Two-sorted algebraic decompositions of Brookes's shared-state denotational semantics

Yotam Dvir[1], Ohad Kammar[2], Ori Lahav[1], and Gordon Plotkin[2]

[1] Tel Aviv University yotamdvir@mail.tau.ac.il orilahav@tau.ac.il
[2] University of Edinburgh ohad.kammar@ed.ac.uk gdp@inf.ed.ac.uk

Abstract. We define a two sorted equational theory of algebraic effects that models concurrent shared state with preemptive interleaving, recovering Brookes's seminal 1996 trace-based model precisely. The decomposition allows us to analyse Brookes's model algebraically in terms of separate but interacting components. The multiple sorts partition terms into layers. We use two sorts: a "hold" sort for layers that disallow interleaving of environment memory accesses, analogous to holding a global lock on the memory; and a "cede" sort for the opposite. The algebraic signature comprises of independent interlocking components: two new operators that switch between these sorts, delimiting the atomic layers, thought of as acquiring and releasing the global lock; non-deterministic choice; and state-accessing operators. The axioms similarly divide cleanly: the delimiters behave as a closure pair; all operators are strict, and distribute over non-empty non-deterministic choice; and non-deterministic global state obeys Plotkin and Power's presentation of global state. Our representation theorem expresses the free algebras over a two-sorted family of variables as sets of traces with suitable closure conditions. When the held sort has no variables, we recover Brookes's trace semantics. We define several other single- and two-sorted theories to elucidate the connection to Brookes's model via translation embeddings and equivalences.

Keywords: shared state · concurrency · denotational semantics · monads · algebraic effects · equational theory · multi-sorted algebra · trace semantics · representability · join semilattices · closure pairs · mnemoids · global state

1 Introduction

We decompose Brookes's pioneering denotational model of concurrent shared state under preemptive interleaving [6] using algebraic effects [30]. This model possesses several desirable features in the area of denotational models for programming languages with concurrent features. (I) It is based on traces, an elementary sequential gadget. (II) It is fully compositional, as in traditional denotational semantics for shared-state [e.g. 15, 17]. Each syntactic programming construct, including parallel composition, has a corresponding semantic operation combining the meanings of its constituents. Such full compositionality

contrasts with some recent models in this area that require additional 'semantic post-processing': some form of quotient, pruning of auxiliary mathematical constructs, reasoning up-to behavioural equivalence; or capture only sequential blocks, reasoning about the parallel composition on a separate layer [e.g. 7, 8, 19, 21]. (III) Subsequent variations and extensions [4, 38, 39], as well as adaptations to relaxed memory models [12, 13, 21], attest to its versatility, making it a cornerstone in the denotational semantics for concurrent languages with side-effects. (IV) It achieves a high level of abstraction, evident in the many compiler transformations that the model supports, including the most common memory access introductions and eliminations, and the laws of parallel programming. Moreover, Brookes showed the model to be fully abstract in a language extended with the `await` construct, which blocks execution until all memory locations contain a given tuple of values, and then atomically updates them to contain another tuple of values. This construct is not a natural programming construct, but is clearly suggested by Brookes's semantics.

Plotkin and Power's modern theory of *algebraic effects* [30] refines Moggi's monadic approach [26] with algebraic theories. The algebraic approach informs the monadic structure by identifying semantic counterparts to syntactic constructs and axiomatising their semantics equationally. The monadic structure emerges through the well-established connection between algebraic theories and monads [23] via *representation theorems*. For example: global state emerges by axiomatising memory lookup and update [30] and a representation theorem involving the state monad; non-determinism emerges by axiomatising semi-lattices and a representation theorem involving the powerdomains [15, 28]; and so on. The algebraic perspective may offer insights into the making of the denotational semantics. It can suggest methods for combining different effects and modularly augment a semantics with a given computational effect [17].

The connection between algebraic effects and concurrency has long been emphasised. For example, the ability to use algebraic effects, without any axioms, and their *effect handlers* [3, 32, 33] to allow users to define their own schedulers was the original motivation for their implementation in the OCaml programming language [9, 10, 35]. Nonetheless, exhibiting abstract models such as Brookes's algebraically via equational axiomatisation of syntactic constructs has proved challenging. Our own previous algebraic model [11] invalidates a key transformation, reflecting a fundamental limitation.

To overcome this limitation, we use multi-sorted algebraic theories, a direction that was raised in personal discussions since the earliest work on algebraic effects [30]. A multi-sorted algebraic term decomposes into layers. Our two sorts represent two modes of interaction between a program fragment and its concurrent environment. A "hold" sort ($\bullet$) provides a reasoning layer in which the environment may not interfere, whereas in the "cede" sort ($\circ$) it may. We provide two operators that switch between these sorts. Our core idea is to axiomatise these operators as a *closure pair*, an established order-theoretic special Galois-connection, the dual to the domain-theoretic embedding-projection pairs [1]. Additionally, we axiomatise strict distributivity of the closure pair over non-

determinism. The remaining axioms, all in the hold sort, are strikingly independent from these axioms. In our shared-state theory $\mathbb{S}$, the remaining axioms are precisely those of non-deterministic global state.

We prove, twice over, that $\mathbb{S}$ recovers Brookes's model in the cede sort. First, using sets of traces akin to Brookes's, we define a representation of $\mathbb{S}$. The representation recovers Brookes's model via the adjunction that forgets the hold sort. Second, we define three algebraic theories for Brookes's `await` and its sequential variant, relating them to global-state, shared-state, and each other via embeddings and equivalences. The theory for concurrent `await` is straightforwardly represented by Brookes's model, and embeds in the cede sort of $\mathbb{S}$.

Caveats In our development, we opt for mathematical simplicity whenever possible. For example, we use countable-join semilattices instead of finite-join semilattices to represent non-determinism. This choice streamlines the development leading up to the representation theorem, allowing us to use countable sets instead of finitely generated ones. We also do not treat recursion to avoid the complexity that a domain-theoretic account incurs. The resulting model—identical to Brookes's—coincides with the elided domain-theoretic model over discrete predomains. This model also supports iteration (i.e. `while`-loops) without change thanks to countable-joins. It also supports first-order recursion without change by equipping it with a domain-theoretic structure. These compromises let us focus on the core concepts, and provide a relatively elementary exposition and a clear presentation of the underlying idea, motivating future inquiry.

2 Overview

Equational theories study terms constructed from algebraic operators (§3.1). In Plotkin and Power's algebraic theory of effects, the operators represent fundamental program effects, and their arguments represent continuations. Equational axioms reflect fundamental relationships between the operators. The equations that hold in the theory, reflecting the semantics as a whole, are those that follow from its axiomatic presentation by equational logic (§3.2).

In the global-state theory, a theory for sequential stateful computation, the operators U and L represent updating and looking up bits in memory. For example, consider the global-state term $\mathsf{U}_{\mathtt{y},0}\,\mathsf{L}_{\mathtt{y}}(3, \mathsf{U}_{\mathtt{x},1}\,\mathsf{U}_{\mathtt{y},1}\,7)$. After updating `y` to `0`, the computation looks `y` up: if it finds `0`, it returns 3; if it finds `1`, it updates `x` and `y` to `1` in succession, then returns 7. Between the update and the lookup, the value at `y` cannot change. Therefore, the computation finds the value `0`, and takes the left-hand continuation. The global-state axiom (UL) $\mathsf{U}_{\ell,b}\,\mathsf{L}_{\ell}(x_0, x_1) = \mathsf{U}_{\ell,b}\,x_b$ reflects this fact. By (UL), $\mathsf{U}_{\mathtt{y},0}\,\mathsf{L}_{\mathtt{y}}(3, \mathsf{U}_{\mathtt{x},1}\,\mathsf{U}_{\mathtt{y},1}\,7) = \mathsf{U}_{\mathtt{y},0}\,3$ holds in global state.

A representing monadic model for an equational theory (§3.3) interprets the algebraic operators as corresponding operations over the model's domain; such that each term, up-to equality in the theory, is represented uniquely in the domain. Interpretations respect the theory; that is, applying an operation to representations of terms results in the representation of the corresponding operator applied to said terms: $[\![O]\!]_{\mathrm{op}}([\![t_1]\!]_{\mathrm{term}}, \ldots, [\![t_\alpha]\!]_{\mathrm{term}}) = [\![O(t_1, \ldots, t_\alpha)]\!]_{\mathrm{term}}$.

For example, global-state terms are represented by memory-manipulating functions in the state-monad model. This model interprets update by precomposing a state update $[\![\mathsf{U}_{\ell,b}]\!]_{\mathrm{op}} f = f \circ [\ell \mapsto b]$; and interprets lookup by passing the input memory state σ along to the σ_ℓ-continuation $[\![\mathsf{L}_\ell]\!]_{\mathrm{op}}(f_0, f_1) = \lambda\sigma.\, f_{\sigma_\ell}\sigma$. In this way, global state recovers the (historically precedent) state monad.

The state monad does not account for concurrent interference. The monad underlying Brookes's denotational semantics does, by using sequences of transitions to denote potential behaviours (§6.1). Each transition $\langle \sigma, \rho \rangle$ in sequence means that the computation, by relying on exclusive access to the memory at state σ, can guarantee to provide the state ρ and yield to the environment.

Following the tradition of algebraic effects, we wish to recover Brookes's model using an equational theory for shared state. This is rather straightforward using a single-sorted theory B (§6.3) in which transitions appear as operators. However, this theory is dissatisfying, for two reasons. (I) Transitions do not correspond to familiar programming constructs, but to Brookes artificial `await` construct. (II) Conceptualising shared state as global state with concurrent interference, we expect the global-state effects to be present in a theory of shared state, and the equations between them to hold when interference is prohibited.

A more appropriate approach adds an operator Y to global state for yielding control to the environment. Otherwise, the computation has exclusive access to memory. This direction lead to a Brookes-like model [11]. However, unlike Brookes's model, it does not validate the Irrelevant Read Introduction (IRI) program transformation, which introduces a read instruction that possibly yields to the environment and discards the value it read. The IRI transformation is useful as a stepping stone to other practical transformations, such as common-subexpression elimination from conditionals, and consequently loops. Invalidating IRI seems to be a fundamental limitation of the yield-operator approach, as we show in the extended manuscript [14, §A]. In retrospect, we can pinpoint the issue to Y both releasing exclusive access to memory, and acquiring it back. Our key insight is that the mode of computation that cedes access to memory needs to be explicit, decomposing Y into a pair of mode-switching operators.

The remaining structure falls into place straightforwardly and naturally, in our proposed two-sorted theory for shared state $\mathbb{S}$ (§4). Each sort represents a computation mode: *hold* ($\bullet$) represents the computation's exclusive access to memory; *cede* ($\circ$) represents positions in which the environment may interfere. Each operator has a sort and expects each continuation to have a specific sort. In $\mathbb{S}$, update $\mathsf{U} : \bullet\langle\bullet\rangle$ and lookup $\mathsf{L} : \bullet\langle\bullet, \bullet\rangle$ are $\bullet$-sorted and expect $\bullet$-sorted continuations, allowing us to reason about interference-free stateful interactions.

The theory $\mathbb{S}$ also supports non-deterministic choice in both sorts. For example, the term $(\mathsf{U}_{\mathtt{x},1}\, 2 \vee \mathsf{U}_{\mathtt{x},1}\, 5)$ either updates $\mathtt{x}$ to $\mathtt{1}$ and returns 2, or updates $\mathtt{x}$ to $\mathtt{1}$ and returns 5. We axiomatise $\mathbb{S}$ such that each operator distributes over non-deterministic choice, e.g. $(\mathsf{U}_{\mathtt{x},1}\, 2 \vee \mathsf{U}_{\mathtt{x},1}\, 5) = \mathsf{U}_{\mathtt{x},1}(2 \vee 5)$. We order terms by potential behaviours $(l \geq r) := (l = l \vee r)$, a partial-order for $\mathbb{S}$-equality (§3.2).

The mode-switching operators of $\mathbb{S}$ are $\lhd : \circ\langle\bullet\rangle$ and $\rhd : \bullet\langle\circ\rangle$. That is, $\lhd$ is $\circ$-sorted and expects a $\bullet$-sorted continuation, and vice versa for $\rhd$. We think of

them as delimiting atomic blocks, or acquiring and releasing an abstract global lock. We axiomatise them in $\mathbb{S}$ by strict distributivity over countable joins, i.e. (ND-$\lhd$) $\bigvee_{i<\alpha} \lhd x_i = \lhd \bigvee_{i<\alpha} x_i$ and (ND-$\rhd$) $\bigvee_{i<\alpha} \rhd x_i = \rhd \bigvee_{i<\alpha} x_i$; and:

Empty $(\lhd \rhd y = y)$**.** An empty atomic block has no observable effect.
Fuse $(\rhd \lhd x \geq x)$**.** Fusing atomic blocks eliminates potential interference.

These axiomatise $\lhd$ and $\rhd$ as an *(insertion)-closure pair* [e.g. 1].

Each $\mathbb{S}$-term denotes a set of sort-delimited traces (§5.1), which generalise Brookes's traces. For example, $t := \mathsf{U}_{\mathtt{y},0} \rhd \lhd \mathsf{L}_{\mathtt{y}}(3, \mathsf{U}_{\mathtt{x},1}\, \mathsf{U}_{\mathtt{y},1} \rhd 7)$ denotes a set that includes $\bullet\langle \binom{\mathtt{x}\mapsto 1}{\mathtt{y}\mapsto 1}, \binom{\mathtt{x}\mapsto 1}{\mathtt{y}\mapsto 0}\rangle\langle \binom{\mathtt{x}\mapsto 1}{\mathtt{y}\mapsto 1}, \binom{\mathtt{x}\mapsto 0}{\mathtt{y}\mapsto 0}\rangle\circ 7$. Indeed, we can read off a corresponding computation from t, initially holding the lock in the state $\binom{\mathtt{x}\mapsto 1}{\mathtt{y}\mapsto 1}$ of both bits `1`: the computation updates `y` to `0`, then yields to the environment before looking `y` up; finding `1`, it updates `x` and `y` to `1`, then releases the lock and returns 7. Brookes's original traces correspond to those delimited by $\circ$ on both ends.

With these traces we define our two-sorted generalisation of Brookes's model, and prove that it represents $\mathbb{S}$ (§5.2). The $\circ$-sorted $\circ$-valued fragment (§6.2), which represents the "block-closed" terms, is Brookes's original model (§6.1).

We also provide an algebraic perspective on the representation, by $\circ$-embedding (§6.4) the transitions-theory B into $\mathbb{S}$ (§5.5). This embedding maps $\langle \sigma, \rho \rangle$ to $\lhd \{\!| \sigma, \rho |\!\} \rhd$, where $\{\!| \sigma, \rho |\!\} : \bullet\langle \bullet \rangle$ is defined by global-state operators (§5.2).

3 Preliminaries

We present a standard treatment of countably-infinitary multi-sorted equational theories and their free models [e.g. 2, 37], straightforwardly generalising the single-sorted case by assigning sorts to functions and their arguments. The reader may choose to skim/skip this section, consulting it as necessary.

3.1 Terms

We define the logical language of multi-sorted equational logic. The basic vocabulary of multi-sorted algebra is parameterised by a set $\mathbf{sort}$ whose elements $\square, \Diamond$ we call *sorts*. We will mostly focus on the *single-sorted* case ($\mathbf{sort} = \{\star\}$) and the *two-sorted* case ($\mathbf{sort} = \{\bullet, \circ\}$). A $\mathbf{sort}$*-scheme* $\vec{\square} \in \mathrm{Scheme}\,\mathbf{sort}$ is a countable sequence of sorts from $\mathbf{sort}$, i.e. a finite sequence $\vec{\square} = \langle \square_0, \ldots, \square_{n-1} \rangle$ of length n, or countably infinite sequence $\vec{\square} = \langle \square_0, \square_1, \ldots \rangle$ of length ω, where $\square_i \in \mathbf{sort}$ for all i. For example: the empty scheme $\mathbf{0} := \langle\rangle$ of length 0; and the constant schemes $\alpha \cdot \square := \langle \square \rangle_{i<\alpha}$ of length α. We write $\square$ for the scheme $1 \cdot \square$.

A $\mathbf{sort}$*-sorted signature* $\Sigma = \langle \mathbf{op}_\Sigma, \mathbf{ar}_\Sigma \rangle$ consists of a set of *operators* $\mathbf{op}_\Sigma$ and an *arity* assignment $\mathbf{ar}_\Sigma : \mathbf{op}_\Sigma \to \mathbf{sort} \times \mathrm{Scheme}\,\mathbf{sort}$. For $O \in \mathbf{op}_\Sigma$ with $\mathbf{ar}_\Sigma O = \langle \square, \langle \Diamond_i \rangle_i \rangle$, we write $(O : \square\, \langle \Diamond_i \rangle_{i<\alpha}) \in \Sigma$. The operator O will allow us to construct a $\square$-sort term with a tuple of terms, with the i^{th} subterm having sort $\Diamond_i$. For single-sorted arities ($\mathbf{sort} = \{\star\}$), we write $O : \alpha$ for $O : \star\,(\alpha \cdot \star)$. A *signature* is a set $\mathbf{sort}_\Sigma$ and a $\mathbf{sort}_\Sigma$-sorted signature we also denote by Σ.

We will use the following signature to model non-deterministic choice:

Example 1. The *join semilattice* single-sorted signature $\mathtt{J}$ consists of two operators: *join* $\vee : 2$, i.e. $\vee : \star\,\langle\star, \star\rangle$; and *bottom* $\bot : 0$, i.e. $\bot : \star\,\langle\rangle$. □

To simplify the formulation of our representation theorem later, we generalise the signature to countable non-deterministic choice operators:

Example 2. The *countable-join semilattice* single-sorted signature $\mathtt{V}$ consists of an α-ary *choice* operator $\bigvee_\alpha : \alpha$ for every $\alpha \leq \omega$. In particular, the signature $\mathtt{J}$ is included with $\alpha = 2$ (join) and $\alpha = 0$ (bottom). □

The final example demonstrates the treatment for multiple sorts:

Example 3. The *finite dimensional transformations* signature $\mathtt{M}$ consists of a sort for each pair of natural numbers $\mathbf{sort}_{\mathtt{M}} := \{\mathbf{Hom}\,(m,n) \mid m,n \in \mathbb{N}\}$, an identity operator $\mathrm{Id}_n : \mathbf{Hom}\,(n,n)\,\langle\rangle$ for each $n \in \mathbb{N}$, and, for each triple $m,n,k \in \mathbb{N}$, a composition operator $(\circ_{m,n,k}) : \mathbf{Hom}\,(m,k)\,\langle \mathbf{Hom}\,(n,k)\,, \mathbf{Hom}\,(m,n)\rangle$. □

A signature generates a language of algebraic terms as follows. A $\mathbf{sort}$-*family* $\boldsymbol{X} \in \mathbf{Set}^{\mathbf{sort}}$ is an assignment of a set $\boldsymbol{X}_{\square}$, to each sort $\square \in \mathbf{sort}$. We identify $\mathbf{Set}^{\{\star\}} \cong \mathbf{Set}$, and use a set-like notation to specify families, e.g. $\boldsymbol{X} := \{x : \bullet, y, z : \circ\}$ is the two-sorted family $\boldsymbol{X}_\bullet := \{x\}$ and $\boldsymbol{X}_\circ := \{y, z\}$. We can turn every $\mathbf{sort}$-family $\boldsymbol{X}$ into the set $\oint \boldsymbol{X} := \coprod_{\square \in \mathbf{sort}} \boldsymbol{X}_{\square}$ equipped with the injections $\mathrm{in}_{\square} : \boldsymbol{X}_{\square} \to \oint \boldsymbol{X}$. This construction is a special case of the Grothendieck construction, and lets us track the distinction between sets and families.

For a signature Σ and $\mathbf{sort}_\Sigma$-family $\boldsymbol{X} \in \mathbf{Set}^{\mathbf{sort}_\Sigma}$, define the $\mathbf{sort}_\Sigma$-family of Σ-*terms over* $\boldsymbol{X}$: $\mathrm{Term}^\Sigma \boldsymbol{X} \in \mathbf{Set}^{\mathbf{sort}_\Sigma}$, $\mathrm{Term}^\Sigma_{\square} \boldsymbol{X} := \{t \mid \boldsymbol{X} \vdash_\Sigma t : \square\}$ inductively:

$$\frac{(x : \square) \in \boldsymbol{X}}{\boldsymbol{X} \vdash_\Sigma x : \square} \qquad \frac{(O : \square\,\langle \Diamond_i \rangle_{i<\alpha}) \in \Sigma \qquad \forall i.\, \boldsymbol{X} \vdash_\Sigma t_i : \Diamond_i}{\boldsymbol{X} \vdash_\Sigma O\,\langle t_i \rangle_{i<\alpha} : \square}$$

Here, the elements $x \in \boldsymbol{X}_{\square}$, written $(x : \square) \in \boldsymbol{X}$, represent variables of sort $\square$. We may drop the set-brackets left of a trunstile, e.g. write $x : \bullet, y, z : \circ \vdash_\Sigma y : \circ$; and omit the sorts, especially in the single-sorted case, e.g. write $x, y \vdash_{\mathtt{J}} x \vee \bot$. For $t \in \mathrm{Term}^\Sigma_{\square} \boldsymbol{X}$, we write $\boldsymbol{X} \vdash_\Sigma \psi := t : \square$ to define ψ as t, e.g. $x, y \vdash_{\mathtt{J}} \psi := x \vee \bot$.

A $\mathbf{sort}$-*sorted map* $f : \boldsymbol{X} \to \boldsymbol{Y}$ is a $\mathbf{sort}$-indexed tuple of functions between the corresponding sets: $f_{\square} : \boldsymbol{X}_{\square} \to \boldsymbol{Y}_{\square}$, for every $\square \in \mathbf{sort}$. Our development utilises sorted maps extensively. A *(simultaneous) substitution* $\boldsymbol{X} \vdash_\Sigma \theta : \boldsymbol{Y}$ is a sorted function $\theta : \boldsymbol{Y} \to \mathrm{Term}^\Sigma \boldsymbol{X}$, specifying which $\square$-term $\boldsymbol{X} \vdash_\Sigma \theta_{\square} y : \square$ to substitute for each variable $y \in \boldsymbol{Y}_{\square}$. Each such substitution determines a sorted map $[\theta] : \mathrm{Term}\boldsymbol{Y} \to \mathrm{Term}\boldsymbol{X}$ inductively, which we write in post-fix notation:

$$(\boldsymbol{Y} \vdash_\Sigma y : \square)\,[\theta] := (\boldsymbol{X} \vdash_\Sigma \theta_{\square} y : \square) \qquad (\boldsymbol{Y} \vdash_\Sigma O\,\langle t_i \rangle_i)\,[\theta] := (\boldsymbol{X} \vdash_\Sigma O\,\langle t_i\,[\theta] \rangle_i)$$

3.2 Equational logic

A $\square$-*sorted* Σ-*equation in context* $\boldsymbol{X}$ is a pair $\langle l, r\rangle \in \mathrm{Term}^\Sigma_{\square} \boldsymbol{X}$ of $\square$-sorted Σ-terms over $\boldsymbol{X}$. We write this situation as $\boldsymbol{X} \vdash_\Sigma l = r : \square$, or just $l = r$, and call l the left-hand side (LHS) and r the right-hand side (RHS) of the equation. A *presentation* $\mathfrak{p}$ consists of a signature $\Sigma_{\mathfrak{p}}$ and *axioms*: a set $\mathrm{Ax}_{\mathfrak{p}}$ of Σ-equations.

$$\frac{\boldsymbol{X} \vdash_{\Sigma_{\mathfrak{p}}} t : \square}{\boldsymbol{X} \vdash_{\mathfrak{p}} t = t : \square} \qquad \frac{\boldsymbol{X} \vdash_{\mathfrak{p}} t_2 = t_1 : \square}{\boldsymbol{X} \vdash_{\mathfrak{p}} t_1 = t_2 : \square} \qquad \frac{\boldsymbol{X} \vdash_{\mathfrak{p}} t_1 = t_2 : \square \qquad \boldsymbol{X} \vdash_{\mathfrak{p}} t_2 = t_3 : \square}{\boldsymbol{X} \vdash_{\mathfrak{p}} t_1 = t_3 : \square}$$

$$\frac{(\boldsymbol{X} \vdash_{\Sigma_{\mathfrak{p}}} t_1 = t_2 : \square) \in \mathrm{Ax}_{\mathfrak{p}}}{\boldsymbol{X} \vdash_{\mathfrak{p}} t_1 = t_2 : \square} \qquad \frac{\boldsymbol{Y} \vdash_{\mathfrak{p}} t_1 = t_2 : \square \qquad \boldsymbol{X} \vdash_{\Sigma_{\mathfrak{p}}} \theta : \boldsymbol{Y}}{\boldsymbol{X} \vdash_{\mathfrak{p}} t_1 [\theta] = t_2 [\theta] : \square}$$

$$\frac{\boldsymbol{Y} \vdash_{\Sigma_{\mathfrak{p}}} t : \square \qquad \boldsymbol{X} \vdash_{\Sigma_{\mathfrak{p}}} \theta, \theta' : \boldsymbol{Y} \qquad \forall (y : \diamondsuit) \in \boldsymbol{Y}. \boldsymbol{X} \vdash_{\mathfrak{p}} \theta_{\diamondsuit} y = \theta'_{\diamondsuit} y : \diamondsuit}{\boldsymbol{X} \vdash_{\mathfrak{p}} t [\theta] = t [\theta'] : \square}$$

Fig. 1. Multi-sorted equational logic with countable arities

Example 4. The *join semilattice* presentation J consists of the signature $\Sigma_{\mathsf{J}} := \mathtt{J}$ of example 1, and the axioms Ax_{J} below:

(Associativity) $x \vee (y \vee z) = (x \vee y) \vee z$ (Idempotency) $x \vee x = x$
(Commutativity) $x \vee y = y \vee x$ (Neutrality) $x \vee \bot = x$ □

Example 5. The *countable-join semilattice* presentation V consists of the signature $\Sigma_{\mathsf{V}} := \mathtt{V}$ of example 2, and the axioms Ax_{V}:

(ND-return) $\bigvee_{i<1} x_i = x_0$
(ND-squash) $\bigvee_{i<\alpha} \bigvee_{j<\beta_i} x_{i,j} = \bigvee_{k<\gamma} x_{fk}$ where $f : \gamma \twoheadrightarrow \coprod_{i<\alpha} \beta_i$ □

Example 6. The *finite dimensional transformations* presentation M consists of the signature $\Sigma_{\mathsf{M}} := \mathtt{M}$ of example 3 and the axioms Ax_{M} below, suppressing the sort indices (each axiom scheme includes every possible instantiation):

(L-Id) $\mathrm{Id} \circ f = f$ (R-Id) $f \circ \mathrm{Id} = f$ (Assoc) $f \circ (g \circ h) = (f \circ g) \circ h$ □

Figure 1 presents the deductive system called *equational logic*. We say that a presentation $\mathfrak{p}$ *proves* an equation, writing $\boldsymbol{X} \vdash_{\mathfrak{p}} t_1 = t_2 : \square$, when it is derivable from $\mathrm{Ax}_{\mathfrak{p}}$ using these standard equational reasoning rules, namely: reflexivity, symmetry, transitivity, use of an axiom, substitution, and congruence. This logic is monotone: assuming more axioms allows us to prove more equations. The *algebraic theory* of a presentation $\mathfrak{p}$ is the smallest derivation-closed set of equations containing the axioms. We denote the theory of $\mathfrak{p}$ by $\mathfrak{p}$ as well.

Example 7. We can prove $\{x, y : \star\} \vdash_{\mathsf{J}} (x \vee \bot) \vee y = x \vee y : \star$ using an instance of Neutrality and reflexivity with the following instance of congruence:

$$\{z, y : \star\} \vdash_{\mathsf{J}} t := z \vee y : \star \qquad \theta_\star := \begin{pmatrix} z \mapsto x \vee \bot \\ y \mapsto y \end{pmatrix} \qquad \theta'_\star := \begin{pmatrix} z \mapsto x \\ y \mapsto y \end{pmatrix} \qquad \square$$

When a presentation $\mathfrak{p}$ proves the semi-lattice axioms in one of its sorts $\square$, then the encoding $(\boldsymbol{X} \vdash_{\Sigma_{\mathfrak{p}}} l \leq r : \square) := (\boldsymbol{X} \vdash_{\Sigma_{\mathfrak{p}}} l \vee r = r : \square)$ of inequations as equations in this sort is a preorder that is a partial order w.r.t. $\mathfrak{p}$-equality, i.e. $(\boldsymbol{X} \vdash_{\mathfrak{p}} s \leq t : \square) \wedge (\boldsymbol{X} \vdash_{\mathfrak{p}} t \leq s : \square) \implies (\boldsymbol{X} \vdash_{\mathfrak{p}} s = t : \square)$. We encode $(\geq)$ similarly. Due to the monotonicity property of equational logic, once we have

included an axiomatisation of semi-lattices through a subset of the axioms, we may proceed to postulate inequations.

We also use a generalisation of distributivity axioms [18], reproducing familiar arithmetic distributivity equations such as $x \cdot \max\{y_1, y_2\} = \max\{x \cdot y_1, x \cdot y_2\}$, the distributivity of $(\cdot)$ over max in the right-hand-side position. The extended manuscript [14, §B] details the straightforward, but technical generalisation. The main message is as follows. In a given presentation $\mathfrak{p}$, if all operators distribute over binary joins in every position, the congruence rule is valid for inequations:

$$\frac{\boldsymbol{Y} \vdash_{\Sigma_{\mathfrak{p}}} t : \square \qquad \boldsymbol{X} \vdash_{\Sigma_{\mathfrak{p}}} \theta, \theta' : \boldsymbol{Y} \qquad \forall (y : \Diamond) \in \boldsymbol{Y}. \boldsymbol{X} \vdash_{\mathfrak{p}} \theta_{\Diamond} y \leq \theta'_{\Diamond} y : \Diamond}{\boldsymbol{X} \vdash_{\mathfrak{p}} t\,[\theta] \leq t\,[\theta'] : \square}$$

If a presentation $\mathfrak{p}$ supports semi-lattices in every sort and they distribute over binary joins in every positions, then we say that $\mathfrak{p}$ *supports inequational reasoning*. The theory of $\mathfrak{p}$ then admits Bloom's logic for ordered algebraic theories [5]. We let future work determine the most appropriate variety of inequational logic [29].

Going forward, all of our presentations support inequational reasoning in this sense, and all operators distribute over arbitrary non-empty joins, not just the binary ones. Moreover, they are all strict: $O(\bot, \dots, \bot) = \bot$ for every operator $(O : \square \langle \Diamond_i \rangle_{i<\alpha}) \in \Sigma_{\mathfrak{p}}$. Such theories 'absorb' side-effects when their continuations diverge, an inherent 'partial correctness' property of Brookes's model.

3.3 Algebras and models

After presenting the proof theory—equational logic—let's turn to the model theory of universal algebra. A Σ-*algebra* $\mathbf{A}$ consists of a $\mathbf{sort}_\Sigma$-family $\underline{\mathbf{A}} \in \mathbf{Set}^{\mathbf{sort}_\Sigma}$, the *carrier*, and an assignment $\mathbf{A}[\![-]\!]_{\mathrm{op}}$, for each operator $(O : \square \langle \Diamond_i \rangle_{i<\alpha}) \in \Sigma$, of an *operation* over this carrier: $\mathbf{A}[\![O]\!]_{\mathrm{op}} : (\prod_{i<\alpha} \underline{\mathbf{A}}_{\Diamond_i}) \to \underline{\mathbf{A}}_{\square}$.

Example 8. For any set X, define the $\mathtt{V}$-algebra $\mathbf{V}X$ by taking the carrier to be the set of countable (finite or infinite) X-subsets $\underline{\mathbf{V}X} := \mathcal{P}^{\aleph_0}(X)$, and interpret choice as union $\mathbf{V}X[\![\bigvee_\alpha]\!]_{\mathrm{op}}\langle D_i\rangle_{i<\alpha} := \bigcup_{i<\alpha} D_i$. □

Example 9. Define the $\mathtt{M}$-algebra $\mathbf{M}$ by taking the carrier to be the set of real-valued matrices of the corresponding dimensions, $\underline{\mathbf{M}}_{\mathbf{Hom}(m,n)} := \mathbb{M}^{\mathbb{R}}_{m\times n}$, interpret the identity $\mathbf{M}[\![\mathrm{Id}_n]\!]_{\mathrm{op}} := I_n \in \mathbb{M}^{\mathbb{R}}_{n\times n}$ as the identity matrix, and composition $\mathbf{M}[\![(\circ)]\!]_{\mathrm{op}} := (\cdot)$ as matrix multiplication.

Let $\mathbf{A}$ be an $\mathtt{M}$-algebra. Define the *opposite* algebra $\mathbf{A}^{\mathsf{op}}$ by exchanging dimensions. So $\underline{\mathbf{A}^{\mathsf{op}}_{\mathbf{Hom}(m,n)}} := \underline{\mathbf{A}}_{\mathbf{Hom}(n,m)}$, the same identity $\mathbf{A}^{\mathsf{op}}[\![\mathrm{Id}_n]\!]_{\mathrm{op}} := \mathbf{A}[\![\mathrm{Id}_n]\!]_{\mathrm{op}}$, and reversing composition $\mathbf{A}^{\mathsf{op}}[\![(\circ)]\!]_{\mathrm{op}}(A, B) := \mathbf{A}[\![(\circ)]\!]_{\mathrm{op}}(B, A)$. □

Example 10 (term algebra). The Σ-terms with variables from $\boldsymbol{X}$ carry a canonical algebra structure $\mathbf{F}^\Sigma\boldsymbol{X}$, given by $\underline{\mathbf{F}^\Sigma\boldsymbol{X}} := \mathrm{Term}^\Sigma\boldsymbol{X}$, with each O-term constructor as the corresponding O-operation: $(\mathbf{F}^\Sigma\boldsymbol{X})[\![O]\!]_{\mathrm{op}}\langle t_i\rangle_i := O\langle t_i\rangle_i$. □

A Σ-*algebra homomorphism* $\varphi : \mathbf{A} \to \mathbf{B}$ is a sorted-function $\varphi : \underline{\mathbf{A}} \to \underline{\mathbf{B}}$ that preserves the operations: $\varphi_{\square}(\mathbf{A}[\![O]\!]_{\mathrm{op}}(a_1, \dots, a_\alpha)) = \mathbf{B}[\![O]\!]_{\mathrm{op}}(\varphi_{\Diamond_1} a_1, \dots, \varphi_{\Diamond_\alpha} a_\alpha)$.

Example 11. Transposing real-valued matrices $(-)^\top : \mathbb{M}^{\mathbb{R}}_{m\times n} \to \mathbb{M}^{\mathbb{R}}_{n\times m}$ is a homomorphism $(-)^\top : \mathbf{M} \to \mathbf{M}^{\mathsf{op}}$, by the well-known identity $(A \cdot B)^\top = B^\top \cdot A^\top$. □

A Σ-algebra allows us to interpret every Σ-term, by assigning values to its variables. Formally, let $\mathbf{A}$ be a Σ-algebra. An $\boldsymbol{X}$-*environment in* $\mathbf{A}$ is a sorted function $e : \boldsymbol{X} \to \underline{\mathbf{A}}$. Given such an environment, interpret terms by induction:

$$\mathbf{A}\,[\![\boldsymbol{X} \vdash_\Sigma x : \square]\!]_{\mathrm{term}}\, e := e_\square x \qquad \mathbf{A}\,[\![O\,\langle t_i\rangle_i]\!]_{\mathrm{term}}\, e := \mathbf{A}\,[\![O]\!]_{\mathrm{op}}\,\langle \mathbf{A}\,[\![t_i]\!]_{\mathrm{term}}\, e\rangle_i$$

Example 12 (substitution). An $\boldsymbol{X}$-environment in $\mathbf{F}^\Sigma\boldsymbol{X}$ amounts to a substitution, and interpreting terms in $\mathbf{F}^\Sigma\boldsymbol{X}$ amounts to substitution. □

Example 13 (evaluation homomorphism). Evaluation using an $\boldsymbol{X}$-environment $e : \boldsymbol{X} \to \underline{\mathbf{A}}$ in a Σ-algebra $\mathbf{A}$ is a homomorphism $\mathbf{A}[\![-]\!]_{\mathrm{term}} e : \mathbf{F}^\Sigma\boldsymbol{X} \to \mathbf{A}$. □

A Σ-algebra $\mathbf{A}$ *validates* the equation $\boldsymbol{X} \vdash_\Sigma l = r : \square$ when evaluation in all environments equates its sides: $\mathbf{A}[\![l]\!]_{\mathrm{term}} e = \mathbf{A}[\![r]\!]_{\mathrm{term}} e$ for all $e : \boldsymbol{X} \to \underline{\mathbf{A}}$. We then write $\mathbf{A} \vdash \boldsymbol{X} \vdash_\Sigma l = r : \square$. A $\mathfrak{p}$-*model* is an algebra validating all of $\mathrm{Ax}_\mathfrak{p}$. The soundness theorem of equational logic states that every $\mathfrak{p}$-model validates all the equations in the algebraic theory of $\mathfrak{p}$.

Example 14. Referring to previous examples, the algebras $\mathbf{V}X$ are V-models, the algebras $\mathbf{M}$ and $\mathbf{M}^{\mathsf{op}}$ are M-models, and algebras of terms are $\emptyset$-models. □

Example 15. Consider the Σ_J-algebra $\mathbf{A}$ for which the carrier is the set of natural numbers $\underline{\mathbf{A}} := \mathbb{N}$, join interprets as addition $\mathbf{A}[\![\vee]\!]_{\mathrm{op}}(m,n) := m+n$, and bottom as zero $\mathbf{A}[\![\bot]\!]_{\mathrm{op}} := 0$. This is *not* a J-model, since, taking $e : \{x : \star\} \to \underline{\mathbf{A}}$ with $ex = 1$, we get $\mathbf{A}[\![x \vee x]\!]_{\mathrm{term}} e \neq \mathbf{A}[\![x]\!]_{\mathrm{term}} e$; and so $\mathbf{A} \not\vdash x : \star \vdash_\mathsf{J} x \vee x = x : \star$. □

We end this section with representations of free models. These are $\mathfrak{p}$-models whose elements represent the $\Sigma_\mathfrak{p}$-terms up-to provable equality in $\mathfrak{p}$.

A $\mathfrak{p}$-*model* $\langle \mathbf{A}, e\rangle$ *over a family* $\boldsymbol{X}$ consists of a $\mathfrak{p}$-model $\mathbf{A}$ and an $\boldsymbol{X}$-environment in it $e : \boldsymbol{X} \to \underline{\mathbf{A}}$. A *free* $\mathfrak{p}$-model $\langle \mathbf{A}, \mathrm{return}\rangle$ over a family $\boldsymbol{X}$ is then a $\mathfrak{p}$-model over $\boldsymbol{X}$ such that every environment in every $\mathfrak{p}$-model $e : \boldsymbol{X} \to \underline{\mathbf{B}}$ extends uniquely along return to a $\mathfrak{p}$-homomorphism $e^\# : \mathbf{A} \to \mathbf{B}$, i.e., for all $x \in \boldsymbol{X}_\square$, we have: $e^\#_\square(\mathrm{return}_\square\, a) = ea$. We then say that the algebra $\mathbf{A}$ *represents* $\boldsymbol{X}$-environments via the assignment $e \mapsto e^\#$, the corresponding *representation*.

The algebraic theory of effects [30] emphasises the role free models play in denotational semantics for programming languages with effects. In particular, given a free $\mathfrak{p}$-model over $\boldsymbol{X}$ for every family $\boldsymbol{X}$, one standardly obtains a monad suitable for the denotational semantics of a language with computational effects conforming to the operators in $\mathfrak{p}$.

Example 16. For any set X, the V-algebra $\mathbf{V}X$ given by the countable powerset in example 8 represents X-environments; together with return $x := \{x\}$ it forms a free V-model over X. The representation assigns $e : X \to \underline{\mathbf{B}}$ to $e^\# : \mathbf{V}X \to \mathbf{B}$, defined $e^\# D := \mathbf{B}[\![\bigvee_{|D|}]\!]_{\mathrm{op}}\langle ex\rangle_{x\in D}$; how it enumerates D doesn't matter since $\mathbf{B}$ is a V-model. The data $\langle X \mapsto \underline{\mathbf{V}X}, \mathrm{return}, (-)^\#\rangle$ is a monad. □

4 Shared state

To define the equational theory of shared state, we first recall the standard, single sorted *(non-deterministic) global state* theory G [17, 25, 30]. The variant we present here has countable non-determinism, and the global state operators manipulate a common memory store $\mathbb{S} := \mathbb{L} \to \mathbb{B}$ with a finite set of locations $\mathbb{L} \neq \emptyset$ each storing a bit $\mathbb{B} := \{\mathtt{0}, \mathtt{1}\}$. A larger finite set of storable-values would not be conceptually different. Infinite sets of storable-values or locations work similarly with more involved representation theorems. In concrete examples, we let $\mathbb{L} = \{\mathtt{x}, \mathtt{y}\}$ and use non-bracketed vectors for stores, e.g. $\begin{smallmatrix}1\\0\end{smallmatrix}$ denotes $\left(\begin{smallmatrix}\mathtt{x}\mapsto 1\\ \mathtt{y}\mapsto 0\end{smallmatrix}\right)$.

The signature Σ_{G} consists of the countable-join semilattice operators (example 2), as well as two kinds of memory-access operators: *lookup* operators $\mathsf{L}_\ell : 2$, to look a location $\ell \in \mathbb{L}$ up and branch according to the value found; and *update* operators $\mathsf{U}_{\ell,b} : 1$, to update a location $\ell \in \mathbb{L}$ to the value $b \in \mathbb{B}$. The global state axioms Ax_{G} consists of the countable-join semilattice axioms (example 5), as well as the following:

Non-deterministic global state (omitting semilattice axioms)

(UL)	$\mathsf{U}_{\ell,b}\,\mathsf{L}_\ell(x_0, x_1) = \mathsf{U}_{\ell,b}\,x_b$	(LU)	$\mathsf{L}_\ell(\mathsf{U}_{\ell,0}\,x, \mathsf{U}_{\ell,1}\,x) = x$
(UU)	$\mathsf{U}_{\ell,b'}\,\mathsf{U}_{\ell,b}\,x = \mathsf{U}_{\ell,b}\,x$	(ND-U)	$\bigvee_{i<\alpha} \mathsf{U}_{\ell,b}\,x_i = \mathsf{U}_{\ell,b} \bigvee_{i<\alpha} x_i$
(UUc)	$\mathsf{U}_{\ell,b}\,\mathsf{U}_{\ell',b'}\,x = \mathsf{U}_{\ell',b'}\,\mathsf{U}_{\ell,b}\,x$	where $\ell \neq \ell'$	

The induced algebraic theory G includes axioms of less succinct presentations of the same theory [25]. For example, lookup also distributes over binary join, so the theory admits inequational reasoning; consecutively looking the same location up can be merged, e.g. $x_0, x_1, y \vdash_{\mathsf{G}} \mathsf{L}_\ell(\mathsf{L}_\ell(x_0, x_1), y) = \mathsf{L}_\ell(x_0, y)$; and other combinations of looking-up and updating different locations commute, e.g. for any $\ell \neq \ell'$ we have $x_0, x_1 \vdash_{\mathsf{G}} \mathsf{L}_\ell(\mathsf{U}_{\ell',b}\,x_0, \mathsf{U}_{\ell',b}\,x_1) = \mathsf{U}_{\ell',b}\,\mathsf{L}_\ell(x_0, x_1)$.

Our two-sorted presentation $\mathbb{SS}$ of *shared state* extends global state. Its sorts are $\mathbf{sort}_{\Sigma_{\mathbb{SS}}} = \{\bullet, \circ\}$. The *hold* sort ($\bullet$) represents an uninterrupted sequence of memory accesses, whereas the *cede* sort ($\circ$) allows control to pass to the environment. The operators and the arities of the signature $\Sigma_{\mathbb{SS}}$ consist of a copy of Σ_{G} at $\bullet$, a copy of Σ_{V} at $\circ$, and new operators $\lhd : \circ\langle\bullet\rangle$ and $\rhd : \bullet\langle\circ\rangle$.

The intuitive reading for algebraic effects is from the outside in. With this intuition, one interpretation of the operators $\lhd$ and $\rhd$ is to acquire and release a global lock. The hold sort ($\bullet$) represents the lock being held by one of the threads in the program. The cede sort ($\circ$) represents points in the execution in which one of the threads in the concurrent environment may acquire the lock. The sorts ensure exclusive access to the lock, and therefore to the store. In an alternative interpretation, these operators delimit atomic blocks; their sorts prevent nesting.

The shared state axioms $\mathrm{Ax}_{\mathbb{SS}}$ include a copy of the (non-deterministic) global state axioms Ax_{G} at $\bullet$ and a copy of the countable-join semilattice axioms Ax_{V} at $\circ$. In particular, $\mathbb{SS}$ proves the semi-lattice axioms in both sorts. It further includes standard strict distributivity axioms for the new unary operators:

Strict distributivity of $\lhd$ and $\rhd$

(ND-$\lhd$) $\bigvee_{i<\alpha} \lhd x_i = \lhd \bigvee_{i<\alpha} x_i$ (ND-$\rhd$) $\bigvee_{i<\alpha} \rhd x_i = \rhd \bigvee_{i<\alpha} x_i$

With these axioms, $\mathbb{SS}$ supports inequational reasoning, which represents the semantic refinement relation used to validate program transformations [e.g. 11].

Finally, $\mathrm{Ax}_{\mathbb{SS}}$ axiomatises $\lhd$ and $\rhd$ as an *(insertion)-closure pair* [e.g. 1]:

Closure pair (Empty) $\lhd \rhd y = y$ (Fuse) $\rhd \lhd x \geq x$

They are compatible with the global-lock interpretation:

Empty $(\lhd \rhd y = y)$**.** Acquiring and immediately releasing the lock has no effect on the sequence of effects that can occur as a result of arbitrary interleavings.

Fuse $(\rhd \lhd x \geq x)$**.** Releasing and immediately acquiring the lock only allows more behaviours. The environment may or may not interleave there.

To summerise, $\mathrm{Ax}_{\mathbb{SS}} := \mathrm{Ax}^{\bullet}_{\mathsf{G}} \cup \mathrm{Ax}^{\circ}_{\mathsf{V}} \cup \{\text{ND-}\rhd, \text{ND-}\lhd\} \cup \{\text{Empty}, \text{Fuse}\}$.

Example 17. The $\Sigma_{\mathbb{SS}}$-equations appearing below are named after corresponding transformations that may or may not be valid, depending on the setting (e.g. is there concurrency, and under what assumptions), all $\circ$-sorted over $\{x : \circ\}$:

$$\lhd \mathsf{L}_\ell(\rhd x, \rhd x) = x \qquad \text{(Irrelevant Read Intro \& Elim)}$$
$$\lhd \mathsf{U}_{\ell,b_1} \rhd \lhd \mathsf{U}_{\ell,b_2} \rhd x \geq \lhd \mathsf{U}_{\ell,b_2} \rhd x \qquad \text{(Write Elim)}$$
$$\lhd \mathsf{U}_{\ell,b_1} \rhd \lhd \mathsf{U}_{\ell,b_2} \rhd x \leq \lhd \mathsf{U}_{\ell,b_2} \rhd x \qquad \text{(Write Intro)}$$

Intuitively, Irrelevant Read Intro & Elim should be valid in our setting, as looking a value up is not observable by the environment, and the computation itself disregards the value. Write Elim should be valid too, because it is possible that the environment does not look ℓ up at the interference point between the updates on the LHS, covering the behaviour denoted by the RHS. On the other hand, Write Intro should be invalid in our setting because only on the LHS can a concurrently running thread look ℓ up and find b_1. Formally, we will show $\mathbb{SS}$ does not prove Write Intro in example 25. Here we show $\mathbb{SS}$ proves the other two:

$$\begin{aligned}
\lhd \mathsf{L}_\ell\left(\rhd x, \rhd x\right) &\overset{\text{LU}}{=} \lhd \mathsf{L}_\ell\left(\mathsf{U}_{\ell,0}\, \mathsf{L}_\ell\left(\rhd x, \rhd x\right), \mathsf{U}_{\ell,1}\, \mathsf{L}_\ell\left(\rhd x, \rhd x\right)\right) \\
&\overset{\text{UL}}{=} \lhd \mathsf{L}_\ell\left(\mathsf{U}_{\ell,0} \rhd x, \mathsf{U}_{\ell,1} \rhd x\right) \overset{\text{LU}}{=} \lhd \rhd x \overset{\text{Empty}}{=} x \\
\lhd \mathsf{U}_{\ell,b_1} \rhd \lhd \mathsf{U}_{\ell,b_2} \rhd x &\overset{\text{Fuse}}{\geq} \lhd \mathsf{U}_{\ell,b_1}\, \mathsf{U}_{\ell,b_2} \rhd x \overset{\text{UU}}{=} \lhd \mathsf{U}_{\ell,b_2} \rhd x
\end{aligned}$$

□

5 Representation

We now establish the representation theorem describing a free $\mathbb{SS}$-model over any $\boldsymbol{X} \in \mathbf{Set}^{\{\bullet,\circ\}}$. Following Brookes [6], we use sets of traces to denote behaviours.

5.1 Sorted traces

A *sorted trace* starts with a sort ($\bullet$ or $\circ$) followed by a non-empty sequence of state transitions, and ending in a sorted value. The initial sort in the trace and the initial store in each transition represent assumptions the trace relies on from its concurrent and sequential environment. The final sort and value and the final store in each transition represent guarantees the trace makes to its environment.

Formally, a *(state) transition* is a pair $\langle \sigma, \rho \rangle \in \mathbb{S} \times \mathbb{S}$. Let $\xi^? \in (\mathbb{S} \times \mathbb{S})^*$ range over possibly empty sequences of transitions, and $\xi \in (\mathbb{S} \times \mathbb{S})^+$ range over non-empty ones. For any set X, define the set of *X-valued Brookes traces* $\mathsf{T}X := (\mathbb{S} \times \mathbb{S})^+ \times X$, also used in Brookes's model (§6). For any family $\boldsymbol{X} \in \mathbf{Set}^{\{\bullet,\circ\}}$ define the $\{\bullet,\circ\}$-sorted family $\mathbf{T}\boldsymbol{X}$ of *traces* $(\mathbf{T}\boldsymbol{X})_{\square} := \mathsf{T} \oint \boldsymbol{X}$. Then, for any sorted family $\boldsymbol{X} \in \mathbf{Set}^{\{\bullet,\circ\}}$, we define the set of *sorted traces over* $\boldsymbol{X}$ by:

$$\mathbb{T}\boldsymbol{X} := \oint \mathbf{T}\boldsymbol{X} = \{\bullet, \circ\} \times (\mathbb{S} \times \mathbb{S})^+ \times \coprod_{\diamondsuit \in \{\bullet,\circ\}} \boldsymbol{X}_{\diamondsuit}$$

A *$\square$-sorted $\diamondsuit$-valued trace* is one of the form $\square \xi \diamondsuit x := \langle \square, \xi, \mathrm{in}_{\diamondsuit}\, x \rangle$ in the set $\mathbb{T}\boldsymbol{X}$.

Example 18. $\bullet\langle {}^{1}_{1}, {}^{1}_{0}\rangle\langle {}^{1}_{1}, {}^{0}_{0}\rangle\circ 7 \in \mathbb{T}\boldsymbol{X}$, with $\boldsymbol{X}_{\circ} = \mathbb{N}$, is $\bullet$-sorted and $\circ$-valued. □

Intuitively, the trace $\square \xi \diamondsuit x$ models a potential behaviour, or protocol, that a shared-state program phrase under preemptive interleaving concurrency can exhibit, or adhere to, given as a rely/guarantee sequence.

Example 19. The behaviour denoted by $\bullet\langle {}^{1}_{1}, {}^{1}_{0}\rangle\langle {}^{1}_{1}, {}^{0}_{0}\rangle\circ 7$ relies on the preceding environment for ${}^{1}_{1}$ and for the sequential environment to hold access to the store; then guarantees ${}^{1}_{0}$; then relies on ${}^{1}_{1}$; and finally guarantees ${}^{0}_{0}$, and returns 7 to the succeeding sequential environment, ceding exclusive store access. □

One can make these trace-semantic concepts more formal, for example, when formulating an adequacy proof w.r.t. an operational semantics. We will not define these concepts formally since we will not need the additional level of rigour, for example, because we appeal to the well-established adequacy of Brookes's model.

We implicitly understand the exclusive access to the store is ceded ($\circ$) between transitions. For example, for the trace $\bullet\langle {}^{1}_{1}, {}^{1}_{0}\rangle\langle {}^{1}_{1}, {}^{0}_{0}\rangle\circ 7$, we could write $\bullet\langle {}^{1}_{1}, {}^{1}_{0}\rangle\circ\langle {}^{1}_{1}, {}^{0}_{0}\rangle\circ 7$ for emphasis. A hypothetical $\bullet\langle {}^{1}_{1}, {}^{1}_{0}\rangle\bullet\langle {}^{1}_{1}, {}^{0}_{0}\rangle\circ 7$ would denote an impossible behaviour, making intermediate sorts redundant.

One of Brookes's innovations is that sets of traces should be closed under what we now call *(trace) deductions*. Specifically, Brookes identified two such deductions, given as binary relations called $\mathsf{stutter}$ ($\xrightarrow{\mathtt{st}}$) and mumble ($\xrightarrow{\mathtt{mu}}$), defined in such a way that if the program phrase can adhere to the source protocol (left of arrow), then it can adhere to the target protocol (right of arrow).

We define these deductions in our two-sorted setting. For convenience, we write $\square \xi_1^? \circ \xi_2^? \diamondsuit x$ for the trace $\square \xi_1^? \xi_2^? \diamondsuit x$ in which, intuitively, the lock is ceded ($\circ$) at the marked spot. Formally, we require that both (a) if $\xi_1^?$ is empty, then $\square = \circ$; and (b) if $\xi_2^?$ is empty, then $\diamondsuit = \circ$. In particular, the requirement holds when both $\xi_1^?$ and $\xi_2^?$ are non-empty, where we implicitly assume the ceded sort between them; and in the case of a $\circ$-sorted $\circ$-valued trace, i.e. $\square = \circ = \diamondsuit$.

Example 20. We have the following valid/invalid notations for $\bullet\langle {}^{1}_{1}, {}^{1}_{0}\rangle\langle {}^{1}_{1}, {}^{0}_{0}\rangle\circ 7$:

valid: $\bullet\langle {}^{1}_{1}, {}^{1}_{0}\rangle\circ\langle {}^{1}_{1}, {}^{0}_{0}\rangle\circ 7$ $\quad \bullet\langle {}^{1}_{1}, {}^{1}_{0}\rangle\langle {}^{1}_{1}, {}^{0}_{0}\rangle\circ\circ 7$ $\qquad$ invalid: $\bullet\circ\langle {}^{1}_{1}, {}^{1}_{0}\rangle\langle {}^{1}_{1}, {}^{0}_{0}\rangle\circ 7$ $\quad \square$

We define the following *sorted* stutter *and* mumble *deductions*:

$$\square\xi_1^?\circ\xi_2^?\Diamond x \xrightarrow{\texttt{st}} \square\xi_1^?\langle\sigma,\sigma\rangle\xi_2^?\Diamond x \qquad \square\xi_1^?\langle\sigma,\rho\rangle\langle\rho,\theta\rangle\xi_2^?\Diamond x \xrightarrow{\texttt{mu}} \square\xi_1^?\langle\sigma,\theta\rangle\xi_2^?\Diamond x$$

The condition on stutter's source rules out deductions which implicitly cede access to the store to the concurrent environment at the ends of the trace. We will compare these deductions to Brookes's in §6.

Example 21. These deductions are valid, highlighting the change to the trace:

$$\bullet\langle {}^{1}_{1}, {}^{1}_{0}\rangle\langle {}^{1}_{1}, {}^{0}_{0}\rangle\circ 7 \xrightarrow{\texttt{st}} \bullet\langle {}^{1}_{1}, {}^{1}_{0}\rangle\langle {}^{1}_{1}, {}^{0}_{0}\rangle\langle {}^{0}_{1}, {}^{0}_{1}\rangle\circ 7 \qquad \bullet\langle {}^{1}_{1}, {}^{1}_{0}\rangle\langle {}^{1}_{0}, {}^{0}_{0}\rangle\circ 7 \xrightarrow{\texttt{mu}} \bullet\langle {}^{1}_{1}, {}^{0}_{0}\rangle\circ 7$$

However, thanks to the condition on stutter's source, this deduction is invalid:

$$\bullet\langle {}^{1}_{1}, {}^{1}_{0}\rangle\langle {}^{1}_{1}, {}^{0}_{0}\rangle\circ 7 \not\xrightarrow{\texttt{st}} \bullet\langle {}^{0}_{1}, {}^{0}_{1}\rangle\langle {}^{1}_{1}, {}^{1}_{0}\rangle\langle {}^{1}_{1}, {}^{0}_{0}\rangle\circ 7$$

The source protocol relies on the preceding sequential environment for ${}^{1}_{1}$. We prohibit relaxing the protocol to rely on the concurrent environment for it. $\square$

The stutter and mumble deductions follow the rely/guarantee intuition:

Stuttering ($\square\xi_1^?\circ\xi_2^?\Diamond x \xrightarrow{\texttt{st}} \square\xi_1^?\langle\sigma,\sigma\rangle\xi_2^?\Diamond x$) means a thread-pool also obeys the protocol that guarantees a state σ by relying on its environment for σ.

Mumbling ($\square\xi_1^?\langle\sigma,\rho\rangle\langle\rho,\theta\rangle\xi_2^?\Diamond x \xrightarrow{\texttt{mu}} \square\xi_1^?\langle\sigma,\theta\rangle\xi_2^?\Diamond x$) means a thread-pool that guarantees the store ρ it later relies on also obeys the protocol in which we exclude the environment's access to the store ρ at that point.

Sets of traces represent a non-deterministic choice between the behaviours that a program phrase may exhibit. For such a set K, define its *closure* under trace deduction $K^\dagger$ as the least set K' such that: $K \subseteq K'$; and if $\tau_1 \in K'$ and $\tau_1 \xrightarrow{\texttt{x}} \tau_2$ for $\texttt{x} \in \{\texttt{st}, \texttt{mu}\}$, then $\tau_2 \in K'$. According to the rely/guarantee intuition above, a program phrase that is compatible with a set of traces is also compatible with its closure. We therefore represent program phrases as *closed* sets, i.e. sets K such that $K = K^\dagger$. The closure $K^\dagger$ of a countable K is countably infinite—by stuttering indefinitely—unless K is a finite set of single-transition $\bullet$-sorted $\bullet$-valued traces, in which case K is already closed.

For a set of traces U and sort $\square \in \{\bullet, \circ\}$, define a $\{\bullet, \circ\}$-sorted family $\mathcal{P}^{\aleph_0}(U)$ by taking its $\square$ component to be the set $\mathcal{P}^{\aleph_0}_{\square}(U)$ of countable subsets of U whose elements are all $\square$-sorted. Similarly, define $\mathcal{P}^{\dagger}_{\square}(U) \subseteq \mathcal{P}^{\aleph_0}_{\square}(U)$ to be the set of *closed* countable subsets of U whose elements are all $\square$-sorted.

The *prefixing* function adds the given transition to each $\bullet$-sorted trace:

$$(\sigma,\rho) : \mathcal{P}^{\aleph_0}_{\bullet}(\mathbb{T}\boldsymbol{X}) \to \mathcal{P}^{\aleph_0}_{\bullet}(\mathbb{T}\boldsymbol{X}) \quad (\sigma,\rho)\,K := \{\bullet\langle\sigma,\theta\rangle\xi^?\Diamond x \mid \bullet\langle\rho,\theta\rangle\xi^?\Diamond x \in K\}$$

It lifts to closed sets, i.e. $K \in \mathcal{P}^{\dagger}_{\bullet}(\mathbb{T}\boldsymbol{X})$ implies that $(\sigma,\rho)\,K \in \mathcal{P}^{\dagger}_{\bullet}(\mathbb{T}\boldsymbol{X})$.

5.2 Representation theorem

For $\boldsymbol{X} \in \mathbf{Set}^{\{\bullet,\circ\}}$, define the $\Sigma_{\mathbb{S}}$-algebra of $\boldsymbol{X}$-*valued closed trace-sets* $\mathbf{R}\boldsymbol{X}$ as:

$$\underline{\mathbf{R}\boldsymbol{X}}_{\square} \coloneqq \mathcal{P}^{\dagger}_{\square}(\mathbb{T}\boldsymbol{X}) \qquad [\![\mathsf{U}_{\ell,b}]\!]_{\mathrm{op}} K \coloneqq \bigcup_{\sigma\in\mathbb{S}} (\sigma, \sigma[\ell \mapsto b])\, K$$
$$[\![\bigvee_{i<\alpha}]\!]_{\mathrm{op}} K_i \coloneqq \bigcup_{i<\alpha} K_i \qquad [\![\mathsf{L}_{\ell}]\!]_{\mathrm{op}}(K_0, K_1) \coloneqq \bigcup_{\sigma\in\mathbb{S}} (\sigma,\sigma)\, K_{\sigma_\ell}$$
$$[\![\triangleleft]\!]_{\mathrm{op}} K \coloneqq \{\circ\xi\Diamond x \mid \bullet\xi\Diamond x \in K\}^{\dagger} \qquad [\![\triangleright]\!]_{\mathrm{op}} K \coloneqq \left\{\bullet\langle\sigma,\sigma\rangle\xi\Diamond x \;\middle|\; \sigma\in\mathbb{S}, \circ\xi\Diamond x \in K\right\}^{\dagger}$$

Additionally, define return $: \boldsymbol{X} \to \underline{\mathbf{R}\boldsymbol{X}}$ by $\mathrm{return}_{\square}\, x \coloneqq \{\square\langle\sigma,\sigma\rangle\square x \mid \sigma\in\mathbb{S}\}^{\dagger}$.

The rest of this section establishes that the algebra $\langle \mathbf{R}\boldsymbol{X}, \mathrm{return}\rangle$ over $\boldsymbol{X}$ is a free $\mathbb{S}$-model over $\boldsymbol{X}$. A key ingredient is *reification*: for any $\{\bullet,\circ\}$-sorted family $\boldsymbol{X}$, we define a sorted-function $\mathrm{reify} : \mathcal{P}^{\aleph_0}(\mathbb{T}\boldsymbol{X}) \to \mathrm{Term}^{\Sigma_{\mathbb{S}}}\boldsymbol{X}$, choosing a representative term $t_2 \coloneqq \mathrm{reify}\,[\![\boldsymbol{X} \vdash t_1]\!]_{\mathrm{term}}$ such that $\boldsymbol{X} \vdash_{\mathbb{S}} t_1 = t_2$. This use of countable choice is inessential, the mere existence of the defining term t_2 suffices.

First define for any $\ell \in \mathbb{L}$ and $b \in \mathbb{B}$ the *cell assertion* term $x : \bullet \vdash_{\Sigma_{\mathbb{S}}} \mathsf{A}_{\ell,b}\, x : \bullet$ that looks ℓ up and only continues if it holds b:

$$x : \bullet \vdash_{\Sigma_{\mathbb{S}}} \mathsf{A}_{\ell,0}\, x \coloneqq \mathsf{L}_{\ell}(x, \bot) : \bullet \qquad x : \bullet \vdash_{\Sigma_{\mathbb{S}}} \mathsf{A}_{\ell,1}\, x \coloneqq \mathsf{L}_{\ell}(\bot, x) : \bullet$$

Next, for any $\sigma, \rho \in \mathbb{S}$ we define the *open transition* $x : \bullet \vdash_{\Sigma_{\mathbb{S}}} \{\!|\sigma,\rho|\!\}\, x : \bullet$, as a term that asserts the state is σ, then updates the state to ρ, and returns x:

$$x : \bullet \vdash_{\Sigma_{\mathbb{S}}} \{\!|\sigma,\rho|\!\}\, x \coloneqq \mathsf{A}_{\mathtt{l}_1,\sigma_{\mathtt{l}_1}} \ldots \mathsf{A}_{\mathtt{l}_n,\sigma_{\mathtt{l}_n}}\, \mathsf{U}_{\mathtt{l}_1,\rho_{\mathtt{l}_1}} \ldots \mathsf{U}_{\mathtt{l}_n,\rho_{\mathtt{l}_n}}\, x : \bullet \quad (\mathbb{L} = \{\mathtt{l}_1, \ldots, \mathtt{l}_n\})$$

Now we can represent traces as terms. Define the $\Sigma_{\mathbb{S}}$-term reifying a trace $x : \Diamond \vdash_{\Sigma_{\mathbb{S}}} \underline{\square\xi\Diamond x} : \square$ by sequencing open transition as they are in ξ, separated by $\triangleright\,\triangleleft$; and delimited by $\triangleleft$ on the left if $\square = \circ$ and by $\triangleright$ on the right if $\Diamond = \circ$.

Example 22. $x : \circ \vdash_{\Sigma_{\mathbb{S}}} \underline{\bullet\langle\sigma,\rho\rangle\langle\sigma',\rho'\rangle\circ x} \coloneqq \{\!|\sigma,\rho|\!\} \triangleright \triangleleft \{\!|\sigma',\rho'|\!\} \triangleright x : \bullet$ □

Trace deductions are sound w.r.t. this encoding, in the following sense:

Proposition 23. *Assume that τ_1 and τ_2 are $\square$-sorted traces over $\{x : \Diamond\}$, such that $\tau_1 \xrightarrow{\mathtt{x}} \tau_2$ for $\mathtt{x} \in \{\mathtt{st}, \mathtt{mu}\}$. Then $x : \Diamond \vdash_{\Sigma_{\mathbb{S}}} \underline{\tau_1} \geq \underline{\tau_2} : \square$.*

Finally, we reify a trace set by reifying its traces in a chosen enumeration:

$$\mathrm{reify} : \mathcal{P}^{\aleph_0}(\mathbb{T}\boldsymbol{X}) \to \mathrm{Term}^{\Sigma_{\mathbb{S}}}\boldsymbol{X} \qquad \mathrm{reify}_{\square}\, K \coloneqq \left(\boldsymbol{X} \vdash_{\Sigma_{\mathbb{S}}} \bigvee_{\tau\in K} \underline{\tau} : \square\right)$$

By proposition 23, closure preserves reification: $\boldsymbol{X} \vdash_{\mathbb{S}} \mathrm{reify}_{\square}\, K = \mathrm{reify}_{\square}\, K^{\dagger} : \square$.

Using reification, we state the representation theorem [proof in 14, §C].

Theorem 24 ($\mathbb{S}$-representation). *The pair $\langle \mathbf{R}\boldsymbol{X}, \mathrm{return}\rangle$ is a free $\mathbb{S}$-model over $\boldsymbol{X}$. Its representation sends environments $e : \boldsymbol{X} \to \underline{\mathbf{A}}$ to $\mathbb{S}$-homomorphisms $e^{\#} : \mathbf{R}\boldsymbol{X} \to \mathbf{A}$ by $e^{\#}_{\square} K \coloneqq \mathbf{R}\boldsymbol{X}[\![\mathrm{reify}_{\square}\, K]\!]_{\mathrm{term}} e$. Moreover, for $\mathbf{A} = \mathbf{R}\boldsymbol{Y}$ we have:*

$$e^{\#}_{\square} K = \left\{\square\xi_1\xi_2\Diamond y \;\middle|\; \begin{array}{l}\square\xi_1\circ x \in K,\\ \circ\xi_2\Diamond y \in e_{\Diamond} x\end{array}\right\}^{\dagger} \cup \left\{\square\xi_1\langle\sigma,\theta\rangle\xi_2\Diamond y \;\middle|\; \begin{array}{l}\square\xi_1\langle\sigma,\rho\rangle\bullet x \in K,\\ \bullet\langle\rho,\theta\rangle\xi_2\Diamond y \in e_{\Diamond} x\end{array}\right\}^{\dagger}.$$

Example 25. The model $\mathbf{R}\{x : \circ\}$ invalidates Write Intro:

$$\mathbf{R}\{x : \circ\}[\![\triangleleft \mathsf{U}_{\ell,b_1} \triangleright \triangleleft \mathsf{U}_{\ell,b_2} \triangleright x]\!]_{\mathrm{term}}\mathrm{return} \neq \mathbf{R}\{x : \circ\}[\![\triangleleft \mathsf{U}_{\ell,b_2} \triangleright x]\!]_{\mathrm{term}}\mathrm{return}$$

Every trace in the right-hand set has at most one state-changing transition. The left-hand set has traces with two. Therefore, $\mathbb{S}$ does not prove Write Intro. □

6 Recovering Brookes's model

The theory $\mathfrak{S}$ recovers Brookes's model (§6.1). We recover it twice, using different strategies that offer different perspectives. The first transforms the monad induced by the representation of §5.2 along a right adjoint $(-)_\circ : \mathbf{Set}^{\{\bullet,\circ\}} \to \mathbf{Set}$ sending each $\{\bullet,\circ\}$-family $\boldsymbol{X}$ to the set $\boldsymbol{X}_\circ := \{x \mid (x : \circ) \in \boldsymbol{X}\}$ (§6.2). In the second, we define a single-sorted theory of transitions B that recovers Brookes's model straightforwardly (§6.3). In this theory, the transition operators correspond to Brookes's `await` construct. After swiftly introducing embedding translations (§6.4), we show that B embeds into $\mathfrak{S}$. The embedding factors through another, two-sorted, theory of transitions Tr (§6.5).

6.1 Brookes's model

We designed our notions of traces, deduction, etc. from §5.1 based on the following model of Brookes [6], in which traces cannot hold exclusive memory access at their ends. In this model, ceding access is implicit.

For any set $X \in \mathbf{Set}$, recall the set of Brookes traces $\mathsf{T}X := (\mathbb{S} \times \mathbb{S})^+ \times X$ from §5.1. Writing ξx for $\langle \xi, x \rangle$, Brookes's $\mathsf{stutter}$ and mumble deductions are:

$$\xi_1^? \xi_2^? x \xrightarrow{\texttt{st}} \xi_1^? \langle \sigma, \sigma \rangle \xi_2^? x \qquad \xi_1^? \langle \sigma, \rho \rangle \langle \rho, \theta \rangle \xi_2^? x \xrightarrow{\texttt{mu}} \xi_1^? \langle \sigma, \theta \rangle \xi_2^? x$$

We reuse the notation $(-)^\dagger$ for closure under these deductions.

The difference between Brookes's deductions and our multi-sorted deductions is the maintenance of the sort on each end of the trace. In particular, Brookes's $\mathsf{stutter}$ does not need to explicitly allow interleaving at the relevant position in the source, because the environment may always interleave on either end.

Brookes's semantic domain $BX := \mathcal{P}^\dagger(\mathsf{T}X)$ forms a monad. The monadic unit is $\mathrm{return} : X \to BX$, $\mathrm{return}\, x := \{\langle \sigma, \sigma \rangle x \mid \sigma \in \mathbb{S}\}^\dagger$. The Kleisli extension $e^\# : BX \to BY$ of every $e : X \to BY$ is $e^\# K := \{\xi_1 \xi_2 y \mid \xi_1 x \in K, \xi_2 y \in ex\}^\dagger$. It interprets memory accesses, dereferencing ($\ell!$) and mutation ($\ell := b$), as follows:

$$[\![\ell!]\!] : \mathbb{1} \xrightarrow{\{\langle \sigma, \sigma \rangle \sigma_\ell \mid \sigma \in \mathbb{S}\}^\dagger} B\mathbb{B} \qquad [\![\ell := b]\!] : \mathbb{1} \xrightarrow{\{\langle \sigma, \sigma[\ell \mapsto b] \rangle \langle\rangle \mid \sigma \in \mathbb{S}\}^\dagger} B\mathbb{1}$$

These *generic effects* [31] correspond to these monadic algebraic operations:

$$\begin{array}{lll} [\![\mathsf{R}_\ell]\!] & : (BX)^2 \to BX & [\![\mathsf{R}_\ell]\!](K_0, K_1) := \{\langle \sigma, \sigma \rangle \xi x \mid \sigma \in \mathbb{S}, \xi x \in K_{\sigma_\ell}\}^\dagger \\ [\![\mathsf{W}_{\ell,b}]\!] & : BX \to BX & [\![\mathsf{W}_{\ell,b}]\!] K := \{\langle \sigma, \sigma[\ell \mapsto b] \rangle \xi x \mid \sigma \in \mathbb{S}, \xi x \in K\}^\dagger \end{array}$$

6.2 Recovery via an adjunction

In Brookes's model, yielding to the concurrent environment is implicit, and always allowed. From our two-sorted point-of-view, we expect the traces in Brookes's model to represent $\circ$-sorted $\circ$-valued traces.

There is an abstract construction that recovers the monad and its operations in §6.2 from our $\{\bullet,\circ\}$-sorted model. The functor $(-)_\circ : \mathbf{Set}^{\{\bullet,\circ\}} \to \mathbf{Set}$

has a left-adjoint $(-)^\circ : \mathbf{Set} \to \mathbf{Set}^{\{\bullet,\circ\}}$. This functor sends each set X to the $\{\bullet,\circ\}$-family $X^\circ := \{x : \circ \mid x \in X\}$, using the set-like notation for families we introduced in §3.1. Monads transform along adjoints, and transforming the monad obtained standardly from the representation of §5.2 along the adjunction above results in Brookes's model. Explicitly, denoting $B_\circ X := \underline{\mathbf{R}X^\circ}_\circ = \mathcal{P}^\dagger_\circ(\mathbb{T}X^\circ)$, the resulting monad over $\mathbf{Set}$ is $\langle B_\circ, \mathrm{return}_\circ, (-)^\#_\circ\rangle$. This monad is isomorphic to Brookes's $\langle B, \mathrm{return}, (-)^\#\rangle$ above by way of removing $\circ$ from both ends of every trace. Thus, the Brookes model amounts to the free $\mathbb{S}$-model from §5.2 transformed along the adjunction $(-)^\circ \dashv (-)_\circ$. The monad $\mathbf{R}$ supports the following generic effects. The adjunction transforms them, via its natural bijection on homsets, into Brookes's generic effects for memory access:

$$[\![\ell!]\!] : \mathbb{1}^\circ \xrightarrow{[\![\lhd \mathsf{L}_\ell(\rhd 0, \rhd 1)]\!]} \mathbf{R}\mathbb{B}^\circ \qquad [\![\ell := b]\!] : \mathbb{1}^\circ \xrightarrow{[\![\lhd \mathsf{U}_{\ell,b} \rhd \langle\rangle]\!]} \mathbf{R}\mathbb{1}^\circ$$

6.3 The single-sorted theory of transitions

There is a more direct, single-sorted presentation B for Brookes's model. It uses transitions as operators rather than lookup and update operators. The signature Σ_B consists of countable-joins Σ_V and a unary transition operator $\langle\sigma,\rho\rangle$ for every $\sigma,\rho \in \mathbb{S}$. The axioms Ax_B consist of the countable-join semilattice axioms Ax_V, strict distributivity axioms (ND-B) $\langle\sigma,\rho\rangle \bigvee_{i<\alpha} x_i = \bigvee_{i<\alpha} \langle\sigma,\rho\rangle x_i$, and:

Trace closure

(M) $\langle\sigma,\rho\rangle\langle\rho,\theta\rangle x \geq \langle\sigma,\theta\rangle x$ (S) $x \geq \langle\sigma,\sigma\rangle x$ (H) $\bigvee_{\sigma\in\mathbb{S}} \langle\sigma,\sigma\rangle x \geq x$

The first two axiom schemes are algebraic counterparts to mumble and stutter. These alone do not recover Brookes's model—the representation theorem for the theory without the (H) axioms includes potentially-empty traces. The axiom (H) fails in this model, but holds in Brookes's. In the representation theorem for B it is tempting to require, along with closure under Brookes's mumble and stutter trace deductions, closure under hush: presented in fig. 2 for a set of traces K. However, there is no need, due to the non-emptiness of the traces. Indeed, either $\xi_1^?$ or $\xi_2^?$ must be non-empty for the rule to apply. Take σ to match an adjacent transition, and apply the mumble closure rule to obtain the required consequence. This nuanced observation exposing the hush rule would be hard to notice without this algebraic analysis.

$$\frac{\forall\sigma.\, \xi_1^?\langle\sigma,\sigma\rangle\xi_2^? x \in K}{\xi_1^?\xi_2^? x \in K}$$

Fig. 2. The hush rule

To conclude, we formulate the representation theorem for B. Let $X \in \mathbf{Set}$. Define the Σ_B-algebra $\mathbf{B}X$ with carrier $\underline{\mathbf{B}X} := \mathcal{P}^\dagger(\mathsf{T}X)$ and interpretations:

$$\mathbf{B}X[\![\textstyle\bigvee_{i<\alpha}]\!]_{\mathrm{op}} K_i := \bigcup_{i<\alpha} K_i \qquad \mathbf{B}X[\![\langle\sigma,\rho\rangle]\!]_{\mathrm{op}} K := \{\langle\sigma,\rho\rangle\tau \mid \tau \in K\}^\dagger$$

Additionally, define $\mathrm{return} : X \to \underline{\mathbf{B}X}$ by $\mathrm{return}\, x := \lambda x.\, \{\langle\sigma,\sigma\rangle x \mid \sigma \in \mathbb{S}\}^\dagger$.

To prove that this is a free B-model, we use reification as in §5.2, though here reification is more straightforward. A trace is reified as itself, and sets of

traces use countable-joins as before: $\mathrm{reify}\, K \coloneqq \left(\boldsymbol{X} \vdash_{\Sigma_{\mathsf{B}}} \bigvee_{\tau \in K} \underline{\tau} : \star\right)$. The monad obtained from the next proposition is Brookes's model:

Proposition 26. *The pair* $\langle \mathbf{B}X, \mathrm{return}\rangle$ *is a free* B*-model over* X*, for which the representation sends* $e : X \to \underline{\mathbf{A}}$ *to* $e^{\#} : \mathbf{B}X \to \mathbf{A}$ *by* $e^{\#}_{\square} K \coloneqq \mathbf{B}X [\![\mathrm{reify}_{\square}\, K]\!]_{\mathrm{term}} e$.

6.4 Translations and equivalences

We will need the following notions for relating presentations. Consider a map between two sort sets $\epsilon : \mathbf{sort}_1 \to \mathbf{sort}_2$. It lifts to $\epsilon : \mathbf{Set}^{\mathbf{sort}_2} \to \mathbf{Set}^{\mathbf{sort}_1}$ by precomposition: $(\epsilon \boldsymbol{Y})_{\square} \coloneqq \boldsymbol{Y}_{\epsilon \square}$. It forms the object part of a geometric morphism between (pre)sheaf toposes, i.e., it has left and right adjoints. The left adjoint $\epsilon^* : \mathbf{Set}^{\mathbf{sort}_1} \to \mathbf{Set}^{\mathbf{sort}_2}$ is in this case $(\epsilon^* \boldsymbol{X})_{\diamondsuit} \coloneqq \coprod_{\epsilon \square = \diamondsuit} \boldsymbol{X}_{\square}$. When ϵ is injective, the left adjoint is given by the simpler formula $\epsilon^* \boldsymbol{X} \coloneqq \{x : \epsilon \square \mid x \in \boldsymbol{X}_{\square}\}$.

Example 27. The geometric morphism for the map $\star \mapsto \circ : \{\star\} \rightarrowtail \{\bullet, \circ\}$ is the forgetful functor $(-)_{\circ} : \mathbf{Set}^{\{\bullet,\circ\}} \to \mathbf{Set}^{\{\star\}} \cong \mathbf{Set}$. As we saw in §6.2, its left adjoint is $(-)^{\circ} : \mathbf{Set}^{\{\star\}} \to \mathbf{Set}^{\{\bullet,\circ\}}$. □

Let Σ_1 and Σ_2 be signatures and $\epsilon : \mathbf{sort}_{\Sigma_1} \to \mathbf{sort}_{\Sigma_2}$ a map between their sort sets. A *translation of signatures* $\mathbf{E} : \Sigma_1 \rightarrowtail \Sigma_2$ *along* ϵ is an assignment, to each $(O : \square \langle \diamondsuit_i \rangle_{i<\alpha}) \in \Sigma_1$, of a term $\mathbf{E}O \in \mathrm{Term}^{\Sigma_2}_{\epsilon\square} \{x_i : \epsilon \diamondsuit_i \mid i < \alpha\}$. Such a translation yields a functor $\mathbf{E}_{\mathrm{tln}} : \mathbf{Alg}\Sigma_2 \to \mathbf{Alg}\Sigma_1$, mapping a Σ_2-algebra $\mathbf{B}$ to:

$$\underline{\mathbf{E}_{\mathrm{tln}}\mathbf{B}} \coloneqq \epsilon \underline{\mathbf{B}} \qquad \mathbf{E}_{\mathrm{tln}}\mathbf{B} [\![O : \square \langle \diamondsuit_i \rangle_{i<\alpha}]\!]_{\mathrm{op}} \langle b_i \rangle \coloneqq \mathbf{B} [\![\mathbf{E}O]\!]_{\mathrm{term}} \langle x_i \mapsto b_i \rangle_{i<\alpha}$$

For a given family $\boldsymbol{Y} \in \mathbf{Set}^{\mathbf{sort}_{\Sigma_2}}$, such a translation therefore extends uniquely to a Σ_1-homomorphism $(\mathbf{E}_{\mathrm{tln}})_{\boldsymbol{Y}} : F_{\Sigma_1} \epsilon \boldsymbol{Y} \to \mathbf{E}_{\mathrm{tln}} F_{\Sigma_2} \boldsymbol{Y}$.

Example 28. We have a translation $\mathbf{E} : \Sigma_{\mathsf{G}} \rightarrowtail \Sigma_{\mathbb{S}}$ along $\star \mapsto \bullet : \{\star\} \rightarrowtail \{\bullet, \circ\}$ that translates the Σ_{G}-operators using their respective copies in the $\bullet$ sort:

$$\begin{aligned}
\mathbf{E}(\textstyle\bigvee_\alpha : \alpha) &\coloneqq (\{x_i : \bullet \mid i < \alpha\} \vdash_{\Sigma_{\mathbb{S}}} \textstyle\bigvee_{i<\alpha} x_i \quad : \bullet) \\
\mathbf{E}(\mathsf{L}_\ell : \mathbf{2}) &\coloneqq (\{x_0, x_1 : \bullet\} \quad \vdash_{\Sigma_{\mathbb{S}}} \mathsf{L}_\ell(x_0, x_1) : \bullet) \\
\mathbf{E}(\mathsf{U}_{\ell,b} : \mathbf{1}) &\coloneqq (\{x_0 : \bullet\} \quad \vdash_{\Sigma_{\mathbb{S}}} \mathsf{U}_{\ell,b}\, x_0 \quad : \bullet)
\end{aligned}$$ □

A translation of *presentations* $\mathbf{E} : \mathfrak{p}_1 \rightarrowtail \mathfrak{p}_2$ along ϵ is a translation of their signatures along ϵ that, moreover, preserves the provability of axioms:

$$(\boldsymbol{X} \vdash_{\Sigma_{\mathfrak{p}_1}} t_1 = t_2 : \square) \in \mathrm{Ax}_{\mathfrak{p}_1} \implies \epsilon^* \boldsymbol{X} \vdash_{\mathfrak{p}_2} \mathbf{E}_{\mathrm{tln}} t_1 = \mathbf{E}_{\mathrm{tln}} t_2 : \epsilon \square$$

Example 29. The translation of global state into shared state from example 28 is a translation of presentations $\mathbf{E} : \mathsf{G} \rightarrowtail \mathbb{S}$. □

Translations along composable sort maps compose via substitution, and a translation $\mathbf{E} : \mathfrak{p} \rightarrowtail \mathfrak{p}$ along $\mathsf{id}_{\Sigma_{\mathfrak{p}}}$ is an *identity* translation when, for all terms $t \in \mathrm{Term}^{\Sigma_{\mathfrak{p}}}_{\square} \boldsymbol{X}$, we have $\boldsymbol{X} \vdash_{\mathfrak{p}} \mathbf{E}_{\mathrm{tln}} t = t : \square$. A translation $\mathbf{E} : \mathfrak{p}_1 \rightarrowtail \mathfrak{p}_2$ along ϵ is an *equivalence* if ϵ is a bijection, and there exists an embedding $\mathbf{E}^{-1} : \mathfrak{p}_2 \rightarrowtail \mathfrak{p}_1$ along ϵ^{-1}, such that $\mathbf{E} \circ \mathbf{E}^{-1}$ and $\mathbf{E}^{-1} \circ \mathbf{E}$ are identity translations. We then write $\mathfrak{p}_1 \simeq \mathfrak{p}_2$ and say that the presentations are *equivalent*. Two multi-sorted theories are equivalent iff their associated free-model monads are isomorphic.

6.5 Translation through the two-sorted theory of transitions

We define a two-sorted presentation Tgs of the *open* transitions $\{\!\{\sigma, \rho\}\!\}$ as sequential operators. The signature Σ_{Tgs} consists of countable-joins Σ_{V} and a unary open transition operator $(\!|\sigma, \rho|\!)$ for $\sigma, \rho \in \mathbb{S}$. The axioms $\mathrm{Ax}_{\mathsf{Tgs}}$ consist of the countable-join semilattice axioms Ax_{V}, strict distributivity axioms (ND-$\mathtt{T}$) $(\!|\sigma, \rho|\!) \bigvee_{i<\alpha} x_i = \bigvee_{i<\alpha} (\!|\sigma, \rho|\!)\, x_i$, and:

Open transition axioms

(HS) $x = \bigvee_{\sigma \in \mathbb{S}} (\!|\sigma, \sigma|\!)\, x$

(Seq$^{=}$) $(\!|\sigma, \rho|\!)\, (\!|\rho, \theta|\!)\, x = (\!|\sigma, \theta|\!)\, x$

(Seq$^{\neq}$) $(\!|\sigma, \rho|\!)\, (\!|\mu, \theta|\!)\, x = \bot \qquad \rho \neq \mu$

Translate $\mathbf{E}_{\mathsf{G}} : \mathsf{Tgs} \rightarrowtail \mathsf{G}$ by interpreting transitions as the open transitions from §5.2: $\mathbf{E}_{\mathsf{G}}\, (\!|\sigma, \rho|\!) := (x_0 \vdash_{\Sigma_{\mathsf{G}}} \{\!\{\sigma, \rho\}\!\}\, x_0)$. Conversely, translate $\mathbf{E}_{\mathsf{Tgs}} : \mathsf{G} \rightarrowtail \mathsf{Tgs}$ as follows, similar to the representation of update and lookup from §5.2:

$$\mathbf{E}_{\mathsf{Tgs}}\mathsf{U}_{\ell,b} := (x_0 \vdash_{\Sigma_{\mathsf{Tgs}}} \textstyle\bigvee_{\sigma \in \mathbb{S}} (\!|\sigma, \sigma[\ell \mapsto b]|\!)\, x_0) \quad \mathbf{E}_{\mathsf{Tgs}}\mathsf{L}_{\ell} := (x_0, x_1 \vdash_{\Sigma_{\mathsf{Tgs}}} \textstyle\bigvee_{\sigma \in \mathbb{S}} (\!|\sigma, \sigma|\!)\, x_{\sigma_\ell})$$

Using the equivalence $\mathsf{Tgs} \simeq \mathsf{G}$ that these translations witness we can translate $\mathsf{B} \rightarrowtail \mathbb{SS}$ along $\star \mapsto \mathsf{o}$. We define a two-sorted presentation Tr, mimicking the definition of $\mathbb{SS}$ but replacing the operators and axioms of G with those of Tgs in the hold ($\bullet$) sort: $\mathrm{Ax}_{\mathsf{Tr}} := \boxed{\mathrm{Ax}^{\bullet}_{\mathsf{Tgs}}} \cup \mathrm{Ax}^{\circ}_{\mathsf{V}} \cup \{\text{ND-}\triangleright, \text{ND-}\triangleleft\} \cup \{\text{Empty}, \text{Fuse}\}$. Extending the translations $\mathbf{E}_{\mathsf{Tgs}}$ and $\mathbf{E}_{\mathsf{G}}$ to all of the operators gives an equivalence $\mathsf{Tr} \simeq \mathbb{SS}$. So Tr induces the same monad as $\mathbb{SS}$, recovering Brookes's model.

Define the translation $\mathbf{E}_{\mathsf{Tr}} : \mathsf{B} \rightarrowtail \mathsf{Tr}$ along $\star \mapsto \mathsf{o}$ by sending transitions to their delimited open counterparts: $\mathbf{E}_{\mathsf{Tr}}\langle \sigma, \rho \rangle := (x_0 : \mathsf{o} \vdash_{\Sigma_{\mathsf{Tr}}} \triangleleft\, (\!|\sigma, \rho|\!)\, \triangleright\, x_0 : \mathsf{o})$. Using $\mathsf{Tr} \simeq \mathbb{SS}$ we get $\mathsf{B} \rightarrowtail \mathbb{SS}$ (fig. 3). Brookes's model, as a free B-model, is thus the o-sorted fragment of $\mathbb{SS}$ over o-variables, formally.

$$\begin{array}{ccc} \mathsf{Tgs} & \simeq & \mathsf{G} \\ \downarrow & \overset{\bullet}{\updownarrow} & \downarrow \\ \mathsf{B} \overset{\star \mapsto \mathsf{o}}{\rightarrowtail} & \mathsf{Tr} & \simeq \mathbb{SS} \end{array}$$

Fig. 3. Th. chart

7 Conclusion and further work

We presented an equational theory for shared state ($\mathbb{SS}$). It separates reasoning into two layers. In the held layer ($\bullet$), we prohibit the concurrent environment from accessing memory, and we can reason about memory accesses by a pool of threads sequentially. In the ceded layer (o), the concurrent environment may interleave, and local memory access is forbidden. We also presented theories of transitions (B, Tgs, & Tr) and formally related them to (non-deterministic) global state (G) and shared state ($\mathbb{SS}$). The single-sorted theory B recovers Brookes's model, but it does so by using Brookes's `await` construct, which we find unnatural; and it does not admit global state explicitly as a component of the theory. We believe that admitting global state will inform modelling other effects in the concurrent setting. Our theory $\mathbb{SS}$ addresses these concerns. It admits the global state theory as-is, and axiomatises the mode-switching operators ($\triangleleft/\triangleright$) without explicit interaction with global state. This theory recovers Brookes's model exactly, in a principled manner: by transforming a monad and its operations along an adjunction; and, independently, through algebraic translations.

Our theory uses countable-join semilattices to recover Brookes's model. They can express iteration (i.e. `while`-loops). The same model admits first-order recursion, i.e. least-fixpoints of mutually-defined first-order functions, using the ω-complete partial order structure of the refinement order and the Scott-continuity of the semantics. We can support higher-order recursion by recourse to domain-theory, generalising algebraic theories using order-enriched theories. There are several standard variants, each with subtle logical trade-offs [29]. We can also restrict the semantics to terminating languages by restricting to finite joins, and using finitely-generated closed subsets for the representation.

We want to analyse Brookes's parallel composition operator algebraically. Brookes composed programs in parallel by interleaving traces from each thread. Initial results show we can define Brookes's parallel composition by simultaneous induction over terms. However, we would like to provide a more abstract account, by recourse to the universal property of free models. This abstraction may expose special properties of global state, or lead to a general parallel composition operation satisfying the expected laws of concurrent programming [16, 27, 34].

We would like to model more effects within this modular multi-sorted algebraic framework. These effects include: more advanced notions of state, such as dynamic allocation [20], higher-order memory cells [24, 36], and weak memory [12, 13]; control-flow effects such as exceptions and effect handlers [3]; and probabilistic programming with shared state [22].

If the multi-sorted approach does indeed generalise to more sophisticated effects, then it will be instructive to review its assumptions. For example, the strictness axioms impose a partial-correctness discipline: the semantics says nothing about the effect a diverging program has on its memory. Relaxing or removing strictness may give a model that allows us to reason about diverging programs.

Our two sorts limit access to the whole store. We would like to explore finer granularity. For example, a theory with per-location access limitation, with sorts for every finite subset $s \subseteq \mathbb{L}$ of locations, and operators $(\lhd_\ell : s\backslash\{\ell\}\ \langle s \cup \{\ell\}\rangle)$ and $(\rhd_\ell : s\cup\{\ell\}\ \langle s\backslash\{\ell\}\rangle)$. We expect the axiomatisation's design to require subtlety.

It may be interesting to to expose the sort discipline in the surface language through typing judgements, explicating regions that rule out data-races with the environment. It seems such judgements would rule out deadlocks structurally, and so may limit expressiveness. Whether this idea is useful remains to be seen.

In conclusion, our two-sorted decomposition of Brookes's seminal model provides new insights into its assumptions and components, and reveals new directions for modelling more advanced features involving concurrent shared state.

Acknowledgments. Supported by the Israel Science Foundation (grant number 814/22) and the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement no. 851811); and by a Royal Society University Research Fellowship and Enhancement Award. For the purpose of Open Access the authors have applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission. We thank Danel Ahman, Andrej Bauer, Martín Escardó, Justus Matthiesen, Sam Staton, and Rob van Glabbeek for interesting and useful discussions and suggestions.

Bibliography

[1] Abramsky, S., Jung, A.: Domain Theory. In: Handbook of Logic in Computer Science, Oxford University Press (04 1995), ISBN 9780198537625, https://doi.org/10.1093/oso/9780198537625.003.0001

[2] Adámek, J., Rosický, J., Vitale, E.M.: Algebraic Theories: A Categorical Introduction to General Algebra. Cambridge Tracts in Mathematics, Cambridge University Press (2010)

[3] Bauer, A., Pretnar, M.: Programming with algebraic effects and handlers. J. Log. Algebraic Methods Program. **84**(1) (2015), https://doi.org/10.1016/J.JLAMP.2014.02.001

[4] Benton, N., Hofmann, M., Nigam, V.: Effect-dependent transformations for concurrent programs. In: PPDP, ACM (2016), https://doi.org/10.1145/2967973.2968602

[5] Bloom, S.L.: Varieties of ordered algebras. Journal of Computer and System Sciences **13**(2) (1976), ISSN 0022-0000, https://doi.org/10.1016/S0022-0000(76)80030-X

[6] Brookes, S.D.: Full abstraction for a shared-variable parallel language. Inf. Comput. **127**(2) (1996), https://doi.org/10.1006/inco.1996.0056

[7] Castellan, S., Clairambault, P., Winskel, G.: The parallel intensionally fully abstract games model of pcf. In: LICS (2015), https://doi.org/10.1109/LICS.2015.31

[8] Dodds, M., Batty, M., Gotsman, A.: Compositional verification of compiler optimisations on relaxed memory. In: ESOP, ETAPS, LNCS, vol. 10801, Springer (2018), https://doi.org/10.1007/978-3-319-89884-1_36

[9] Dolan, S., Eliopoulos, S., Hillerström, D., Madhavapeddy, A., Sivaramakrishnan, K.C., White, L.: Concurrent system programming with effect handlers. In: TFP, LNCS, vol. 10788, Springer (2017), https://doi.org/10.1007/978-3-319-89719-6_6

[10] Dolan, S., White, L., Sivaramakrishnan, K.C., Yallop, J., Madhavapeddy, A.: Effective concurrency with algebraic effects (2015), OCaml Workshop

[11] Dvir, Y., Kammar, O., Lahav, O.: An algebraic theory for shared-state concurrency. In: APLAS, LNCS, vol. 13658, Springer (2022), https://doi.org/10.1007/978-3-031-21037-2_1

[12] Dvir, Y., Kammar, O., Lahav, O.: A denotational approach to release/acquire concurrency. In: ESOP, ETAPS, LNCS, vol. 14577, Springer (2024), https://doi.org/10.1007/978-3-031-57267-8_5

[13] Dvir, Y., Kammar, O., Lahav, O.: A brookes-style denotational semantics for release/acquire concurrency. TOPLAS (Jan 2025), ISSN 0164-0925, https://doi.org/10.1145/3715096, just Accepted

[14] Dvir, Y., Kammar, O., Lahav, O., Plotkin, G.: Two-sorted algebraic decompositions of brookes's shared-state denotational semantics (2025), URL https://arxiv.org/abs/2501.15104

[15] Hennessy, M.C.B., Plotkin, G.D.: Full abstraction for a simple parallel programming language. In: Mathematical Foundations of Computer Science, Springer, Berlin, Heidelberg (1979), ISBN 978-3-540-35088-0
[16] Hoare, T.: Laws of programming: The algebraic unification of theories of concurrency. In: CONCUR 2014 – Concurrency Theory, Springer, Berlin, Heidelberg (2014), ISBN 978-3-662-44584-6
[17] Hyland, M., Plotkin, G.D., Power, J.: Combining effects: Sum and tensor. Theor. Comput. Sci. **357**(1-3) (2006), https://doi.org/10.1016/j.tcs.2006.03.013
[18] Hyland, M., Power, J.: Discrete Lawvere theories and computational effects. Theoretical Computer Science **366**(1) (2006), ISSN 0304-3975, https://doi.org/10.1016/j.tcs.2006.07.007, algebra and Coalgebra in Computer Science
[19] Jeffrey, A., Riely, J.: On Thin Air Reads: Towards an Event Structures Model of Relaxed Memory. LMCS **Volume 15, Issue 1** (Mar 2019), https://doi.org/10.23638/LMCS-15(1:33)2019
[20] Kammar, O., Levy, P.B., Moss, S.K., Staton, S.: A monad for full ground reference cells. In: LICS (2017), https://doi.org/10.1109/LICS.2017.8005109
[21] Kavanagh, R., Brookes, S.: A denotational semantics for SPARC TSO. In: MFPS, ENTCS, vol. 341, Elsevier (2018), https://doi.org/10.1016/j.entcs.2018.03.025
[22] Kozen, D.: Semantics of probabilistic programs. Journal of Computer and System Sciences **22**(3) (1981), ISSN 0022-0000, https://doi.org/10.1016/0022-0000(81)90036-2
[23] Lawvere, F.W.: Functorial Semantics of Algebraic Theories and Some Algebraic Problems in the context of Functorial Semantics of Algebraic Theories. Ph.D. thesis, Department of Mathematics (1963)
[24] Levy, P.B.: Possible world semantics for general storage in call-by-value. In: Computer Science Logic, Springer, Berlin, Heidelberg (2002), ISBN 978-3-540-45793-0
[25] Melliès, P.: Local states in string diagrams. In: RTA-TLCA, LNCS, vol. 8560, Springer (2014), https://doi.org/10.1007/978-3-319-08918-8_23
[26] Moggi, E.: Notions of computation and monads. Inf. Comput. **93**(1) (1991), https://doi.org/10.1016/0890-5401(91)90052-4
[27] Paquet, H., Saville, P.: Effectful semantics in bicategories: strong, commutative, and concurrent pseudomonads. LICS, Association for Computing Machinery, New York, NY, USA (2024), ISBN 9798400706608, https://doi.org/10.1145/3661814.3662130
[28] Plotkin, G.D.: A powerdomain for countable non-determinism. In: Automata, Languages and Programming, Springer, Berlin, Heidelberg (1982), ISBN 978-3-540-39308-5
[29] Plotkin, G.D.: Some Varieties of Equational Logic. Springer, Berlin, Heidelberg (2006), ISBN 978-3-540-35464-2, https://doi.org/10.1007/11780274_8

[30] Plotkin, G.D., Power, J.: Notions of computation determine monads. In: FOSSACS, ETAPS, LNCS, vol. 2303, Springer (2002), https://doi.org/10.1007/3-540-45931-6_24
[31] Plotkin, G.D., Power, J.: Algebraic operations and generic effects. Applied Categorical Structures **11**(3) (2003), ISSN 1572-9095, https://doi.org/10.1023/A:1023064908962
[32] Plotkin, G.D., Pretnar, M.: Handlers of algebraic effects. In: ESOP, ETAPS, LNCS, vol. 5502, Springer (2009), https://doi.org/10.1007/978-3-642-00590-9_7
[33] Plotkin, G.D., Pretnar, M.: Handling algebraic effects. Log. Methods Comput. Sci. **9**(4) (2013), https://doi.org/10.2168/LMCS-9(4:23)2013
[34] Rivas, E., Jaskelioff, M.: Monads with merging (Jun 2019), URL https://inria.hal.science/hal-02150199, working paper or preprint
[35] Sivaramakrishnan, K.C., Dolan, S., White, L., Kelly, T., Jaffer, S., Madhavapeddy, A.: Retrofitting effect handlers onto ocaml. In: PLDI, ACM (2021), https://doi.org/10.1145/3453483.3454039
[36] Sterling, J., Gratzer, D., Birkedal, L.: Denotational semantics of general store and polymorphism (2023), URL https://arxiv.org/abs/2210.02169
[37] Tarlecki, A.: Some nuances of many-sorted universal algebra: A review. Bull. EATCS **104** (2011), URL http://eatcs.org/beatcs/index.php/beatcs/article/view/121
[38] Turon, A.J., Wand, M.: A separation logic for refining concurrent objects. In: POPL, ACM (2011), https://doi.org/10.1145/1926385.1926415
[39] Xu, Q., de Roever, W.P., He, J.: The rely-guarantee method for verifying shared variable concurrent programs. Formal Aspects Comput. **9**(2) (1997), https://doi.org/10.1007/BF01211617