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 use a two sorted equational theory of algebraic effects to model 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.

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 [7] using algebraic effects [33]. 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 [14, 16, e.g.]. 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. 8, 9, 18, 23]. (III) Subsequent variations and extensions [5, 42, 43], as well as adaptations to relaxed memory models [13, 23], 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* [33] refines Moggi's monadic approach [28] 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 [25] via *representation theorems*. For example: global state emerges by axiomatising memory lookup and update [33] and a representation theorem involving the state monad; non-determinism emerges by axiomatising semi-lattices and a representation theorem involving the powerdomains [14, 30]; 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 [16].

Contribution Our main conceptual contribution is to exhibit Brookes's model algebraically. 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* [4, 35, 36] to allow users to define their own schedulers was the original motivation for their implementation in the OCaml programming language [10, 11, 38]. Nonetheless, exhibiting abstract models such as Brookes's algebraically via equational axiomatisation of syntactic constructs has proved challenging. Our own previous algebraic model [12] invalidates a key transformation, reflecting a fundamental limitation of it.

Our main technical innovation is to use multi-sorted algebraic theories, a direction that was raised in personal discussions since the earliest work on algebraic effects [33]. 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 provides a reasoning layer in which the environment may not interfere, whereas in the "cede" sort it may. We provide two operators that switch between these sorts, allowing our axioms to specify the uninterruptable effects. 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 [2]. The remaining axioms are strikingly independent from these axioms, and cover the strict distributive interaction of global state with non-determinism and the strict distributivity of the closure pair over non-determinism. Our main technical contribution is the representation of this theory, which uses sets of traces akin to Brookes's, recovering Brookes's model precisely in the "cede" sort.

Summarising, our contributions are as follows:

- A two-sorted algebraic theory for shared-state, $\mathbb{S}$.
- A representation theorem for $\mathbb{S}$ via Brookes-style trace sets.
- A decomposition of Brookes's model using $\mathbb{S}$ and a geometric morphism.
- A single-sorted algebraic theory for Brookes's `await`, embedding into $\mathbb{S}$.
- The first use of multi-sorted theories for algebraic effects

Caveats Throughout the development, we opt for mathematical simplicity wherever 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 main technical contribution—the representation theorem—allowing us to use countable sets instead of finitely generated ones. We also do not treat recursion to avoid the complexity a domain-theoretic account will incur. The resulting model—identical to Brookes's—coincides with the elided domain-theoretic model over discrete pre-domains. 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 mathematical exposition and a clear presentation of the underlying idea, motivating future inquiry.

Outline In §2 we recap notions of multi-sorted algebra. In §3 we present our two-sorted theory of shared state. In §4 we build a free-model representation of this theory, an adaptation of Brookes's model. In §5 we recover Brookes's model precisely, using two different methods that offer different perspectives: model-theoretically, via an adjunction with the representation; and algebraically, via an embedding of a single-sorted theory of transitions for Brookes's model. Finally, we conclude in §6, where we discuss related work, as well as further research opportunities our contributions enables.

The supplementary material also includes in appendix A some "no-go" results concerning single-sorted theories, motivating the use of a multi-sorted theory to solve the problem at hand. For example, it shows why a natural single-sorted theory—axiomatising yielding as closure operator—cannot work.

2 Preliminaries

In the algebraic effects approach to denotational semantics, we: express core effectful programming constructs as corresponding algebraic operations; express core equational axioms between them as axioms for algebraic structures; and derive a monad by representing the free-model over sets of variables, and define a denotational semantics with it. This section is a standard treatment of countably-infinitary multi-sorted equational theories and their free models [3, 41, e.g.].

2.1 Terms

We define the logical language of multi-sorted equational logic. The basic vocabulary of multi-sorted algebra is parameterised by a set **sort** whose elements $\boxbox, \diamonddiamond$ 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 *sorting scheme* $\vec{\boxbox} \in \mathrm{Scheme}\,\mathbf{sort}$ is a countable sequence of sorts, e.g. a finite sequence $\vec{\boxbox} = \langle \boxbox_0, \dots, \boxbox_{n-1} \rangle$ of length n, or countably infinite sequence $\vec{\boxbox} = \langle \boxbox_0, \boxbox_1, \dots \rangle$ of length ω. For example: the empty scheme $\mathbf{0} \coloneqq \langle\rangle$ of length 0; and the constant schemes $\alpha \cdot \boxbox \coloneqq \langle \boxbox \rangle_{i<\alpha}$ of length α. We write $\boxbox$ for the scheme $1 \cdot \boxbox$.

A **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 $\mathbf{ar}_\Sigma\, O = \left\langle \boxbox, \langle \diamonddiamond_i \rangle_i \right\rangle$, we write $(O : \boxbox \, \langle \diamonddiamond_i \rangle_{i<\alpha}) \in \Sigma$. The operator O will allow us to construct a $\boxbox$-sort term with a tuple of terms, with the i^{th} subterm having sort $\diamonddiamond_i$. 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 `J` consists of two operators: *join* $\vee : \mathbf{2}$, i.e. $\vee : \star \, \langle \star, \star \rangle$ and *bottom* $\bot : \mathbf{0}$, i.e., $\bot : \star \, \langle\rangle$. $\square$

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

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

The final example demonstrates the treatment for multiple sorts:

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

A signature generates a language of algebraic terms as follows. A **sort**-*family* $\boldsymbol{X} \in \mathbf{Set}^{\mathbf{sort}}$ is an assignment of a set $\boldsymbol{X}_{\boxbox}$, to each sort $\boxbox \in \mathbf{sort}$. We identify $\mathbf{Set}^{\{\star\}} \cong \mathbf{Set}$, and use a set-like notation to specify families, e.g. $\boldsymbol{X} \coloneqq \{x : \bullet, y, z : \circ\}$ is the two-sorted family $\boldsymbol{X}_\bullet \coloneqq \{x\}$ and $\boldsymbol{X}_\circ \coloneqq \{y, z\}$. We can turn[3] every **sort**-family $\boldsymbol{X}$ into the set $\oint \boldsymbol{X} \coloneqq \coprod_{\boxbox \in \mathbf{sort}} \boldsymbol{X}_{\boxbox}$ equipped with the injections $\mathrm{in}_{\boxbox} : \boldsymbol{X}_{\boxbox} \to \oint \boldsymbol{X}$.

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* $\boldsymbol{X}$: $\mathrm{Term}^\Sigma \boldsymbol{X} \in \mathbf{Set}^{\mathbf{sort}_\Sigma}$, $\mathrm{Term}^\Sigma_{\boxbox} \boldsymbol{X} \coloneqq \{t \mid \boldsymbol{X} \vdash_\Sigma t : \boxbox\}$ inductively:

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

---

[3] This simple construction is a special case of the Grothendieck construction, and lets us track the distinction between sets and families.

Here, the elements $x \in \boldsymbol{X}_{\boxbox}$, written $(x : \boxbox) \in \boldsymbol{X}$, represent variables of sort $\boxbox$.

A **sort**-*sorted map* $f : \boldsymbol{X} \to \boldsymbol{Y}$ is a **sort**-indexed tuple of functions between the corresponding sets: $f_{\boxbox} : \boldsymbol{X}_{\boxbox} \to \boldsymbol{Y}_{\boxbox}$, for every $\boxbox \in \mathbf{sort}$. Most of our development will utilise such sorted maps, and for now we will use them to define the standard notion of simultaneous substitution. A *substitution* $\boldsymbol{X} \vdash_{\Sigma} \theta : \boldsymbol{Y}$ is a sorted function $\theta : \boldsymbol{Y} \to \mathrm{Term}^{\Sigma}\boldsymbol{X}$, specifying which $\boxbox$-term $\boldsymbol{X} \vdash_{\Sigma} \theta_{\boxbox} y : \boxbox$ to substitute for each variable $y \in \boldsymbol{Y}_{\boxbox}$. 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:

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

2.2 Equational logic

A $\boxbox$-*sorted* Σ-*equation in context* $\boldsymbol{X}$ consists of a pair $\langle l, r\rangle \in \mathrm{Term}^{\Sigma}_{\boxbox}\boldsymbol{X}$ of $\boxbox$-sorted Σ-terms over $\boldsymbol{X}$. We write this situation as $\boldsymbol{X} \vdash_{\Sigma} l = r : \boxbox$, 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.

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, where variables and sorts are omitted:

(Associativity) $x \vee (y \vee z) = (x \vee y) \vee z$ (Idompotency) $x \vee x = x$
(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}, omitting variables and sorts:

(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, omitting variables and sorts, as well as 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 $\boldsymbol{X} \vdash_{\mathfrak{p}} t_1 = t_2 : \boxbox$ 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 deduction-closed set of equations containing the axioms.

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:

$$\{z, y : \star\} \vdash_{\mathsf{J}} t := z \vee y \qquad \theta_1 := \begin{pmatrix} z \mapsto x \vee \bot \\ y \mapsto y \end{pmatrix} \qquad \theta_2 := \begin{pmatrix} z \mapsto x \\ y \mapsto y \end{pmatrix}$$ □

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

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

$$\frac{\boldsymbol{Y} \vdash_{\Sigma_{\mathfrak{p}}} t : ⧈ \qquad \boldsymbol{X} \vdash_{\Sigma_{\mathfrak{p}}} \theta_1, \theta_2 : \boldsymbol{Y} \qquad \forall (y : ◈) \in \boldsymbol{Y}. \boldsymbol{X} \vdash_{\mathfrak{p}} \theta_1 y = \theta_2 y : ◈}{\boldsymbol{X} \vdash_{\mathfrak{p}} t\,[\theta_1] = t\,[\theta_2] : ⧈}$$

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

When a presentation $\mathfrak{p}$ proves the semi-lattice axioms in one of its sorts ⧈, then the encoding $(\boldsymbol{X} \vdash_{\Sigma_{\mathfrak{p}}} l \leq r : ⧈) \coloneqq (\boldsymbol{X} \vdash_{\Sigma_{\mathfrak{p}}} l \vee r = r : ⧈)$ of inequations as equations in this sort is a preorder w.r.t. $\mathfrak{p}$-equality, i.e.

$$(\boldsymbol{X} \vdash_{\mathfrak{p}} s \leq t \leq s : ⧈) \implies (\boldsymbol{X} \vdash_{\mathfrak{p}} s = t : ⧈)$$

We use similar encoding for $(\geq)$. Due to the monotonicity property of equational logic, once we have included an axiomatization of semi-lattices through a subset of the axioms, we may proceed to postulate inequations.

We will also use a generalisation of distributivity axioms, 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 generalization is straightforward, but technical. The main message: in a given presentation $\mathfrak{p}$, if all operators distribute over binary joins in every position, the congruence rule is valid for inequations:

$$\frac{\boldsymbol{Y} \vdash_{\Sigma_{\mathfrak{p}}} t : ⧈ \qquad \boldsymbol{X} \vdash_{\Sigma_{\mathfrak{p}}} \theta_1, \theta_2 : \boldsymbol{Y} \qquad \forall (y : ◈) \in \boldsymbol{Y}. \boldsymbol{X} \vdash_{\mathfrak{p}} \theta_1 y \leq \theta_2 y : ◈}{\boldsymbol{X} \vdash_{\mathfrak{p}} t\,[\theta_1] \leq t\,[\theta_2] : ⧈}$$

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 [6]. We let future work determine the most appropriate variety of inequational logic [32].

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: $O(\bot, \ldots, \bot) = \bot$ for every operator $(O : ⧈ \langle ◈_i \rangle_{i<\alpha}) \in \Sigma_{\mathfrak{p}}$. Such theories 'absorb' side-effects when their continuations diverge, an inherent 'partial correctness' property of Brookes's model.

The rest of this section is devoted to the technical definition of distributivity. Let Σ be a multi-sorted signature, $(P : ⧈ \langle ◈_i \rangle_{i<\alpha}) \in \Sigma$ be an operator, and $i_0 < \alpha$ be one of the positions in P's scheme. Assume further such that both $◈_{i_0}$ and ⧈ have 'single-sorted' operators $(S : ◈_{i_0} \left(\beta \cdot ◈_{i_0}\right)), (S' : ⧈ \, (\beta \cdot ⧈)) \in \Sigma$ with the same arity length β. We define the following *distributivity* axiom [17]:

$$\{x_i : \Diamond_i \mid i_0 \neq i < \alpha\} \cup \left\{y_j : \Diamond_{i_0} \,\middle|\, j < \beta\right\} \vdash_\Sigma$$
$$P\left\langle \begin{cases} i \neq i_0 : & x_i \\ i = i_0 : & S\left\langle y_j\right\rangle_j \end{cases} \right\rangle_i = S'\left\langle P\left\langle \begin{cases} i \neq i_0 : & x_i \\ i = i_0 : & y_j \end{cases} \right\rangle_i \right\rangle_j : \Box$$

which we call the *distributivity of P over S, S' in the i_0-component.*

Distributivity over binary joins implies monotonicity, in the following sense. Let $\mathfrak{p}$ be a presentation, $(O : \Box \left\langle \Diamond_i \right\rangle_{i<\alpha}) \in \Sigma_{\mathfrak{p}}$ be an operator, and $i_0 < \alpha$ an index into its sorting scheme. Assume $\Box$, $\Diamond_{i_0}$ include the theory of semilattices, and that O distributes over the binary joins of $\Diamond_{i_0}$ and $\Box$ in the i_0^{th} component. Then O is monotone in this component w.r.t. the semilattice preorder, i.e., the following deduction rule is admissible:

$$\frac{\boldsymbol{Y} \vdash_{\mathfrak{p}} l \leq r : \Diamond_{i_0}}{\{x_i : \Diamond_i \mid i_0 \neq i < \alpha\} \cup \boldsymbol{Y} \vdash_{\mathfrak{p}} O\left\langle \begin{cases} i \neq i_0 : & x_i \\ i = i_0 : & l \end{cases} \right\rangle_i \leq O\left\langle \begin{cases} i \neq i_0 : & x_i \\ i = i_0 : & r \end{cases} \right\rangle_i}$$

Specifically, if $\mathfrak{p}$ includes the theory of semilattices in all sorts, and every operator distributes over binary joins, then the congruence rule for inequations is valid.

2.3 Algebras and models

After presenting the proof theory—equational logic—lets 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 $\mathbf{A}\left[\!\left[-\right]\!\right]_{\text{op}}$, for each operator $(O : \Box \left\langle \Diamond_i \right\rangle_{i<\alpha}) \in \Sigma$, of an *operation* over this carrier: $\mathbf{A}\left[\!\left[O\right]\!\right]_{\text{op}} : (\prod_{i<\alpha} \underline{\mathbf{A}}_{\Diamond_i}) \to \underline{\mathbf{A}}_{\Box}$.

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 $\underline{\mathbf{V}X} := \mathbf{P}^{\aleph_0}(X)$, and interpret choice as union $\mathbf{L}X[\![\bigvee_\alpha]\!]_{\text{op}}\langle D_i\rangle_{i<\alpha} := \bigcup_{i<\alpha} D_i$. ☐

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 $\mathbf{M}[\![\mathrm{Id}_n]\!]_{\text{op}} := I_n \in \mathbb{M}^{\mathbb{R}}_{n\times n}$ as the identity matrix, and composition $\mathbf{M}[\![(\circ)]\!]_{\text{op}} := (\cdot)$ 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 $\mathbf{A}^{\mathsf{op}}[\![\mathrm{Id}_n]\!]_{\text{op}} := \mathbf{A}[\![\mathrm{Id}_n]\!]_{\text{op}}$, and reversing composition $\mathbf{A}^{\mathsf{op}}[\![(\circ)]\!]_{\text{op}}(A, B) := \mathbf{A}[\![(\circ)]\!]_{\text{op}}(B, A)$. ☐

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: $(\mathbf{F}^\Sigma\boldsymbol{X})\left[\!\left[O\right]\!\right]_{\text{op}} \left\langle t_i\right\rangle_i := O\left\langle t_i\right\rangle_i$. ☐

A Σ-algebra allows us to interpret every Σ-term, given values for 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, we can interpret every term by induction:

$$\mathbf{A}\left[\!\left[\boldsymbol{X} \vdash_\Sigma x : \Box\right]\!\right]_{\text{term}} e := e_{\Box} x \qquad \mathbf{A}\left[\!\left[O\left\langle t_i\right\rangle_i\right]\!\right]_{\text{term}} e := \mathbf{A}\left[\!\left[O\right]\!\right]_{\text{op}} \left\langle \mathbf{A}\left[\!\left[t_i\right]\!\right]_{\text{term}} e\right\rangle_i$$

Example 11 (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. $\square$

A Σ-algebra $\mathbf{A}$ *validates* the equation $\boldsymbol{X} \vdash_{\Sigma} l = r : \boxdot$ 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 $\mathbf{A} \vdash \boldsymbol{X} \vdash_{\Sigma} l = r : \boxdot$. 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 12. 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 the algebra of terms is an $\emptyset$-model. $\square$

Example 13. 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 $\mathbf{A} \not\vdash x : \star \vdash_{\mathsf{J}} x \vee x = x : \star$. $\square$

2.4 Representability

The final concept we need is the representation of free models. It specifies when the elements in a given $\mathfrak{p}$-model represent the $\Sigma_{\mathfrak{p}}$-terms up-to provable equality in $\mathfrak{p}$. Our main technical contribution (§4) is to show that Brookes's trace semantics, generalised appropriately, is the free model for a two-sorted algebraic theory.

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: $\varphi(\mathbf{A}\,[\![O]\!]_{\mathrm{op}}\,(a_1, \dots, a_\alpha)) = \mathbf{B}\,[\![O]\!]_{\mathrm{op}}\,(\varphi a_1, \dots, \varphi a_\alpha)$.

Example 14. 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 $(A \cdot B)^{\top} = B^{\top} \cdot A^{\top}$. $\square$

Example 15 (evaluation homomorphism). Evaluation using any $\boldsymbol{X}$-environment $e : \boldsymbol{X} \to \underline{\mathbf{A}}$ in a Σ-algebra $\mathbf{A}$ is a homomorphism $\mathbf{A}[\![-]\!]_{\mathrm{term}}e : \mathbf{F}^{\Sigma}\boldsymbol{X} \to \mathbf{A}$. $\square$

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 $x \in \boldsymbol{X}_{\boxdot}$, we have: $e^{\#}_{\boxdot}(\mathrm{return}_{\boxdot}\, a) = ea$. 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 [33] 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 $e^{\#} : \mathbf{V}X \to \mathbf{B}$, $e^{\#}D := \bigcup_{x \in D} ex$. The data $\langle X \mapsto \underline{\mathbf{V}X}, \mathrm{return}, (-)^{\#}\rangle$ is a monad. $\square$

3 Shared state

To define the equational theory of shared state, we first recall the standard, single sorted *(non-deterministic) global state* theory G [16, 27, 33]. 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 $\mathbb{L} = \{\mathtt{l}_1, \mathtt{l}_2\}$ and use non-bracketed vectors for stores, e.g. $\begin{smallmatrix}\mathtt{1}\\ \mathtt{0}\end{smallmatrix}$ denotes $\left(\begin{smallmatrix}\mathtt{l}_1 \mapsto \mathtt{1}\\ \mathtt{l}_2 \mapsto \mathtt{0}\end{smallmatrix}\right)$.

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

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

The induced algebraic theory [33] includes other familiar axioms [27]. 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 $\boldsymbol{\mathbb{S}}$ of *shared state* extends global state. Its sorts are $\mathbf{sort}_{\Sigma_{\boldsymbol{\mathbb{S}}}} = \{\bullet, \circ\}$. 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 $\Sigma_{\boldsymbol{\mathbb{S}}}$ 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 $\mathrm{Ax}_{\boldsymbol{\mathbb{S}}}$ 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, $\boldsymbol{\mathbb{S}}$ 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, $\mathbb{S}$ supports inequational reasoning, which represents the semantic refinement relation used to validate program transformations [e.g. 12]. Finally, $\mathrm{Ax}_{\mathbb{S}}$ axiomatises $\lhd$ and $\rhd$ as an *(insertion)-closure pair* [e.g. 2]:

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

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.

Connect $(\rhd \lhd x \geq x)$**.** Releasing and immediately acquiring the lock only allows more behaviours, as the environment is not obliged to interleave.

To summerise, $\mathrm{Ax}_{\mathbb{S}} \coloneqq \mathrm{Ax}^{\bullet}_{\mathsf{G}} \cup \mathrm{Ax}^{\circ}_{\mathsf{V}} \cup \{\text{ND-}\rhd, \text{ND-}\lhd\} \cup \{\text{Empty}, \text{Connect}\}$.

Example 17. The $\Sigma_{\mathbb{S}}$-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 discards 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 $\mathbb{S}$ does not prove Write Intro in example 25. Here we show $\mathbb{S}$ proves the other two:

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

$$\lhd \mathsf{U}_{\ell,b_1} \rhd \lhd \mathsf{U}_{\ell,b_2} \rhd x \overset{\text{Connect}}{\geq} \lhd \mathsf{U}_{\ell,b_1}\, \mathsf{U}_{\ell,b_2} \rhd x \overset{\texttt{UU}}{=} \lhd \mathsf{U}_{\ell,b_2} \rhd x \qquad \square$$

4 Representation

We now establish the representation theorem describing a free $\mathbb{S}$-model over any $\boldsymbol{X} \in \mathbf{Set}^{\{\bullet,\circ\}}$. Following Brookes [7], we use sets of traces to denote behaviours.

4.1 Sorted traces

A *sorted trace* starts with a sort (● or ○) 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 (§5). For any family $\boldsymbol{X} \in \mathbf{Set}^{\{●,○\}}$ define the $\{●, ○\}$-sorted family $\boldsymbol{\mathsf{T}}\boldsymbol{X}$ of *traces* $(\boldsymbol{\mathsf{T}}\boldsymbol{X})_{⧈} := \mathsf{T} \oint \boldsymbol{X}$. Then, for any sorted family $\boldsymbol{X} \in \mathbf{Set}^{\{●,○\}}$, we define the set of *sorted traces over $\boldsymbol{X}$* by:

$$\mathbb{T}\boldsymbol{X} := \oint \boldsymbol{\mathsf{T}}\boldsymbol{X} = \{●, ○\} \times (\mathbb{S} \times \mathbb{S})^{+} \times \coprod\nolimits_{◈ \in \{●,○\}} \boldsymbol{X}_{◈}$$

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

Example 18. $●\langle \substack{1\\1}, \substack{1\\0} \rangle \langle \substack{1\\1}, \substack{0\\0} \rangle ○7 \in \mathbb{T}\boldsymbol{X}$, with $\boldsymbol{X}_{○} = \mathbb{N}$, is ●-sorted and ○-valued. ☐

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

Example 19. The behaviour denoted by $●\langle \substack{1\\1}, \substack{1\\0} \rangle \langle \substack{1\\1}, \substack{0\\0} \rangle ○7$ relies on the preceding environment for $\substack{1\\1}$ and for the sequential environment to hold access to the store; then guarantees $\substack{1\\0}$; then relies on $\substack{1\\1}$; and finally guarantees $\substack{0\\0}$, 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 (○) between transitions. For example, for the trace $●\langle \substack{1\\1}, \substack{1\\0} \rangle \langle \substack{1\\1}, \substack{0\\0} \rangle ○7$, we could write $●\langle \substack{1\\1}, \substack{1\\0} \rangle ○ \langle \substack{1\\1}, \substack{0\\0} \rangle ○7$ for emphasis. A hypothetical $●\langle \substack{1\\1}, \substack{1\\0} \rangle ● \langle \substack{1\\1}, \substack{0\\0} \rangle ○7$ 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 stutter ($\xrightarrow{\mathsf{st}}$) and mumble ($\xrightarrow{\mathsf{mu}}$), 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 $⧈\xi^{?}_{1}○\xi^{?}_{2}◈x$ for the trace $⧈\xi^{?}_{1}\xi^{?}_{2}◈x$ in which, intuitively, the lock is ceded (○) at the marked spot. Formally, we require that both (a) if $\xi^{?}_{1}$ is empty, then $⧈ = ○$; and (b) if $\xi^{?}_{2}$ is empty, then $◈ = ○$. 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 ○-sorted ○-valued trace, i.e. $⧈ = ○ = ◈$.

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

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

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

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

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 §5.

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

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

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

$$\bullet\langle\substack{1\\1},\substack{1\\0}\rangle\langle\substack{1\\1},\substack{0\\0}\rangle\circ 7 \cancel{\xrightarrow{\texttt{st}}} \bullet\langle\substack{0\\1},\substack{0\\1}\rangle\langle\substack{1\\1},\substack{1\\0}\rangle\langle\substack{1\\1},\substack{0\\0}\rangle\circ 7$$

The source protocol relies on the preceding sequential environment for $\substack{1\\1}$. 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 ($\boxdot\xi_1^?\circ\xi_2^?\Diamond x \xrightarrow{\texttt{st}} \boxdot\xi_1^?\langle\sigma,\sigma\rangle\xi_2^?\Diamond x$) means a thread-pool also obeys the protocol that guarantees a state σ by relying on its environment for σ.

Mumbling ($\boxdot\xi_1^?\langle\sigma,\rho\rangle\langle\rho,\theta\rangle\xi_2^?\Diamond x \xrightarrow{\texttt{mu}} \boxdot\xi_1^?\langle\sigma,\theta\rangle\xi_2^?\Diamond x$) means a thread-pool which 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 $\boxdot \in \{\bullet,\circ\}$, define a $\{\bullet,\circ\}$-sorted family $\mathbf{P}^{\aleph_0}(U)$ by taking its $\boxdot$ component to be the set $\mathbf{P}^{\aleph_0}_{\boxdot}(U)$ of countable subsets of U whose elements are all $\boxdot$-sorted. Similarly, define $\mathbf{P}^{\dagger}_{\boxdot}(U) \subseteq \mathbf{P}^{\aleph_0}_{\boxdot}(U)$ to be the set of *closed* countable subsets of U whose elements are all $\boxdot$-sorted.

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

$$(\sigma,\rho) : \mathbf{P}^{\aleph_0}_{\bullet}(\mathbb{T}\boldsymbol{X}) \to \mathbf{P}^{\aleph_0}_{\bullet}(\mathbb{T}\boldsymbol{X}) \qquad (\sigma,\rho)\,K := \left\{\bullet\langle\sigma,\theta\rangle\xi^?\Diamond x \;\middle|\; \bullet\langle\rho,\theta\rangle\xi^?\Diamond x \in K\right\}$$

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

4.2 Representation theorem

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

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

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

The rest of this section establishes that the algebra $\langle \mathbf{R}\boldsymbol{X}, \mathrm{return}\rangle$ over $\boldsymbol{X}$ is a free $\$$-model over $\boldsymbol{X}$. A key ingredient is *reification*: for any $\{\bullet,\circ\}$-sorted family $\boldsymbol{X}$, we define a sorted-function $\mathrm{reify} : \mathbf{P}^{\aleph_0}(\mathbb{T}\boldsymbol{X}) \to \mathrm{Term}^{\Sigma_{\$}}\boldsymbol{X}$, choosing a representative term $t_2 := \mathrm{reify}\,[\![\boldsymbol{X} \vdash t_1]\!]_{\mathrm{term}}$ such that $\boldsymbol{X} \vdash_{\$} t_1 = t_2$. 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 $x : \bullet \vdash_{\Sigma_{\$}} \mathsf{A}_{\ell,b}\, x : \bullet$ that looks ℓ up and only continues if it holds b:

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

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

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

Define the $\Sigma_{\$}$-term reifying a trace $x : \Diamond \vdash_{\Sigma_{\$}} \underline{\square\xi\Diamond x} : \square$ by sequencing open transition as they are in ξ, separated by $\rhd \lhd$; and delimited by $\lhd$ on the left if $\square = \circ$ and by $\rhd$ on the right if $\Diamond = \circ$.

Example 22. $x : \circ \vdash_{\Sigma_{\$}} \underline{\bullet\langle\sigma,\rho\rangle\langle\sigma',\rho'\rangle\circ x} := \lbrace\!\!\lbrace \sigma, \rho \rbrace\!\!\rbrace \rhd \lhd \lbrace\!\!\lbrace \sigma', \rho' \rbrace\!\!\rbrace \rhd x : \bullet$ ☐

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 $\{x : \Diamond\} \vdash_{\Sigma_{\$}} \underline{\tau_1} \geq \underline{\tau_2} : \square$.*

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

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

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

With reification defined, we are ready to state the representation theorem.

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

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

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

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

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

5 Recovering Brookes's model

The theory $\boldsymbol{\mathbb{S}}$ recovers Brookes's model (§5.1). We recover it twice, using different strategies that offer different perspectives. First, we transform the monad induced by the representation of §4.2 along a right adjoint $\mathbf{Set}^{\{\bullet,\circ\}} \to \mathbf{Set}$ (§5.2). Then, we define an embedding translation from a single-sorted theory of transitions into $\boldsymbol{\mathbb{S}}$ (§5.4), corresponding to Brookes's `await` construct (§5.3).

5.1 Brookes's model

We designed our notions of traces, deduction, etc. from §4.1 based on the following model of Brookes [7]. For any set $X \in \mathbf{Set}$, recall the set of Brookes traces $\mathsf{T}X := (\mathbb{S} \times \mathbb{S})^{+} \times X$ from §4.1. Writing ξx for $\langle \xi, x \rangle$, Brookes's stutter and mumble trace deductions are:

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

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

The difference between Brookes's and our multi-sorted deuctions is the maintenance of the sort in the ends of the trace. In particular, Brookes's stutter does not need to assume the 'cede' sort ($\circ$) at the stuttering position in the source. In Brookes's model, the environment may always interleave in either end.

Brookes's semantic domain $BX := \mathbf{P}^{\dagger}(\mathsf{T}X)$ 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:

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

These *generic effects* [34] correspond to these monadic algebraic operations:

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

5.2 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 to represent $\circ$-sorted $\circ$-valued traces.

There is an abstract construction that recovers the monad and its operations in §5.2 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 §2.1. Monads transform along adjoints, and transforming the monad obtained standardly from the representation of §4.2 along the adjunction above results in Brookes's model. Explicitly, denoting $B_{\circ}X \coloneqq \underline{\mathbf{R}X^{\circ}}_{\circ} = \mathbf{P}^{\dagger}_{\circ}(\mathbb{T}X^{\circ})$, the resulting monad over **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 §4.2 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:

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

5.3 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-join semilattice Σ_{V} and a unary operator $\langle\sigma, \rho\rangle$ for every $\sigma, \rho \in \mathbb{S}$. The axioms Ax_{B} consists of countable-join semilattice Ax_{V}, commutativity axioms (ND-B) $\langle\sigma, \rho\rangle \bigvee_{i<\alpha} x_i = \bigvee_{i<\alpha} \langle\sigma, \rho\rangle x_i$, and:

Trace closure

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

The first two axiom schemes are algebraic counterparts to mumble and $\mathsf{stutter}$. 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 of sets of traces K to be closure under, in addition to Brookes's mumble and $\mathsf{stutter}$ trace deductions, the following closure condition:

$$\frac{\forall\sigma.\, \xi^{?}_{1}\langle\sigma, \sigma\rangle\xi^{?}_{2}x \in K}{\xi^{?}_{1}\xi^{?}_{2}x \in K}(\mathsf{hush})$$

The closure rule hush is admissible for trace-deduction closed K, due to the non-emptiness of the traces and closure under mumble. 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 would be hard to notice without this algebraic analysis.

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

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

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

To prove that this is a free B-model, we use reification as in §4.2, though here reification is more straightforward. A trace is reified as itself, and sets of traces use countable-join 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* $e^{\#}_{\square} K := \mathbf{B}X[\![\mathrm{reify}_{\square}\, K]\!]_{\mathrm{term}} e$*.*

5.4 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: $(\epsilon \boldsymbol{Y})_{\square} := \boldsymbol{Y}_{\epsilon\square}$. 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 $(\epsilon^* \boldsymbol{X})_{\Diamond} := \coprod_{\epsilon\square=\Diamond} \boldsymbol{X}_{\square}$. When ϵ is injective, the left adjoint is given by the simpler formula $\epsilon^* \boldsymbol{X} := \{x : \epsilon\square \mid x \in \boldsymbol{X}_{\square}\}$.

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 §5.2, 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 $(O : \square\langle \Diamond_i\rangle_{i<\alpha}) \in \Sigma_1$, of a term $\mathbf{E}O \in \mathrm{Term}^{\Sigma_2}_{\epsilon\square} \{x_i : \epsilon\Diamond_i \mid i < \alpha\}$. 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:

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

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:

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

☐

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.

5.5 Translation through the two-sorted theory of transitions

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

Open transition axioms

$$\begin{aligned}
&(\mathrm{HS})\ \ x = \textstyle\bigvee_{\sigma \in \mathbb{S}} \llparenthesis \sigma, \sigma \rrparenthesis\, x
&&(\mathrm{Seq}^{=})\ \ \llparenthesis \sigma, \rho \rrparenthesis\, \llparenthesis \rho, \theta \rrparenthesis\, x = \llparenthesis \sigma, \theta \rrparenthesis\, x \\
& &&(\mathrm{Seq}^{\neq})\ \ \llparenthesis \sigma, \rho \rrparenthesis\, \llparenthesis \mu, \theta \rrparenthesis\, x = \bot \qquad \rho \neq \mu
\end{aligned}$$

Define the translation $\mathbf{E}_{\mathsf{G}} : \mathsf{Tgs} \rightarrowtail \mathsf{G}$ by interpreting transitions as the open transitions from §4.2: $\mathbf{E}_{\mathsf{G}_{\mathrm{tln}}} \llparenthesis \sigma, \rho \rrparenthesis := \Lbag \sigma, \rho \Rbag\, x_0$. Conversely, $\mathbf{E}_{\mathsf{Tgs}} : \mathsf{G} \rightarrowtail \mathsf{Tgs}$ by interpreting lookup and update as follows, similar to the representation of §4.2:

$$\mathbf{E}_{\mathsf{Tgs}_{\mathrm{tln}}} \mathsf{U}_{\ell,b} := \textstyle\bigvee_{\sigma \in \mathbb{S}} \llparenthesis \sigma, \sigma[\ell \mapsto b] \rrparenthesis\, x_0 \qquad \mathbf{E}_{\mathsf{Tgs}_{\mathrm{tln}}} \mathsf{L}_{\ell} := \textstyle\bigvee_{\sigma \in \mathbb{S}} \llparenthesis \sigma, \sigma \rrparenthesis\, x_{\sigma_\ell}$$

These witness an equivalence: $\mathsf{G} \simeq \mathsf{Tgs}$.

This equivalence lets us use Tgs instead of G in the atomic block layer of $\boldsymbol{\mathbb{S}}$. In detail, the presentation Tr of the two-sorted theory of transitions is given by $\mathrm{Ax}_{\mathsf{Tr}} := \boxed{\mathrm{Ax}^{\bullet}_{\mathsf{Tgs}}} \cup \mathrm{Ax}^{\circ}_{\mathsf{V}} \cup \{\text{ND-}\triangleright, \text{ND-}\triangleleft\} \cup \{\text{Empty}, \text{Connect}\}$. Extending the translations $\mathbf{E}_{\mathsf{Tgs}}$ and $\mathbf{E}_{\mathsf{G}}$ to all of the operators gives an equivalence $\mathsf{Tr} \simeq \boldsymbol{\mathbb{S}}$, and so they induce the same monad, and recover Brookes's model.

Define the translation $\mathbf{E} : \mathsf{B} \rightarrowtail \mathsf{Tr}$ along $\star \mapsto \circ$ by sending transitions to their delimited open counterparts: $\mathbf{E}_{\mathrm{tln}} \langle \sigma, \rho \rangle := \triangleleft \llparenthesis \sigma, \rho \rrparenthesis \triangleright x_0$. By post-composition with the above equivalence, the single-sorted theory of transitions translate to shared state $\mathsf{B} \rightarrowtail \boldsymbol{\mathbb{S}}$. Brookes's model, being a free B-model, is thus the $\circ$-sorted fragment of $\boldsymbol{\mathbb{S}}$ over $\circ$-variables, formally.

6 Conclusion and further work

We presented an equational theory for shared state ($\boldsymbol{\mathbb{S}}$). 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 ($\circ$), the concurrent environment may interleave, but memory access is forbidden. We also presented theories of transitions (Tr and Tgs) and formally related them to the shared state theory. One of these theories (Tr) is a single-sorted theory that recovers Brookes's model. We find this theory unsatisfying for a conceptual and a technical reason. Conceptually, it is a theory of Brookes's `await` construct, which we find unnatural. Technically, Tr does not admit global state as an explicit component of the theory. We believe understanding how global state fits as a component will inform modelling other effects in the concurrent setting. The theory of shared state addresses these concerns. On the one hand, it admits the global state theory as-is, and axiomatizes the interleaving-enabling/disabling operators ($\triangleleft/\triangleright$) without explicit interaction with global state. On the other hand, this theory recovers

Brookes's model precisely in a principled manner: by transforming a monad and its operations along an adjunction, and through algebraic translations.

Our theory uses countable-join semilattices. In the resulting—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 [32]. We can also restrict the semantics to terminating languages by using finite-join semilattice instead of countable joins. The resulting representation theorem then uses finitely-generated closed subsets.

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 general parallel composition operation satisfying the expected laws of concurrent programming [15, 29, 37].

We want to model more effects similarly, within this modular multi-sorted algebraic framework. These effects include: more advanced notions of state, such as dynamic allocation [20], higher-order memory cells [26, 39], and weak memory [13]; control-flow effects such as exceptions and effect handlers [4]; and probabilistic programming with shared state [24].

Our two sorts limit access to the whole store. We would like to explore limiting access in finer granularity, and per-location in the first instance. In this direction, the theory has: sorts for every finite subset $s \subseteq \mathbb{L}$ of locations; and operators:

$$\lhd_{\ell} : s \setminus \{\ell\} \, \langle s \cup \{\ell\} \rangle \qquad \rhd_{\ell} : s \cup \{\ell\} \, \langle s \setminus \{\ell\} \rangle$$

One needs care in designing the appropriate (in)equations for these operators.

It may be interesting to design programming language constructs that expose the sort discipline in the surface language. It is natural to expose them as locking/unlocking, while tracking the capability to call the lock in typing judgements. This construct explicates regions that rule out data-races with the environment. It seems such typing judgements would rule out deadlocks structurally, and so may limit program expressiveness, or be hard to use. It remains to be seen whether such abstractions are useful.

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.

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

Bibliography

[1] Abadi, M., Plotkin, G.D.: A model of cooperative threads. Log. Methods Comput. Sci. **6**(4) (2010), https://doi.org/10.2168/LMCS-6(4:2)2010

[2] 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

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

[4] 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

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

[6] 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

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

[8] 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

[9] 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

[10] 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

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

[12] 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

[13] 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

[14] 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

[15] 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

[16] 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

[17] 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

[18] 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

[19] Kammar, O.: Algebraic theory of type-and-effect systems. Ph.D. thesis, University of Edinburgh, UK (2014), URL http://hdl.handle.net/1842/8910

[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] Kammar, O., McDermott, D.: Factorisation systems for logical relations and monadic lifting in type-and-effect system semantics. In: MFPS, Electronic Notes in Theoretical Computer Science, vol. 341, Elsevier (2018), https://doi.org/10.1016/j.entcs.2018.11.012

[22] Kammar, O., Plotkin, G.D.: Algebraic foundations for effect-dependent optimisations. In: POPL, ACM (2012), https://doi.org/10.1145/2103656.2103698

[23] 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

[24] 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

[25] 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)

[26] 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

[27] 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

[28] Moggi, E.: Notions of computation and monads. Inf. Comput. **93**(1) (1991), https://doi.org/10.1016/0890-5401(91)90052-4

[29] 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

[30] Plotkin, G.D.: A powerdomain for countable non-determinism. In: Automata, Languages and Programming, Springer, Berlin, Heidelberg (1982), ISBN 978-3-540-39308-5

[31] Plotkin, G.D.: Hennessy-plotkin-brookes revisited. In: FSTTCS, Lecture Notes in Computer Science, vol. 4337, Springer (2006), `https://doi.org/10.1007/11944836_2`

[32] 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`

[33] 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`

[34] 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`

[35] 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`

[36] 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`

[37] Rivas, E., Jaskelioff, M.: Monads with merging (Jun 2019), URL `https://inria.hal.science/hal-02150199`, working paper or preprint

[38] 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`

[39] Sterling, J., Gratzer, D., Birkedal, L.: Denotational semantics of general store and polymorphism (2023), URL `https://arxiv.org/abs/2210.02169`

[40] Svyatlovskiy, M., Mermelstein, S., Lahav, O.: Compositional semantics for shared-variable concurrency. Proc. ACM Program. Lang. **8**(PLDI) (2024), `https://doi.org/10.1145/3656399`

[41] 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`

[42] Turon, A.J., Wand, M.: A separation logic for refining concurrent objects. In: POPL, ACM (2011), `https://doi.org/10.1145/1926385.1926415`

[43] 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`

Acknowledgments. This work was funded by a Royal Society University Research Fellowship. 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 the following people for interesting and useful discussions and suggestions: Danel Ahman, Andrej Bauer, Martín Escardó, Justus Matthiesen, Sam Staton, and Rob van Glabbeek.

A No-go results

We can present Brookes's model using a single-sorted presentation (§5.3). However, we found this presentation unsatisfactory, and so propose a two-sorted account. Our use of the two-sorted approach follows a relatively thorough investigation into alternative single-sorted approaches, and we can provide some crisp results that certain single-sorted approaches fail. These no-go results, together with the perspectives on future work the two-sorted decomposition suggests (§6), are evidence for the merit of our two-sorted approach. They may also inform future search for a single-sorted presentation that we have overlooked.

Single-sorted transitions present Brookes's model in terms of the `await` construct. This presentation highlights `await`'s importance for reasoning in Brookes's model and why `await` is a key ingredient in Brookes's full abstraction result. Without `await`, Brookes's model is not fully abstract at 1^{st}-order:

No-go 1 (Svyatlovskiy et al. [40]). *Brookes's model is not fully-abstract w.r.t. the operational semantics in which differentiating contexts can only read and mutate single memory cells atomically.*

Moreover, every single-sorted presentation of Brookes's model must involve operators other than the interpretation of read and write, considered as generic effects [34]. Formally, given a family of algebraic operations and a monad, we can construct the sub-monad generated by a set of operations [19, 21, 22].

No-go 2. *The sub-monad generated by the semantics of read and write, and by union, differs from the Brookes model.*

Proof. The trace-sets generated by read and write always contain a trace in which at most one cell changes within each transition. Brookes's model includes other subsets, definable via the `await` construct. ☐

The traces in Brookes's model explicitly yield control to their concurrent environment. Following Abadi and Plotkin [1], we investigated adding an additional unary operator Y for yielding control to the concurrent environment. It is natural to interpret Y as adding a no-op transition $\langle\sigma,\sigma\rangle$ before every trace in its argument, modelling a possible interference by the environment. An alternative choice is to add such no-op transitions and also keep the original traces, modelling a *possibility* for a yield in the previous sense. Both of these options trivialize in Brookes's model:

No-go 3. *Consider the following interpretations of* Y *in Brookes's model:*

$$[\![\mathsf{Y}]\!]^1_{\mathrm{op}} K := \{\langle\sigma,\sigma\rangle\tau \mid \tau \in K\} \qquad [\![\mathsf{Y}]\!]^2_{\mathrm{op}} K := K \cup [\![\mathsf{Y}]\!]^1_{\mathrm{op}} K$$

Then $[\![\mathsf{Y}]\!]^i_{\mathrm{op}} K = K$ *for both* $i \in \{1,2\}$*, for any closed* K*.*

Proof. K is closed under $\mathsf{stutter}$ and hush. ☐

Even though Brookes's model does not support this intuition, we explored where the yield approach leads. With this yield operator, lookup and update can represent interference-free memory-access as axiomatized in the global-state theory, and surface-language level read and write can be modelled by some combination of the algebraic operators. Formally, let Res be a presentation that includes non-deterministic global state, and the yield operator Y, which is Res-provably strict and distributes over joins.

Option 1 (Dvir et al.'s presentation [12]). For a previous theory of ours, we took a minimal Res satisfying our restrictions, and defined the algebraic representation of read:

$$\mathsf{R}_\ell(x_0, x_1) \coloneqq \left(x_0, x_1 \vdash_{\Sigma_{\mathsf{Res}}} \mathsf{L}_\ell((x_0 \vee \mathsf{Y}\, x_0), (x_1 \vee \mathsf{Y}\, x_1))\right)$$

Reading *may* admit an interference point after looking the value up in memory.

Option 2 (Plotkin's presentation [31]). Another natural option is to take Res to also prove that Y is a closure operator, i.e. $x \vdash_{\mathsf{Res}} \mathsf{Y}\,\mathsf{Y}\, x = \mathsf{Y}\, x \geq x$. In this option, the intuition for Y is that of a *possible* yield, and possibly yielding twice is the same as once. This theory allows the algebraic representation of read to be a bit more natural:

$$\mathsf{R}_\ell(x_0, x_1) \coloneqq \left(x_0, x_1 \vdash_{\Sigma_{\mathsf{Res}}} \mathsf{Y}\,\mathsf{L}_\ell(\mathsf{Y}\, x_0, \mathsf{Y}\, x_1)\right)$$

Both options prove (Irrelevant Read Elim), but not (Irrelevant Read Intro):

$$x \vdash_{\mathsf{Res}} \mathsf{R}_\ell(x, x) \geq x \qquad \text{(Irrelevant Read Elim)}$$
$$x \nvdash_{\mathsf{Res}} \mathsf{R}_\ell(x, x) \leq x \qquad \text{(Irrelevant Read Intro)}$$

Brookes's model validates (Irrelevant Read Intro), so the proposed theories are both not abstract enough. Adding (Irrelevant Read Intro) as an axiom in either version is problematic, as it implies the following inequation:

$$x \vdash_{\Sigma_{\mathsf{Res}}} \mathsf{R}_\ell(\mathsf{R}_\ell(x_{0,0}, x_{0,1}), \mathsf{R}_\ell(x_{1,0}, x_{1,1})) \leq \mathsf{R}_\ell(x_{0,0}, x_{1,1}) \qquad \text{(Same Read Intro)}$$

The corresponding program transformation is invalid in our setting because the environment can interfere, mutating ℓ between the successive reads.

We summarise this intermediate result:

No-go 4. *Let* Res *be either Dvir et al.'s or Plotkin's presentation, and define* R_ℓ *accordingly. if* (Irrelevant Read Elim) *and* (Irrelevant Read Intro) *are valid in* Res*, then so is* (Same Read Intro)*.*

Another approach is to add unary operators $\lhd'$ and $\rhd'$ that delimit the memory accesses. Formally, let Del be a presentation that includes non-deterministic global state, and the delimiting operators $\lhd'$ and $\rhd'$, which are Del-provably strict and distribute over joins. Define the algebraic representation of read:

$$\mathsf{R}_\ell(x_0, x_1) \coloneqq \left(x_0, x_1 \vdash_{\Sigma_{\mathsf{Res}}} \lhd'\, \mathsf{L}_\ell(\rhd'\, x_0, \rhd'\, x_1)\right) \qquad (\star)$$

This approach subsumes the two Res options suggested above, by using the axioms $x \vdash \lhd' x = x$ and $x \vdash \rhd' x = x \vee \mathsf{Y}\, x$ for Dvir et al.'s presentations; and using $x \vdash \lhd' x = \mathsf{Y}\, x$ and $x \vdash \rhd' x = \mathsf{Y}\, x$ for Plotkin's presentation. In both cases, and more generally whenever $\lhd'$ and $\rhd'$ are given by a combination of joins and yields, they commute:

Lemma 30. *Let* t_1 *and* t_2 *be* $\{\vee, \mathsf{Y}\}$*-term over* $\{x\}$*. If* $x \vdash_{\mathsf{Del}} \lhd' x = t_1$ *and* $x \vdash_{\mathsf{Del}} \rhd' x = t_2$*, then* $x \vdash_{\mathsf{Del}} \lhd' \rhd' x = \rhd' \lhd' x$*.*

Proof. Using the semilattice axioms and distributivity of Y over joins, every $\{\vee, \mathsf{Y}\}$-term t over $\{x\}$ is Del-equal to a non-deterministic choice between terms of the form $\mathsf{Y}^n\, x$ for $n \in N_t \subseteq \mathbb{N}$. Both terms above are equal to the same term of this form, with $N = \{n_1 + n_2 \mid n_1 \in N_{\lhd' x}, n_2 \in N_{\rhd' x}\}$. ☐

Any alternative of Del for which $\lhd'$ and $\rhd'$ commute is not satisfactory:

No-go 5. *Let* Del *be a presentation that includes non-deterministic global state, and the unary operators* $\lhd'$ *and* $\rhd'$*, which* Del *proves to be strict, distribute over joins, and commute. With read from* ($\star$)*, if* Del *proves* (Irrelevant Read Elim) *and* (Irrelevant Read Intro)*, then it proves* (Same Read Intro)*.*

Proof. Combining (Irrelevant Read Elim) and (Irrelevant Read Intro), we have $x \vdash_{\mathsf{Del}} \mathsf{R}_\ell(x, x) = x$. Using global-state, we have $x \vdash_{\mathsf{Del}} \mathsf{R}_\ell(x, x) = \lhd' \rhd' x$. Therefore, $x \vdash_{\mathsf{Del}} \lhd' \rhd' x = x$. They commute, so $x \vdash_{\mathsf{Del}} \rhd' \lhd' x = x$. Using global-state, we prove (Same Read Intro) in Del. ☐

Therefore, any such theory Del is either unsound, or it fails to validate a transformation that Brookes's model does. Thus, when picking Del, we need to make sure that $\lhd'$ and $\rhd'$ do not commute.

As a final option we cover here, we could take the axioms $x \vdash \lhd' \rhd' x = x$ and $x \vdash \rhd' \lhd' x \geq x$. These are like the closure pair axioms of our shared-state presentation $\mathbb{S}$, but without the sort discipline. The single-sorted signature allows ill-bracketed terms such as $x \vdash \lhd' \lhd' x$. Though it may be possible to axiomatize that all such terms are equal to $\bot$, a more principled way to avoid such terms is to use a two-sorted theory as we have.

The analysis we offered in this section does not rule out the possibility of a satisfactory single-sorted theory of shared-state. We hope that these considerations could inform future pursuit of this theory, or a tighter no-go result.

B Proof of the representation theorem

To start, we first prove proposition 23, soundness of encoded trace deductions:

Proof. First, standardly in G we have $x : \star \vdash_{\mathsf{G}} \{\!\!\{\sigma, \rho\}\!\!\} \{\!\!\{\rho', \theta\}\!\!\}\, x \geq \{\!\!\{\sigma, \theta\}\!\!\}\, x : \star$ and $x : \star \vdash_{\mathsf{G}} \{\!\!\{\sigma, \sigma\}\!\!\}\, x \geq x : \star$, which are included in the $\bullet$ sort in $\mathbb{S}$.

- The former, combined with Connect, leads to soundness of mumble.

- The latter, combined with Empty, leads to soundness of `stutter`. ☐

That reification is indifferent to closure follows:

Proposition 31. *For* $K \in \mathbf{P}_{\boxdot}^{\aleph_0}(\mathbb{T}\boldsymbol{X})$, $\boldsymbol{X} \vdash_{\boldsymbol{\mathbb{S}}} \mathrm{reify}_{\boxdot} K = \mathrm{reify}_{\boxdot} K^{\dagger} : \boxdot$.

Proof. Follows from proposition 23 by inequational reasoning. ☐

To prove the $\boldsymbol{\mathbb{S}}$-Rep. Thm., let $\boldsymbol{X} \in \mathbf{Set}^{\{\bullet,\circ\}}$. We start by giving alternative formulas to the interpretations of the lock operators.

Lemma 32. *Denote the set of sequences of transitions, where each transition has equal components* $\mathbb{S}^{*}_{=} := \{\langle\sigma,\sigma\rangle \mid \sigma \in \mathbb{S}\}^{*}$. *The following hold:*

$$\mathbf{R}\boldsymbol{X}\,[\![\lhd]\!]_{\mathrm{op}}\,K = \left\{\circ\xi_0^?\xi\Diamond x \,\middle|\, \xi_0^? \in \mathbb{S}^{*}_{=}, \bullet\xi\Diamond x \in K\right\}$$
$$\mathbf{R}\boldsymbol{X}\,[\![\rhd]\!]_{\mathrm{op}}\,K = \{\bullet\xi\Diamond x, \bullet\langle\sigma,\sigma\rangle\xi\Diamond x \mid \sigma \in \mathbb{S}, \circ\xi\Diamond x \in K\}$$

Proof sketch. The fact that K is closed means that most trace deductions afforded in the interpretations as defined in the $\boldsymbol{\mathbb{S}}$-Rep. Thm. are redundant.

- In $\mathbf{R}\boldsymbol{X}\,[\![\lhd]\!]_{\mathrm{op}}\,K$, the only application of a trace deduction that results in a trace that would is not in the set before taking the closure is one of `stutter` at the start of the trace.
- In $\mathbf{R}\boldsymbol{X}\,[\![\rhd]\!]_{\mathrm{op}}\,K$, the only application of a trace deduction that results in a trace that would is not in the set before taking the closure is one of `mumble` at the start of the trace. ☐

Lemma 33. $\mathbf{R}\boldsymbol{X}$ *is an* $\boldsymbol{\mathbb{S}}$*-model.*

Proof. This amounts to showing that $\mathbf{R}\boldsymbol{X}$ validates every $\boldsymbol{\mathbb{S}}$-axiom.

- The countable-join semilattice ones follow standardly for sets and unions.
- Commutativity follows from the fact that interpretations are all defined by direct images.
- The global state equations validate as they did in the model from Dvir et al. [12], where they were interpreted in a similar manner.

This leaves Empty:

$$\begin{aligned}[\![\lhd]\!]\;[\![\rhd]\!]\,K &= [\![\lhd]\!]\,\{\bullet\xi\Diamond x, \bullet\langle\sigma,\sigma\rangle\xi\Diamond x \mid \sigma \in \mathbb{S}, \circ\xi\Diamond x \in K\}\\ &= \left\{\circ\xi_0^?\xi\Diamond x \,\middle|\, \xi_0^? \in \mathbb{S}^{*}_{=}, \bullet\xi\Diamond x \in K\right\} = K\end{aligned}$$

where the last step is due to K being closed; and Connect:

$$\begin{aligned}[\![\rhd]\!]\;[\![\lhd]\!]\,K &= [\![\rhd]\!]\left\{\circ\xi_0^?\xi\Diamond x \,\middle|\, \xi_0^? \in \mathbb{S}^{*}_{=}, \bullet\xi\Diamond x \in K\right\}\\ &= \left\{\bullet\xi_0^?\xi\Diamond x, \bullet\langle\sigma,\sigma\rangle\xi_0^?\xi\Diamond x \,\middle|\, \xi_0^? \in \mathbb{S}^{*}_{=}, \bullet\xi\Diamond x \in K\right\} \supseteq K\end{aligned}$$

where the last step is by taking an empty $\xi_0^?$ in the first element. ☐

We mention some equations regarding open transitions provable in $\mathbb{S}$.

Lemma 34. $x : \bullet \vdash_{\mathbb{S}} \bigvee_{\sigma \in \mathbb{S}} ⦃\sigma, \sigma⦄\, x = x : \bullet$

Proof. Follows from the global state validity: $x : \star \vdash_{\mathsf{G}} \bigvee_{\sigma \in \mathbb{S}} ⦃\sigma, \sigma⦄\, x = x : \star$. ☐

Lemma 35. $x : \circ \vdash_{\mathbb{S}} \bigvee_{\sigma \in \mathbb{S}} \lhd ⦃\sigma, \sigma⦄ \rhd x = x : \circ$

Proof. Follows from ND-$\lhd$, lemma 34, and Empty. ☐

Let's turn to the extension of environments along return. Let $\mathbf{A}$ be an $\mathbb{S}$-algebra, and let $e : \boldsymbol{X} \to \underline{\mathbf{A}}$ be an $\boldsymbol{X}$-environment in $\mathbf{A}$. Then:

Lemma 36. *$e^{\#}$ is homomorphic.*

Proof. By evaluating both sides, it suffices to show that for every operator $(O : \square\langle\square_1, \dots, \square_\alpha\rangle) \in \Sigma_{\mathbb{S}}$, and all $K_i \in \underline{\mathbf{R}\boldsymbol{X}}_{\square_i}$:

$$\boldsymbol{X} \vdash_{\mathbb{S}} \mathrm{reify}(\mathbf{R}\boldsymbol{X}\, [\![O]\!]_{\mathrm{op}}\, (K_1, \dots, K_\alpha)) = O(\mathrm{reify}\, K_1, \dots, \mathrm{reify}\, K_\alpha) : \square$$

As in the proof of lemma 33, most follow as in Dvir et al.'s model [12], and we focus again on the interesting cases of $\lhd$ and $\rhd$. In both cases, we use proposition 31 to simplify. For the treatment of the $\rhd$ case below, we use lemma 34 in the third equation:

$$\begin{aligned}
\boldsymbol{X} \vdash_{\mathbb{S}} \mathrm{reify}(\mathbf{R}\boldsymbol{X}\, [\![\rhd]\!]_{\mathrm{op}}\, K) &= \mathrm{reify}\, \{\bullet\langle\sigma, \sigma\rangle\xi\Diamond x \mid \sigma \in \mathbb{S}, \circ\xi\Diamond x \in K\} \\
&= \bigvee\nolimits_{\sigma \in \mathbb{S}, \circ\xi\Diamond x \in K} ⦃\sigma, \sigma⦄ \rhd \underline{\circ\xi\Diamond x} \\
&= \bigvee\nolimits_{\circ\xi\Diamond x \in K} \rhd \underline{\circ\xi\Diamond x} \\
&= \rhd \bigvee\nolimits_{\circ\xi\Diamond x \in K} \underline{\circ\xi\Diamond x} = \rhd(\mathrm{reify}\, K) : \bullet \\
\boldsymbol{X} \vdash_{\mathbb{S}} \mathrm{reify}(\mathbf{R}\boldsymbol{X}\, [\![\lhd]\!]_{\mathrm{op}}\, K) &= \mathrm{reify}\, \{\circ\xi\Diamond x \mid \bullet\xi\Diamond x \in K\} \\
&= \bigvee\nolimits_{\bullet\xi\Diamond x \in K} \lhd \underline{\bullet\xi\Diamond x} \\
&= \lhd \bigvee\nolimits_{\bullet\xi\Diamond x \in K} \underline{\bullet\xi\Diamond x} = \lhd(\mathrm{reify}\, K) : \circ
\end{aligned}$$

☐

Lemma 37. $e = e^{\#} \circ \mathrm{return}$ *for all* $x \in \boldsymbol{X}$.

Proof. By evaluating in e the equations $x : \square \vdash_{\mathbb{S}} \mathrm{reify}_{\square}(\mathrm{return}_{\square}\, x) = x : \square$, which are easily verified in light of proposition 31, using lemmas 34 and 35. ☐

Lemma 38. $\mathrm{return}^{\#} : \mathbf{R}\boldsymbol{X} \to \mathbf{R}\boldsymbol{X}$ *is the identity.*

Proof sketch. Follows by calculation, mainly by showing that for any $K \in \underline{\mathbf{R}\boldsymbol{X}}_{\bullet}$, we have that $\mathbf{R}\, \{x : \bullet\}\, [\![⦃\sigma, \rho⦄\, x]\!]_{\mathrm{term}}\, (x \mapsto K) = (\sigma, \rho)\, K$. ☐

Finally, we show uniqueness. Let $f : \mathbf{R}\boldsymbol{X} \to \mathbf{A}$ be a homomorphism. Then:

Lemma 39. *If* $e = f \circ \mathrm{return}$ *then* $f = e^{\#}$.

Proof. We use the following notation. For any $\mathbb{S}$-algebra $\mathbf{B}$ and $\tilde{e} : \boldsymbol{X} \to \underline{\mathbf{B}}$, we denote $\mathrm{eval}(\tilde{e}) \coloneqq \mathbf{B}[\![-]\!]_{\mathrm{term}}\tilde{e} : \mathrm{Term}^{\Sigma_{\mathbb{S}}}\boldsymbol{X} \to \mathbf{B}$. Thus, $\tilde{e}^{\#} = \mathrm{eval}(\tilde{e}) \circ \mathrm{reify}$.

Since $\mathrm{eval}(f \circ \mathrm{return}) : \mathrm{Term}^{\Sigma_{\mathbb{S}}}\boldsymbol{X} \to \mathbf{A}$ is the only homomorphic extension of $f \circ \mathrm{return} : \boldsymbol{X} \to \mathbf{A}$ along the inclusion $\iota : \boldsymbol{X} \hookrightarrow \mathrm{Term}^{\Sigma_{\mathbb{S}}}\boldsymbol{X}$, we have that $\mathrm{eval}(f \circ \mathrm{return}) = f \circ \mathrm{eval}(\mathrm{return})$. Using lemma 38:

$$e^{\#} = \mathrm{eval}(e) \circ \mathrm{reify} = \mathrm{eval}(f \circ \mathrm{return}) \circ \mathrm{reify} = f \circ \mathrm{eval}(\mathrm{return}) \circ \mathrm{reify} = f \quad \square$$

Putting everything together, $\langle \mathbf{R}\boldsymbol{X}, \mathrm{return}\rangle$ is a $\mathbb{S}$-model over $\boldsymbol{X}$ (lemma 33) such that every environment homomorphically (lemma 36) extends along return (lemma 37), and does so uniquely (lemma 39). So $\langle \mathbf{R}\boldsymbol{X}, \mathrm{return}\rangle$ is a *free* $\mathbb{S}$-model over $\boldsymbol{X}$, proving the $\mathbb{S}$-Rep. Thm.