Two-sorted Algebraic Decompositions of Brookes's Shared-State Denotational Semantics

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

[1]Tel Aviv University

[2]University of Edinburgh

07 May 2025 | FoSSaCS | Hamilton, ON, Canada

Outline of the Talk

* Sequential setting — introduction to algebraic effects[P & Power 2002]
* Concurrent setting:
  - There's an algebraic effects theory for cooperative concurrency[P 2006]
  - Using it for preemptive concurrency lacks abstraction[DKL 2022]
* Highly abstract denotational model for preemptive concurrency[Brookes 1996, BHN 2016]
* Two-sorted algebraic effects for concurrency:
  - The ○-sort adjunction recovers preemptive concurrency — goal achieved!
  - The ●-sort adjunction recovers cooperative concurrency — nice perk

Section 1

The Sequential Setting

Small-Step Semantics

- Consider a sequential programming language
- Core Language: sequencing $(-;-)$, branching $(\textbf{ifz} - \textbf{then} - \textbf{else}-)$, etc.
- Effects: writing $(l := v)$ and reading $(l?)$ bits $\mathbb{B} = \{0, 1\}$ to storage locations $\mathbb{L}$

$$\sigma \in \mathbb{S} \triangleq \mathbb{L} \to \mathbb{B}$$

$$\begin{aligned}
&\sigma, (l := 0 \; ; \textbf{ifz } l? \textbf{ then } \text{“ok”} \textbf{ else } \text{“bug”}) \\
\to\; &\sigma[l \mapsto 0], (\textbf{ifz } l? \textbf{ then } \text{“ok”} \textbf{ else } \text{“bug”}) \\
\to\; &\sigma[l \mapsto 0], (\textbf{ifz } 0 \textbf{ then } \text{“ok”} \textbf{ else } \text{“bug”}) \\
\to\; &\sigma[l \mapsto 0], (\text{“ok”}) \quad \blacksquare
\end{aligned}$$

Moggi's Monad-based Compositional (Denotational) Semantics[1991]

Domain: state transformers $\underline{TX} \triangleq (\mathbb{S} \to \mathbb{S} \times X)$

$$[\![l := \mathsf{0} \; ; \; \mathbf{ifz} \; l? \; \mathbf{then} \; \text{“}\mathsf{ok}\text{”} \; \mathbf{else} \; \text{“}\mathsf{bug}\text{”}]\!]_{\text{prog}} = \lambda \sigma. \, \langle \sigma[l \mapsto \mathsf{0}], \text{“}\mathsf{ok}\text{”} \rangle \in \underline{T\text{String}}$$

Plotkin & Power's Algebraic Effects Semantics[2002]

Monadic semantics proves contextual equivalences (adequacy theorem):

$$[\![l := 0 \ ; \ \mathbf{ifz}\ l?\ \mathbf{then}\ \text{"ok"}\ \mathbf{else}\ \text{"bug"}]\!]_{\text{prog}} = \lambda\sigma.\, \langle \sigma[l \mapsto 0], \text{"ok"} \rangle = [\![l := 0 \ ; \ \text{"ok"}]\!]_{\text{prog}}$$

$$\qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \quad || \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad ||$$

Underlying algebraic reasoning: $[\![\mathsf{U}_{l,0}\, \mathsf{L}_l\, (\text{"ok"}, \text{"bug"})]\!]_{\text{term}} \overset{(\texttt{UL})}{=} [\![\mathsf{U}_{l,0}\ \text{"ok"}]\!]_{\text{term}}$

The algebraic effects theory of global state G has:

- Operators for updating $\mathsf{U}_{l,v} : 1$ and looking up $\mathsf{L}_l : 2$ bits in storage $(O : \text{arity})$
- Axioms such as $(\texttt{UL})\ \ \mathsf{U}_{l,v}\, \mathsf{L}_l(x_0, x_1) = \mathsf{U}_{l,v}\, x_v$

Plotkin & Power's Algebraic Effects Semantics[2002]

Monadic semantics proves contextual equivalences (adequacy theorem):

$$[\![l := 0 \,;\, \mathbf{ifz}\ l?\ \mathbf{then}\ \text{“ok”}\ \mathbf{else}\ \text{“bug”}]\!]_{\mathrm{prog}} = \lambda\sigma.\, \langle \sigma[l \mapsto 0], \text{“ok”}\rangle = [\![l := 0 \,;\, \text{“ok”}]\!]_{\mathrm{prog}}$$

$$\| \qquad\qquad\qquad\qquad \|$$

Underlying algebraic reasoning:

$$[\![\mathsf{U}_{l,0}\, \mathsf{L}_l\, (\text{“ok”}, \text{“bug”})]\!]_{\mathrm{term}} \overset{(\mathtt{UL})}{=} [\![\mathsf{U}_{l,0}\ \text{“ok”}\ ]\!]_{\mathrm{term}}$$

Interpret operat**ors** as operat**ions** over the domain:

$$[\![\mathsf{U}_{l,v}]\!]_{\mathrm{op}} f \triangleq \lambda\sigma \in \mathbb{S}.\, f\,(\sigma[l \mapsto v]) \qquad [\![\mathsf{L}_l]\!]_{\mathrm{op}}(f_0, f_1) \triangleq \lambda\sigma \in \mathbb{S}.\, f_{\sigma_l}\sigma$$

$$[\![\mathsf{U}_{l,v}\langle\rangle]\!]_{\mathrm{term}} = [\![l := v]\!]_{\mathrm{prog}} \qquad [\![\mathsf{L}_l(0, 1)]\!]_{\mathrm{term}} = [\![l?]\!]_{\mathrm{prog}}$$

$$[\![t]\!]_{\mathrm{term}} = [\![r]\!]_{\mathrm{term}} \quad \iff \quad t \overset{\mathsf{G}}{=} r$$

Adding Non-deterministic Choice

The theory of non-deterministic global state takes global state G and adds:

* Operators for choice: binary $\vee : 2$ and empty $\bot : 0$
* Axioms of semilattice, e.g.: (Symmetry) $x \vee y = y \vee x$ (Neutrality) $x \vee \bot = x$
* Axioms of interaction, e.g.: ($\vee$-U) $\mathsf{U}_{l,v}(x \vee y) = (\mathsf{U}_{l,v}\, x) \vee (\mathsf{U}_{l,v}\, y)$ ($\bot$-U) $\mathsf{U}_{l,v}\, \bot = \bot$

It is standard to:

* Generalize to larger cardinalities, e.g. countable choice
* Order by choices: $t \geq r \triangleq t \vee r = t$ (t includes every choice r does)

Section 2

The Concurrent Setting

Shared-State Concurrency Small-Step

Cooperative scheduling (program permits scheduler to switch a thread):

$$
\begin{aligned}
& \sigma, (l := 1 \;\|\; l := 0 \,;\, \textbf{yield} \,;\, \textbf{ifz}\ l?\ \textbf{then}\ \text{“ok”}\ \textbf{else}\ \text{“bug”}) \\
\to\ & \sigma, (l := 1 \;\|\rangle\; l := 0 \,;\, \textbf{yield} \,;\, \textbf{ifz}\ l?\ \textbf{then}\ \text{“ok”}\ \textbf{else}\ \text{“bug”}) \\
\to\ & \sigma[l \mapsto 0], (l := 1 \;\|\rangle\; \textbf{yield} \,;\, \textbf{ifz}\ l?\ \textbf{then}\ \text{“ok”}\ \textbf{else}\ \text{“bug”}) \\
\to\ & \sigma[l \mapsto 0], (l := 1 \;\|\; \textbf{ifz}\ l?\ \textbf{then}\ \text{“ok”}\ \textbf{else}\ \text{“bug”}) \\
\to\ & \ldots
\end{aligned}
$$

Cooperative Concurrency: Resumptions[Abadi & Plotkin 2010]

$$[\![l := 0 \,;\, \textbf{ifz}\ l?\ \textbf{then}\ \text{“ok”}\ \textbf{else}\ (\textbf{yield} \,;\, \text{“bug”})]\!]_{\text{prog}} = [\![\mathsf{U}_{l,0}\,\mathsf{L}_l\,(\text{“ok”}, \mathsf{Y}\,\text{“bug”})]\!]_{\text{term}}$$

$$\stackrel{(\texttt{UL})}{=} [\![\mathsf{U}_{l,0}\,\text{“ok”}]\!]_{\text{term}} = [\![l := 0 \,;\, \text{“ok”}]\!]_{\text{prog}}$$

The theory of resumptions Res takes non-deterministic global state and adds:

- Operator for yielding to the concurrent environment $\mathsf{Y} : 1$
- Axioms of closure: (Pure) $\mathsf{Y}\,x \geq x$ (Join) $\mathsf{Y}\,\mathsf{Y}\,x = \mathsf{Y}\,x$
- Axioms of interaction: ($\vee$-$\mathtt{Y}$) $\mathsf{Y}(x \vee y) = (\mathsf{Y}\,x) \vee (\mathsf{Y}\,y)$ ($\bot$-$\mathtt{Y}$) $\mathsf{Y}\,\bot = \bot$

$$[\![l := 0 \,;\, \textbf{yield} \,;\, \textbf{ifz}\ l?\ \textbf{then}\ \text{“ok”}\ \textbf{else}\ \text{“bug”}]\!]_{\text{prog}} = [\![\mathsf{U}_{l,0}\,\mathsf{Y}\,\mathsf{L}_l\,(\text{“ok”}, \text{“bug”})]\!]_{\text{term}}$$

$$\stackrel{(\text{Pure})}{\geq} [\![\mathsf{U}_{l,0}\,\mathsf{L}_l\,(\text{“ok”}, \text{“bug”})]\!]_{\text{term}} \stackrel{(\texttt{UL})}{=} [\![\mathsf{U}_{l,0}\,\text{“ok”}]\!]_{\text{term}} = [\![l := 0 \,;\, \text{“ok”}]\!]_{\text{prog}}$$

Preemptive Concurrency Small-Step

Preemptive scheduling (non-deterministic interleaving):

$$
\begin{aligned}
& \sigma, (l := 1 \;\|\; l := 0 \,;\, \textbf{ifz } l? \textbf{ then } \text{“ok”} \textbf{ else } \text{“bug”}) \\
\to\; & \sigma[l \mapsto 0], (l := 1 \;\|\; \textbf{ifz } l? \textbf{ then } \text{“ok”} \textbf{ else } \text{“bug”}) \\
\to\; & \sigma[l \mapsto 1], (\langle\rangle \;\|\; \textbf{ifz } l? \textbf{ then } \text{“ok”} \textbf{ else } \text{“bug”}) \\
\to\; & \ldots
\end{aligned}
$$

Preemptive Concurrency: Also Resumptions?[P 2006; DKL 2022]

Use resumptions for preemptive concurrency by yielding implicitly?

(i.e. using operator Y without **yield** construct)

Fundamental issue (no-go theorem): does $[\![l?]\!]_{\mathrm{prog}}$ implicitly yield?

* If so, e.g. $[\![l?]\!]_{\mathrm{prog}} = [\![\mathsf{Y}\,\mathsf{L}_l(\mathsf{Y}\,0, \mathsf{Y}\,1)]\!]_{\mathrm{term}}$ — abstraction issue:

$$[\![\mathbf{ifz}\ l?\ \mathbf{then}\ \text{“ok”}\ \mathbf{else}\ \text{“ok”}]\!]_{\mathrm{prog}} \neq [\![\text{“ok”}]\!]_{\mathrm{prog}}$$

* If not, e.g. $[\![l?]\!]_{\mathrm{prog}} = [\![\mathsf{L}_l(0, 1)]\!]_{\mathrm{term}}$ — soundness issue:

$$[\![\mathbf{ifz}\ l?\ \mathbf{then}\ l?\ \mathbf{else}\ 0]\!]_{\mathrm{prog}} = [\![0]\!]_{\mathrm{prog}}$$

Historical Precedent: Reverse Engineering

Monads came first (1991) — Algebraic effects recovered them (2002)

> *the process is a kind of reverse engineering*
>
> — *Hyland & Power [2007]*

We target the Brookes monad based on sequences of atomic state transitions

- Highly Abstract: e.g. has $[\![\textbf{ifz}\ l?\ \textbf{then}\ \text{“ok”}\ \textbf{else}\ \text{“ok”}]\!]_{\text{prog}} = [\![\text{“ok”}]\!]_{\text{prog}}$
- Extensible: e.g. infinite executions, type-and-effect systems, allocations, relaxed memory

Historical Precedent: Reverse Engineering

Monads came first (1991) — Algebraic effects recovered them (2002)

Setting	Sequential	Cooperative Concurrency	Preemptive Concurrency
Monad	State Transformers	...	Brookes Monad
Alg. Theory	Global State	Resumptions	??

We target the Brookes monad based on sequences of atomic state transitions

* Highly Abstract: e.g. has $[\![\textbf{ifz}\ l?\ \textbf{then}\ \text{“ok”}\ \textbf{else}\ \text{“ok”}]\!]_{\text{prog}} = [\![\text{“ok”}]\!]_{\text{prog}}$
* Extensible: e.g. infinite executions, type-and-effect systems, allocations, relaxed memory

Section 3

Brookes Monad

Brookes's Trace-Based Denotational Model[1996]

- Denotations $[\![M]\!]_{\mathrm{prog}}$ are sets of traces
- Trace — a protocol that the pool of threads in M can adhere to

Example (Rely/Guarantee Intuition for Traces)

(1) relies on access — (2) to guarantee access

$$\left\langle {1 \atop 1}, {1 \atop 0} \right\rangle \left\langle {1 \atop 1}, {0 \atop 0} \right\rangle x \in \mathsf{T}X \qquad \text{where } x \in X$$

then (3) relies on access — (4) to guarantee access and return value

add reliance

$$\left\langle {1 \atop 1}, {1 \atop 0} \right\rangle \left\langle {1 \atop 1}, {0 \atop 0} \right\rangle x \in [\![M]\!]_{\mathrm{prog}} \overset{\text{stutter}}{\Longrightarrow} \left\langle {1 \atop 1}, {1 \atop 0} \right\rangle \left\langle {1 \atop 1}, {0 \atop 0} \right\rangle \left\langle {0 \atop 1}, {0 \atop 1} \right\rangle x \in [\![M]\!]_{\mathrm{prog}}$$

$$\left\langle {1 \atop 1}, {1 \atop 0} \right\rangle \left\langle {1 \atop 0}, {0 \atop 0} \right\rangle x \in [\![M]\!]_{\mathrm{prog}} \overset{\text{mumble}}{\Longrightarrow} \left\langle {1 \atop 1}, {0 \atop 0} \right\rangle x \in [\![M]\!]_{\mathrm{prog}}$$

remove guarantee

Reasoning in the Brookes Monad

* Brookes's model has a monadic presentation B
  * Domain: closed sets of traces $\underline{BX} \triangleq \mathcal{P}^{\dagger}(\mathsf{T}X)$
* Supports program refinements, e.g.:

$$
\begin{aligned}
&[\![l := 0 \;;\; \textbf{ifz}\; l?\; \textbf{then}\; \text{“ok”}\; \textbf{else}\; \text{“bug”}]\!]_{\text{prog}} \\
&\qquad = \{\langle\sigma, \sigma[l \mapsto 0]\rangle\langle\rho, \rho\rangle\text{“ok”} \mid \sigma, \rho \in \mathbb{S}, \rho_l = 0\}^{\dagger} \cup \{\cdots \text{“bug”} \mid \cdots \rho_l = 1\}^{\dagger} \\
(\rho = \sigma[l \mapsto 0]) &\qquad \supseteq \{\langle\sigma, \sigma[l \mapsto 0]\rangle\langle\sigma[l \mapsto 0], \sigma[l \mapsto 0]\rangle\text{“ok”} \mid \sigma \in \mathbb{S}\}^{\dagger} \\
(\dagger) &\qquad = \{\langle\sigma, \sigma[l \mapsto 0]\rangle\text{“ok”} \mid \sigma \in \mathbb{S}\}^{\dagger} \\
&\qquad = [\![l := 0 \;;\; \text{“ok”}]\!]_{\text{prog}}
\end{aligned}
$$

* And equivalences, e.g.: $[\![\textbf{ifz}\; l?\; \textbf{then}\; \text{“ok”}\; \textbf{else}\; \text{“ok”}]\!]_{\text{prog}} = [\![\text{“ok”}]\!]_{\text{prog}}$

Reverse Engineering the Brookes Monad

Setting	Sequential	Cooperative Concurrency	Preemptive Concurrency
$[\![l := 0]\!]_{\mathrm{prog}}$	$\lambda\sigma.\langle\sigma[l \mapsto 0], \langle\rangle\rangle$	...	$\{\langle\sigma, \sigma[l \mapsto 0]\rangle \langle\rangle \mid \sigma \in \mathbb{S}\}^{\dagger}$
Alg. Rep.	$[\![\mathsf{U}_{l,0}\langle\rangle]\!]_{\mathrm{term}}$	$[\![\mathsf{U}_{l,0}\langle\rangle]\!]_{\mathrm{term}}$	??

Reverse Engineering the Brookes Monad

Setting	Sequential	Cooperative Concurrency	Preemptive Concurrency
$[\![l := 0]\!]_{\text{prog}}$	$\lambda\sigma.\,\langle \sigma[l \mapsto 0], \langle\rangle\rangle$	...	$\{\langle\sigma, \sigma[l \mapsto 0]\rangle\,\langle\rangle \mid \sigma \in \mathbb{S}\}^{\dagger}$
Alg. Rep.	$[\![\mathsf{U}_{l,0}\langle\rangle]\!]_{\text{term}}$	$[\![\mathsf{U}_{l,0}\langle\rangle]\!]_{\text{term}}$	$[\![\triangleleft\, \mathsf{U}_{l,0} \triangleright\langle\rangle]\!]_{\text{term}}$

Section 4

Our Two-sorted Shared State Theory

Our Two-sorted Theory of Shared State $\mathbb{S}$

- Sorts: Hold ($\bullet$) & Cede ($\circ$)
- Operators:
  - $\bullet$-sorted update $\mathsf{U}_{l,v} : \bullet\langle\bullet\rangle$ and lookup $\mathsf{L}_l : \bullet\langle\bullet,\bullet\rangle$
  - choice in each sort
  - acquire $\lhd : \circ\langle\bullet\rangle$ release $\rhd : \bullet\langle\circ\rangle$
- Axioms:
  - $\bullet$-copy of the global state axioms
  - Standard choice axioms (including distributivity and strictness)
  - Closure pair axioms: (Empty) $\lhd \rhd x = x$ (Fuse) $\rhd \lhd x \geq x$
- Represented by a two-sorted generalization $B^{\{\bullet,\circ\}}$ of the Brookes monad B
  - $[\![\lhd \mathsf{U}_{l,v} \rhd\langle\rangle]\!]_{\text{term}} \cong [\![l := v]\!]_{\text{prog}}$ $[\![\lhd \mathsf{L}_l(\rhd 0, \rhd 1)]\!]_{\text{term}} \cong [\![l?]\!]_{\text{prog}}$

Reasoning with Shared State $\mathbb{S}$

$$[\![l := 0 \,;\, \textbf{ifz } l? \textbf{ then } \text{“ok” } \textbf{else } \text{“bug”}]\!]_{\text{prog}} \cong [\![\triangleleft\, \mathsf{U}_{l,0} \triangleright \triangleleft\, \mathsf{L}_l\, (\triangleright\text{“ok”}, \triangleright\text{“bug”})]\!]_{\text{term}}$$

$$\overset{(\text{Fuse})}{\supseteq} [\![\triangleleft\, \mathsf{U}_{l,0}\, \mathsf{L}_l\, (\triangleright\text{“ok”}, \triangleright\text{“bug”})]\!]_{\text{term}}$$

$$\overset{(\mathtt{UL})}{=} [\![\triangleleft\, \mathsf{U}_{l,0} \triangleright\text{“ok”}]\!]_{\text{term}} \cong [\![l := 0 \,;\, \text{“ok”}]\!]_{\text{prog}}$$

(Fuse) $(\triangleright \triangleleft\, x \geq x)$: fusing atomic blocks eliminates potential interference

$$[\![\textbf{ifz } l? \textbf{ then } \text{“ok” } \textbf{else } \text{“ok”}]\!]_{\text{prog}} \cong [\![\triangleleft\, \mathsf{L}_l\, (\triangleright\text{“ok”}, \triangleright\text{“ok”})]\!]_{\text{term}}$$

$$\overset{\mathsf{G}}{=} [\![\triangleleft \triangleright \text{“ok”}]\!]_{\text{term}} \overset{(\text{Empty})}{=} [\![\text{“ok”}]\!]_{\text{term}} \cong [\![\text{“ok”}]\!]_{\text{prog}}$$

(Empty) $(\triangleleft \triangleright x = x)$: empty atomic blocks have no observable effect

Section 5

Our Two-sorted Brookes Monad

Two-sorted Brookes Traces

- Each family $\boldsymbol{X} \in \mathbf{Set}^{\{\bullet,\circ\}}$ has a $\bullet$-component $\boldsymbol{X}_\bullet \in \mathbf{Set}$ and a $\circ$-component $\boldsymbol{X}_\circ \in \mathbf{Set}$
- Generalize to sorted traces: $\boxdot\langle \sigma_1, \rho_1 \rangle \ldots \langle \sigma_i, \rho_i \rangle \ldots \langle \sigma_n, \rho_n \rangle \diamondsuit x$ is $\boxdot$-sorted and $\diamondsuit$-valued

Example (Rely/Guarantee Intuition for Sorted Traces)

relies on predecessor holding

guarantees releasing

$\bullet$-sorted and $\circ$-valued: $\bullet \langle {1 \atop 1}, {1 \atop 0} \rangle \langle {1 \atop 1}, {0 \atop 0} \rangle \circ x \in (\mathbb{T}\boldsymbol{X})_\bullet$ where $x \in \boldsymbol{X}_\circ$

- Same closure rules $(\xrightarrow{\text{stutter}})$ & $(\xrightarrow{\text{mumble}})$ except: no stutter next to $\bullet$

Example (Disallowed stutter)

$$\bullet \langle {1 \atop 1}, {1 \atop 0} \rangle \langle {1 \atop 1}, {0 \atop 0} \rangle \circ x \not\xrightarrow{\text{stutter}} \bullet \langle {0 \atop 1}, {0 \atop 1} \rangle \langle {1 \atop 1}, {1 \atop 0} \rangle \langle {1 \atop 1}, {0 \atop 0} \rangle \circ x$$

Representation of our Theory of Shared State $\mathbb{S}$

- Monad $B^{\{\bullet,\circ\}}$ represents the shared-state theory $\mathbb{S}$:
  - Domain for each sort $\square$: closed sets of $\square$-sorted traces $\underline{B^{\{\bullet,\circ\}}\boldsymbol{X}}_{\square} \triangleq \mathcal{P}^{\dagger}((\mathbb{T}\boldsymbol{X})_{\square})$
  - Interpretations:

$$[\![\bigvee]\!]_{\mathrm{op}} \triangleq \bigcup \qquad \text{(Choice)}$$

$$[\![\triangleleft]\!]_{\mathrm{op}} K \triangleq \{\circ\xi \lozenge x \mid \bullet\xi \lozenge x \in K\}^{\dagger} \qquad \text{(Acquire)}$$

$$[\![\triangleright]\!]_{\mathrm{op}} K \triangleq \{\bullet\langle\sigma,\sigma\rangle\xi \lozenge x \mid \sigma \in \mathbb{S}, \circ\xi \lozenge x \in K\}^{\dagger} \qquad \text{(Release)}$$

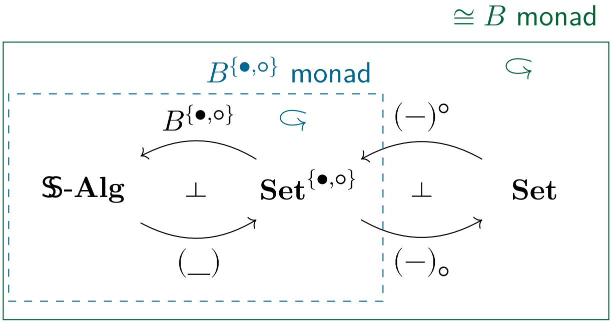
$$[\![\mathsf{U}_{l,v}]\!]_{\mathrm{op}} K \triangleq \bigcup_{\sigma\in\mathbb{S}} (\sigma, \sigma[l \mapsto v])\, K \qquad \text{(Update)}$$

$$[\![\mathsf{L}_{l}]\!]_{\mathrm{op}} (K_0, K_1) \triangleq \bigcup_{\sigma\in\mathbb{S}} (\sigma, \sigma)\, K_{\sigma_l} \qquad \text{(Lookup)}$$

where $(\sigma, \rho)\, K \triangleq \{\bullet\langle\sigma,\theta\rangle\xi \lozenge x \mid \bullet\langle\rho,\theta\rangle\xi \lozenge x \in K\}$

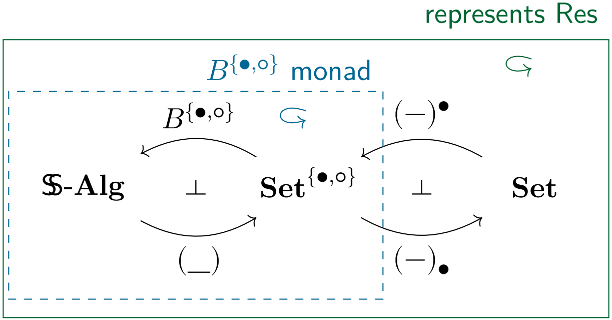
Recovery Along the Inclusion-Projection Adjunction

- Monad $B^{\{\bullet,\circ\}}$ transformed along $(-)^{\circ} \dashv (-)_{\circ}$ $\cong$ Brookes's monad B
  - » $X^{\circ} \triangleq \{x : \circ \mid x \in X\}$ — $\underline{B^{\{\bullet,\circ\}} X^{\circ}}_{\circ} = \mathcal{P}^{\dagger}((\mathbb{T}X^{\circ})_{\circ}) \cong \mathcal{P}^{\dagger}(\mathsf{T}X) = \underline{BX}$
  - » $[\![\lhd\, \mathsf{U}_{l,v} \rhd \langle\rangle]\!]_{\mathrm{term}} \cong [\![l := v]\!]_{\mathrm{prog}}$ $[\![\lhd\, \mathsf{L}_{l}(\rhd 0, \rhd 1)]\!]_{\mathrm{term}} \cong [\![l?]\!]_{\mathrm{prog}}$



Recovery Along the Inclusion-Projection Adjunction

- Monad $B^{\{\bullet,\circ\}}$ transformed along $(-)^{\bullet} \dashv (-)_{\bullet}$ represents the resumptions theory Res
  - Closure axioms ($\mathsf{Y} \mapsto \triangleright \triangleleft$): (Pure) $\triangleright \triangleleft x \geq x$ (Join) $\triangleright \triangleleft \triangleright \triangleleft x = \triangleright \triangleleft x$
  - $[\![\triangleright \triangleleft \langle\rangle]\!]_{\text{term}} \cong [\![\mathbf{yield}]\!]_{\text{prog}}$ $[\![\mathsf{U}_{l,v}\langle\rangle]\!]_{\text{term}} \cong [\![l := v]\!]_{\text{prog}}$ $[\![\mathsf{L}_l(0,1)]\!]_{\text{term}} \cong [\![l?]\!]_{\text{prog}}$



High-Level Outline of the Solution

A two-sorted algebraic effects theory for **shared state concurrency** $\mathbb{S}$:
(the first example of a multi-sorted algebraic effects theory)

- The sorts Hold $\bullet$ & Cede $\circ$ declare exclusive access to memory
- Classic algebraic effects theories:
  - » Global State G in $\bullet$
  - » Choice (semilattice) in both sorts
- The Closure Pair theory C for managing access:
  - » Operators: Acquire $\lhd : \circ\langle\bullet\rangle$ & Release $\rhd : \bullet\langle\circ\rangle$
  - » Closure pair axioms: (Empty) $\lhd \rhd x = x$ (Fuse) $\rhd \lhd x \geq x$
- Represented by a two-sorted model recovering known models in each sort:
  - » The $\circ$-incl $\dashv$ $\circ$-proj adjunction **recovers Brookes's model (preemptive concurrency)**
  - » The $\bullet$-incl $\dashv$ $\bullet$-proj adjunction **represents Resumptions (cooperative concurrency)**

Thank you!