow{1 \cdot b} e \otimes s, \mathbf{0} \times \mathbf{0}}\ \textsc{Prod}}{R \oplus S, E + E \xrightarrow{1 \cdot b} e \otimes s, \mathbf{0} \times \mathbf{0}}\ \textsc{Sum}_2.$$

Following these evolutions, the resulting resources will either be of the form $s \otimes e$ or of the form $e \otimes s$. If this bunched resource were combined with the process $E + E$, then no progress could be made, as the structure of the resources doesn't match the structure of the processes. Hence, following the above evolutions, if we plan to make use the resources with another iteration of $E + E$, we must redistribute the resources, to reintroduce the relevant structure. This can be done through a redistribution function:

$$\delta(z) = \begin{cases} T & \text{if } z = s \otimes e \text{ or } z = e \otimes s \\ z & \text{otherwise.} \end{cases}$$

Note that this redistribution function can be defined in terms of redistribution operators that add resources (cf. Definition 7).

We then can define the following system, where processes contend repeatedly for access to the semaphore:

$$F = \text{fix}\, X.((E + E) :_\delta X).$$

Using the rule PrefixOne, the system T, F can evolve as

$$T, F \xrightarrow{a\cdot 1} s \otimes e, (\mathbf{0} \times \mathbf{0}) :_\delta F.$$

Following this, the resulting state can evolve as:

$$\dfrac{s \otimes e, \mathbf{0} \times \mathbf{0} \not\twoheadrightarrow \quad \dfrac{\dots}{\delta(s \otimes e), F \xrightarrow{1\cdot b} e \otimes s, (\mathbf{0} \times \mathbf{0}) :_\delta F}\,\text{Rec} \quad \delta(s \otimes e) = R \oplus S}{s \otimes e, (\mathbf{0} \times \mathbf{0}) :_\delta F \xrightarrow{1\cdot b} e \otimes s, (\mathbf{0} \times \mathbf{0}) :_\delta F}\,\text{PrefixTwo}. \qquad \square$$

The use of process prefixing, rather than action prefixing, is a deliberate design decision, made so that states can more intuitively reflect the structure of the system they abstract. In the above example, the redistribution function δ maps $s \otimes e$ to a $\oplus$-bunched resource, which creates a non-deterministic choice concerning which of the subprocess is allocated the semaphore. We cannot, however, define some resource U, redistribution functions δ_1 and δ_2, and process

$$G = (\text{fix}\, X.((1 :_{\delta_1} X) + (a :_{\delta_1} X))) \times (\text{fix}\, Y.((1 :_{\delta_2} Y) + (b :_{\delta_2} Y))),$$

such that the state U, G is bisimilar to T, F in Example 15: the redistribution functions δ_1 and δ_2 would only have access to the resources on either side of the $\otimes$-bunched resource (s and e, respectively) and hence neither would be able to redistribute the resources on the other side of the $\otimes$-bunch. We could define a redistribution function δ'' and process H

$$\delta''(s) = s \oplus (s \oplus s) \qquad H = \text{fix}\, X.((1 :_{\delta''} X) + ((a :_{\delta''} X) + (b :_{\delta''} X))),$$

which, with resource $s \oplus (s \oplus s)$, is bisimilar to T, E. This system, however, loses the intuitive structure of two agents contending for a resource, and looks essentially like a state created via an expansion theorem (as in Lemma 30, below). In order to permit redistribution of resources between agents, and to prevent the states collapsing to an 'expanded' form, we permit process prefixing.

Example 16 *(Mutual exclusion).* (See [14].) A slightly more complex example is a system where two subprocesses use a semaphore to ensure that only one is acting in its critical region at any given time. Here we use only one action that accesses the semaphore, a. Let $\mathbf{Act} = \{a\}$, $\mathbf{Res} = \{e, s\}$, with empty resource e,

$$R = (s \oplus (s \oplus s)) \otimes (e \oplus (e \oplus e)) \qquad S = (e \oplus (e \oplus e)) \otimes (s \oplus (s \oplus s)) \qquad T = R \oplus S,$$

and $\mathbf{R}$ be the least set such that $T \in \mathbf{R}$ and the constraints in Definition 2 hold. Let the modification function be the least such that $\mu(a, s) = s$ and the constraints in Definition 5 hold. We define the following redistribution functions:

$$\delta(z) = \begin{cases} e \oplus (e \oplus e) & \text{if } z = e \\ s \oplus (s \oplus s) & \text{if } z = s \\ T & \text{if } z = s \otimes e \text{ or } z = e \otimes s \\ z & \text{otherwise,} \end{cases}$$

and

$$\delta'(z) = \begin{cases} e \oplus e & \text{if } z = e \\ r \oplus r & \text{if } z = r \\ z & \text{otherwise.} \end{cases}$$

We define the processes of the system as follows:

$$E = \text{fix}\, X.((E_1 \times E_1) + (E_1 \times E_1)) :_{\delta'} X$$

$$E_1 = \text{fix}\, Y.(1 :_\delta Y + (1 + (a :_{\delta'} E_2))) \qquad E_2 = \text{fix}\, Z.(a :_{\delta'} Z + a).$$

The process E consists of a choice between two identical subprocesses in parallel. Each subprocess attempts to acquire the semaphore. When it does so, it enters its critical region, E_2:

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{\dfrac{\mu(a,s)=s}{s,a \xrightarrow{a} s,\mathbf{0}}\,\textsc{Act}}{s,a :_{\delta'} E_2 \xrightarrow{a} s,\mathbf{0} :_{\delta'} E_2}\,\textsc{PrefixOne}
}{s\oplus s, 1+(a :_{\delta'} E_2) \xrightarrow{a} s,\mathbf{0} :_{\delta'} E_2}\,\textsc{Sum}_2
}{s\oplus(s\oplus s), 1 :_{\delta} E_1 + (1+(a :_{\delta'} E_2)) \xrightarrow{a} s,\mathbf{0} :_{\delta'} E_2}\,\textsc{Sum}_2
\quad
\dfrac{
\dfrac{\dfrac{\mu(1,e)=e}{e,1 \xrightarrow{1} e,\mathbf{0}}\,\textsc{Act}}{e,1 :_{\delta} E_1 \xrightarrow{1} e,\mathbf{0} :_{\delta} E_1}\,\textsc{PrefixOne}
}{e\oplus(e\oplus e), 1 :_{\delta} E_1 + (1+(a :_{\delta'} E_2)) \xrightarrow{1} e,\mathbf{0} :_{\delta} E_1}\,\textsc{Sum}_1
}{R, E_1 \times E_1 \xrightarrow{a\cdot 1} s\otimes e, \mathbf{0} :_{\delta'} E_2 \times \mathbf{0} :_{\delta} E_1}\,\textsc{Prod}
}{R \oplus S, (E_1\times E_1)+(E_1\times E_1) \xrightarrow{a\cdot 1} s\otimes e, \mathbf{0} :_{\delta'} E_2 \times \mathbf{0} :_{\delta} E_1}\,\textsc{Sum}_1
}{R \oplus S, ((E_1\times E_1)+(E_1\times E_1)) :_{\delta} E \xrightarrow{a\cdot 1} s\otimes e, (\mathbf{0} :_{\delta'} E_2 \times \mathbf{0} :_{\delta} E_1) :_{\delta} E}\,\textsc{PrefixOne}.
$$

While there, the other process may only tick, and cannot acquire the semaphore:

$$
\dfrac{
\dfrac{
\dfrac{
s,\mathbf{0}\nrightarrow \quad
\dfrac{
\dfrac{
\dfrac{\dfrac{\mu(a,s)=s}{s,a \xrightarrow{a} s,\mathbf{0}}\,\textsc{Act}}{s,a :_{\delta'} E_2 \xrightarrow{a} s,\mathbf{0} :_{\delta'} E_2}\,\textsc{PrefixOne}
}{s\oplus s, (a :_{\delta'} E_2)+a \xrightarrow{a} s,\mathbf{0} :_{\delta'} E_2}\,\textsc{Sum}_1
}{\delta'(s), \mathrm{fix}\, Z.((a :_{\delta'} Z)+a) \xrightarrow{a} s,\mathbf{0} :_{\delta'} E_2}\,\textsc{Rec}
}{s,\mathbf{0} :_{\delta'} E_2 \xrightarrow{a} s,\mathbf{0} :_{\delta'} E_2}\,\textsc{PrefixTwo}
\quad
\dfrac{
e,\mathbf{0}\nrightarrow \quad
\dfrac{
\dfrac{
\dfrac{\dfrac{\mu(1,e)=e}{e,1 \xrightarrow{1} e,\mathbf{0}}\,\textsc{Act}}{e,1 :_{\delta} E_1 \xrightarrow{1} e,\mathbf{0} :_{\delta} E_1}\,\textsc{PrefixOne}
}{e\oplus(e\oplus e), 1 :_{\delta} E_1 + (1+(a :_{\delta} E_2)) \xrightarrow{1} e,\mathbf{0} :_{\delta} E_1}
}{\delta(e), E_1 \xrightarrow{1} e,\mathbf{0} :_{\delta} E_1}
}{e,\mathbf{0} :_{\delta} E_1 \xrightarrow{1} e,\mathbf{0} :_{\delta} E_1}\,\textsc{PrefixTwo}
}{s\otimes e, \mathbf{0} :_{\delta'} E_2 \times \mathbf{0} :_{\delta} E_1 \xrightarrow{a\cdot 1} s\otimes e, \mathbf{0} :_{\delta'} E_2 \times \mathbf{0} :_{\delta} E_1}\,\textsc{Prod}
}{s\otimes e, (\mathbf{0} :_{\delta'} E_2 \times \mathbf{0} :_{\delta} E_1) :_{\delta} E \xrightarrow{a\cdot 1} s\otimes e, (\mathbf{0} :_{\delta'} E_2 \times \mathbf{0} :_{\delta} E_1) :_{\delta} E}\,\textsc{PrefixOne}.
}{}
$$

When the process that has the semaphore decides to exit its critical region (by choosing the process a instead of the process $a : E_2$), both prefix processes can terminate:

$$
\dfrac{
\dfrac{
s,\mathbf{0}\nrightarrow \quad
\dfrac{
\dfrac{\dfrac{\mu(a,s)=s}{s,a \xrightarrow{a} s,\mathbf{0}}\,\textsc{Act}}{s\oplus s, (a :_{\delta'} E_2)+a \xrightarrow{a} s,\mathbf{0}}\,\textsc{Sum}_2
}{\delta'(s), \mathrm{fix}\, Z.((a :_{\delta'} Z)+a) \xrightarrow{a} s,\mathbf{0}}\,\textsc{Rec}
}{s,\mathbf{0} :_{\delta'} E_2 \xrightarrow{a} s,\mathbf{0}}\,\textsc{PrefixTwo}
\quad
\dfrac{
e,\mathbf{0}\nrightarrow \quad
\dfrac{
\dfrac{
\dfrac{\dfrac{\mu(1,e)=e}{e,1 \xrightarrow{1} e,\mathbf{0}}\,\textsc{Act}}{e\oplus e, 1+(a :_{\delta} E_2) \xrightarrow{1} e,\mathbf{0}}\,\textsc{Sum}_1
}{e\oplus(e\oplus e), 1 :_{\delta} E_1 + (1+(a :_{\delta} E_2)) \xrightarrow{1} e,\mathbf{0}}\,\textsc{Sum}_2
}{\delta(e), E_1 \xrightarrow{1} e,\mathbf{0}}\,\textsc{Rec}
}{e,\mathbf{0} :_{\delta} E_1 \xrightarrow{1} e,\mathbf{0}}\,\textsc{PrefixTwo}
}{
\dfrac{s\otimes e, \mathbf{0} :_{\delta'} E_2 \times \mathbf{0} :_{\delta} E_1 \xrightarrow{a\cdot 1} s\otimes e, \mathbf{0}\times\mathbf{0}}{s\otimes e, (\mathbf{0} :_{\delta'} E_2 \times \mathbf{0} :_{\delta} E_1) :_{\delta} E \xrightarrow{a\cdot 1} s\otimes e, (\mathbf{0}\times\mathbf{0}) :_{\delta} E}\,\textsc{PrefixOne}.
}\,\textsc{Prod}
$$

In this example, one of the processes may obtain the semaphore and proceed to the critical region, but both cannot. Moreover, once one process has gone into its critical region, the other process cannot pre-empt it, through the use of the semaphore, to enter its critical region, until the former chooses to exit its critical region. □

Example 17 *(Resource transfer).* (See [14].) A similar example is given by a system in which only one of the parallel tasks is 'active' at any one time, and that, instead of the subprocesses non-deterministically entering their critical regions, they take turns. This can be achieved through the passing of tokens, in the form of resources, back and forth between the processes. These resources are known as 'permits', which are 'produced' by the actions p_1 and p_2, and are 'gotten' by the actions g_1 and g_2. The resource r_i is required for the g_i action to be performed, and is used to guard the ith process's critical region. Let $\mathbf{Act} = \{g_1, g_2, p_1, p_2\}$, $\mathbf{Res} = \{e, r_1, r_2\}$, with empty resource e,

$$R_1 = (r_1 \oplus (r_1 \oplus r_2)) \otimes (e \oplus (e \oplus e)) \qquad R_2 = (e \oplus (e \oplus e)) \otimes (r_2 \oplus (r_2 \oplus r_2))$$

and $\mathbf{R}$ be the least set such that $R_1, R_2 \in \mathbf{R}$, and the constraints in Definition 2 hold. Let the modification function be the least such that

$$\mu(p_1, e) = r_2 \;\; \mu(p_2, e) = r_1 \;\; \mu(g_1, r_1) = e \;\; \mu(g_2, r_2) = e,$$

and the constraints in Definition 5 hold.

A process may either 'get' the relevant permit (if available) and proceed to its critical region, tick and loop, or tick and terminate. Once within the critical region, a process may perform its task (here we simply perform a tick action and loop), or produce a token that enables the other process and leave its critical region. As the permit for one processes is generated by the other, following an evolution, the processes need to exchange their resources. This is performed by the redistribution function δ:

$$\delta(z)=\begin{cases}(e\oplus(e\oplus e)) & \text{if } z=e\\ (r_1\oplus(r_1\oplus r_1)) & \text{if } z=r_1\\ (r_2\oplus(r_2\oplus r_2)) & \text{if } z=r_2\\ R_2 & \text{if } z=r_2\otimes e\\ R_1 & \text{if } z=e\otimes r_1.\end{cases}$$

We also make use of one other redistribution function, δ':

$$\delta'(z)=\begin{cases}e\oplus e & \text{if } z=e\\ r_1\oplus r_1 & \text{if } z=r_1\\ r_2\oplus r_2 & \text{if } z=r_2\\ z & \text{otherwise.}\end{cases}$$

We define the processes of the system as follows:

$$E=\text{fix } X.((E_1\times E_2):_\delta X)$$

$$E_1=\text{fix } Y_1.(g_1:_{\delta'}E_1'+((1:_\delta Y_1)+1))\qquad E_1'=\text{fix } Z_1.((1:_{\delta'}Z_1)+p_1)$$

$$E_2=\text{fix } Y_2.(g_2:_{\delta'}E_2'+((1:_\delta Y_2)+1))\qquad E_2'=\text{fix } Z_2.((1:_{\delta'}Z_2)+p_2).$$

The system R_1, E represents the scenario where the subprocess E_1 goes first. It performs its ‘get’ action, consumes its permit, and enters its critical region, E_1':

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mu(g_1,r_1)=e}{r_1,g_1\xrightarrow{g_1}e,\mathbf{0}}\,\textsc{Act}}{r_1,g_1:_{\delta'}E_1'\xrightarrow{g_1}e,\mathbf{0}:_{\delta'}E_1'}\,\textsc{PrefixOne}}{r_1\oplus(r_1\oplus r_2),g_1:_{\delta'}E_1'+((1:_\delta E_1)+1)\xrightarrow{g_1}e,\mathbf{0}:_{\delta'}E_1'}\,\textsc{Sum}_1\quad \dfrac{\dfrac{\dfrac{\dfrac{\mu(1,e)=e}{e,1\xrightarrow{1}e,\mathbf{0}}\,\textsc{Act}}{e,1:_\delta E_2\xrightarrow{1}e,\mathbf{0}:_\delta E_2}\,\textsc{PrefixOne}}{e\oplus e,(1:_\delta E_2)+1\xrightarrow{1}e,\mathbf{0}:_\delta E_2}\,\textsc{Sum}_1}{e\oplus(e\oplus e),g_2:_{\delta'}E_2'+((1:_\delta E_2)+1)\xrightarrow{1}e,\mathbf{0}:_\delta E_2}\,\textsc{Sum}_2}{(r_1\oplus(r_1\oplus r_2))\otimes(e\oplus(e\oplus e)),E_1\times E_2\xrightarrow{g_1\cdot 1}e\otimes e,(\mathbf{0}:_{\delta'}E_1')\times(\mathbf{0}:_\delta E_2)}\,\textsc{Prod}}{R_1,(E_1\times E_2):_\delta E\xrightarrow{g_1\cdot 1}e\otimes e,((\mathbf{0}:_{\delta'}E_1')\times(\mathbf{0}:_\delta E_2)):_\delta E}\,\textsc{PrefixOne}.$$

The first process can evolve through its critical region, and finishes with its p_1 action, which produces the permit for the second process to proceed:

$$\dfrac{\dfrac{\dfrac{e,\mathbf{0}\nrightarrow\quad\dfrac{\dfrac{\dfrac{\mu(p_1,e)=r_2}{e,p_1\xrightarrow{p_1}r_2,\mathbf{0}}\,\textsc{Act}}{e\oplus e,(1:_{\delta'}E_1')+p_1\xrightarrow{p_1}r_2,\mathbf{0}}\,\textsc{Sum}_2}{\delta'(e),E_1'\xrightarrow{p_1}r_2,\mathbf{0}}\,\textsc{Rec}}{e,\mathbf{0}:_{\delta'}E_1'\xrightarrow{p_1}r_2,\mathbf{0}}\,\textsc{PrefixTwo}\quad\dfrac{e,\mathbf{0}\nrightarrow\quad\dfrac{\dfrac{\dots}{e\oplus(e\oplus e),g_2:_{\delta'}E_2'+((1:_\delta E_2)+1)\xrightarrow{1}e,\mathbf{0}}\,\textsc{Sum}_2}{\delta(e),E_2\xrightarrow{1}e,\mathbf{0}}\,\textsc{Rec}}{e,\mathbf{0}:_\delta E_2\xrightarrow{1}e,\mathbf{0}}\,\textsc{PrefixTwo}}{e\otimes e,((\mathbf{0}:_{\delta'}E_1')\times(\mathbf{0}:_\delta E_2))\xrightarrow{p_1\cdot 1}r_2\otimes e,\mathbf{0}\times\mathbf{0}}\,\textsc{Prod}}{e\otimes e,((\mathbf{0}:_{\delta'}E_1')\times(\mathbf{0}:_\delta E_2)):_\delta E\xrightarrow{p_1\cdot 1}r_2\otimes e,(\mathbf{0}\times\mathbf{0}):_\delta E}\,\textsc{PrefixOne}.$$

When the prefix terminates, the system can apply the redistribution function δ to the resources and evolve according to the suffix. This transfers the permit r_2 to the second process, so that it can proceed and enter its own critical region:

$$\dfrac{r_2\otimes e,(\mathbf{0}\times\mathbf{0})\nrightarrow\qquad\dfrac{\dfrac{\dfrac{\dfrac{\dots}{(e\oplus(e\oplus e)),E_1\xrightarrow{1}e,\mathbf{0}:_\delta E_1}\,\textsc{Rec}\quad\dfrac{\dfrac{\dfrac{\mu(g_2,r_1)=e}{r_2,g_2\xrightarrow{g_2}e,\mathbf{0}}\,\textsc{Act}}{r_2,g_2:_{\delta'}E_2'\xrightarrow{g_2}e,\mathbf{0}:_{\delta'}E_2'}\,\textsc{PrefixOne}}{(r_2\oplus(r_2\oplus r_2)),E_2\xrightarrow{g_2}e,\mathbf{0}:_{\delta'}E_2'}\,\textsc{Sum}_1}{R_2,E_1\times E_2\xrightarrow{1\cdot g_2}e\otimes e,(\mathbf{0}:_\delta E_1)\times(\mathbf{0}:_{\delta'}E_2')}\,\textsc{Prod}}{R_2,(E_1\times E_2):_\delta E\xrightarrow{1\cdot g_2}e\otimes e,((\mathbf{0}:_\delta E_1)\times(\mathbf{0}:_{\delta'}E_2')):_\delta E}\,\textsc{PrefixOne}}{\delta(r_2\otimes e),E\xrightarrow{1\cdot g_2}e\otimes e,((\mathbf{0}:_\delta E_1)\times(\mathbf{0}:_{\delta'}E_2')):_\delta E}\,\textsc{Rec}}{r_2\otimes e,(\mathbf{0}\times\mathbf{0}):_\delta E\xrightarrow{1\cdot g_2}e\otimes e,((\mathbf{0}:_\delta E_1)\times(\mathbf{0}:_{\delta'}E_2')):_\delta E}\,\textsc{PrefixTwo}.$$

This example can be straightforwardly extended to 'round-robin' schedulers, and demonstrates the complex communication and scheduling patterns that can be represented using the resource–process paradigm. □

Example 18 *(Handshaking).* (See [14]). A different form of distributed task coordination is that where two processes only proceed together, and may not proceed individually. This scenario (known as a 'join') can be implemented through the use of a pair of semaphores. Each process may either choose to 'go', which is denoted by the g_i action, or to 'wait', which is denoted by the w_i action.

Let $\mathbf{Act} = \{g_1, g_2, w_1, w_2\}$, $\mathbf{Res} = \{e, r_1, r_2\}$, e be the empty resource,

$$R = (r_1 \oplus r_1) \otimes (r_2 \oplus r_2) \qquad S = (r_2 \oplus r_2) \otimes (r_1 \oplus r_1) \qquad T = R \oplus S$$

and $\mathbf{R}$ be the least set such that $R, S, T \in \mathbf{R}$, and the constraints in Definition 2 hold. Let the modification function be the least such that

$$\mu(g_1, r_1) = r_1 \quad \mu(g_2, r_2) = r_2 \quad \mu(w_1, r_2) = r_2 \quad \mu(w_2, r_1) = r_1,$$

and the constraints in Definition 5 hold. The 'go' action of one process uses the same semaphore as the 'wait' action of the other.

The resource R denotes the scenario where the first process has the first semaphore, and the second process has the second semaphore. The resource S denotes the scenario where the second process has the first semaphore, and the first process has the second semaphore. The resource T denotes the scenario where resources can be allocated according to R or according to S, but where neither both processes nor a single process can have both semaphores. We make use of the redistribution function

$$\delta(z) = \begin{cases} r_1 \oplus r_1 & \text{if } z = r_1 \\ r_2 \oplus r_2 & \text{if } z = r_2 \\ T & \text{if } z = r_1 \otimes r_2 \text{ or } z = r_2 \otimes r_1. \end{cases}$$

We define the processes of the system as

$$\begin{aligned} E &= \text{fix}\, X.(((E_1 \times E_2) + (E_1 \times E_2)) :_\delta X) \\ E_1 &= w_1 + (g_1 :_\delta E_1') \qquad E_2 = w_2 + (g_2 :_\delta E_2'), \end{aligned}$$

where E_1' and E_2' are the critical regions of each process (which we elide for the purposes of this example).

The process $\big(w_1 + (g_1 :_\delta E_1')\big) \times \big(w_2 + (g_2 :_\delta E_2')\big)$ consists of two subprocesses, each of which can either choose to wait, or to go (and to proceed to the subprocess's critical region). When combined with resource T, the process can either evolve through the use of the resource R or through the use of the resource S.

In the latter case, each process can only perform its 'wait' action.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mu(w_1, r_2) = r_2}{r_2, w_2 \xrightarrow{w_1} r_2, \mathbf{0}}\,\textsc{Act}}{r_2 \oplus r_2, w_1 + (g_1 :_\delta E_1') \xrightarrow{w_1} r_2, \mathbf{0}}\,\textsc{Sum}_1 \qquad \dfrac{\dfrac{\mu(w_2, r_1) = r_1}{r_1, w_2 \xrightarrow{w_2} r_1, \mathbf{0}}\,\textsc{Act}}{r_1 \oplus r_1, w_2 + (g_2 :_\delta E_2') \xrightarrow{w_2} r_1, \mathbf{0}}\,\textsc{Sum}_1}{(r_2 \oplus r_2) \otimes (r_1 \oplus r_1), E_1 \times E_2 \xrightarrow{w_1 \cdot w_2} r_2 \otimes r_1, \mathbf{0} \times \mathbf{0}}\,\textsc{Prod}}{R \oplus S, ((E_1 \times E_2) + (E_1 \times E_2)) \xrightarrow{w_1 \cdot w_2} r_2 \otimes r_1, \mathbf{0} \times \mathbf{0}}\,\textsc{Sum}_2}{T, ((E_1 \times E_2) + (E_1 \times E_2)) :_\delta E \xrightarrow{w_1 \cdot w_2} r_2 \otimes r_1, (\mathbf{0} \times \mathbf{0}) :_\delta E}\,\textsc{PrefixOne}.$$

In the second case, each process can only perform its 'go' action.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mu(g_1, r_1) = r_1}{r_1, g_1 \xrightarrow{g_1} r_1, \mathbf{0}}\,\textsc{Act}}{r_1, g_1 :_\delta E_1' \xrightarrow{g_1} r_1, \mathbf{0} :_\delta E_1'}\,\textsc{PrefixOne}}{r_1 \oplus r_1, w_1 + (g_1 :_\delta E_1') \xrightarrow{g_1} r_1, \mathbf{0} :_\delta E_1'}\,\textsc{Sum}_2 \qquad \dfrac{\dfrac{\dfrac{\mu(g_2, r_2) = r_2}{r_2, g_2 \xrightarrow{g_2} r_2, \mathbf{0}}\,\textsc{Act}}{r_2, g_2 :_\delta E_2' \xrightarrow{g_2} r_2, \mathbf{0} :_\delta E_2'}\,\textsc{PrefixOne}}{r_2 \oplus r_2, w_2 + (g_2 :_\delta E_2') \xrightarrow{g_2} r_2, \mathbf{0} :_\delta E_2'}\,\textsc{Sum}_2}{(r_1 \oplus r_1) \otimes (r_2 \oplus r_2), E_1 \times E_2 \xrightarrow{g_1 \cdot g_2} r_1 \otimes r_2, (\mathbf{0} :_\delta E_1') \times (\mathbf{0} :_\delta E_2')}\,\textsc{Prod}}{R \oplus S, ((E_1 \times E_2) + (E_1 \times E_2)) \xrightarrow{g_1 \cdot g_2} r_1 \otimes r_2, (\mathbf{0} :_\delta E_1') \times (\mathbf{0} :_\delta E_2')}\,\textsc{Sum}_1}{T, ((E_1 \times E_2) + (E_1 \times E_2)) :_\delta E \xrightarrow{g_1 \cdot g_2} r_1 \otimes r_2, ((\mathbf{0} :_\delta E_1') \times (\mathbf{0} :_\delta E_2')) :_\delta E}\,\textsc{PrefixOne}.$$

When the second of the resources is chosen, the subprocesses must perform the 'wait' actions and loop back to the starting process. Notice that the two processes can either wait or proceed together, but cannot proceed independently. □

Example 19 *(Asynchronous resource transfer without exclusion).* (See [14].) Some distributed coordination approaches don't make use of exclusion: the most common form is a producer–consumer system, where one process generates work that another process can handle at a later point. We make use of the action p, which produces an additional resource to be handled, and c, which consumes a resource. Let $\mathbf{Act} = \{p, c\}$, $\mathbf{Res} = \mathbb{N}$, the empty resource be 0, $\mathbf{R}$ be the least set such that, for all $m, n \in \mathbb{N}$, $(m \oplus m) \otimes (n \oplus n) \in \mathbf{R}$ and the constraints in Definition 2 hold. Let the modification function be the least such that

$$\mu(p,n) = n+1 \qquad \mu(c, n+1) = n,$$

and the constraints in Definition 5 hold. We make use of the redistribution function

$$\delta(z) = (0 \oplus 0) \otimes (n' \oplus n') \quad \text{if } z = m \otimes n \text{ and } n' = m + n.$$

We define the process of the system as

$$E = \text{fix}\, X.(((1+p) \times (1+c)) :_\delta X).$$

The process $(1+p) \times (1+c)$ consists of two subprocesses. The first can either choose to produce a resource or to tick. The second can either choose to consume a resource, if available, or to tick. When combined with the bunched resource $(0 \oplus 0) \otimes (2 \oplus 2)$, the subprocesses can both perform either action. When combined with the bunched resource $(0 \oplus 0) \otimes (0 \oplus 0)$, the first can either produce or tick, while the latter can only tick. The first evolution of the system produces a work package:

$$\dfrac{\dfrac{\dfrac{\dfrac{\mu(p,0)=1}{0, p \xrightarrow{p} 1, \mathbf{0}}\,\text{Act}}{0 \oplus 0, 1+p \xrightarrow{p} 1, \mathbf{0}}\,\text{Sum}_2 \qquad \dfrac{\dfrac{\mu(1,0)=0}{0, 1 \xrightarrow{1} 0, \mathbf{0}}\,\text{Act}}{0 \oplus 0, 1+c \xrightarrow{1} 0, \mathbf{0}}\,\text{Sum}_1}{(0 \oplus 0) \otimes (0 \oplus 0), (1+p) \times (1+c) \xrightarrow{p \cdot 1} 1 \otimes 0, \mathbf{0} \times \mathbf{0}}\,\text{Prod}}{(0 \oplus 0) \otimes (0 \oplus 0), (1+p) \times (1+c) :_\delta E \xrightarrow{p \cdot 1} 1 \otimes 0, (\mathbf{0} \times \mathbf{0}) :_\delta E}\,\text{PrefixOne}.$$

Following this, the work package is transferred to the consumer, which can consume it:

$$\dfrac{1 \otimes 0, \mathbf{0} \times \mathbf{0} \nrightarrow \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mu(1,0)=0}{0, 1 \xrightarrow{1} 0, \mathbf{0}}\,\text{Act}}{0 \oplus 0, 1+p \xrightarrow{1} 0, \mathbf{0}}\,\text{Sum}_1 \qquad \dfrac{\dfrac{\mu(c,1)=0}{1, c \xrightarrow{c} 0, \mathbf{0}}\,\text{Act}}{1 \oplus 1, 1+c \xrightarrow{c} 0, \mathbf{0}}\,\text{Sum}_2}{(0 \oplus 0) \otimes (1 \oplus 1), (1+p) \times (1+c) \xrightarrow{1 \cdot c} 0 \otimes 0, \mathbf{0} \times \mathbf{0}}\,\text{Prod}}{(0 \oplus 0) \otimes (1 \oplus 1), ((1+p) \times (1+c)) :_\delta E \xrightarrow{1 \cdot c} 0 \otimes 0, (\mathbf{0} \times \mathbf{0}) :_\delta E}\,\text{PrefixOne}}{\delta(1 \otimes 0), E \xrightarrow{1 \cdot c} 0 \otimes 0, (\mathbf{0} \times \mathbf{0}) :_\delta E}\,\text{Rec}}}{1 \otimes 0, (\mathbf{0} \times \mathbf{0}) :_\delta E \xrightarrow{1 \cdot c} 0 \otimes 0, (\mathbf{0} \times \mathbf{0}) :_\delta E}\,\text{PrefixOne}.$$

The following evolutions are possible, demonstrating the asynchronous nature of the transfer.

$$\begin{aligned}
(0 \oplus 0) \otimes (0 \oplus 0), ((1+p) \times (1+c)) :_\delta E &\xrightarrow{p \cdot 1} 1 \otimes 0, (\mathbf{0} \times \mathbf{0}) :_\delta E \\
&\xrightarrow{p \cdot 1} 1 \otimes 1, (\mathbf{0} \times \mathbf{0}) :_\delta E \\
&\xrightarrow{1 \cdot 1} 0 \otimes 2, (\mathbf{0} \times \mathbf{0}) :_\delta E \\
&\xrightarrow{1 \cdot c} 0 \otimes 1, (\mathbf{0} \times \mathbf{0}) :_\delta E \\
&\xrightarrow{p \cdot c} 1 \otimes 0, (\mathbf{0} \times \mathbf{0}) :_\delta E \\
&\;\ldots \qquad\qquad\qquad\qquad . \quad \square
\end{aligned}$$

Example 20 *(Generalized hiding).* Previous work on hiding simply composes (multiplicatively) additional resources onto the existing resources. In our work, however, the reduction of a resource–process pair depends critically on both the structure of the bunched resource and of the process. Two bisimilar states may have vastly different internal structures to their resource components, so composing additional resources onto each in the same way is unlikely to respect said resource bunched structure. This example demonstrates why we need a generalized hiding bijection.

Let $\mathbf{Act} = \{a, b, c\}$, $\mathbf{Res} = \{e, s\}$, the empty resource be e,

$$\begin{array}{ll}
R = (e \oplus s) \otimes e & R' = (s \oplus s) \otimes e \\
S = (e \otimes e) \oplus (s \otimes e) & S' = (s \otimes e) \oplus (s \otimes e) \\
T = e \otimes e & T' = s \otimes e,
\end{array}$$

and let $\mathbf{R}$ be the least set such that $R, R', S, S', R, T' \in \mathbf{R}$ and the constraints in Definition 2 hold. Let the modification function be the least such that

$$\mu(a,s) = s \qquad \mu(b,s) = s \qquad \mu(c,e) = e,$$

and the constraints in Definition 5 hold.

We define two processes, which exemplify the notion of distributivity.

$$E = (a+b) \times c \qquad F = (a \times c) \ + \ (b \times c)$$

We also define a hiding function.

$$\begin{array}{lll} h(R) = R' & h(S) = S' & h(T) = T' \\ h(R') = R & h(S') = S & h(T') = T \end{array}$$

$$h(z) = z \quad \text{otherwise}$$

Note that h is a bijection. Here, the states R, E and S, F can perform the action $b \cdot c$, but not the action $a \cdot c$. The states R', E and S', F can, however, perform the action $a \cdot c$. The hiding function h transforms R to R', S to S', and T to T'. The state $R, \nu h.E$ then reduces as

$$\dfrac{\dfrac{\dfrac{\dfrac{\mu(a,s)=s}{s, a \xrightarrow{a} s, \mathbf{0}}\,\textsc{Act}}{s \oplus s, a+b \xrightarrow{a} s, \mathbf{0}}\,\textsc{Sum}_1 \quad \dfrac{\mu(c,e)=e}{e, c \xrightarrow{c} e, \mathbf{0}}\,\textsc{Act}}{h(R), (a+b) \times c \xrightarrow{a \cdot c} h(T), \mathbf{0} \times \mathbf{0}}\,\textsc{Prod}}{R, \nu h.E \xrightarrow{\nu h.(a \cdot c)} T, \nu h.(\mathbf{0} \times \mathbf{0})}\,\textsc{Hide},$$

where $h(R) = (s \oplus s) \otimes e$ and $h(T) = s \otimes e$, and the state $S, \nu h.F$ reduces as

$$\dfrac{\dfrac{\dfrac{\dfrac{\mu(a,s)=s}{s, a \xrightarrow{a} s, \mathbf{0}}\,\textsc{Act} \quad \dfrac{\mu(c,e)=e}{e, c \xrightarrow{c} e, \mathbf{0}}\,\textsc{Act}}{s \otimes e, a \times c \xrightarrow{a \cdot c} s \otimes e, \mathbf{0} \times \mathbf{0}}\,\textsc{Prod}}{h(S), (a \times c) \ + \ (b \times c) \xrightarrow{a \cdot c} h(T), \mathbf{0} \times \mathbf{0}}\,\textsc{Sum}_1}{S, \nu h.F \xrightarrow{\nu h.(a \cdot c)} T, \nu h.(\mathbf{0} \times \mathbf{0})}\,\textsc{Hide},$$

where $h(S) = (s \otimes e) \oplus (s \otimes e)$. Here, in order for the processes E and F to perform the action $a \cdot c$, they need the relevant resources in different parts of their resource bunches. Simply adding additional resources, either using $\otimes$-bunching or $\oplus$-bunching, would not respect the structure required for either process.

While the states R', E and S', F can perform the action $a \cdot c$, by the Hide rule, the states $R, \nu h.E$ and $S, \nu h.F$, which use the hiding function h to obtain additional resources, perform the $\nu h.(a \cdot c)$ action. Intuitively, $\nu h.(a \cdot c)$ denotes the largest action which can be performed on the non-hidden resources.

In order to determine the result of applying the function νh to action $a \cdot c$, recall Definition 13. There are four sub-actions of $a \cdot c$, namely $1 \cdot 1$, $a \cdot 1$, $1 \cdot c$, and $a \cdot c$. The set $A_{a \cdot c, h}$ comprises the sub-actions x of $a \cdot c$ such that, for all resources U, $\mu(a \cdot c, h(U)) = h(V)$ implies that $\mu(x, U) = V$. The only U such that $\mu(a \cdot c, h(U))$ is defined is $U = T$. In that case, $h(T) = T'$, $\mu(a \cdot c, h(U)) = h(T)$, and we can tabulate the definition of $\mu(x, U)$, and whether the above implication holds.

x	$\mu(x,U)$	$\mu(a \cdot c, h(U)) = h(V)$ implies $\mu(x,U) = V$
$1 \cdot 1$	T	$\top$
$a \cdot 1$	$\uparrow$	$\bot$
$1 \cdot c$	T	$\top$
$a \cdot c$	$\uparrow$	$\bot$

Here we have that $A_{a \cdot c, h} = \{1 \cdot 1, 1 \cdot c\}$, the supremum of which, under action inclusion, is $1 \cdot c$. This fits with our intuition of hiding actions that are only defined with the additional resources; action a requires the additional resource r, but c is defined on resource e. $\square$

Example 21 *(Weak memory consistency).* In distributed systems, it is costly (in terms of time and communication) to ensure that all updates to shared variables are propagated to all the nodes atomically. A weaker requirement is weak memory consistency; that is, if no further updates are made to the shared variable, then at some point in the future, all nodes will see only the most recent value. In this example, we define a system where any value assigned by any node to a shared variable can be read, up to a synchronization point, after which only the most recent value can be read. For simplicity, we assume a shared, discrete clock, to order the writes: we see no reason why a more complex system such as vector clocks [26] could not be used to replace the shared clock. Such a system is of a decidedly distributed systems character, as opposed to the single-chip concurrency of weak memory models.

We consider a single shared variable. Let $\tau \mapsto v$ denote the value $v \in \mathbb{Z}$ assigned to that variable, at time $\tau \in \mathbb{N}$. Let **Res** be the set of all triples $(\tau, \tau' \mapsto v', T)$, where τ denotes the current time in the state, $\tau' \mapsto v'$ denotes the assignment to be read, and T denotes the set of all (non-flushed) assignments seen by a single process in the course of an execution. The bunch $(\tau, \tau' \mapsto v', T) \oplus (\tau, \tau'' \mapsto v'', T)$ denotes two writes to the shared variable that are visible to a given process. When some process attempts to read the value of the shared variable, it will see either the value v' or the value v''. Note that the value τ and the set T are the same in both cases. Let **R** be the least set such that, for all τ, $\tau_1 \mapsto v_1$, …, $\tau_m \mapsto v_m$, $\tau_{m+1} \mapsto v_{m+1}$, …, $\tau_n \mapsto v_n$, T, T',

$$(\Sigma\{(\tau, \tau_i \mapsto v_i, T) \mid i \in 1 \ldots m\})) \otimes (\Sigma\{(\tau, \tau_j \mapsto v_j, T') \mid j \in (m+1) \ldots n\}) \in \mathbf{R},$$

and the constraints in Definition 2 hold. Given such a bunch, a left hand process would have m visible writes to the shared variable, $\tau_1 \mapsto v_1, \ldots, \tau_m \mapsto v_m$, whereas a right hand process have $n-m$ visible writes, $\tau'_{m+1} \mapsto v'_{m+1}, \ldots, \tau'_n \mapsto v'_n$. Note that they must agree on the clock value τ, but can have different values for T and T'. Unless explicitly distributed, the writes performed by each process are not visible to the other.

Let $\mathbf{Act} = \{w1, w2, w3, r1, r2, r3\}$ consist of actions that write and read the values 1, 2, and 3, to and from the shared variable, respectively. Let the modification function be the least such that

$$\begin{aligned}
\mu(rv, (\tau, \tau' \mapsto v, T)) &= \Sigma\{(\tau+1, \tau_i \mapsto v_i, T) && \mid \tau_i \mapsto v_i \in T\} \\
\mu(wv, (\tau, \tau' \mapsto v', T)) &= \Sigma\{(\tau+1, \tau_i \mapsto v_i, T \cup \{\tau \mapsto v\}) && \mid \tau_i \mapsto v_i \in T \cup \{\tau \mapsto v\}\},
\end{aligned}$$

and the constraints in Definition 5 hold. Note that the modification function for action rv and resource $(\tau, \tau' \mapsto v', T))$ is only defined when $v = v'$.

We make use of the distribution functions δ and δ'. The first, δ, copies all the possible writes to both sides of a $\otimes$-bunch.

$$\begin{aligned}
&\delta((\Sigma\{(\tau, \tau_i \mapsto v_i, T) \mid i \in 1 \ldots m\}) \otimes (\Sigma\{(\tau, \tau_j \mapsto v_j, T') \mid j \in (m+1) \ldots n\})) \\
&\quad = (\Sigma\{(\tau, \tau_k \mapsto v_k, T \cup T') \mid k \in 1 \ldots n\}) \otimes (\Sigma\{(\tau, \tau_k \mapsto v_k, T \cup T') \mid k \in 1 \ldots n\})
\end{aligned}$$

The second, δ', removes all writes except the most recent (in the case of a tie for the most recent, we take the smallest value).

$$\begin{aligned}
&\delta'((\Sigma\{(\tau, \tau_i \mapsto v_i, T) \mid i \in 1 \ldots m\}) \otimes (\Sigma\{(\tau, \tau_j \mapsto v_j, T') \mid j \in (m+1) \ldots n\}) \\
&\quad = (\tau, \tau_k \mapsto v_k, \{\tau_k \mapsto v_k\}) \otimes (\tau, \tau_k \mapsto v_k, \{\tau_k \mapsto v_k\}),
\end{aligned}$$

where $k \in 1, \ldots, n$, and for all $l \in 1 \ldots n$, $\tau_l \leq \tau_k$ and $\tau_l = \tau_k$ implies $v_k \leq v_l$.

We define the processes of the system as

$$E = E_1 :_\delta E_2 :_{\delta'} E_3 \qquad E_1 = (w1 \times w2) \qquad E_2 = (r_0 + (r1 + r2)) \times w3 \qquad E_3 = r_3 \times r_3.$$

The process system consists of two parallel components, at each stage of a sequential process. In the first stage, the components each write different values to the shared variable. In the second stage, after the writes are distributed one to the other, the first process attempts to read a value, while the second process writes a new value. In the third stage, after all previous writes have been synchronized, each attempts to read a value.

Let $r = (1, 0 \mapsto 0, \{0 \mapsto 0\})$, which denotes that, at starting time 1, the variable only has one possible value, 0, and this value was assigned at time point 0, before be beginning of the execution. Furthermore, let $R_1 = r \otimes r$, $T_2^1 = \{0 \mapsto 0, 1 \mapsto 1\}$, $T_2^2 = \{0 \mapsto 0, 1 \mapsto 2\}$, and

$$R'_1 = ((2, 0 \mapsto 0, T_2^1) \oplus (2, 1 \mapsto 1, T_2^1)) \otimes ((2, 0 \mapsto 0, T_2^2) \oplus (2, 1 \mapsto 2, T_2^2)).$$

In the first evolution of the system, both processes write different values to the shared variable. They end up with different possible views of what values the shared variable can take. The first can see the initial write, and its write of the value 1. The second can see the initial write, and its write of the value 2.

$$\dfrac{\dfrac{\mu(w1, r) = (2, 0 \mapsto 0, T_2^1) \oplus (2, 1 \mapsto 1, T_2^1)}{r, w1 \xrightarrow{w1} (2, 0 \mapsto 0, T_2^1) \oplus (2, 1 \mapsto 1, T_2^1), \mathbf{0}}\,\textsc{Act} \qquad \dfrac{\mu(w2, r) = (2, 0 \mapsto 0, T_2^2) \oplus (2, 1 \mapsto 2, T_2^2)}{r, w2 \xrightarrow{w2} (2, 0 \mapsto 0, T_2^2) \oplus (2, 1 \mapsto 2, T_2^2), \mathbf{0}}\,\textsc{Act}}{\dfrac{r \otimes r, w1 \times w_2 \xrightarrow{w1 \cdot w2} ((2, 0 \mapsto 0, T_2^1) \oplus (2, 1 \mapsto 1, T_2^1)) \otimes ((2, 0 \mapsto 0, T_2^2) \oplus (2, 1 \mapsto 2, T_2^2)), \mathbf{0} \times \mathbf{0}}{R_1, E \xrightarrow{w1 \cdot w2} R'_1, (\mathbf{0} \times \mathbf{0}) :_\delta E_2 :_{\delta'} E_3}}$$

In the second evolution of the system, the redistribution function δ is applied, copying all of the writes to both sides of the concurrent composition. Let $T_2 = T_2^1 \cup T_2^2$, $T_3 = T_2 \cup \{2 \mapsto 3\}$, and

$$\begin{aligned}
S_2 &= ((2,0\mapsto 0,T_2)\oplus((2,1\mapsto 1,T_2)\oplus(2,1\mapsto 2,T_2)))\\
S'_2 &= ((3,0\mapsto 0,T_2)\oplus((3,1\mapsto 1,T_2)\oplus(3,1\mapsto 2,T_2)))\\
S''_2 &= ((3,0\mapsto 0,T_3)\oplus((3,1\mapsto 1,T_3)\oplus((3,1\mapsto 2,T_3)\oplus(3,2\mapsto 3,T_3)))),
\end{aligned}$$

and $R_2 = S_2 \otimes S_2$. Note that $\delta(R'_1) = R_2$. Following the redistribution, the first process reads a value, and the second writes an additional value, 3.

$$\dfrac{R'_1, (\mathbf{0}\times\mathbf{0}) \nrightarrow \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mu(r1,S_2)=S'_2}{(2,1\mapsto 1,T_2), r_1 \xrightarrow{r1} S'_2, \mathbf{0}}\ \textsc{Act}}{(2,1\mapsto 1,T_2)\oplus(2,1\mapsto 2,T_2), r_1+r_2 \xrightarrow{r1} S'_2, \mathbf{0}}\ \textsc{Sum}_1}{S_2, (r_0+(r1+r2)) \xrightarrow{r_1} S'_2, \mathbf{0}}\ \textsc{Sum}_2 \quad \dfrac{\mu(w3,S_2)=S''_2}{S_2, w3 \xrightarrow{w3} S''_2, \mathbf{0}}\ \textsc{Act}}{R_2, (r_0+(r1+r2))\times w3 \xrightarrow{r1\cdot w3} S'_2\otimes S''_2, \mathbf{0}\times\mathbf{0}}\ \textsc{Prod}}{\delta(R'_1), E_2 :_{\delta'} E_3 \xrightarrow{r1\cdot w3} S'_2\otimes S''_2, (\mathbf{0}\times\mathbf{0}) :_{\delta'} E_3}\ \textsc{PrefixOne}}{R'_1, (\mathbf{0}\times\mathbf{0}) :_{\delta} E_2 :_{\delta'} E_3 \xrightarrow{r1\cdot w3} S'_2\otimes S''_2, (\mathbf{0}\times\mathbf{0}) :_{\delta'} E_3}\ \textsc{PrefixTwo}.$$

In the third evolution of the system, the redistribution function δ' is applied, which removes all assignments except for the most recent, which is copied to both sides of the $\otimes$-bunch. Let $R^1_3 = R^2_3(3, 2\mapsto 3, \{2\mapsto 3\})$ Note that $\delta'(S'_2\otimes S''_2) = R^1_3\otimes R^1_3$.

$$\dfrac{S'_2\otimes S''_2, \mathbf{0}\times\mathbf{0} \nrightarrow \quad \dfrac{\dfrac{}{R^1_3, r3 \xrightarrow{r3} (4,2\mapsto 3,\{2\mapsto 3\}), \mathbf{0}}\ \textsc{Act} \quad \dfrac{}{R^2_3, r3 \xrightarrow{r3} (4,2\mapsto 3,\{2\mapsto 3\}), \mathbf{0}}\ \textsc{Act}}{\delta'(S'_2\otimes S''_2), E_3 \xrightarrow{r3\cdot r3} (4,2\mapsto 3,\{2\mapsto 3\})\otimes(4,2\mapsto 3,\{2\mapsto 3\}), \mathbf{0}\times\mathbf{0}}\ \textsc{Prod}}{S'_2\otimes S''_2, (\mathbf{0}\times\mathbf{0}) :_{\delta'} E_3 \xrightarrow{r3\cdot r3} (4,2\mapsto 3,\{2\mapsto 3\})\otimes(4,2\mapsto 3,\{2\mapsto 3\}), \mathbf{0}\times\mathbf{0}}\ \textsc{PrefixTwo}.$$

Note that the only write to the shared variable left in the state is that of the value 3. □

The standard notion of bisimulation is that two states in a system are bisimilar if they can perform the same actions, and, after those reductions, remain bisimilar. Let R and S be resources, and E and F be agents. Then we have the following:

Definition 22 *(Bisimulation).* A relation $\mathcal{R}$ is a bisimulation relation if, for all closed states $(R, E)\mathcal{R}(S, F)$, then

- if $R, E \xrightarrow{a} R', E'$, then there exist S', F', such that $S, F \xrightarrow{a} S', F'$, and $(R', E')\mathcal{R}(S', F')$, and
- if $S, F \xrightarrow{a} S', F'$, then there exist R', E', such that $R, E \xrightarrow{a} R', E'$, and $(R', E')\mathcal{R}(S', F')$. □

Let $\sim \subseteq \mathbf{CState}\times\mathbf{CState}$ be the union of all bisimulations. The union of any two bisimulations is also a bisimulation. Hence $\sim$ is well defined, and a bisimulation.

Note that the bisimulation relation is defined on closed states. While it is possible to define an operational semantics for open states, an appropriate notion of bisimulation for open states in a calculus with bunched resources is an open problem.

There are various well-formedness conditions that we use in the remainder of the paper. Note that all of the following results are for a fixed $(\mathbf{Act}, \mathbf{Res}, \mathbf{R}, \mu, \Delta, \mathbf{H})$ structure.

Definition 23 *(Image-finite).* A state R, E is image-finite if it has finitely many derivatives. □

From this point onwards, all states are assumed to be image-finite.

Consider two states R, E and S, F. In order to perform a sequential composition between these states, we need to take account of how the resource bunches relate to each other. Consider some state $R, E :_{\delta} F$, and recall the PrefixTwo rule. When the prefixed process E can no longer be reduced with the accompanying resource bunch R, the resources are transformed by the application of the redistribution function δ, and the postfixed process F is reduced alongside the transformed resource bunch $\delta(R)$. In order to be able to sensibly consider the prefixing of the state R, E onto the state S, F (rather than onto the *process* F), with respect to a redistribution function δ, we must ensure, for all states R', E' to which R, E can reduce, that, if the new state cannot reduce, then the reordered resource bunch $\delta(R')$ is equal to S. Hence, whenever the state R, E reduces to some state R', E' that cannot reduce, the sequential composition $R', E' :_{\delta} F$ then behaves as S, F.

Definition 24. A state R, E and a bunched resource S are δ-compatible if, for all transition sequences $R, E \rightarrow^* R', E'$, we have that $R', E' \nrightarrow$ implies $\delta(R') = S$. □

The notion of prefixing of a state R, E onto a state S, F is a technical one, used mostly in Theorem 27. From this point onwards, for any sequential composition of states R, E and S, F according to the redistribution function δ, $R, E :_\delta F$, we assume that R, E and S are δ-sequence compatible.

Consider two pairs of bisimilar states, $R_1, E_1 \sim S_1, F_1$ and $R_2, E_2 \sim S_2, F_2$. In order to perform a concurrent composition between these states, it is necessary to take account of the partiality of the resource model. It may be possible that $R_1 \otimes R_2$ is defined, but that $S_1 \otimes S_2$ is not defined. Were that to be the case, we would not have congruence of concurrent composition with respect to bisimulation.

Example 25. Let $\mathbf{Res} = \{r, s, t, e\}$, with empty resource e, and $\mathbf{R}$ be the smallest resource model such that $r \otimes t \in \mathbf{R}$ and the constraints in Definition 2 hold. Let $\mathbf{Act} = \{a, b\}$, and μ be the smallest modification function μ such that

$$\mu(a, r) = r \qquad \mu(a, s) = s \qquad \mu(b, t) = t,$$

and the constraints in Definition 5 hold. Consider states (r, a), (s, a), and (t, b). We obviously have that $r, a \sim s, a$. But, we then have that

$$r \otimes t, a \times b \xrightarrow{a \cdot b} r \otimes t, \mathbf{0} \times \mathbf{0} \qquad s \otimes t, a \times b \overset{a \cdot b}{\not\rightarrow},$$

as $s \otimes t$ is not defined (that is, $s \otimes t \notin \mathbf{R}$). Hence, with this resource model, concurrent composition is not a congruence. $\square$

As a result, we require the following property of our calculi.

Definition 26 *($\sim$-Resource-closed CBRP).* A calculus is $\sim$-resource-closed if, for all R_1, E_1, S_1, F_1, R_2, E_2, S_2, F_2, if $R_1, E_1 \sim S_1, F_1$ and $R_2, E_2 \sim S_2, F_2$, then $R_1 \otimes R_2$ (respectively, $R_1 \oplus R_2$) is defined if and only if $S_1 \otimes S_2$ (respectively, $S_1 \oplus S_2$) is defined. $\square$

For the remainder of this section, all calculi are assumed to be $\sim$-resource-closed.

We can obtain a useful property for reasoning compositionally: that bisimulation is a congruence; that is, it is an equivalence relation that is respected by the state constructors (excepting the fixed point constructor).

Theorem 27 *(Bisimulation congruence). The relation $\sim$ is a congruence for concurrent, non-deterministic, and sequential composition, and hiding: for all closed states (R_i, E_i), (S_i, F_i), (R, E), (S, F), distribution functions $\delta, \delta' \in \Delta$, and hiding functions $h \in \mathbf{H}$, if $R_i, E_i \sim S_i, F_i$, $h(R), E \sim h(S), F$, R_1, E_1, and R_2 are δ-sequence compatible, S_1, F_1, S_2 are δ'-sequence compatible, and $R_1 \otimes R_2$, $S_1 \otimes S_2$, $R_1 \oplus R_2$, and $S_1 \oplus S_2$ are defined, then,*

$$R_1 \otimes R_2, E_1 \otimes E_2 \sim S_1 \otimes S_2, F_1 \otimes F_2 \quad R_1 \oplus R_2, E_1 + E_2 \sim S_1 \oplus S_2, F_1 + F_2$$

$$R_1, E_1 :_\delta E_2 \sim S_1, F_1 :_{\delta'} F_2 \qquad\qquad R, \nu h.E \sim S, \nu h.E.$$

Proof. The bisimulation relation $\sim$ is the largest bisimulation relation, and contains all other bisimulation relations. In order to show that the above properties hold, it is sufficient, therefore, to define a relation $\mathcal{R}$, for which the required properties hold, and to show that the relation $\mathcal{R}$ is a bisimulation.

A congruence is reflexive, symmetric, transitive, and preserved under the above constructions. The reflexivity property follows immediately. Consider the symmetricity property. Let

$$\mathcal{R} = \{((S, F), (R, E)) \mid R, E \sim S, F\}.$$

The relation $\mathcal{R}$ is a bisimulation if and only if, for all $(S, F)\mathcal{R}(R, E)$, then the following holds: if $S, F \xrightarrow{a} S', F'$, then there exist R', E', such that $R, E \xrightarrow{a} R', E'$, and $(S', F')\mathcal{R}(R', E')$; and, if $R, E \xrightarrow{a} R', E'$, then there exist S', F', such that $S, F \xrightarrow{a} S', F'$, and $(S', F')\mathcal{R}(R', E')$.

Consider the case in which $S, F \xrightarrow{a} S', F'$. Then, by Definition 22, we have that there exist R', E', such that $R, E \xrightarrow{a} R', E'$, and $(R', E') \sim (S', F')$. Hence we have that $(S', F')\mathcal{R}(R', E')$.

Consider the case in which $R, E \xrightarrow{a} R', E'$. Then, by Definition 22, we have that there exist S', F', such that $S, F \xrightarrow{a} S', F'$, and $(R', E') \sim (S', F')$. Hence we have that $(S', F')\mathcal{R}(R', E')$.

The transitivity property follows similarly to the symmetricity property.

(1) Consider concurrent composition. Let

$$\mathcal{R} = \{((R_1 \otimes R_2, E_1 \times E_2), (S_1 \otimes S_2, F_1 \times F_2)) \mid R_1, E_1 \sim S_1, F_1, R_2, E_2 \sim S_2, F_2, \\ \text{and } R_1 \otimes R_2 \text{ and } S_1 \otimes S_2 \text{ are defined}\}.$$

If $R_1 \otimes R_2, E_1 \times E_2 \xrightarrow{a} R'_1 \otimes R'_2, E'_1 \times E'_2$, then the last rule used in the derivation of this reduction must be the PROD rule. By the PROD rule, we have that $R_1, E_1 \xrightarrow{a_1} R'_1, E'_1$, $R_2, E_2 \xrightarrow{a_2} R'_2, E'_2$, $a = a_1 \cdot a_2$, and $R'_1 \otimes R'_2$ is defined. By

Definition 22, we have that there exist S_1', F_1', S_2', F_2', such that $S_1, F_1 \xrightarrow{a_1} S_1', F_1'$, $S_2, F_2 \xrightarrow{a_2} S_2', F_2'$, $(R_1', E_1') \sim (S_1', F_1')$, and $(R_2', E_2') \sim (S_2', F_2')$. As $R_1' \otimes R_2'$ is defined, by Definition 26, we have that $S_1' \otimes S_2'$ is defined. By the Prod rule, we have that $S_1 \otimes S_2, F_1 \times F_2 \xrightarrow{a_1 \cdot a_2} S_1' \otimes S_2', F_1' \times F_2'$. As $(R_1', E_1') \sim (S_1', F_1')$ and $(R_2', E_2') \sim (S_2', F_2')$, we have that $(R_1' \otimes R_2', E_1' \times E_2')\mathcal{R}(S_1' \otimes S_2', F_1' \times F_2')$.

The other case is similar. Hence $\mathcal{R}$ is closed and a bisimulation.

(2) Consider non-deterministic choice. Let

$$\mathcal{R} = \{((R_1 \oplus R_2\,,\, E_1 + E_2), (S_1 \oplus S_2\,,\, F_1 + F_2)) \mid R_1, E_1 \sim S_1, F_1 \text{ and } R_2, E_2 \sim S_2, F_2\} \cup \sim .$$

If $R_1 \oplus R_2, E_1 + E_2 \xrightarrow{a} R', E'$, then the last rule used in the derivation of this reduction must be the Sum rule. By the Sum rule, we have that, for some $i \in \{1, 2\}$, $R_i, E_i \xrightarrow{a} R_i', E_i'$, $R', E' = R_i', E_i'$, and $R_1 \oplus R_2$ is defined. By Definition 22, we have that there exist S_i', F_i', such that $S_i, F_i \xrightarrow{a} S_i', F_i'$ and $(R_i', E_i') \sim (S_i', F_i')$. By the Sum rule, we have that $S_1 \oplus S_2, F_1 + F_2 \xrightarrow{a} S_i', F_i'$. As $(R_i', E_i') \sim (S_i', F_i')$, we have that $(R_i', E_i')\mathcal{R}(S_i', F_i')$.

The other case is similar. Hence $\mathcal{R}$ is closed and a bisimulation.

(3) Consider sequential composition. Let

$$\begin{aligned}\mathcal{R} = \{((R_1, E_1 :_\delta E_2), (S_1, F_1 :_{\delta'} F_2)) \mid\ & R_1, E_1 \sim S_1, F_1,\ R_2, E_2 \sim S_2, F_2,\ R_1, E_1, R_2 \text{ are } \delta\text{-sequence compatible,}\\ & S_1, F_1, S_2 \text{ are } \delta'\text{-sequence compatible, and } \delta, \delta' \in \Delta\\ \} \cup \sim .\end{aligned}$$

If $R_1, E_1 :_\delta E_2 \xrightarrow{a} R', E'$, then there are two possibilities for the last rule that is used in the derivation of this reduction.

First, suppose that PrefixOne is the last rule used. Then, we have that $R_1, E_1 \xrightarrow{a} R_1', E_1'$ and $\delta \in \Delta$. By Definition 22, we have that there exist S_1', F_1', such that $S_1, F_1 \xrightarrow{a} S_1', F_1'$ and $(R_1', E_1') \sim (S_1', F_1')$. By Definition 24, we have that R_1', E_1', and R_2 are δ-sequence compatible, and that S_1', F_1', S_2 are δ'-sequence compatible. By the PrefixOne rule, we have that $S_1, F_1 :_{\delta'} F_2 \xrightarrow{a} S_1', F_1' :_{\delta'} F_2$. We then have that $(R_1', E_1' :_\delta E_2)\mathcal{R}(S_1', F_1' :_{\delta'} F_2)$.

Second, suppose that PrefixTwo is the last rule used. Then, we have that $R_1, E_1 \nrightarrow$ and $\delta(R_1), E_2 \xrightarrow{a} R_2', E_2'$. By Definition 22, we have that $S_1, F_1 \nrightarrow$. By Definition 24, we have that $\delta(R_1) = R_2$ and $\delta'(S_1) = S_2$. By Definition 22, we have that there exist S_2', F_2', such that $\delta'(S_1), F_2 \xrightarrow{a} S_2', F_2'$ and $(R_2', E_2') \sim (S_2', F_2')$. By the PrefixTwo rule, we have that $S_1, F_1 :_{\delta'} F_2 \xrightarrow{a} S_2', F_2'$. As $(R_2', E_2') \sim (S_2', F_2')$, we have that $(R_2', E_2')\mathcal{R}(S_2', F_2')$.

The other case is similar. Hence $\mathcal{R}$ is closed and a bisimulation.

(4) Consider the hiding operator. Let

$$\mathcal{R} = \{((R, \nu h.E), (S, \nu h.F)) \mid h(R), E \sim h(S), F\}.$$

If $R, \nu h.E \xrightarrow{a} R', E'$, then the last rule used in the derivation of this reduction must be the Hide rule. Then, we have that $h(R), E \xrightarrow{b} R'', E''$, $R'' = h(R')$, $E' = \nu h.E''$, and $a = \nu h.b$. By Definition 22, we have that there exist S'', F'', such that $h(S), F \xrightarrow{b} S'', F''$ and $R'', E'' \sim S'', F''$. By Definition 10, we have that there exists some S' such that $S'' = h(S')$. By the Hide rule, we have that $S, \nu h.F \xrightarrow{a} S', \nu h.F''$. As $h(R'), E'' \sim h(S'), F''$, we have that $(R', \nu h.E'')\mathcal{R}(S', \nu h.F'')$.

The other case is similar. Hence $\mathcal{R}$ is closed and a bisimulation. □

As non-determinism 'forgets' the resources that are associated with the choice which is not taken, there is not a functional relationship between the resource bunches of the states related by a transition relation $\xrightarrow{a}$. There is, however, a functional relationship between *some* resource bunch and the resource bunch of its reduced state. This result can be used to prove an expansion result for our calculus (Lemma 30).

Lemma 28. *If $R, E \xrightarrow{a} R', E'$, then there exists S such that $\mu(a, S) = R'$.*

Proof. By induction over the derivation of $R, E \xrightarrow{a} R', E'$. Consider which rule is the last used in the derivation of the transition.

Case Act. By the Act rule, we have that $\mu(a, R) = R'$. Let $S = R$, and we are done with this case.

Case Sum. By the Sum rule, we have that $R_i, E_i \xrightarrow{a} R', E'$, for some $i \in \{1, 2\}$. By the induction hypothesis, we have that there exists some S such that $\mu(a, S) = R'$.

Case Prod. By the Prod rule, we have that $R_1, E_1 \xrightarrow{a_1} R_1', E_1'$, $R_2, E_2 \xrightarrow{a_2} R_2', E_2'$, $a = a_1 \cdot a_2$, $R = R_1 \otimes R_2$, and $R' = R_1' \times R_2'$. By the induction hypothesis, there exist S_1 and S_2 such that $\mu(a_1, S_1) = R_1'$ and $\mu(a_2, S_2) = R_2'$. By Definition 5, we have that $S_1 \otimes S_2 \in \mathbf{R}$, and hence that $\mu(a_1 \cdot a_2, S_1 \otimes S_2) = R_1' \otimes R_2'$.

Case PrefixOne. By the PrefixOne rule, we have that $E = E_1 :_\delta E_2$, $R_1, E_1 \xrightarrow{a} R', E_1'$, and $E' = E_1' :_\delta E_2$. By the induction hypothesis, we have that there exists some S such that $\mu(a, S) = R'$.

Case PrefixTwo. By the PrefixTwo rule, we have that $\delta(R), F \xrightarrow{a} R', F'$. By the induction hypothesis, we have that there exists some S such that $\mu(a, S) = R'$.

Case Hide. By the Hide rule, we have that $E = \nu h.F$, $h(R), F \xrightarrow{b} R'', F'$, $a = \nu h.b$, $R'' = h(R')$, and $E' = \nu h.F'$. By the induction hypothesis, we have that there exists some S' such that $\mu(b, S') = h(R')$. By Definition 10, we have that there exists some S such that $h(S) = S'$. By Definition 13, as $\mu(b, h(S)) = h(R')$, we have that $\mu(a, S) = R'$.

Case Rec. By the Rec rule, we have that $R, E[\text{fix } X.E/X] \xrightarrow{a} R', E'$. By the induction hypothesis, we have that there exists some S such that $\mu(a, S) = R'$. □

In order to simplify our examples that include sequential composition, it is helpful to obtain the following lemma. This describes how a sequential composition, which is annotated by the function δ, behaves, when its prefix is bisimilar to the zero process.

Lemma 29 *(Zero prefix). If $R, E \sim R, \mathbf{0}$, then $R, E :_\delta F \sim \delta(R), F$.*

Proof. Let

$$\mathcal{R} = \{((\delta(R), F), (R, E :_\delta F)) \mid R, E \sim R, \mathbf{0}\} \cup \sim .$$

Consider the case where $\delta(R), F \xrightarrow{a} R', F'$. As $R, E \sim R, \mathbf{0}$, then we have that $R, E \not\rightarrow$. By the PrefixTwo rule, as $R, E \not\rightarrow$ and $\delta(R), F \xrightarrow{a} R', F'$, we have that $R, E :_\delta F \xrightarrow{a} R', F'$. By bisimulation is an equivalence relation we have that $R', F' \sim R', F'$, and hence that $(R', E')\mathcal{R}(R', E')$.

If $R, E :_\delta F \xrightarrow{a} R', F'$, then there are two possibilities for the last rule that is used in the derivation of this reduction. If PrefixOne is the last rule used, then, by that rule, we have that $R, \mathbf{0} \xrightarrow{a} R'', E''$ and $R', E' = R'', E'' :_\delta E$. This case is clearly impossible, as there are no transitions for the zero process, irrespective of the accompanying resources. If PrefixTwo is the last rule used, then, by that rule, we have that $\delta(R), F \xrightarrow{a} R', F'$. As bisimulation is an equivalence relation, we have that $R', F' \sim R', F'$, and hence that $(R', F')\mathcal{R}(R', F')$. □

The combinators of most process calculi can be classified as either static or dynamic. This classification comes from the operational semantics of said combinators [28]. Static combinators are those where the combinator is present both before and after the evolution. In our calculus, the static combinators are concurrent product and hiding. Dynamic combinators are those where the combinator is present before, but not after, the evolution. Expansion results describe the behaviour of the static combinators in terms of the dynamic combinators. In order to describe the behaviour of static combinators, we must be able to describe the behaviour of the combinators' subcomponents. The fact that we can do so is expressed by the following lemma:

Lemma 30 *(State expansion). For all closed states R, E, there exist an indexing set I, actions a_i, resources R_i, and agents E'_i such that*

$$R, E \sim \Sigma_I\{R_i, a_i :_{id} E'_i \mid R, E \xrightarrow{a_i} R'_i, E'_i \text{ and } \mu(a_i, R_i) = R'_i\},$$

where $\Sigma_I R_i, E_i$ is syntactic sugar for the state $R_1 \oplus (\ldots \oplus R_n)\ldots), E_1 + (\ldots + E_n)\ldots)$, if $I = \{i\}$, then $\Sigma_I R_i, E_i = R_i, E_i$, and, if $I = \emptyset$, then $\Sigma_I R_i, E_i = e, \mathbf{0}$.

Proof. Let

$$\mathcal{R} = \{((R, E), \Sigma_I(R_i, a_i :_{id} E'_i)) \mid R, E \xrightarrow{a_i} R'_i, E'_i \text{ and } \mu(a_i, R_i) = R'_i\} \cup \sim .$$

Suppose that $R, E \xrightarrow{a} R', E'$. By Lemma 28, we have that there exists S such that $\mu(a, S) = R'$. Then, by the definition of $\Sigma_I\{R_i, a :_{id} E'_i \mid R, E \xrightarrow{a_i} R'_i, E'_i \text{ and } \mu(a_i, R_i) = R'_i\}$, there exist $j \in I$, R_j, a_j, R'_j, E'_j, such that $R_j = S$, $a = a_j$, and $R', E' = R'_j, E'_j$. By the Act rule, we have that $R_j, a_j \xrightarrow{a_j} R'_j, \mathbf{0}$. By the PrefixOne rule, we have that $R_j, a_j :_{id} E'_j \xrightarrow{a_j} R'_j, \mathbf{0} :_{id} E'_j$. By repeated application of the Sum rule, we have that $\Sigma_I(R_i, a_i :_{id} E'_i) \xrightarrow{a_j} R'_j, \mathbf{0} :_{id} E'_j$. By Lemma 29, we have that $R'_j, \mathbf{0} :_{id} E'_j \sim R'_j, E'_j$, and hence that $(R'_j, \mathbf{0} :_{id} E'_j)\mathcal{R}(R'_j, E'_j)$.

Suppose that $\Sigma_I(R_i, a_i :_{id} E'_i) \xrightarrow{a} R', E'$. By repeated application of the Sum rule, we have that there exist $j \in I$, R_j, a_j, R'_j, E''_j such that $R_j, a_j :_{id} E'_j \xrightarrow{a_j} R'_j, E''_j$, $a = a_j$, and $R', E' = R'_j, E''_j$. By the Act and PrefixOne rules, we have that $E''_j = \mathbf{0} :_{id} E'_j$. By Lemma 29, we have that $R'_j, \mathbf{0} :_{id} E'_j \sim R'_j, E'_j$. By the definition of $\Sigma_I(R_i, a_i :_{id} E'_i)$, we have that $R, E \xrightarrow{a_j} R'_j, E'_j$. As $R'_j, \mathbf{0} :_{id} E'_j \sim R'_j, E'_j$, we have that $(R'_j, \mathbf{0} :_{id} E'_j)\mathcal{R}(R'_j, E'_j)$.

Hence $\mathcal{R}$ is closed and a bisimulation. □

Note that in writing this expansion, we assume that all components are defined. If not, then the expansion does not hold.

Given the behaviour of the subcomponents, in terms of dynamic operators, it is possible to describe the behaviour of the concurrent product combinator in terms of dynamic operators.

Proposition 31 *(Product state expansion). If*

$$R, E \sim \Sigma_I\{R_i, a_i :_{id} E'_i \mid R, E \xrightarrow{a_i} R'_i, E'_i \text{ and } \mu(a_i, R_i) = R'_i\}$$
$$S, F \sim \Sigma_J\{S_j, b_j :_{id} F'_j \mid S, F \xrightarrow{b_j} S'_j, F'_j \text{ and } \mu(b_j, S_j) = S'_j\},$$

then

$$R \otimes S, E \times F \sim \Sigma_{I\times J}(R_i \otimes S_j, (a_i \cdot b_j) :_{id} E'_i \times F'_j).$$

Proof. Let

$$\mathcal{R} = \{((R \otimes S, E \times F), (\Sigma_{I\times J}(R_i \otimes S_j, (a_i \cdot b_j) :_{id} E'_i \times F'_j)))\} \cup \sim,$$

where I, R_i, a_i, R'_i, E'_i, J, S_j, b_j, S'_j, and F'_j are defined as above.

Suppose that $R \otimes S, E \times F \xrightarrow{c} T, G$. Then, the last rule used in the derivation must be the Prod rule. By the Prod rule, we have that there exist a, R', E', b, S', F' such that $R, E \xrightarrow{a} R', E'$, $S, F \xrightarrow{b} S', F'$, $c = a \cdot b$, and $T, G = R' \otimes S'$, $E' \times F'$. By Lemma 28, we have that there exist R'' and S'' such that $\mu(a, R'') = R'$ and $\mu(b, S'') = S'$. By the definition of $\Sigma_I(R_i, a_i :_{id} E'_i)$, and repeated application of the Sum rule, we have that there exist $k \in I$, R_k, a_k, R'_k, E''_k, such that $R'' = R_k$, $\Sigma_I(R_i, a_i :_{id} E'_i) \xrightarrow{a_k} R'_k, E''_k$, $a = a_k$, and $R', E' = R'_k, E''_k$. By the Act and PrefixOne rules, we have that $E''_k = \mathbf{0} :_{id} E'_k$. By the definition of $\Sigma_J(S_j, b_j :_{id} F'_j)$, and repeated application of the Sum rule, we have that there exist $l \in J$, S_l, b_l, S'_l, F''_l, such that $S'' = S_l$, $\Sigma_J(S_j, b_j :_{id} F'_j) \xrightarrow{b_l} S'_l, F''_l$, $b = b_l$, and $S', F' = S'_l, F''_l$. By the Act and PrefixOne rules, we have that $F''_l = \mathbf{0} :_{id} F'_l$. By Definition 5, as $\mu(a_k, R_k) = R'_k$ and $\mu(b_l, S_l) = S'_l$, we have that $\mu(a_k \cdot b_l, R_k \otimes S_l) = R'_k \otimes S'_l$. By the Act rule, we have that $R_k \otimes S_l, a_k \cdot b_l \xrightarrow{a_k \cdot b_l} R'_k \otimes S'_l, \mathbf{0}$. By the PrefixOne rule, we have that $R_k \otimes S_l, (a_k \cdot b_l) :_{id} E'_k \times F'_l \xrightarrow{a_k \cdot b_l} R'_k \otimes S'_l, \mathbf{0} :_{id} (E'_k \times F'_l)$. By repeated application of the Sum rule, we have that $\Sigma_{I\times J}(R_i \otimes S_j, (a_i \cdot b_j) :_{id} E'_i \times F'_j) \xrightarrow{a_k \cdot b_l} \mathbf{0} :_{id} (E'_k \times F'_l)$. By Lemma 29, we have that $R'_k \otimes S'_l, \mathbf{0} :_{id} (E'_k \times F'_l) \sim R' \otimes S', E' \times F'$, and hence that $(R'_k \otimes S'_l, \mathbf{0} :_{id} (E'_k \times F'_l))\mathcal{R}(R' \otimes S', E' \times F')$.

Suppose that $\Sigma_{I\times J}(R_i \otimes S_j, (a_i \cdot b_j) :_{id} E'_i \times F'_j) \xrightarrow{c} T, G$. By repeated application of the Sum rule, we have that there exist $k \in I$, $l \in J$ such that $R_k \otimes S_l, (a_k \cdot b_l) :_{id} E'_k \times F'_l \xrightarrow{c} T, G$. By the Act and PrefixOne rules, we have that $G = \mathbf{0} :_{id} (E'_k \times F'_l)$ and $\mu(a_k \cdot b_l, R_k \otimes S_l) = R'_k \otimes S'_l$. By the definition of $(\Sigma_{I\times J}(R_i \otimes S_j, (a_i \cdot b_j) :_{id} E'_i \times F'_j))$, we have that $\mu(a_k, R_k) = R'_k$, and $\mu(b_l, S_l) = S'_l$. By the definition of $\Sigma_I(R_i, a_i :_{id} E'_i)$, we have that $R, E \xrightarrow{a_k} R'_k, E'_k$. By the definition of $\Sigma_J(S_j, b_j :_{id} F'_j)$, we have that $S, F \xrightarrow{b_l} S'_l, F'_l$. By the Prod rule, we have that $R \otimes S, E \times F \xrightarrow{a_k \cdot b_l} R'_k \otimes S'_l, E'_k \times F'_l$. By Lemma 29, we have that $R'_k \otimes S'_l, \mathbf{0} :_{id} (E'_k \times F'_l) \sim R'_k \otimes S'_l, E'_k \times F'_l$, and hence that $(R'_k \otimes S'_l, \mathbf{0} :_{id} (E'_k \times F'_l))\mathcal{R}(R'_k \otimes S'_l, E'_k \times F'_l)$.

Hence $\mathcal{R}$ is closed and a bisimulation. □

3. Embedding SCRP in CBRP

In order to be able to use CBRP as a replacement for SCRP — as defined in [11,12,14] and sketched in Section 1 — we should be able to embed the latter soundly into the former. The embedding, for finite states, is described below.

Recall from our introductory discussion that the essential differences between SCRP and CBRP are the following:

- SCRP: resources are assumed to form a (possibly preordered) monoid (with some coherence conditions) in which there is a single composition operation. Elements R of the monoid of resources are then taken together with process terms E in judgements of the form $R, E \xrightarrow{a} \ldots$ and $R, E \models \phi$;
- CBRP: Resources are not assumed to form a monoid. Rather, resource are combined into bunches using two combining operations, $\oplus$ and $\otimes$. Bunches of resources R are then taken together with process terms E in judgements of the form $R, E \xrightarrow{a} \ldots$, where $\oplus$ is used in consort with $+$ and $\otimes$ is used in consort with $\times$, and $R, E \models \phi$.

Formally, SCRP is parametrized by structures $(\mathbf{Act}, \mathbf{R}, \mu, \nu)$, where $\mathbf{Act}$ is a commutative monoid of actions and $\mathbf{R}$ is a resource monoid [11,12,14]. We refer to $(\mathbf{Act}, \mathbf{R}, \mu, \nu)$-SCRP just as we refer to $(\mathbf{Act}, \mathbf{Res}, \mathbf{R}, \mu, \Delta, \mathbf{H})$-CBRP.

Consider some $(\mathbf{Act}, \mathbf{S}, \mu, \nu)$-SCRP. We define a $(\mathscr{A}, \mathbf{Res}, \mathbf{R}, \mu', \Delta, \mathbf{H})$-CBRP. Let $\mathscr{A}$ be the carrier set of the monoid $\mathbf{Act}$. Let $\mathbf{Res}$ — the carrier set of the resource monoid $\mathbf{S}$ — be the set of atomic resources, the resource model $\mathbf{R}$ be the smallest set such that Definition 2 holds, and the unit of the resource monoid to be the empty resource. Let the modification function μ' be the closure of the SCRP modification function μ under the conditions in Definition 5. The redistribution functions consist the identity function and those that make n-copies of the input, bunched using $\oplus$ — that is, $\Delta = \{\text{n-copy} \mid \text{1-copy} = id, (n+1)\text{-copy}(R) = R \oplus n\text{-copy}(R),\ \text{where } n \in \mathbb{N}, 1 \le n\}$. We make use of no hiding functions — that is, $\mathbf{H} = \emptyset$. For the remainder of this section, we assume fixed SCRP and CBRP structures.

Let $\hat{a}$ denote a SCRP action, $\hat{r}$ denote a SCRP resource, and $\hat{E}$ denote a SCRP process. Here we let $\approx$ denote local (cf. global equivalence [11,12,14]) equivalence for SCRP [11,12,14], which is defined as follows:

Definition 32 *(Local bisimulation).* The local equivalence relation, $\approx$, is the largest binary relation on closed SCRP states such that the condition below holds.

Let $\hat{r}$ and $\hat{s}$ be SCRP resources and $\hat{E}$ and $\hat{F}$ be SCRP processes. If $\hat{r}, \hat{E} \approx \hat{s}, \hat{F}$, then

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

Let $|\_|$ denote the cardinality of a set. We define our embedding on finite SCRP states, those that cannot generate infinite traces. An embedding from $(\mathbf{Act}, \mathbf{S}, \mu, \nu)$-SCRP into $(\mathscr{A}, \mathbf{Res}, \mathbf{R}, \mu', \Delta, \mathbf{H})$-CBRP is then definable. Let $\Sigma_I R_i, E_i = e, \mathbf{0}$, if $I = \emptyset$, and let $\Sigma_I E_i = E_i$, if $I = \{i\}$.

Definition 33 *(SCRP embedding).* The embedding function $[\![\_]\!]$ from finite, closed SCRP states to finite, closed CBRP states is defined as $[\![\hat{r}, \hat{E}]\!] = \Sigma_I \{r, a_i :_{|J_i|\text{-copy}} (\Sigma_{J_i} E''_{j_i}) \mid [\![\hat{r}'_i, \hat{E}'_i]\!] = \Sigma_{J_i}(r'_{j_i}, E''_{j_i})\}$, where $\hat{r}, \hat{E} \approx \hat{r}, \Sigma_I \{\hat{a}_i, \hat{E}'_i \mid \hat{r}, \hat{E} \xrightarrow{\hat{a}_i} \hat{r}'_i, \hat{E}'_i\}$. □

As we consider only finite SCRP states, there are no infinite traces, and hence this function is well defined.

Note that any SCRP resource $\hat{r}$ is an atomic CBRP resource. When such a resource is being considered as a CBRP resource, we refer to it as r rather than $\hat{r}$. An example embedding for mutual exclusion from SCRP to CBRP is given below.

Example 34 *(Embedding mutual exclusion).* We use the free monoid over the atomic actions $\{\hat{a}, \hat{b}\}$, with composition ; and unit $\hat{1}$. We write ab for $a; b$. We use the resource monoid $\mathbf{S} = (\{\hat{e}, \hat{s}\}, \circ, \hat{e})$, where

$$\hat{s} \circ \hat{e} = \hat{e} \circ \hat{s} = \hat{s} \qquad \hat{e} \circ \hat{e} = \hat{e} \qquad \hat{s} \circ \hat{s} \uparrow .$$

We define the SCRP modification function

$$\mu(\hat{1}, \hat{e}) = \hat{e} \qquad \mu(\hat{1}, \hat{s}) = \hat{s} \qquad \mu(\hat{a}, \hat{s}) = \hat{s} \qquad \mu(\hat{b}, \hat{e}) = \hat{e}.$$

We define the SCRP processes

$$\begin{array}{ll} \hat{B} = (\hat{a} : \hat{B}') + (\hat{b} : \hat{B}) & \hat{B}' = (\hat{a} : \hat{B}') + (\hat{a} : \hat{B}) \\ \hat{E} = (\hat{ab} : \hat{F}) + (\hat{bb} : \hat{E}) & \hat{F} = (\hat{ab} : \hat{F}) + (\hat{ab} : \hat{E}). \end{array}$$

The following states are bisimilar.

$$\hat{s}, \hat{B} \times \hat{B} \approx \hat{s}, \hat{E} \qquad \hat{s}, \hat{B} \times \hat{B}' \approx \hat{s}, \hat{F} \qquad \hat{s}, \hat{B}' \times \hat{B} \approx \hat{s}, \hat{F}$$

Let $\mathbf{Res} = \{e, s\}$ be the set of atomic resources, e be the empty resource, $\mathbf{R}$ be the smallest set such that Definition 2 holds, and the modification function μ' be the smallest that contains μ and is closed under the conditions in Definition 5. We define the BCRP processes

$$G = ((ab) :_{2\text{-copy}} H) + ((bb) :_{2\text{-copy}} G) \qquad H = ((ab) :_{2\text{-copy}} H) + ((ab) :_{2\text{-copy}} G).$$

As ab and bb ($a; b$ and $b; b$) are elements of the free monoid, we use them as atomic actions.

By Definition 33, we have that

$$[\![\hat{s}, \hat{E}]\!] = s \oplus s, G \qquad [\![\hat{s}, \hat{F}]\!] = s \oplus s, H$$

Note that whenever an action $\hat{x}$ can be performed by $\hat{s}, \hat{B} \times \hat{B}$ or $\hat{s}, \hat{B} \times \hat{B}'$, the action x can be performed by $s \oplus s, G$ or $s \oplus s, H$, respectively. □

The standard notion of simulation is that one state in a system simulates another if the former can perform the same actions as the latter, and, after those reductions, the resulting states of the former simulate the resulting states of the latter. The simulation relation between CBRP and SCRP is defined as follows below. Let $\hat{r}$ and $\hat{r'}$ be SCRP resources, $\hat{E}$ and $\hat{E}'$ be closed SCRP processes, S and S' be CBRP resources, and F and F' be closed CBRP processes.

Definition 35 *(Simulation).* Let the relation $\lesssim$ be the largest relation $\mathcal{R}$ such that, for all $(\hat{r}, \hat{E})\,\mathcal{R}\,(S, F)$, if $\hat{r}, \hat{E} \xrightarrow{\hat{a}} \hat{r'}, \hat{E}'$, then there exist S', F' such that $S, F \xrightarrow{a} S', F'$ and $(\hat{r'}, \hat{E}')\,\mathcal{R}\,(S', F')$. □

We then have that the embedding of a finite SCRP state simulates that state.

Proposition 36. *Let $\hat{r}, \hat{E}$ be a finite SCRP state. Then, $\hat{r}, \hat{E} \lesssim [\![\hat{r}, \hat{E}]\!]$.*

Proof. Let

$$\mathcal{R} = \{((\hat{r}, \hat{E}), (S, F)) \mid [\![\hat{r}, \hat{E}]\!] \sim S, F\}.$$

Suppose that $\hat{r}, \hat{E} \xrightarrow{\hat{a}} \hat{r}', \hat{E}'$. By [11, Lemma 11], there exist I, $\hat{E}_i$, $\hat{a}_i$, $\hat{r}'_i$, $\hat{E}'_i$ such that $\hat{r}, \hat{E} \approx \hat{r}, \Sigma_I\{\hat{a}_i, \hat{E}'_i \mid \hat{r}, \hat{E} \xrightarrow{\hat{a}_i} \hat{r}'_i, \hat{E}'_i\}$. By Definition 32, we have that there exists $k \in I$ such that $\hat{a} = \hat{a}_k$, $\hat{r}' = \hat{r}'_k$, and $\hat{E}' = \hat{E}'_k$.

By Definition 33, we have that $[\![\hat{r}, \hat{E}]\!] = \Sigma_I\{r, a_i :_{|J_i|\text{-copy}} (\Sigma_{J_i} E''_{j_i}) \mid [\![\hat{r}'_i, \hat{E}'_i]\!] = \Sigma_{J_i}(r'_{j_i}, E''_{j_i})\}$. By [11, Lemma 2], we have that $\hat{r}' = \mu(\hat{a}, \hat{r})$. By the definition of the embedding, we have that $\mu'(a, r) = r'$. By the Act rule, we have that $r, a_k \xrightarrow{a_k} r', \mathbf{0}$. By the PrefixOne rule, we have that $r, a_k :_{|J_k|\text{-copy}} (\Sigma_{J_k} E''_{j_k}) \xrightarrow{a_k} r', \mathbf{0} :_{|J_k|\text{-copy}} (\Sigma_{J_k} E''_{j_k})$. By repeated application of the Sum rule, we have that $[\![\hat{r}, \hat{E}]\!] \xrightarrow{a_k} r', \mathbf{0} :_{|J_k|\text{-copy}} (\Sigma_{J_k} E''_{j_k})$. By Lemma 29, we have that $r', \mathbf{0} :_{|J_k|\text{-copy}} (\Sigma_{J_k} E''_{j_k}) \sim |J_k|\text{-copy}(r'), \Sigma_{J_k} E''_{j_k}$. We straightforwardly have that $|J_k|\text{-copy}(r'), \Sigma_{J_k} E''_{j_k} = \Sigma_{J_k}(r', E''_{j_k})$. By Definition 33, we have that $[\![\hat{r}', \hat{E}']\!] = \Sigma_{J_k}(r', E''_{j_k})$. We then have that $(r', E')\mathcal{R}(r', \mathbf{0} :_{|J_k|\text{-copy}} (\Sigma_{J_k} E''_{j_k}))$, and we are done. □

4. Algebraic properties

In order to reason equationally about processes, it is also useful to establish various algebraic properties concerning concurrent composition and choice. Notable standard algebraic properties of process calculi are commutativity and associativity of concurrent composition, that is, $R \otimes S, E \times F \sim S \otimes R, F \times E$ and $R \otimes (S \otimes T), E \times (F \times G) \sim (R \otimes S) \otimes T, (E \times F) \times G$. For the notion of bisimulation in Definition 22, however, we do not have these properties.

Example 37. Let the set of atomic actions be $\mathbf{Act} = \{a, b\}$, the set of atomic resources be $\mathbf{Res} = \{e, r, s\}$, with empty resource e,

$$\begin{array}{ll} R = r \otimes (s \otimes e) & S = (r \otimes s) \otimes s \\ E = a \times (b \times 1) & F = (a \times b) \times 1, \end{array}$$

and the resource model $\mathbf{R}$ be the least set such that $R, S \in \mathbf{R}$ and Definition 2 holds. Let the modification function $\mu : \mathbf{A} \times \mathbf{R} \rightharpoonup \mathbf{R}$ be the least function (under set inclusion of the domain) such that

$$\mu(a, r) = e \qquad \mu(b, s) = e,$$

and Definition 5 holds.

By the operational semantics, we have that

$$\dfrac{\dfrac{\mu(a,r)=e}{r, a \xrightarrow{a} e, \mathbf{0}}\,\textsc{Act} \quad \dfrac{\mu(b,s)=e}{s, b \xrightarrow{b} e, \mathbf{0}}\,\textsc{Act}}{r \otimes s, a \times b \xrightarrow{a \cdot b} e \otimes e, \mathbf{0} \times \mathbf{0}}\,\textsc{Prod} \qquad \dfrac{\dfrac{\mu(b,s)=e}{s, b \xrightarrow{b} e, \mathbf{0}}\,\textsc{Act} \quad \dfrac{\mu(a,r)=e}{r, a \xrightarrow{a} e, \mathbf{0}}\,\textsc{Act}}{s \otimes r, b \times a \xrightarrow{b \cdot a} e \otimes e, \mathbf{0} \times \mathbf{0}}\,\textsc{Prod}.$$

Note that, despite the fact that $r \otimes s, a \times b$ is the commutation of $s \otimes r, b \times a$, they are not bisimilar. This is as the former performs the action $a \cdot b$, and the latter performs the action $b \cdot a$.

By the operational semantics, we also have that

$$\dfrac{\dfrac{\mu(a,r)=e}{r, a \xrightarrow{a} e, \mathbf{0}}\,\textsc{Act} \quad \dfrac{\dfrac{\mu(b,s)=e}{s, b \xrightarrow{b} e, \mathbf{0}}\,\textsc{Act} \quad \dfrac{\mu(1,e)=e}{e, 1 \xrightarrow{1} e, \mathbf{0}}\,\textsc{Act}}{s \otimes e, b \times 1 \xrightarrow{b \cdot 1} e \otimes e, \mathbf{0} \times \mathbf{0}}\,\textsc{Prod}}{r \otimes (s \otimes e), a \times (b \times 1) \xrightarrow{a \cdot (b \cdot 1)} e \otimes (e \otimes e), \mathbf{0} \times (\mathbf{0} \times \mathbf{0})}\,\textsc{Prod}$$

and

$$\dfrac{\dfrac{\dfrac{\mu(a,r)=e}{r, a \xrightarrow{a} e, \mathbf{0}}\,\textsc{Act} \quad \dfrac{\mu(b,s)=e}{s, b \xrightarrow{b} e, \mathbf{0}}\,\textsc{Act}}{r \otimes s, a \times b \xrightarrow{a \cdot b} e \otimes e, \mathbf{0} \times \mathbf{0}}\,\textsc{Prod} \quad \dfrac{\mu(1,e)=e}{e, 1 \xrightarrow{1} e, \mathbf{0}}\,\textsc{Act}}{(r \otimes s) \otimes e, (a \times b) \times 1 \xrightarrow{(a \cdot b) \cdot 1} (e \otimes e) \otimes e, (\mathbf{0} \times \mathbf{0}) \times \mathbf{0}}\,\textsc{Prod}.$$

Note that, despite the fact that $r \otimes (s \otimes e), a \times (b \times 1)$ can be re-associated to $(r \otimes s) \otimes e, (a \times b) \times 1$, they are not bisimilar. This is as the former performs the action $a \cdot (b \cdot 1)$, and the latter performs the action $(a \cdot b) \cdot 1$. □

We would like to recover the algebraic properties, and to have that the above pairs of states are bisimilar. Note that $\cdot$ structure of an action matches the concurrent structure ($\otimes$ and $\times$) of a state which performs that action. Hence, the notion of bisimulation in Definition 22 requires that the structure of two bisimilar states be very closely aligned.

In order to obtain the desired algebraic properties, it can be useful to disregard some of the structure. We define an equivalence relation on actions.

Definition 38 *(Action equivalence).* The action equivalence relation $\equiv$ is the least relation under reflexivity, symmetry, and transitivity such that the following hold:

$$\frac{}{a \cdot 1 \equiv a}\ (1) \qquad \frac{}{a \cdot b \equiv b \cdot a}\ (2) \qquad \frac{}{a \cdot (b \cdot c) \equiv (a \cdot b) \cdot c}\ (3) \qquad \frac{a \equiv a' \quad b \equiv b'}{a \cdot b \equiv a' \cdot b'}\ (4). \qquad \square$$

Let R and S be resources, and E and F be agents. Then we define bisimulation, up to the equivalence relation $\equiv$.

Definition 39 *(Bisimulation).* A relation $\mathcal{R}_\equiv$ is a bisimulation relation, *up to* the action equivalence $\equiv$, if, for all closed states $(R, E)\mathcal{R}_\equiv(S, F)$, then

- if $R, E \xrightarrow{a} R', E'$, then there exist b, S', F', such that $a \equiv b$, $S, F \xrightarrow{b} S', F'$, and $(R', E')\mathcal{R}_\equiv(S', F')$, and
- if $S, F \xrightarrow{a} S', F'$, then there exist b, R', E', such that $a \equiv b$, $R, E \xrightarrow{b} R', E'$, and $(R', E')\mathcal{R}_\equiv(S', F')$. $\square$

Let $\sim_\equiv \subseteq \textbf{State} \times \textbf{State}$ be the union of all bisimulations. The union of any two bisimulations is also a bisimulation. Hence $\sim_\equiv$ is well defined, and a bisimulation.

With this definition of bisimulation, we can obtain the desired algebraic properties. Note that the fact that we impose the use of the equivalence relation in the notion of bisimulation does not mean we must impose its use in the notion of actions modifying resources. Hence, if we have that $r, a \sim_\equiv s, b$, then we have that $\mu(a, r)$ and $\mu(b, s)$ are defined and that $a \equiv b$, but we do not necessarily have that $\mu(a, s)$ or $\mu(b, r)$ are defined. The distinct (but equivalent) actions can modify the resource components of the two states differently, and the resulting states are bisimilar. Hence, if $R, E \sim_\equiv S, F$, if $R, E \xrightarrow{a}$, then we don't necessarily have that $S, F \xrightarrow{a}$. Bisimilar pairs perform distinct (but equivalent) transitions that lead to bisimilar states. This is possible as, unlike in [11,12,14], the definition of bisimulation permits bisimilar states to have *different* resource components.

Example 40. Recall Example 37. By the operational semantics, we have that

$$\dfrac{\dfrac{\mu(a,r)=e}{r, a \xrightarrow{a} e, \mathbf{0}}\ \textsc{Act} \qquad \dfrac{\mu(b,s)=e}{s, b \xrightarrow{b} e, \mathbf{0}}\ \textsc{Act}}{r \otimes s, a \times b \xrightarrow{a \cdot b} e \otimes e, \mathbf{0} \times \mathbf{0}}\ \textsc{Prod} \qquad \dfrac{\dfrac{\mu(b,s)=e}{s, b \xrightarrow{b} e, \mathbf{0}}\ \textsc{Act} \qquad \dfrac{\mu(a,r)=e}{r, a \xrightarrow{a} e, \mathbf{0}}\ \textsc{Act}}{s \otimes r, b \times a \xrightarrow{b \cdot a} e \otimes e, \mathbf{0} \times \mathbf{0}}\ \textsc{Prod}.$$

As $a \cdot b \equiv b \cdot a$ and $e \otimes e, \mathbf{0} \times \mathbf{0} \not\rightarrow$, we have that $r \otimes s, a \times b \sim_\equiv s \otimes r, b \times a$. Note that, despite the fact that $r \otimes s, a \times b \sim_\equiv s \otimes r, b \times a$ and $\mu(a \cdot b, r \otimes s) \downarrow$, it is *neither* the case that $r \otimes s, a \times b \xrightarrow{b \cdot a}$ nor that $\mu(b \cdot a, r \otimes s) \downarrow$.

By the operational semantics, we also have that

$$\dfrac{\dfrac{\mu(a,r)=e}{r, a \xrightarrow{a} e, \mathbf{0}}\ \textsc{Act} \qquad \dfrac{\dfrac{\mu(b,s)=e}{s, b \xrightarrow{b} e, \mathbf{0}}\ \textsc{Act} \qquad \dfrac{\mu(1,e)=e}{e, 1 \xrightarrow{1} e, \mathbf{0}}\ \textsc{Act}}{s \otimes e, b \times 1 \xrightarrow{b \cdot 1} e \otimes e, \mathbf{0} \times \mathbf{0}}\ \textsc{Prod}}{r \otimes (s \otimes e), a \times (b \times 1) \xrightarrow{a \cdot (b \cdot 1)} e \otimes (e \otimes e), \mathbf{0} \times (\mathbf{0} \times \mathbf{0})}\ \textsc{Prod}$$

and

$$\dfrac{\dfrac{\dfrac{\mu(a,r)=e}{r, a \xrightarrow{a} e, \mathbf{0}}\ \textsc{Act} \qquad \dfrac{\mu(b,s)=e}{s, b \xrightarrow{b} e, \mathbf{0}}\ \textsc{Act}}{r \otimes s, a \times b \xrightarrow{a \cdot b} e \otimes e, \mathbf{0} \times \mathbf{0}}\ \textsc{Prod} \qquad \dfrac{\mu(1,e)=e}{e, 1 \xrightarrow{1} e, \mathbf{0}}\ \textsc{Act}}{(r \otimes s) \otimes e, (a \times b) \times 1 \xrightarrow{(a \cdot b) \cdot 1} (e \otimes e) \otimes e, (\mathbf{0} \times \mathbf{0}) \times \mathbf{0}}\ \textsc{Prod}.$$

As $a \cdot (b \cdot 1) \equiv (a \cdot b) \cdot 1$, $e \otimes (e \otimes e), \mathbf{0} \times (\mathbf{0} \times \mathbf{0}) \not\rightarrow$, and $(e \otimes e) \otimes e, (\mathbf{0} \times \mathbf{0}) \times \mathbf{0} \not\rightarrow$, we have that $r \otimes (s \otimes e), a \times (b \times 1) \sim_\equiv (r \otimes s) \otimes e, (a \times b) \times 1$. Note that, despite the fact that $r \otimes (s \otimes e), a \times (b \times 1) \sim_\equiv (r \otimes s) \otimes e, (a \times b) \times 1$ and $\mu(a \cdot (b \cdot 1), r \otimes (s \otimes e)) \downarrow$, it is *neither* the case that $r \otimes (s \otimes e), a \times (b \times 1) \xrightarrow{(a \cdot b) \cdot 1}$ nor that $\mu((a \cdot b) \cdot 1, r \otimes (s \otimes e)) \downarrow$. $\square$

The structure of actions that a state can perform is directed by the concurrent structure of that state. When bisimulation is defined up to action equivalence, two bisimilar states do not necessarily perform *exactly* the same actions, but that they can always perform *equivalent* actions. In the definition of bisimulation up to equivalence (Definition 39), if $R, E \sim_{\equiv} S, F$, the resources R and S can be entirely unrelated, both in their structure and their constituent atomic resources. The key thing is that both *states* can perform equivalent actions, and remain bisimilar (up to action equivalence).

The definition of bisimulation up to the action equivalence relation, $\equiv$, enables us to prove various algebraic properties of our calculus.

Proposition 41 *(Algebraic properties). For all bunched resources $R, S, T \in \mathbf{R}$ and agents E, F, G,*

$$\begin{array}{ll}
\textit{(Commutativity of choice)} & R \oplus S, E + F \sim_{\equiv} S \oplus R, F + E \\
\textit{(Unit of choice)} & R \oplus S, E + \mathbf{0} \sim_{\equiv} R, E \\
\textit{(Associativity of choice)} & R \oplus (S \oplus T), E + (F + G) \sim_{\equiv} (R \oplus S) \oplus T, (E + F) + G \\
\\
\textit{(Commutativity of product)} & R \otimes S, E \times F \sim_{\equiv} S \otimes R, F \times E \\
\textit{(Unit of product)} & R \otimes S, E \times \mathbf{1} \sim_{\equiv} R, E \\
\textit{(Zero property of product)} & R \otimes S, E \times \mathbf{0} \sim_{\equiv} S, \mathbf{0} \\
\textit{(Associativity of product)} & R \otimes (S \otimes T), E \times (F \times G) \sim_{\equiv} (R \otimes S) \otimes T, (E \times F) \times G \\
\\
\textit{(Distribution of product} & R \otimes (S \oplus T), E \times (F + G) \sim_{\equiv} \\
\quad \textit{over choice)} & \quad (R \otimes S) \oplus (R \otimes T), (E \times F) + (E \times G).
\end{array}$$

Proof. Straightforward, through the definition of the relevant bisimulations, by the operational semantics and Definition 38. As an illustration, we prove associativity of product.

Let

$$\mathcal{R}_{\equiv} = \{((R \otimes (S \otimes T), E \times (F \times G)), ((R \otimes S) \otimes T, (E \times F) \times G)) \mid E, F, G \text{ are agents}\}.$$

Suppose that $R \otimes (S \otimes T), E \times (F \times G) \xrightarrow{d} U, H$. By repeated application of the Prod rule, there exist a, b, c, R', S', T', E', F', G', such that $d = a \cdot (b \cdot c)$, $U, H = R' \otimes (S' \otimes T'), E' \times (F' \times G')$, $R, E \xrightarrow{a} R', E'$, $S, F \xrightarrow{b} S', F'$. and $T, G \xrightarrow{c} T', G'$. By Definition 2, we have that $(R \otimes S) \otimes T$ is defined. By further application of the Prod rule, $(R \otimes S) \otimes T, (E \times F) \times G \xrightarrow{(a \cdot b) \cdot c} (R' \otimes S') \otimes T', (E' \times F') \times G'$. By Definition 38, $a \cdot (b \cdot c) \equiv (a \cdot b) \cdot c$. We then have that $(R' \otimes (S' \otimes T'), E' \times (F' \times G')) \mathcal{R}_{\equiv} ((R' \otimes S') \otimes T', (E' \times F') \times G')$.

The other case is similar. Hence $\mathcal{R}_{\equiv}$ is closed and a bisimulation. □

Corollary 42. *For all bunched resources $R, S, T \in \mathbf{R}$ and agents E, F, G,*

$$\begin{array}{ll}
R \oplus R, E + F \sim_{\equiv} R \oplus R, F + E & R \oplus S, E + E \sim_{\equiv} S \oplus R, E + E \\
R \otimes R, E \times F \sim_{\equiv} R \otimes R, F \times E & R \otimes S, E \times E \sim_{\equiv} S \otimes R, E \times E.
\end{array}$$

Recall the semaphore resource model in Example 3. Suppose some action a and modification function μ such that $\mu(a, s) = s$. We then have that $s \otimes e, a \times 1 \sim_{\equiv} e \otimes s, 1 \times a$, as

$$\dfrac{\dfrac{\mu(a,s) = s}{s, a \xrightarrow{a} s, \mathbf{0}}\ \textsc{Act} \quad \dfrac{\mu(1,e) = e}{e, 1 \xrightarrow{1} e, \mathbf{0}}\ \textsc{Act}}{s \otimes e, a \times 1 \xrightarrow{a \cdot 1} s \otimes e, \mathbf{0} \times \mathbf{0}}\ \textsc{Prod} \qquad \dfrac{\dfrac{\mu(1,e) = e}{e, 1 \xrightarrow{1} e, \mathbf{0}}\ \textsc{Act} \quad \dfrac{\mu(a,s) = s}{s, a \xrightarrow{a} s, \mathbf{0}}\ \textsc{Act}}{e \otimes s, 1 \times a \xrightarrow{1 \cdot a} e \otimes s, \mathbf{0} \times \mathbf{0}}\ \textsc{Prod},$$

and $a \cdot 1 \equiv 1 \cdot a$. We do not, however, have that $s \otimes e, a \times 1 \sim_{\equiv} e \otimes s, a \times 1$. If we commute the resource component of a state, but not the process component, then the resulting state is not necessarily bisimilar to the original state.

Recall the resource model and modification function in Example 6. We trivially have that $2 \otimes 4, i \times 1 \sim_{\equiv} 2 \otimes 4, 1 \times i$ as they can perform equivalent actions, $i \cdot 1$ and $1 \cdot i$, and are bisimilar thereafter (they can perform no behaviour).

It is possible to retain the congruence results for bisimulation up to equivalence. In order to prove that hiding to be a congruence operator, we require that hiding functions preserve action equivalence, and evolve our notion of $\sim$-resource-closed calculi.

Definition 43. A hiding function on resources h preserves action equivalence if, for all $a, b \in \mathbf{A}$, if $a \equiv b$, then $\nu h.a \equiv \nu h.b$. □

We define a notion of $\sim_{\equiv}$-resource-closed calculi.

Definition 44 *($\sim_{\equiv}$-Resource-closed CBRP).* A calculus is $\sim_{\equiv}$-resource-closed if, for all R_1, E_1, S_1, F_1, R_2, E_2, S_2, F_2, such that $R_1, E_1 \sim_{\equiv} S_1, F_1$ and $R_2, E_2 \sim_{\equiv} S_2, F_2$, then $R_1 \otimes R_2$ (respectively, $R_1 \oplus R_2$) is defined if and only if $S_1 \otimes S_2$ (respectively, $S_1 \oplus S_2$) is defined. □

We can then prove that bisimulation up to equivalence is a congruence.

Theorem 45 *(Bisimulation congruence). Suppose a $\sim_{\equiv}$-resource-closed calculus. The relation $\sim_{\equiv}$ is a congruence for concurrent, non-deterministic, and sequential composition, and hiding: for all closed states (R_i, E_i), (S_i, F_i), (R, E), (S, F), if $R_i, E_i \sim_{\equiv} S_i, F_i$, $h(R), E \sim_{\equiv} h(S), F$, $\delta, \delta' \in \Delta$, R_1, E_1, and R_2 are δ-sequence compatible, S_1, F_1, S_2 are δ'-sequence compatible, and $R_1 \otimes R_2$, $S_1 \otimes S_2$, $R_1 \oplus R_2$, and $S_1 \oplus S_2$ are defined, then, for any hiding function $h \in \mathbf{H}$ that preserves action equivalence*

$$R_1 \otimes R_2, E_1 \otimes E_2 \sim_{\equiv} S_1 \otimes S_2, F_1 \otimes F_2 \quad R_1 \oplus R_2, E_1 + E_2 \sim_{\equiv} S_1 \oplus S_2, F_1 + F_2$$

$$R_1, E_1 :_{\delta} E_2 \sim_{\equiv} S_1, F_1 :_{\delta'} F_2 \qquad\qquad R, \nu h.E \sim_{\equiv} S, \nu h.E.$$

Proof. The bisimulation relation $\sim_{\equiv}$ is the largest bisimulation relation, and contains all other bisimulation relations. In order to show that the above properties hold, it is sufficient, therefore, to define a relation $\mathcal{R}_{\equiv}$, for which the required properties hold, and to show that the relation $\mathcal{R}_{\equiv}$ is a bisimulation.

A congruence is reflexive, symmetric, transitive, and preserved under the above constructions. Reflexivity, symmetricity, and transitivity are straightforward to observe.

(1) Consider concurrent composition. Let

$$\mathcal{R}_{\equiv} = \{((R_1 \otimes R_2, E_1 \times E_2), (S_1 \otimes S_2, F_1 \times F_2)) \mid R_1, E_1 \sim_{\equiv} S_1, F_1 \text{ and } R_2, E_2 \sim_{\equiv} S_2, F_2\}.$$

If $R_1 \otimes R_2, E_1 \times E_2 \xrightarrow{a} R_1' \otimes R_2', E_1' \times E_2'$, then the last rule used in the derivation of this reduction must be the Prod rule. By the Prod rule, we have that $R_1, E_1 \xrightarrow{a_1} R_1', E_1'$, $R_2, E_2 \xrightarrow{a_2} R_2', E_2'$, and $a = a_1 \cdot a_2$. By Definition 22, we have that there exist b_1, S_1', F_1', b_2, S_2', F_2', such that $a_1 \equiv b_1$, $a_2 \equiv b_2$, $S_1, F_1 \xrightarrow{b_1} S_1', F_1'$, $S_2, F_2 \xrightarrow{b_2} S_2', F_2'$, $(R_1', E_1') \sim_{\equiv} (S_1', F_1')$, and $(R_2', E_2') \sim_{\equiv} (S_2', F_2')$. By the Prod rule, we have that $S_1 \otimes S_2, F_1 \times F_2 \xrightarrow{b_1 \cdot b_2} S_1' \otimes S_2', F_1' \times F_2'$. By Definition 38, we have that $a_1 \cdot a_2 \equiv b_1 \cdot b_2$. As $(R_1', E_1') \sim_{\equiv} (S_1', F_1')$ and $(R_2', E_2') \sim_{\equiv} (S_2', F_2')$, we have that $(R_1' \otimes R_2', E_1' \times E_2')\mathcal{R}_{\equiv}(S_1' \otimes S_2', F_1' \times F_2')$.

The other case is similar. Hence $\mathcal{R}_{\equiv}$ is closed and a bisimulation for concurrent composition.

(2) Consider non-deterministic choice. Let

$$\mathcal{R}_{\equiv} = \{((R_1 \oplus R_2\,,\, E_1 + E_2), (S_1 \oplus S_2\,,\, F_1 + F_2)) \mid R_1, E_1 \sim S_1, F_1 \text{ and } R_2, E_2 \sim S_2, F_2\} \cup \sim .$$

If $R_1 \oplus R_2, E_1 + E_2 \xrightarrow{a} R', E'$, then the last rule used in the derivation of this reduction must be the Sum rule. By the Sum rule, we have that, for some $i \in \{1, 2\}$, $R_i, E_i \xrightarrow{a} R_i', E_i'$ and $R', E' = R_i', E_i'$. By Definition 22, we have that there exist b, S_i', F_i', such that $S_i, F_i \xrightarrow{b} S_i', F_i'$, $a \equiv b$, and $(R_i', E_i') \sim_{\equiv} (S_i', F_i')$. By the Sum rule, we have that $S_1 \oplus S_2, F_1 + F_2 \xrightarrow{b} S_i', F_i'$. As $(R_i', E_i') \sim_{\equiv} (S_i', F_i')$, we have that $(R_i', E_i')\mathcal{R}_{\equiv}(S_i', F_i')$.

The other case is similar. Hence $\mathcal{R}_{\equiv}$ is closed and a bisimulation for non-deterministic composition.

(3) Consider sequential composition. Let

$$\begin{aligned}\mathcal{R} = \{((R_1, E_1 :_{\delta} E_2), (S_1, F_1 :_{\delta'} F_2)) \mid\ & R_1, E_1 \sim S_1, F_1,\ R_2, E_2 \sim S_2, F_2,\ R_1, E_1, R_2 \text{ are } \delta\text{-sequence compatible},\\ & S_1, F_1, S_2 \text{ are } \delta'\text{-sequence compatible, and } \delta, \delta' \in \Delta\\ \} \cup \sim .\end{aligned}$$

If $R_1, E_1 :_{\delta} E_2 \xrightarrow{a} R', E'$, then there are two possibilities for the last rule that is used in the derivation of this reduction.

First, suppose that PrefixOne is the last rule used. Then, we have that $R_1, E_1 \xrightarrow{a} R_1', E_1'$ and $\delta \in \Delta$. By Definition 22, we have that there exist b, S_1', F_1', such that $S_1, F_1 \xrightarrow{b} S_1', F_1'$, $a \equiv b$, and $(R_1', E_1') \sim_{\equiv} (S_1', F_1')$. By Definition 24, we have that R_1', E_1', and R_2 are δ-sequence compatible, and that S_1', F_1', S_2 are δ'-sequence compatible. By the PrefixOne rule, we have that $S_1, F_1 :_{\delta'} F_2 \xrightarrow{b} S_1', F_1' :_{\delta'} F_2$. We then have that $(R_1', E_1' :_{\delta} E_2)\mathcal{R}_{\equiv}(S_1', F_1' :_{\delta'} F_2)$.

Second, suppose that PrefixTwo is the last rule used. Then, we have that $R_1, E_1 \not\rightarrow$ and $\delta(R_1), E_2 \xrightarrow{a} R_2', E_2'$. By Definition 22, we have that $S_1, F_1 \not\rightarrow$. By Definition 24, we have that $\delta(R_1) = R_2$ and $\delta'(S_1) = S_2$. By Definition 22, we have that there exist b, S_2', F_2', such that $\delta'(S_1), F_2 \xrightarrow{b} S_2', F_2'$, $a \equiv b$, and $(R_2', E_2') \sim_{\equiv} (S_2', F_2')$. By the PrefixTwo rule, we have that $S_1, F_1 :_{\delta'} F_2 \xrightarrow{b} S_2', F_2'$. As $(R_2', E_2') \sim_{\equiv} (S_2', F_2')$, we have that $(R_2', E_2')\mathcal{R}_{\equiv}(S_2', F_2')$.

The other case is similar. Hence $\mathcal{R}_{\equiv}$ is closed and a bisimulation for sequential composition.

(4) Consider the hiding operator. Let

$$\mathcal{R}_{\equiv} = \{((R, \nu h.E), (S, \nu h.F)) \mid h(R), E \sim h(S), F\}.$$

If $R, \nu h.E \xrightarrow{a} R', E'$, then the last rule used in the derivation of this reduction must be the Hide rule. Then, we have that $h(R), E \xrightarrow{b} R'', E''$, $R'' = h(R')$, $E' = \nu h.E''$, and $a = \nu h.b$. By Definition 22, we have that there exist c, S'', F'', such that $h(S), F \xrightarrow{c} S'', F''$, $b \equiv c$, and $R'', E'' \sim_{\equiv} S'', F''$. By Definition 10, we have that there exists some S' such that $S'' = h(S')$. By the Hide rule, we have that $S, \nu h.F \xrightarrow{d} S', \nu h.F''$ and $d = \nu h.c$. By Definition 43, as $b \equiv c$, we have that $\nu h.b \equiv \nu h.c$. As $h(R'), E'' \sim_{\equiv} h(S'), F''$, we have that $(R', \nu h.E'')\mathcal{R}_{\equiv}(S', \nu h.F'')$.

The other case is similar. Hence $\mathcal{R}_{\equiv}$ is closed and a bisimulation for the hiding operator. □

5. A modal logic of resources and processes

In this section, we define a modal logic, here called the modal logic of bunched implications (MBI). We reuse the name MBI from [11,12,14], where it denotes a logic with the same propositional formulae, the semantics of which is given in terms of the transition relation for the calculus SCRP, sketched in Section 1.

Let Prop be a countable set of propositional letters denoting atomic propositions. Let p, q, etc., denote elements of Prop. Recall, from Section 2, that **A**, with elements a, b, etc., denotes a set of actions.

Definition 46 *(Propositional formulae).* We assume a set Prop of propositional letters, with elements denoted p, q, etc. Then the propositional formulae of MBI are given by the following grammar:

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

It is straightforward to add both additive and multiplicative quantifiers — $\exists$, $\forall$ and $\exists_\nu$, $\forall_\nu$, as described in [11,14] — to the logic. Their definition and theoretical treatment works as described in [11,14] and repeating them here would add little to the understanding provided.

The additive modalities are the standard necessarily and possibly modalities familiar from Hennessy–Milner logics for process algebras such as CCS. As such, they implicitly use meta-theoretic quantification to make statements about reachable states. The connectives $*$, $\mathrel{-\!\!*}$, and I are the multiplicative conjunction, implication, and unit, respectively. The multiplicative modalities $\langle a \rangle_\nu$ and $[a]_\nu$ reuse the symbol ν, which is meant to evoke the addition of extra components to the system, as is performed by hiding on resources.

The remainder of the section is structured as follows. In Section 5.1, we describe a semantics for MBI defined in terms of the transitions of the resource–process terms and of the bisimulation relation $\sim$ described in Section 2. We prove the Hennessy–Milner completeness result for the logic with this semantics. In Section 5.2, we describe a semantics for MBI defined in terms of the transitions of the resource–process terms described in Section 2 and of the bisimulation relation $\sim_\equiv$ described in Section 4. We sketch proofs of the Hennessy–Milner completeness result for the logic with this semantics.

5.1. MBI with $\sim$-semantics

In this section, we define a semantics for MBI in terms of the transitions of the resource–process terms and of the bisimulation relation $\sim$ described in Section 2. We prove the Hennessy–Milner completeness result for the logic with this semantics. To illustrate this, we provide an example that demonstrates how the concurrent composition of some bisimilar states, in SCRP, is not bisimilar, and how the corresponding states in our calculus can be concurrently composed in a way that preserves bisimilarity.

We define how atomic propositions are interpreted with respect to resource–process states. The mathematical structure on which we interpret MBI is the set **CState** of states generated by resources and processes. Recall that each state generates a transition structure, via the operational semantics rules (Fig. 1). 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 ($\mathbf{Act}, \mathbf{Res}, \mathbf{R}, \mu, \Delta, \mathbf{H}$) is fixed and $\sim$-resource-closed. Recall the bisimulation relation $\sim$. A set Σ of states is said to be $\sim$-closed if it satisfies the property

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

for all states R, E and S, F.

We now proceed to give an interpretation of the logical calculus on the set **CState** of closed states. Consider the relation $\sim$ restricted to **CState**. Let $\mathcal{P}_\sim(\mathbf{CState})$ be the set of all $\sim$-closed sets of closed states. A valuation is a function

$$\mathcal{V} : \mathrm{Prop} \to \mathcal{P}_\sim(\mathbf{CState})$$

from the set of propositional letters 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 given in Fig. 2, and in which every process that appears is required to be an agent. A model for MBI consists of the set of closed states together with such an interpretation. Satisfaction in a given model is then denoted $R, E \models \phi$, read as 'for the given model, the state R, E has property ϕ'. Clauses for the quantifiers can be adapted directly from the ones given in [11,14].

We define the notion of logical equivalence as follows:

Definition 47 *(Logical equivalence).* $R, E \equiv_{\mathrm{MBI}} S, F$ if and only if, for any model of MBI and all ϕ, $R, E \models \phi$ if and only if $S, F \models \phi$. $\square$

With this set-up, we can prove the forward direction of the Hennessy–Milner completeness theorem.

Theorem 48. *If $R, E \sim S, F$, then $R, E \equiv_{\mathrm{MBI}} S, F$.*

$$\begin{array}{lll}
R, E \models \mathrm{p} & \text{iff} & (R, E) \in \mathcal{V}(\mathrm{p}) \\
R, E \models \bot & & \text{never} \\
R, E \models \top & & \text{always} \\
R, E \models \neg\phi & \text{iff} & R, E \not\models \phi \\
R, E \models \phi_1 \vee \phi_2 & \text{iff} & R, E \models \phi_1 \text{ or } R, E \models \phi_2 \\
R, E \models \phi_1 \wedge \phi_2 & \text{iff} & R, E \models \phi_1 \text{ and } R, E \models \phi_2 \\
R, E \models \phi_1 \rightarrow \phi_2 & \text{iff} & R, E \models \phi_1 \text{ implies } R, E \models \phi_2 \\
R, E \models \langle a\rangle\phi & \text{iff} & \text{there exist } R', E', \text{ such that } R, E \xrightarrow{a} R', E', \text{ and } R', E' \models \phi \\
R, E \models [a]\phi & \text{iff} & \text{for all } R', E', \text{ if } R, E \xrightarrow{a} R', E', \text{ then } R', E' \models \phi \\
R, E \models I & \text{iff} & R, E \sim e, \mathbf{1} \\
R, E \models \phi_1 * \phi_2 & \text{iff} & \text{there exist } R_1, R_2, E_1, E_2, \text{ such that } R, E \sim R_1 \otimes R_2, E_1 \times E_2, \ R_1, E_1 \models \phi_1, \text{ and} \\
 & & \quad R_2, E_2 \models \phi_2 \\
R, E \models \phi_1 \mathrel{-\!\!*} \phi_2 & \text{iff} & \text{for all } S, F, \text{ if } S, F \models \phi_1 \text{ and } R \otimes S \downarrow, \text{ then } R \otimes S, E \times F \models \phi_2 \\
R, E \models \langle a\rangle_\nu\phi & \text{iff} & \text{there exist } S, F, R', S', E', F', \text{ such that } R \otimes S, E \times F \xrightarrow{a} R' \otimes S', E' \times F' \text{ and} \\
 & & \quad R' \otimes S', E' \times F' \models \phi \\
R, E \models [a]_\nu\phi & \text{iff} & \text{for all } S, F, R', S', E', F', \text{ if } R \otimes S, E \times F \xrightarrow{a} R' \otimes S', E' \times F', \text{ then } R' \otimes S', E' \times F' \models \phi.
\end{array}$$

Fig. 2. Satisfaction relation.

Proof. By induction over the structure of the satisfaction relation, $R, E \models \phi$.

Case $\phi = \mathrm{p}$. As $\mathcal{V}$ is a $\sim$-closed set, we have that if $R, E \in \mathcal{V}(p)$ and $R, E \sim S, F$, then $S, F \in \mathcal{V}(\mathrm{p})$. Hence we have that $S, F \models \mathrm{p}$.

Case $\phi = \bot$. As the premisses assume $R, E \models \bot$, we have a contradiction and can disregard this case.

Case $\phi = \top$. We have that $S, F \models \top$, straightforwardly.

Case $\phi = \phi_1 \vee \phi_2$. By the induction hypothesis, we know that $S, F \models \phi_1$ or $S, F \models \phi_2$. Hence we have that $S, F \models \phi_1 \vee \phi_2$.

Case $\phi = \phi_1 \wedge \phi_2$. By the induction hypothesis, we know that $S, F \models \phi_1$ and $S, F \models \phi_2$. Hence we have that $S, F \models \phi_1 \wedge \phi_2$.

Case $\phi = \phi_1 \rightarrow \phi_2$. By the induction hypothesis, we know that $S, F \models \phi_1$ whenever $R, E \models \phi_1$, and $S, F \models \phi_2$ whenever $R, E \models \phi_2$. Hence we have that $S, F \models \phi_1 \rightarrow \phi_2$.

Case $\phi = \langle a\rangle\, \psi$. As there exist R', E' such that $R, E \xrightarrow{a} R', E'$, by the definition of bisimulation, there exist S', F' such that $S, F \xrightarrow{a} S', F'$, and $R', E' \sim S', F'$. As $R', E' \models \psi$, by the induction hypothesis, we have that $S', F' \models \psi$, and hence we have that $S, F \models \langle a\rangle\, \psi$.

Case $\phi = [a]\psi$. Suppose some S', F' such that $S, F \xrightarrow{a} S', F'$. By the definition of bisimulation, there exist R', E', such that $R, E \xrightarrow{a} R', E'$. By the hypothesis, we have that $R', E' \models \psi$. By the induction hypothesis, we have that $S', F' \models \psi$. Hence we have that $S, F \models [a]\psi$.

Case $\phi = I$. By Theorem 27, as $e, \mathbf{1} \sim R, E$ and $R, E \sim S, F$, we have that $S, F \sim e, \mathbf{1}$. Hence we have that $S, F \models I$.

Case $\phi = \phi_1 * \phi_2$. By Theorem 27, as $R_1 \otimes R_2, E_1 \times E_2 \sim R, E$ and $R, E \sim S, F$, we have that $R_1 \otimes R_2, E_1 \times E_2 \sim S, F$. As $R_1, E_1 \models \phi_1$ and $R_2, E_2 \models \phi_2$, we have that $S, F \models \phi_1 * \phi_2$.

Case $\phi = \phi_1 \mathrel{-\!\!*} \phi_2$. Suppose some T, G such that $T, G \models \phi_2$ and $S \otimes T$ is defined.

As $R, E \sim S, F$ and $T, G \sim T, G$, by Definition 26, we have that $R \otimes T$ is defined. By the hypothesis, we have that $R \otimes T, E \times G \models \phi_2$. By Theorem 27, we have that $R \otimes T, E \times G \sim S \otimes T, F \times G$. By the induction hypothesis, we have that $S \otimes T, F \times G \models \phi_2$. Hence, we have that $S, F \models \phi_1 \mathrel{-\!\!*} \phi_2$.

Case $\phi = \langle a\rangle_\nu\psi$. By the hypothesis, there exist T, G, R', T', E', G' such that $R \otimes T, E \times G \xrightarrow{a} R' \otimes T', E' \times G'$ and $R' \otimes T', E' \times G' \models \psi$. As $R, E \sim S, F$, $T, G \sim T, G$, and $R \otimes T$ is defined, by Definition 26, we have that $S \otimes T$ is defined. By Theorem 27, we have that $R \otimes T, E \times G \sim S \otimes T, F \times G$. By the definition of bisimulation, there exist S', T'', F', G'' such that $S \otimes T, F \times G \xrightarrow{a} S' \otimes T'', F' \times G''$ and $R' \otimes T'', E' \times G'' \sim S' \otimes T', F' \times G'$. By the induction hypothesis, we have that $S' \otimes T'', F' \times G'' \models \psi$. Hence we have that $S, T \models \langle a\rangle_\nu\psi$.

Case $\phi = [a]_\nu\psi$. Suppose some T, G, S', T', F', G', such that $S \otimes T, F \times G \xrightarrow{a} S' \otimes T', F' \times G'$. As $R, E \sim S, F$, $T, G \sim T, G$, and $S \otimes T$ is defined, by Definition 26, we have that $R \otimes T$ is defined. By Theorem 27, we have that $R \otimes T, E \times G \sim S \otimes T, F \times G$. By the definition of bisimulation, we have that there exist R', T'', E', G'', such that $R \otimes T, E \times G \xrightarrow{a} R' \otimes T'', E' \times G''$ and $R' \otimes T'', E' \times G'' \sim S' \otimes T', F' \times G'$. By the hypothesis, we have that $R' \otimes T'', E' \times G'' \models \psi$. By the induction hypothesis, we have that $S' \otimes T', F' \times G' \models \psi$. Hence we have that $S, T \models [a]_\nu\psi$. □

Bisimilar states satisfy the same logical statements for the entire logic, including multiplicative implication and the multiplicative modalities. To prove this, it is necessary for bisimulation to be a congruence with respect to concurrent composition. In the set-up of [11,14], that is not the case. There, the operational semantics of concurrent composition does not preserve the allocation of resources to processes. Consequently, Theorem 48 fails to hold for $\langle a\rangle_\nu$, $[a]_\nu$, and $\mathrel{-\!\!*}$, which require congruence.

We demonstrate, with an example, that SCRP bisimulation is not a congruence, and hence how bisimilar SCRP states can satisfy different logical statements. Recall the $\hat{\ }$ notation from Section 3, which is used to denote SCRP resources, actions, and processes.

We use the free monoid over the atomic action $\{\hat{a}\}$, with composition ; and unit $\hat{1}$. We write $\hat{a}\hat{1}$ for $\hat{a};\hat{1}$. We use the resource monoid $\mathbf{S} = (\{\hat{e},\hat{s}\}, \circ, \hat{e})$, where

$$\hat{s}\circ\hat{e} = \hat{e}\circ\hat{s} = \hat{s} \qquad \hat{e}\circ\hat{e} = \hat{e} \qquad \hat{s}\circ\hat{s}\uparrow .$$

We define the SCRP modification function

$$\mu(\hat{1},\hat{e}) = \hat{e} \qquad \mu(\hat{1},\hat{s}) = \hat{s} \qquad \mu(\hat{a},\hat{s}) = \hat{s}.$$

We make use of the following SCRP operational rules:

$$\frac{}{\hat{r},\hat{a}:\hat{E} \xrightarrow{\hat{a}} \mu(\hat{a},\hat{r}),\hat{E}}\ \textsc{Act} \qquad \frac{\hat{r},\hat{E}_i \xrightarrow{\hat{a}} \hat{r}',\hat{E}_i'}{\hat{r},\hat{E}_1+\hat{E}_2 \xrightarrow{\hat{a}} \hat{r}',\hat{E}_i'}\ i\in\{1,2\}\quad \textsc{Sum}$$

$$\frac{\hat{r}=\hat{r}_1\circ\hat{r}_2 \quad \hat{r}_1,\hat{E}_1 \xrightarrow{\hat{a}_1} \hat{r}_1',\hat{E}_1' \quad \hat{r}_2,\hat{E}_2 \xrightarrow{\hat{a}_2} \hat{r}_2',\hat{E}_2'}{\hat{r},\hat{E}_1\times\hat{E}_2 \xrightarrow{\hat{a}_1\hat{a}_2} \hat{r}_1'\circ\hat{r}_2',\hat{E}_1'\times\hat{E}_2'}\ \textsc{Prod}.$$

Consider the processes

$$\hat{E} = (\hat{a}:\hat{\mathbf{0}}) + (\hat{1}:\hat{\mathbf{0}}) \qquad \hat{F} = \hat{1}:\hat{\mathbf{0}}.$$

By Definition 32, we have that $\hat{e},\hat{E}\approx\hat{e},\hat{F}$, as

$$\hat{e},\hat{E}\xrightarrow{\hat{1}}\hat{e},\hat{\mathbf{0}} \qquad \hat{e},\hat{F}\xrightarrow{\hat{1}}\hat{e},\hat{\mathbf{0}} \qquad \hat{e},\hat{E}\overset{\hat{a}}{\not\rightarrow} \qquad \hat{e},\hat{F}\overset{\hat{a}}{\not\rightarrow}.$$

Similarly, we have that $\hat{s},\hat{F}\approx\hat{s},\hat{F}$.

It is not the case, however, that $\hat{e}\circ\hat{s},\hat{E}\times\hat{F}\approx\hat{e}\circ\hat{s},\hat{F}\times\hat{F}$. Note that $\hat{e}\circ\hat{s}=\hat{s}$, and $\hat{a};\hat{1}=\hat{a}$. We can show the reduction

$$\frac{\hat{s}=\hat{s}\circ\hat{e} \quad \dfrac{\dfrac{\dfrac{\mu(\hat{a},\hat{s})=\hat{s}}{\hat{s},\hat{a}:\hat{\mathbf{0}}\xrightarrow{\hat{a}}\hat{s},\hat{\mathbf{0}}}\ \textsc{Act}}{\hat{s},(\hat{a}:\hat{\mathbf{0}})+(\hat{1}:\hat{\mathbf{0}})\xrightarrow{\hat{a}}\hat{s},\hat{\mathbf{0}}}\ \textsc{Sum} \quad \dfrac{\mu(\hat{1},\hat{e})=\hat{e}}{\hat{e},\hat{1}:\hat{\mathbf{0}}\xrightarrow{\hat{1}}\hat{e},\hat{\mathbf{0}}}\ \textsc{Act}}{}}{\hat{s},\hat{E}\times\hat{F}\xrightarrow{\hat{a};\hat{1}}\hat{s},\hat{\mathbf{0}}\times\hat{\mathbf{0}}}\ \textsc{Prod}.$$

We cannot, however, derive that $\hat{e}\circ\hat{s},\hat{F}\times\hat{F}\xrightarrow{\hat{a}}$. This difference can be displayed in the logic. We make use of the following fragment of the SCRP MBI logical interpretation:

$$\begin{array}{lll}
\hat{r},\hat{E} & \models \bot & \text{never} \\
\hat{r},\hat{E} & \models [\hat{a}]\phi & \text{iff for all } \hat{r}',\hat{E}', \text{if } \hat{r},\hat{E}\xrightarrow{\hat{a}}\hat{r}',\hat{E}', \text{then } \hat{r}',\hat{E}'\models\phi, \\
\hat{r},\hat{E} & \models \phi \mathrel{-\!\!*} \psi & \text{iff for all } \hat{s},\hat{F},\ \text{if } \hat{s},\hat{F}\models\phi, \text{then } \hat{r}\circ\hat{s},\hat{E}\times\hat{F}\models\psi.
\end{array}$$

Under the SCRP operational semantics, we have the following: first, $\hat{s},\hat{F}\models[\hat{a}]\bot$, as it cannot perform an $\hat{a}$ action; second, as $\hat{e}\circ\hat{s},\hat{E}\times\hat{F}\xrightarrow{\hat{a}}$, we can show that $\hat{e},\hat{E}\not\models([\hat{a}]\bot)\mathrel{-\!\!*}([\hat{a}]\bot)$. Suppose there exist $\hat{r},\hat{G}$ such that $\hat{r},\hat{G}\models[\hat{a}]\bot$ and $\hat{e}\circ\hat{r},\hat{E}\times\hat{G}\xrightarrow{\hat{a}}$. By the definition of the resource monoid, the modification function, and [11, Lemma 2], we have that $\hat{r}=\hat{s}$. By the SCRP Prod rule, there exists some $\hat{G}'$ such that $\hat{r},\hat{G}\xrightarrow{\hat{a}}\hat{r},\hat{G}'$, which contradicts the assumption that $\hat{r},\hat{G}\models[\hat{a}]\bot$. Hence, there exist no $\hat{r},\hat{G}$ such that $\hat{r},\hat{G}\models[\hat{a}]\bot$ and $\hat{e}\circ\hat{r},\hat{E}\times\hat{G}\xrightarrow{\hat{a}}$. Then, we have that $\hat{e},\hat{F}\models([\hat{a}]\bot)\mathrel{-\!\!*}([\hat{a}]\bot)$. As a result, we can then differentiate between $\hat{e},\hat{E}$ and $\hat{e},\hat{F}$ in the logic.

By contrast, in our calculus, concurrent composition is a congruence with respect to bisimulation. As a demonstration of this congruence property, we describe how we can model the above example in our calculus. Let $\mathbf{Act}=\{a\}$, $\mathbf{Res}=\{s,e\}$, with empty resource e, and $\mathbf{R}$ be the smallest set such that Definition 2 holds. Let the modification function μ' be the smallest such that $\mu'(a,s)=s$ and that is closed under the conditions in Definition 5. Consider the processes

$$E = a+1 \qquad F = 1.$$

We have that $e\oplus e, E\sim e,F$, as

$$e\oplus e,E\xrightarrow{1}e,\mathbf{0} \qquad e,F\xrightarrow{1}e,\mathbf{0} \qquad e,E\overset{a}{\not\rightarrow} \qquad e,F\overset{a}{\not\rightarrow}.$$

It *is* the case that $(e \oplus e) \otimes s, E \times F \sim e \otimes s, F \times F$. This is as neither can perform an action that is equivalent to a. We demonstrate this by witnessing the failed derivations.

$$\dfrac{\dfrac{\dfrac{\mu(a,e)\uparrow}{e, a \overset{a}{\not\rightarrow}}}{e \oplus e, a+1 \overset{a}{\not\rightarrow}} \quad \dfrac{\mu(1,e)=e}{s, F \xrightarrow{1} e, \mathbf{0}}}{(e \oplus e) \otimes s, E \times F \overset{a\cdot 1}{\not\rightarrow}} \qquad \dfrac{e, 1 \overset{a}{\not\rightarrow} \quad \dfrac{\mu(1,s)=s}{s, 1 \xrightarrow{1} s, \mathbf{0}}}{e \otimes s, F \times F \overset{a\cdot 1}{\not\rightarrow}}\ .$$

As a result, we have that both $e \oplus e, E \models ([a]\bot) \multimap\!\!* ([a]\bot)$ and $e, F \models ([a]\bot) \multimap\!\!* ([a]\bot)$.

Note that there exist analogous examples that demonstrate how bisimilar SCRP states do not satisfy logical properties defined in terms of either of the multiplicative modalities, $\langle a\rangle_\nu$ or $[a]_\nu$. Hence, in [11,14], the Hennessy–Milner completeness results only hold for the fragment of the logic excluding $\langle a\rangle_\nu$, $[a]_\nu$, and $\multimap\!\!*$.

The reverse direction of the Hennessy–Milner completeness theorem relies on image-finiteness.

Theorem 49. *If $R, E \equiv_{\mathrm{MBI}} S, F$, then $R, E \sim S, F$.*

Proof. Supposing that $R, E \equiv_{\mathrm{MBI}} S, F$, we require to show that $R, E \sim S, F$. Since $\sim$ is the *largest* relation closed under the conditions in Definition 22, it suffices to show that $\equiv_{\mathrm{MBI}}$ is closed under these conditions. That is (wlog), assuming that $R, E \xrightarrow{a} R', E'$, for some $a \in \mathbf{A}$, we have to exhibit S', F' such that $S, F \xrightarrow{a} S', F'$ and $R', E' \equiv_{\mathrm{MBI}} S', F'$.

Let $\mathscr{F} = \{S', F' \mid S, F \xrightarrow{a} S', F'\}$. If $\mathscr{F}$ is empty, then $R, E \models \langle a\rangle\top$ and $S, F \not\models \langle a\rangle\top$, contradicting $R, E \equiv_{\mathrm{MBI}} S, F$. Hence $\mathscr{F}$ is non-empty, and by the image-finiteness assumption, $\mathscr{F} = \{S_i, F_i \mid 1 \le i \le n\}$, for some finite n. Assume for contradiction that $R', E' \not\equiv_{\mathrm{MBI}} S_i, F_i$, for all $1 \le i \le n$. Thus there exist formulas $\phi_1, \ldots, \phi_n$ such that $R', E' \models \phi_i$ but $S_i, F_i \not\models \phi_i$, for all i. Hence $R, E \models \langle a\rangle(\phi_1 \wedge \ldots \wedge \phi_n)$ and $S, F \not\models \langle a\rangle(\phi_1 \wedge \ldots \wedge \phi_n)$, again contradicting $R, E \equiv_{\mathrm{MBI}} S, F$. Hence indeed $R', E' \equiv_{\mathrm{MBI}} S_i, F_i$, for some $S_i, F_i \in \mathscr{F}$, as required. □

We conclude this section, adopting the evident logical notation, with a useful equivalence between the additive and multiplicative modalities. The key point to note is the role of $\multimap\!\!*$ in introducing additional resources: in its absence, the modalities are distinct. Let $\phi \dashv\models \psi$ denote that, for R and E, $R, E \models \phi$ if and only if $R, E \models \psi$.

Proposition 50. *For any model of MBI, we have the following logical equivalences:*

(1) $\langle a\rangle_\nu \phi \dashv\models \neg(\top \multimap\!\!* \neg\langle a\rangle\phi)$*; and*
(2) $[a]_\nu \phi \dashv\models \top \multimap\!\!* [a]\phi$.

Proof.

(1) We show that, for all R, E we have that $R, E \models \langle a\rangle_\nu \phi$ if and only if $R, E \models \neg\big(\top \multimap\!\!* \neg(\langle a\rangle\phi)\big)$.

Suppose that $R, E \models \langle a\rangle_\nu \phi$. By the interpretation relation, there exist S, F, R', S', E', F', such that $R \otimes S, E \times F \xrightarrow{a}$, $R' \otimes S', E' \times F'$ and $R' \otimes S', E' \times F' \models \phi$. Suppose, for a contradiction, that $R, E \models \top \multimap\!\!* \neg(\langle a\rangle\phi)$. By the interpretation relation, for all T and G, if $T, G \models \top$, then $R \otimes T, E \times G \models \neg(\langle a\rangle\phi)$, and hence there do not exist R', T', E', G', such that $R \otimes T, E \times G \xrightarrow{a} R' \otimes T', E' \times G'$ and $R' \otimes T', E' \times G' \models \phi$. But we have already shown that S, F, S', and F' witness such a transition, equivalence, and logical entailment. Hence our supposition must be false, and we are done.

Conversely, suppose that $R, E \models \neg\big(\top \multimap\!\!* \neg(\langle a\rangle\phi)\big)$. By the interpretation relation, we have that $R, E \not\models \top \multimap\!\!* \neg(\langle a\rangle\phi)$, and hence that there exist some S and F such that $S, F \models \top$ and $R \otimes S, E \times F \not\models \neg(\langle a\rangle\phi)$. Again, by the interpretation relation, we have that there exist R', S', E', and F' such that $R \otimes S, E \times F \xrightarrow{a} R' \otimes S', E' \times F'$ and $R' \otimes S', E' \times F' \models \phi$. Hence we have that $R, E \models \langle a\rangle_\nu \phi$.

(2) We show that, for all R, E, we have that $R, E \models [a]_\nu \phi$ if and only if $R, E \models \top \multimap\!\!* [a]\phi$.

Suppose that $R, E \models [a]_\nu \phi$. By the interpretation relation, for all S, F, R', S', E', F', if $R \otimes S, E \times F \xrightarrow{a} R' \otimes S', E' \times F'$, then $R' \otimes S', E' \times F' \models \phi$. By the interpretation relation, $S, F \models \top$, and hence $R, E \models \top \multimap\!\!* [a]\phi$.

Conversely, suppose that $R, E \models \top \multimap\!\!* [a]\phi$. By the interpretation relation, for all S, F, if $S, F \models \top$, then $R \otimes S, E \times F \models [a]\phi$, and, furthermore, that for all R', S', E', F', if $R \otimes S, E \times F \xrightarrow{a} R' \otimes S', E' \times F'$, then $R' \otimes S', E' \times F' \models \phi$. Again, by the interpretation relation, $S, F \models \top$, and hence we have that $R, E \models [a]_\nu \phi$. □

5.2. MBI with $\sim_\equiv$-semantics

In this section, we demonstrate how the semantics of action modalities in Section 5.1 is very prescriptive in terms of the structure of the states that satisfy the modal formulae. We describe a semantics for MBI defined in terms of the transitions of the resource–process terms described in Section 2 and of the bisimulation relation $\sim_\equiv$ described in Section 4. We demonstrate how this relaxes the structural prescriptiveness of the semantics of the action modalities, and sketch proofs of the Hennessy–Milner completeness result for the logic with this semantics.

Recall that the structure of an action matches the concurrent structure ($\otimes$ and $\times$) of a state which performs that action. Hence, when using the interpretation of the logic in Fig. 2, the action modalities are very prescriptive about the internal structure of the states.

Example 51. Let the set of atomic actions be $\mathbf{Act} = \{a\}$, the set of atomic resources be $\mathbf{Res} = \{e, r\}$, with empty resource e, and the resource model $\mathbf{R}$ be the least set such that Definition 2 holds. Let the modification function $\mu : \mathbf{A} \times \mathbf{R} \rightharpoonup \mathbf{R}$ be the least function (under set inclusion of the domain) such that $\mu(a, r) = e$, and Definition 5 holds.

We have that $r, a \vDash \langle a \rangle \top$, but have that $r \otimes e, a \times 1 \nvDash \langle a \rangle \top$ and $e \otimes r, 1 \times a \nvDash \langle a \rangle \top$, as the former performs $a \cdot 1$ and the latter performs $1 \cdot a$. $\square$

If a is an atomic action, then the action modality $\langle a \rangle$ requires that any state that satisfies the modality does not consist of a concurrent composition. We may wish to define the satisfaction of action modalities in terms of actions that are equivalent to the action specified in the modality, rather than those that are exactly the same as the action specified in the modality. We define an interpretation $\vDash_\equiv$ such that $r \otimes e, a \times 1 \vDash_\equiv \langle a \rangle \top$ and $e \otimes r, 1 \times a \vDash_\equiv \langle a \rangle \top$, as $a \equiv a \cdot 1$ and $a \equiv 1 \cdot a$. The interpretation of a formula at a state is the interpretation of that formula at the corresponding transition structure in the ambient set of states. For the purposes of this section ($\mathbf{Act}, \mathbf{Res}, \mathbf{R}, \mu, \Delta, \mathbf{H}$) is fixed and $\sim_\equiv$-resource-closed. Recall the bisimulation relation $\sim_\equiv$. A set Σ of states is said to be $\sim_\equiv$-closed if it satisfies the property

$$R, E \in \Sigma \text{ and } R, E \sim_\equiv S, F \text{ implies } S, F \in \Sigma,$$

for all states R, E and S, F.

We now proceed to give an interpretation of the logical calculus on the set **CState** of closed states. Consider the relation $\sim_\equiv$ restricted to **CState**. Let $\mathcal{P}_{\sim_\equiv}(\mathbf{CState})$ be the set of all $\sim_\equiv$-closed sets of closed states. A valuation is a function

$$\mathcal{V} : \mathrm{Prop} \to \mathcal{P}_{\sim_\equiv}(\mathbf{CState})$$

from the set of propositional letters to $\sim_\equiv$-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 given in Fig. 3, and in which every process that appears is required to be an agent. An $\equiv$-model for MBI consists of the set of closed states together with such an interpretation. Satisfaction in a given $\equiv$-model is then denoted $R, E \vDash_\equiv \phi$, read as 'for the given $\equiv$-model, the state R, E has property ϕ, up to action equivalence relation $\equiv$'. Clauses for the quantifiers can be adapted directly from the ones given in [11,14].

We define the notion of logical equivalence for $\vDash_\equiv$ as follows:

Definition 52 *(Logical equivalence up to action equivalence).* $R, E \equiv_{\mathrm{MBI}} S, F$ if and only if, for any $\equiv$-model of MBI and all ϕ, $R, E \vDash_\equiv \phi$ if and only if $S, F \vDash_\equiv \phi$. $\square$

With this set-up, we can prove the forward direction of the Hennessy–Milner completeness theorem. Note that we use the congruence result proved in Section 4.

Theorem 53. *If $R, E \sim_\equiv S, F$, then $R, E \equiv_{\mathrm{MBI}} S, F$.*

Proof. By induction over the structure of the satisfaction relation, $R, E \vDash_\equiv \phi$. The proof follows similarly to that of Theorem 48. We provide illustrative cases.

Case $\phi = \langle a \rangle\, \psi$. As there exist b, R', E' such that $R, E \xrightarrow{b} R', E'$ and $a \equiv b$, by the definition of bisimulation, there exist c, S', F' such that $S, F \xrightarrow{c} S', F'$, $b \equiv c$, and $R', E' \sim_\equiv S', F'$. As $R', E' \vDash_\equiv \psi$, by the induction hypothesis, we have that $S', F' \vDash_\equiv \psi$. By Definition 38, we have that $a \equiv c$, and hence we have that $S, F \vDash_\equiv \langle a \rangle\, \psi$.

Case $\phi = \phi_1 \mathrel{-\!\!*} \phi_2$. Suppose some T, G such that $T, G \vDash \phi_2$ and $S \otimes T$ is defined. As $R, E \sim_\equiv S, F$ and $T, G \sim_\equiv T, G$, by Definition 44, we have that $R \otimes T$ is defined. By the hypothesis, we have that $R \otimes T, E \times G \vDash_\equiv \phi_2$. By Theorem 45, we have that $R \otimes T, E \times G \sim_\equiv S \otimes T, F \times G$. By the induction hypothesis, we have that $S \otimes T, F \times G \vDash_\equiv \phi_2$. Hence, we have that $S, F \vDash_\equiv \phi_1 \mathrel{-\!\!*} \phi_2$.

Case $\phi = \langle a \rangle_\nu \psi$. By the hypothesis, there exist T, G, R', T', E', G' such that $R \otimes T, E \times G \xrightarrow{b} R' \otimes T', E' \times G'$, $R' \otimes T', E' \times G' \vDash \psi$, and $a \equiv b$. As $R, E \sim_\equiv S, F$, $T, G \sim_\equiv T, G$, and $R \otimes T$ is defined, by Definition 44, we have that $S \otimes T$ is defined. By Theorem 45, we have that $R \otimes T, E \times G \sim_\equiv S \otimes T, F \times G$. By the definition of bisimulation, there exist c, S', T'', F', G'', such that $S \otimes T, F \times G \xrightarrow{c} S' \otimes T'', F' \times G''$, $b \equiv c$, and $R' \otimes T'', E' \times G'' \sim S' \otimes T', F' \times G'$. By Definition 38, we have that $a \equiv c$. By the hypothesis, we have that $S' \otimes T'', E' \times G'' \vDash_\equiv \psi$. By the induction hypothesis, we have that $S' \otimes T', F' \times G' \vDash_\equiv \psi$. Hence we have that $S, T \vDash_\equiv \langle a \rangle_\nu \psi$. $\square$

Again, the reverse direction of the Hennessy–Milner completeness theorem relies on image-finiteness.

Theorem 54. *If $R, E \equiv_{\mathrm{MBI}} S, F$, then $R, E \sim_\equiv S, F$.*

$$
\begin{array}{lll}
R, E \vDash_{\equiv} \mathrm{p} & \text{iff} & R, E \in \mathcal{V}(\mathrm{p}) \\
R, E \vDash_{\equiv} \bot & & \text{never} \\
R, E \vDash_{\equiv} \top & & \text{always} \\
R, E \vDash_{\equiv} \neg\phi & \text{iff} & R, E \nvDash_{\equiv} \phi \\
R, E \vDash_{\equiv} \phi_1 \vee \phi_2 & \text{iff} & R, E \vDash_{\equiv} \phi_1 \text{ or } R, E \vDash_{\equiv} \phi_2 \\
R, E \vDash_{\equiv} \phi_1 \wedge \phi_2 & \text{iff} & R, E \vDash_{\equiv} \phi_1 \text{ and } R, E \vDash_{\equiv} \phi_2 \\
R, E \vDash_{\equiv} \phi_1 \to \phi_2 & \text{iff} & R, E \vDash_{\equiv} \phi_1 \text{ implies } R, E \vDash_{\equiv} \phi_2 \\
R, E \vDash_{\equiv} \langle a \rangle \phi & \text{iff} & \text{there exist } b, R', E', \text{ such that } R, E \xrightarrow{b} R', E', a \equiv b, \text{ and } R', E' \vDash_{\equiv} \phi \\
R, E \vDash_{\equiv} [a] \phi & \text{iff} & \text{for all } b, R', E', \text{ if } R, E \xrightarrow{b} R', E' \text{ and } a \equiv b, \text{ then } R', E' \vDash_{\equiv} \phi \\
R, E \vDash_{\equiv} I & \text{iff} & R, E \sim_{\equiv} e, \mathbf{1} \\
R, E \vDash_{\equiv} \phi_1 * \phi_2 & \text{iff} & \text{there exist } R_1, R_2, E_1, E_2, \text{ such that } R, E \sim_{\equiv} R_1 \otimes R_2, E_1 \times E_2, \; R_1, E_1 \vDash_{\equiv} \phi_1, \text{ and} \\
& & \quad R_2, E_2 \vDash_{\equiv} \phi_2 \\
R, E \vDash_{\equiv} \phi_1 \mathrel{-\!\!*} \phi_2 & \text{iff} & \text{for all } S, F, \text{ if } S, F \vDash_{\equiv} \phi_1, \text{ then } R \otimes S, E \times F \vDash_{\equiv} \phi_2 \\
R, E \vDash_{\equiv} \langle a \rangle_\nu \phi & \text{iff} & \text{there exist } b, S, F, R', S', E', F', \text{ such that } R \otimes S, E \times F \xrightarrow{b} R' \otimes S', E' \times F', a \equiv b, \text{ and} \\
& & \quad R' \otimes S', E' \times F' \vDash_{\equiv} \phi \\
R, E \vDash_{\equiv} [a]_\nu \phi & \text{iff} & \text{for all } b, S, F, R', S', E', F', \text{ if } R \otimes S, E \times F \xrightarrow{b} R' \otimes S', E' \times F' \text{ and } a \equiv b, \\
& & \quad \text{then } R' \otimes S', E' \times F' \vDash_{\equiv} \phi.
\end{array}
$$

Fig. 3. Satisfaction relation for bisimulation up to equivalence.

Proof. Supposing that $R, E \equiv_{\mathrm{MBI}} S, F$, we require to show that $R, E \sim_{\equiv} S, F$. Since $\sim_{\equiv}$ is the *largest* relation closed under the conditions in Definition 22, it suffices to show that $\equiv_{\mathrm{MBI}}$ is closed under these conditions. That is (wlog), assuming that $R, E \xrightarrow{a} R', E'$, for some $a \in \mathbf{A}$, we have to exhibit S', F' such that $S, F \xrightarrow{a} S', F'$ and $R', E' \equiv_{\mathrm{MBI}} S', F'$.

Let $\mathscr{F} = \{S', F' \mid S, F \xrightarrow{a} S', F'\}$. If $\mathscr{F}$ is empty, then $R, E \vDash \langle a \rangle \top$ and $S, F \nvDash \langle a \rangle \top$, contradicting $R, E \equiv_{\mathrm{MBI}} S, F$. Hence $\mathscr{F}$ is non-empty, and by the image-finiteness assumption, $\mathscr{F} = \{S_i, F_i \mid 1 \leq i \leq n$, for some finite n. Assume for contradiction that $R', E' \not\equiv_{\mathrm{MBI}} S_i, F_i$, for all $1 \leq i \leq n$. Thus there exist formulas $\phi_1, \ldots, \phi_n$ such that $R', E' \vDash \phi_i$ but $S_i, F_i \nvDash \phi_i$, for all i. Hence $R, E \vDash \langle a \rangle (\phi_1 \wedge \ldots \wedge \phi_n)$ and $S, F \nvDash \langle a \rangle (\phi_1 \wedge \ldots \wedge \phi_n)$, again contradicting $R, E \equiv_{\mathrm{MBI}} S, F$. Hence indeed $R', E' \equiv_{\mathrm{MBI}} S_i, F_i$, for some $S_i, F_i \in \mathscr{F}$, as required. □

6. Discussion

This work suggests that the original ideas of resource semantics, though useful and influential in, say, separation logic, may warrant further exploration.

Specifically, we have shown that a technical difficulty present in an earlier formulation of the relationship between resources and processes — that is, the lack of the Hennessy–Milner completeness theorem for the full logic — can be resolved by moving to a version of resource semantics in which there is a closer combinatory match between the structure carried by resources and that carried by processes.

Nevertheless, the resulting resource semantics continues to support the semantics of connectives of the bunched logic BI [30,17] in the evident way. That is, for example,

$$R \vDash \phi_1 * \phi_2 \text{ iff there are } R_1 \text{ and } R_2 \text{ such that } R = R_1 \otimes R_2 \text{ and } R_1 \vDash \phi_1 \text{ and } R_2 \vDash \phi_2$$

and

$$R \vDash \phi_1 \wedge \phi_2 \text{ iff } R \vDash \phi_1 \quad \text{and} \quad R \vDash \phi_2.$$

Thus the additional combinatory structure does not appear to involve any loss of semantic utility from the motivating logical perspective.

Some conceptual and technical issues, beyond our present scope, remain to be addressed, however. In recent work in logic [16], one of us has considered a generalization of resource semantics to admit multi-dimensional satisfaction relations of the form, for example,

$$w, r \vDash \phi,$$

in which $w \in W$ are taken to be Kripke worlds (ordered by $\sqsubseteq$, say) in the sense of classical modal logic and $r \in R$, where R carries monoidal structure (with composition $\circ$, say), are interpreted as resources. In this set-up, we can define, informally for now, a modality $\diamond_s$ as

$$w, r \vDash \diamond_s \phi \text{ iff there is a world } w \sqsubseteq v \text{ such that } v, r \circ s \vDash \phi.$$

Such a modality is highly expressive and, among other things, generalizes the usual S4 modality [6]. It may be possible to define an analogous action modality, $\langle a \rangle_{S,F}$, which generalises our multiplicative modality $\langle a \rangle_\nu$:

$$R, E \models \langle a \rangle_{S,F}\, \phi \text{ iff there exist } R', S', E', F' \text{ such that } R \otimes S, E \otimes F \xrightarrow{a} R' \otimes S', E' \otimes F' \text{ and } R' \otimes S', E' \otimes F' \models \phi.$$

Note that, unlike in the previous definition, we add both a resource and a process component. We conjecture that the transition system employed in the body of this paper and the construction described above are both examples of a more general treatment of a more general multi-dimensional semantics that will have interpretations as a resource semantics. There would seem to be much to explore here.

A further question concerns the relationship between our work and concurrent separation logic [29]. Concurrent separation logic is built upon the resource semantics of bunched logic and handles concurrent processes in the style of Hoare logic. We conjecture that our treatment of resource semantics can be used to support CSL too.

In general, there is a more-or-less straightforward relationship between Hoare-style presentations of program logics and logically more standard presentations based on a satisfaction relation between a model and a propositional formula. Hoare-style systems are based on assertions of the form

$$\{\phi\}\, C\, \{\psi\},$$

for logical formulæ ϕ and ψ and program commands C, with inference rules — using an essentially Hilbert-style system — such as Composition and Consequence, respectively,

$$\frac{\{\phi\}S\{\psi\} \quad \{\psi\}T\{\chi\}}{\{\phi\}S;T\{\chi\}} \quad \text{and} \quad \frac{\phi_1 \to \phi_2 \quad \{\phi_2\}S\{\psi_2\} \quad \psi_1 \to \psi_2}{\{\phi_1\}S\{\psi_1\}}$$

and Conditional

$$\frac{\{\chi \wedge \phi\}S\{\psi\} \quad \{\neg\chi \wedge \phi\}T\{\psi\}}{\{\phi\}\,\text{if}\ \chi\ \text{then}\ S\ \text{else}\ T\{\psi\}},$$

for programs S and T.

Semantic presentations are formulated along the lines of

$$w \models_{\mathcal{M}} \phi,$$

where $\mathcal{M}$ is a model and w is a choice of world. In establishing the relationship between this view and Hoare-style presentations, we take a model with worlds given by program states (S, T, etc.) and consider how states evolve as programs perform actions C by executing commands; that is, $S \xrightarrow{C} T$. To see how this works we need to consider how such commands generate logical modalities. Define

$$S \models_{\mathcal{M}} [C]\phi \text{ iff for every evolution } S \xrightarrow{C} T,\ T \models_{\mathcal{M}} \phi,$$

which asserts that the program must have property ϕ after executing command c provided that whenever C evolves S to T, the state T has property ϕ. Thus, a Hoare-style assertion, $\{\phi\}C\{\psi\}$, in which the command C evolves the program state from S to T essentially corresponds to a semantic assertion

$$S \models_{\mathcal{M}} \phi \to [C]\psi.$$

Separation Logic [34] (Hoare-style presentation) and Pointer Logic [25] (semantic presentation) enrich this view of reasoning about programs by introducing the BI's concept of resource semantics in order to reason about mutable data structures.

In concurrent separation logic, the rule for the concurrent product of $n \geq 2$ commands has the form

$$\frac{\{\phi_1\}C_1\{\psi_1\}\ldots\{\phi_n\}C_n\{\psi_n\}}{\{\phi_1 * \ldots * \phi_n\}C_1 \times \cdots \times C_n\{\psi_1 * \ldots * \psi_n\}},$$

where no variable free in ϕ_i or ψ_i is changed in C_j when $j \neq i$.

In our setting, as explained in Section 5, the multiplicative conjunction is also intimately connected to concurrent product:

$$\begin{aligned} R, E \models \phi_1 * \phi_2 \text{ iff there exist } & R_1, E_1, R_2, E_2 \text{ such that } R, E \sim R_1 \otimes R_2, E_1 \times E_2 \\ & \text{and } R_1, E_1 \models \phi_1 \text{ and } R_2, E_2 \models \phi_2. \end{aligned}$$

By exploring the relationship between Hoare-style and semantic presentations of the program logic sketched above, we conjecture that it will be possible to give a systematic resource semantics for a wide range of concurrent phenomena (cf. [23]), including a synchronous semantics for concurrent separation logic (in contrast to Brookes' interleaving semantics [8]). Such a programme lies beyond the scope of this paper.

Acknowledgements

We are grateful to Jade Alglave, James Brotherston, Matthew Collinson, and Peter O'Hearn for their advice in writing this paper. This work has been partially supported by the UK Technology Strategy Board project TS/I002502/1, 'Trust Domains', and the UK EPSRC project EP/K033042/1, 'Algebra and Logic for Policy and Utility in Information Security'.

References

[1] G. Anderson, J. Brotherston, D. Pym, Hennessy–Milner completeness in resource–process calculus, Manuscript, UCL, 2015, http://www.cs.ucl.ac.uk/staff/D.Pym/resource_process_HM.pdf.
[2] A. Beautement, R. Coles, J. Griffin, C. Ioannidis, B. Monahan, D. Pym, A. Sasse, M. Wonham, Modelling the human and technological costs and benefits of USB memory stick security, in: M. Eric Johnson (Ed.), Managing Information Risk and the Economics of Security, Springer, 2008, pp. 141–163.
[3] Y. Beres, J. Griffin, S. Shiu, M. Heitman, D. Markle, P. Ventura, Analysing the performance of security solutions to reduce vulnerability exposure windows, in: Proc. Annual Computer Security Applications Conference, IEEE, Anaheim, California, 2008, pp. 33–42.
[4] Y. Beres, D. Pym, S. Shiu, Decision support for systems security investment, in: Proc. Business-Driven IT Management (BDIM), IEEE Xplore, 2010, pp. 118–125.
[5] J. Bergstra, J. Klop, Algebra of communicating processes with abstraction, Theoret. Comput. Sci. 37 (1) (1985) 77–121.
[6] B. Chellas, Modal Logic: An Introduction, Cambridge University Press, 1980.
[7] G. Birtwistle, Discrete Event Modelling on Simula, Springer, 1987.
[8] S. Brookes, A semantics for concurrent separation logic, Theoret. Comput. Sci. 375 (1–3) (2007) 227–270.
[9] T. Caulfield, D. Pym, J. Williams, Compositional Security Modelling: Structure, Economics, and Behaviour, LNCS, vol. 8533, 2014, pp. 233–245.
[10] T. Caulfield, D. Pym, Modelling and simulating systems security policy, in: Proc. 8th SIMUTools, ACM Digital Library, 2015.
[11] M. Collinson, D. Pym, Algebra and logic for resource-based systems modelling, Math. Structures Comput. Sci. 19 (5) (2009) 959–1027.
[12] M. Collinson, B. Monahan, D. Pym, A logical and computational theory of located resource, J. Logic Comput. 19 (2009) 1207–1244.
[13] M. Collinson, B. Monahan, D. Pym, Semantics for structured systems modelling and simulation, in: Proc. SIMUTools, vol. 34, ACM Digital Library, 2010, pp. 1–10.
[14] M. Collinson, B. Monahan, D. Pym, A Discipline of Mathematical Systems Modelling, College Publications, 2012.
[15] G. Coulouris, J. Dollimore, T. Kindberg, Distributed Systems: Concepts and Design, 3rd ed., Addison Wesley, 2000.
[16] J.-R. Courtault, D. Galmiche, D. Pym, A logic of separating modalities, Manuscript, UCL, 2015.
[17] D. Galmiche, D. Méry, D. Pym, The semantics of BI and resource tableaux, Math. Structures Comput. Sci. 15 (2005) 1033–1088.
[18] Hewlett–Packard Laboratories, A brief introduction to structured modelling with Core Gnosis, http://www.hpl.hp.com/research/systems_security/gnosis.html.
[19] M. Hennessy, A calculus for costed computations, Log. Methods Comput. Sci. 7 (1) (2011) 1–35.
[20] M. Hennessy, G. Plotkin, On observing nondeterminism and concurrency, in: LNCS, vol. 85, 1980, pp. 299–308.
[21] M. Hennessy, R. Milner, Algebraic laws for nondeterminism and concurrency, J. ACM 32 (1) (1985) 137–161.
[22] Hewlett–Packard Laboratories, Towards a science of risk analysis, http://www.hpl.hp.com/news/2011/oct-dec/security_analytics.html.
[23] T. Hoare, Generic models of the laws of programming, in: LNCS, vol. 8051, 2013, pp. 213–226.
[24] T. Hoare, P. O'Hearn, Separation logic semantics for communicating processes, Electron. Notes Theor. Comput. Sci. 212 (2008) 3–25, http://dx.doi.org/10.1016/j.entcs.2008.04.050.
[25] S. Ishtiaq, P. O'Hearn, BI as an assertion language for mutable data structures, in: C. Hankin, D. Schmidt (Eds.), Proc. 28th ACM POPL, ISBN 1-58113-336-7, 2001, pp. 14–26.
[26] F. Mattern, Virtual time and global states of distributed systems, Parallel Distrib. Algorithm 1 (23) (1989) 215–226.
[27] R. Milner, Calculi for synchrony and asynchrony, Theoret. Comput. Sci. 25 (3) (1983) 267–310.
[28] R. Milner, Communication and Concurrency, Prentice Hall, New York, 1989.
[29] P. O'Hearn, Resources, concurrency, and local reasoning, Theoret. Comput. Sci. 375 (1–3) (2007) 271–307.
[30] P. O'Hearn, D. Pym, The logic of bunched implications, Bull. Symbolic Logic 5 (2) (June 1999) 215–244.
[31] G. Plotkin, A structural approach to operational semantics, Technical report DAIMI FN-19, Department of Computer Science, Aarhus University, 1981.
[32] D. Pym, P. O'Hearn, H. Yang, Possible worlds and resources: the semantics of BI, Theoretical Computer Science 315 (1) (2004) 257–305. Erratum: p. 285, l. -12: ', for some P', $Q = P; P'$' should be '$P \vdash Q$'.
[33] S. Read, Relevant Logic, Basil Blackwell, 1988.
[34] J. Reynolds, Separation logic: a logic for shared mutable data structures, in: Proceedings of the 17th IEEE Symposium on Logic in Computer Science, IEEE, 2002, pp. 55–74.
[35] R. de Simone, Higher-level synchronising devices in Meije-SCCS, Theoret. Comput. Sci. 37 (1985) 245–267.
[36] W. Vogels, Eventually consistent, Commun. ACM 52 (1) (2009) 40–44, http://dx.doi.org/10.1145/1435417.1435432.