Memoryful Geometry of Interaction
From Coalgebraic Components to Algebraic Effects

Naohiko Hoshino
RIMS, Kyoto University

Koko Muroya Ichiro Hasuo
Dept. Computer Science, University of Tokyo

Abstract—Girard’s Geometry of Interaction (GoI) is interaction-based semantics of linear logic proofs and, via suitable translations, of functional programs in general. Its mathematical cleanness identifies essential structures in computation; moreover its use as a compilation technique from programs to state machines—“GoI implementation,” so to speak—has been worked out by Mackie, Ghica and others. In this paper, we develop Abramsky’s idea of resumption based GoI systematically into a generic framework that accounts for computational effects (nondeterminism, probability, exception, global states, interactive I/O, etc.). The framework is categorical: Plotkin & Power’s algebraic operations provide an interface to computational effects; the framework is built on the categorical axiomatization of GoI by Abramsky, Haghverdi and Scott; and, by use of the coalgebraic formalization of component calculus, we describe explicit construction of state machines as interpretations of functional programs. The resulting interpretation is shown to be sound with respect to equations between algebraic operations, as well as to Moggi’s equations for the computational lambda calculus. We illustrate the construction by concrete examples.

I. INTRODUCTION

Geometry of Interaction (GoI) is introduced by Girard [1] as semantics of proofs—i.e. programs, under the Curry-Howard correspondence—for the study of dynamics and invariants of the cut elimination process (i.e. program execution). Girard’s original presentation of GoI is in the language of C*-algebras; Mackie’s alternative presentation [2] as token machines initiated another important application of GoI, namely as a compilation technique. There GoI provides translation of programs into state machines; and the machines’ execution results are invariant under cut elimination. Dynamics in such machines can be understood as a mathematical counterpart of control flow in program execution, and in this way, GoI connects mathematics (denotational semantics), program evaluation (operational semantics) and state based computation (low-level languages/implementations). Applications of GoI are widespread: implementation of (imperative) functional programming languages [2], [3]; relationship to Krivine abstract machines [4] and to defunctionalization [5]; optimal graph reduction for the lambda calculus [6]; and the design of a functional programming language for sublinear space [7].

Categorical GoI: This remarkable level of integration in GoI—of operational and denotational/structural semantics—is further exemplified by its categorical axiomatics (categorical GoI) developed by Abramsky, Haghverdi and Scott [8]. There a general construction is given from a traced monoidal category—together with additional constructs, altogether called a GoI situation—to a combinatorial algebra. One can then apply the realizability construction (see e.g. [9]) that turns a combinatorial algebra (an “untyped” model) into a categorical model of a typed calculus, from which one extracts realizers as concrete interpretations. The latter are sound by construction.

In a big picture, the current work is one of the attempts to instantiate this general methodology of categorical GoI to concrete situations. Our starting point is the previous work [10] where we extend the above workflow by a step prior to it. The extension comes from the following observation (a folklore result, see Lemma IV.3; see also [11]): many traced monoidal categories arise as a Kleisli category of a monad with a suitable order structure. The resulting extended workflow is as follows.

\[
\begin{align*}
\text{(a Set-monad } T \text{ whose Kleisli category is Cppo-enriched)} & \\
\text{— } & \\
\text{(a traced monoidal category)} & \\
\text{— } & \\
\text{(a combinatorial algebra)} & \\
\text{— } & \\
\text{(a GoI interpretation of a typed calculus)} & \end{align*}
\]

In [10] we pursued use of this extended general workflow that is parameterized by \( T \): in order to interpret a calculus with a certain additional feature, we start with a monad \( T \) equipped with the same feature, and the generic constructions would yield a suitable GoI interpretation. In [10], specifically, we considered a calculus for quantum computation.

Effects and Resumption Based GoI: However, following this naive scenario turned out to be far from straightforward: in [10] we ended up using a complicated continuation monad that keeps track of all the the measurement outcomes. In fact the same kind of difficulty is already with nondeterminism—one of the most basic computational effects—as we exhibit now. Here we shall speak on the intuitive level, using the game-theoretic terms of queries and answers instead of the categorical language for GoI.

Consider the call-by-value evaluation of the program

\[
(\lambda x : \text{nat}. \ x + x) (0 \sqcup 1)
\]

where the subterm \( 0 \sqcup 1 \) returns 0 or 1 nondeterministically. Then the whole program is expected to return 0 or 2. However, the usual GoI interpretation of (2) may return an unexpected value 1, as the result of the following interaction.

1) We ask the value of the left occurrence of \( x \) in \( x + x \).
2) The subterm \( 0 \sqcup 1 \) answers 0 or 1 nondeterministically.
3) We ask the value of the right occurrence of $x$ in $x + x$.
4) The subterm $0 \sqcup 1$ answers $0$ or $1$ nondeterministically.
5) We add the values of the left $x$ and the right $x$.

The difficulty is as follows. After the first query 2), the nondeterminism in the subterm $0 \sqcup 1$ is resolved, with the subterm reduced to $0$ or $1$. The second query 4) must stick to this choice; however, most instances of GoI—presented in terms of $C^*$-algebra, token machines or categories—lack such an explicit “memory” mechanism. The lack of memories in GoI causes similar difficulties with other computational effects. Even more, it seems to be also the reason why additive connectives are far less trivial to interpret in GoI: additive slices—a common tool for additive connectives in GoI [12], [13]—indeed have a strong flavor of memories. Here one may wonder if the call-by-value evaluation strategy is to blame in the previous example. This is unlikely; see Remark I.1.

The memory mechanism needed in the above example can be provided in the form of a Mealy machine, or a (nondeterministic) transducer. The term $0 \sqcup 1$ is now interpreted as

$$\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
q_0 \xrightarrow{0} x_0 \xrightarrow{0} \quad q_0' \xrightarrow{1} x_1 \xrightarrow{1} q_1
\end{array}
\end{array}
\end{array}$$

where the machine can initially respond to a query $q$ with 0 or 1; however, after that the machine sticks to the same choice by remembering the choice by means of its internal state.

The idea of using such memories (or states) in GoI is not new. In [14], [8], an instance of categorical GoI is given by the category of resumptions—roughly speaking, a resumption from a set $A$ to $B$ is a “stateful computation” from $A$ to $B$; more precisely it is a suitable equivalence class (e.g. modulo bisimilarity) of transducers from $A$ to $B$. In [8] it is also characterized as an element of a final coalgebra.

**Remark I.1.** Lack of memories in effectful situations causes difficulties in GoI regardless of evaluation strategies. Consider the equation $(t \sqcup s) u = t u \sqcup s u$; we expect it to hold regardless of evaluation strategies. In the GoI interpretation of $(t \sqcup s) u$, when $u$ receives a query from the $t$ part in $t \sqcup s$, we must make sure that the response goes back to $t$, not to $s$. It is not clear how to enforce this without using memories, as one would see trying to interpret the program in (any presentation of) GoI.

**Contributions—from Coalgebraic Components to Algebraic Effects:** Building on Abramsky’s idea of resumption based GoI, the current paper aims at a generic framework that yields GoI interpretations—or rather GoI implementations, since they are given concretely as state based transducers like in (3)—of calculi with various computational effects.

More specifically: we model an effect by a monad $T$ following Moggi [15]; and as a syntactic interface we use algebraic operations like $\sqcup$ for the powerset monad $\mathcal{P}$, following Plotkin & Power [16]. Concrete interpretations are given by (state based) transducers with the same effect $T$, much like the nondeterministic one in (3). Assuming that $T$ comes with a suitable $\text{Copro}$ structure—many effects qualify by the slight modification of adding partiality—we show that the category $\text{Res}(T)$ of $T$-resumptions is traced monoidal.

Then the general workflow in (1), starting from its second step, yields a GoI interpretation of a calculus, with transducers as realizers. The resulting interpretation is sound with respect to the algebraic axioms (e.g. associativity of $\sqcup$) as well as Moggi’s equations for the computational lambda calculus.

This overall procedure—from a monad $T$ and algebraic operations to a GoI interpretation in the form of $T$-transducers—we wish to call memoryful GoI, emphasizing the role of memories (“memories” here are the same as “internal states”; the choice is to distinguish from the global “state” monad). A novelty is the use of memories (in the form of internal states of transducers) that allows us to describe interpretations of algebraic effect operations in a generic yet concrete manner.

We describe the construction of transducers concretely in terms of coalgebraic component calculus. Component calculi are heavily studied in software engineering (see e.g. [17]) as means of composing software components. A major concern there is compositionality, much like in the study of process calculi (see e.g. [18]); and successful (co)algebraic techniques have been developed for the latter [19], [20] as well as for the former [21]. We rely on the categorical formalization of component calculi in [21], [22] where components are coalgebras; categorical genericity is needed since our transducers are parametrized by a monad $T$. In this way, the current work pursues “some convergence and unification”—suggested in the paper [14], from which we draw inspirations—of GoI and coalgebra, in program semantics.

**Organization of This Paper:** In Section II we recall categorical GoI, and in Section III we summarize notations used in this paper. In Section IV we introduce component calculus on transducers; in Section V we quotient components by behavioral equivalence and define a traced symmetric monoidal category of sets and resumptions. In Section VI, we sketch construction of categorical models of the computational lambda calculus based on categorical GoI with transducers; the resulting GoI interpretation is described in Section VII with concrete examples.

**II. CATEGORICAL GEOMETRY OF INTERACTION**

We recall a categorical formulation of GoI called GoI situation introduced by Abramsky, Haghverdi and Scott and pin down the obstacle explained in the introduction in mathematical terms. A crucial notion in their categorical axiomatics is that of traced symmetric monoidal category.

**Definition II.1.** A traced symmetric monoidal category is a symmetric monoidal category $(\mathcal{C}, \otimes, I)$ with a family of maps $\text{tr}_{A,B}^C : \mathcal{C}(A \otimes C, B \otimes C) \to \mathcal{C}(A, B)$ subject to certain conditions (see [23], [24]).

**Definition II.2.** Let $\mathcal{C}$ be a traced symmetric monoidal category. A traced symmetric monoidal functor $F : \mathcal{C} \to \mathcal{C}$ is a strong symmetric monoidal functor such that

$$\text{tr}_{F(A,B)}^C(m_{B,C}^{-1} \circ F \circ m_{A,C}) = F(\text{tr}_{A,B}^C(f))$$
for all $C$-morphisms $f : A \otimes C \to B \otimes C$ where $m_{A,B} : FA \otimes FB \to F(A \otimes B)$ is the coherence isomorphism of $F$.

For example, the category $(\text{Rel}, +, \emptyset)$ of sets and relations forms a traced symmetric monoidal category: for a $\text{Rel}$-morphism $f : A + C \to B + C$, we define the trace operator $\text{tr}_{A,B}^C(f) : A \to C$ by the execution formula

$$\text{tr}_{A,B}^C(f) = f_{AB} \cup \bigcup_{n \geq 0} f_{CB} \circ f_{AC}$$

where $f_{XY} : X \to Y$ is the restriction of $f$ to a relation between $X$ and $Y$.

**Definition II.3.** A GoI situation is a list $(C, U, F, \phi, \psi, u, v)$ consisting of a traced symmetric monoidal category $(C, \otimes, I)$, a $C$-object $U$ and a traced symmetric monoidal functor $F : C \to \mathcal{C}$ with retractions $\phi : U \otimes U \to U$ and $u : F \cup U \to U$ together with the following retractions

$$n : I \otimes U : n' e_A : A \otimes FA : e'_A d_A : FFA \otimes FA : d'_A$$

$$c_A : FA \otimes FA \otimes FA : c'_A w_A : I \otimes FA : w'_A$$

such that $e_A, d_A, c_A$ and $w_A$ are natural in $A$.

The retraction $\phi$ and $(n, n')$ provides GoI interpretation of the multiplicative fragment of linear logic, and the traced symmetric monoidal functor $F$ with the remaining retractions provide GoI interpretation of the exponential fragment of linear logic. In [8], a GoI situation is shown to yield a linear combinatory algebra; via the Girard translation, we obtain an SK-algebra that is a model of intensional logic.

**Proposition II.4.** [8] Let $(C, U, F, \phi, \psi, u, v)$ be a GoI situation. The set $C(U, U)$ with the binary application $a \circ b$ on $C(U, U)$ given by

$$a \circ b = \text{tr}_{U,U}^U((U \otimes (u \circ Fb \circ v)) \circ \psi \circ a \circ \phi)$$

forms an SK-algebra: there exist $S, K \in C(U, U)$ such that

$$S \circ a \circ b \circ c = a \circ c \circ (b \circ c), \quad K \circ a \circ b = a$$

where we assume that the binary application is left associative.

On the categorical level, the obstacle in the introduction stems from the fact that the trace operator $\text{tr}$ on $\text{Rel}$ does not preserve the union of relations:

$$\text{tr}_{A,B}^C(f \cup g) = \text{tr}_{A,B}^C(f) \cup \text{tr}_{A,B}^C(g) \cup f_{AC} \cup \cdot \cdot \cdot$$

Failure of preservation of the trace results in failure of the equation:

$$(a \cup b) \circ c \neq (a \circ c) \cup (b \circ c)$$

in the SK-algebra $(\text{Rel}(U, U), \bullet)$ constructed from a GoI situation $(\text{Rel}, U, F, \phi, \psi, u, v)$. The unexpected value returned by the program (2) appears in the extra summands in (4).

**Remark II.5.** In the original definition of GoI situation in [8], the retractions $e_A, d_A, c_A, w_A$ and their left inverses are required to be monoidal natural transformations. In this paper, we only require the injection side of retractions to be natural since this is enough to prove Proposition II.4. This relaxation is needed in our concrete examples.

III. NOTATIONS

We summarize several notations used in this paper. Let $\text{Set}$ be the category of sets and maps (i.e. functions). We write

$$A \xrightarrow{\text{inl}_{A,B}} A + B \xleftarrow{\text{inr}_{A,B}} B, \quad A + A \xrightarrow{\gamma_A} A$$

for the injections and the codiagonal map. We write

$$A \times B \xrightarrow{\sigma_{A,B}} B \times A, \quad A \times B + A \times C \xrightarrow{\delta_{A,B,C}} A \times (B + C)$$

for the canonical bijections. We write $T_A : A \to 1$ and $\perp_A : \emptyset \to A$ for the unique maps. For sets $I$ and $A$, we write $A^I$ for the $I$-fold product of $A$.

Let $(T, \eta, \mu)$ be a monad on $\text{Set}$. In order to distinguish maps from morphisms in the Kleisli category $\text{Set}_T$, we write $f : X \to_T Y$ when $f$ is a $\text{Set}_T$-morphism from $X$ to $Y$. Since $T$ is a monad on $\text{Set}$, we have tensorial strengths

$$TA \times B \xrightarrow{st_{A,B}} T(A \times B), \quad A \times TB \xrightarrow{st_{A,B}} T(A \times B).$$

For $\text{Set}_T$-morphisms $f : A \to_T B$ and $g : B \to_T C$, a map $h : A \to B$ and a set $D$, we define $\text{Set}_T$-morphisms $g \circ_T f$ by

$$g \circ_T f = \mu_C \circ Tg \circ f : A \to_T C,$$

$$f \circ_D = st_{D,D} \circ (f \times D) : A \times D \to_T B \times D,$$

$$D \circ f = st'_{D,D} \circ (D \times f) : D \times A \to_T D \times B,$$

$$h^* = \eta_B \circ h : A \to_T B.$$}

The first construction is the composition of Kleisli morphisms. The second and the third constructions are the premonoidal products in $\text{Set}_T$. See [25] for premonoidal category, although further familiarity will not be needed. For $\text{Set}_T$-morphisms $f : A \to_T B$ and $g : C \to_T D$ such that

$$(f \circ D) \circ_T (A \circ g) = (B \circ g) \circ_T (f \circ C),$$

we write $f \circ g$ for $(f \circ D) \circ_T (A \circ g)$; this happens when $T$ is a commutative monad. The last construction $(-)^*$ is the Kleisli inclusion from $\text{Set}$ to $\text{Set}_T$, which lifts the (finite) coproducts $\emptyset, +, \text{inl}^*, \text{inr}^*$ of $\text{Set}$ to (finite) coproducts $\emptyset, +, \text{nlin}^*, \text{nir}^*$ of $\text{Set}_T$.

For legibility, we omit some obvious isomorphisms in the remainder of this paper. For example, we write $\eta_A$ for a map from $1 \times A$ to $T(1 \times A)$ obtained by composing $\eta_A$ with obvious isomorphisms.

IV. TRANSDUCERS AND A COMPONENT CALCULUS

A. Transducer

Transducers are “functions with internal states.”

**Definition IV.1.** Let $T$ be a monad on $\text{Set}$. For sets $A$ and $B$, a $T$-transducer from $A$ to $B$ is a pair $(X, c)$ consisting of a set $X$ together with a $\text{Set}_T$-morphism $c : X \times A \to_T X \times B$. A pointed $T$-transducer is a triple $(X, c, x)$ consisting of a $T$-transducer $(X, c)$ and a map $x : 1 \to X$. We often drop the word ‘pointed’ in ‘pointed $T$-transducer.’ When $(X, c, x)$ is a $T$-transducer from $A$ to $B$, we write $(X, c, x) : A \to B$. 


A $T$-transducer $(X, c, x)$ is a machine consisting of a set of (internal) states $X$, an initial state $x$ and a transition rule $c$. For example, when $T$ is the identity functor, an equation $c(y, a) = (y', g')$ means that if an input is $a$ and the current internal state of the machine is $y$, then the machine outputs $a'$ and the next internal state is $y'$. The monad $T$ enables us to consider various effects of transition rules: when $T$ is the powerset monad, transition rules are nondeterministic.

**Requirement IV.2.** Throughout the paper we require that the symmetric monoidal category $(\text{Set}_T, +, 0, \emptyset)$ has a trace operator $\text{tr}$ that satisfies the following restricted uniformity [26]: for all $h: C \to D, f: A + C \to T B + C$ and $g: A + D \to T B + D$, if $(B + h^*) \circ_T f = g \circ_T (A + h^*)$, then $\text{tr}_{A,B}^C (f) = \text{tr}_{A,B}^D (g)$.

It is typical in particle style GoI [8] that the underlying traced symmetric monoidal category of a GoI situation is a Kleisli category with a trace operator that is uniform in the above restricted sense. The next lemma is useful for checking Requirement IV.2. We write $\text{Cppo}$ for the category of pointed complete posets (cpo) and continuous maps. We consider $\text{Cppo}$-enrichment with respect to the symmetric monoidal structure given by the finite products. A $\text{Cppo}$-enriched cocartesian category $A$ is a $\text{Cppo}$-enriched category whose underlying category has finite coproducts such that the coproduct $f + g: A + B \to C + D$ is continuous on $f$ and $g$.

**Lemma IV.3** (Lemma A.2). If the Kleisli category $\text{Set}_T$ is a $\text{Cppo}$-enriched cocartesian category such that the bottom morphisms $\bot_{A,B}: A \to T B$ satisfy the following conditions:

- $f \circ_T \bot_{A,B} = \bot_{A,B'}$ for all $f: B \to T B'$
- $\bot_{A,B} \circ_T g^* = \bot_{A',B}$ for all $g: A' \to A$

then $(\text{Set}_T, +, 0, \emptyset)$ satisfies Requirement IV.2.

In the following examples, we can use Lemma IV.3 to check Requirement IV.2: we combine partiality with the standard definitions of monads so that the Kleisli categories are enriched over $\text{Cppo}$.

**Example IV.4.** We give leading examples of monads that satisfy Requirement IV.2.

- The **lift** monad $\mathcal{L}A = 1 + A$.
- The **(full) powerset** monad $\mathcal{P}A = 2^A$ and the **countable powerset** monad $\mathcal{P}_\omega A = \{a \subseteq A \mid a \text{ is countable}\}$.
- The **probabilistic substitution** monad $D A = \{d: A \to [0, 1] \mid \sum_{a \in A} da \leq 1\}$ where $[0, 1]$ is the unit interval.
- A **global state** monad $S A = (1 + A \times V^L)^V$ where $V$ and $L$ are (countable) sets.
- A **writer** monad $T A = 1 + M \times A$ where $M$ is a monoid.
- An **exception** monad $E A = 1 + E + A$ where $E$ is a set.
- A **continuation** monad $T A = R^A \Rightarrow R$ where $R$ is a pointed cpo and $R^A \Rightarrow R$ is the set of continuous maps from the $A$-fold product of $R$ to $R$.
- An $I/O$ monad $T A = \mu D. (O \times D + D^I + A)_{\bot}$ where $O$ and $I$ are (countable) sets.

In the last example, we regard a set as a cpo with the discrete order, and $D_{\bot}$ is the pointed cpo obtained by adding a bottom element to a cpo $D$. For an endo-functor $F$ on $\text{Cppo}$, the fixed point $\mu D. FD$ denotes an initial $F$-algebra in $\text{Cppo}$.

**B. A Component Calculus**

We shall extend some constructions on $\text{Set}_T$ to constructions on $T$-transducers, namely the sequential composition $f \circ_T g$, the traced symmetric monoidal structure $(\text{Set}_T, +, 0, \emptyset, \text{tr})$ and algebraic operations on $T$. These extensions will organize $T$-transducers into a “traced symmetric monoidal category”, on which we will define a “GoI situation”. Here the quotation marks (“so to speak”) are because the equational axioms of traced symmetric monoidal category and GoI situation hold only up-to suitable equivalences between $T$-transducers. In Section V, we will (properly) introduce a traced symmetric monoidal category and a GoI situation as quotients of the “traced symmetric monoidal category” and the “GoI situation” in this section.

1) **Identity and Composition:** For a set $A$, we define an “identity” on $A$ to be the obvious one-state $T$-transducer

$$(1, \eta_A, \text{id}_1): A \to A.$$  

Generalizing the above “identity,” we have a construction $J$ from a $\text{Set}_T$-morphism $f: A \to T B$ to a $T$-transducer $J f: A \to B$ defined by (1, $f, \text{id}_1$). For a map $g: A \to B$, we define a $T$-transducer $J g: A \to B$ as the composition of $J$ and the Kleisli inclusion, namely $(1, g^*, \text{id}_1)$.

For $T$-transducers $(X, c, x): A \to B$ and $(Y, d, y): B \to C$, we define a “composition”

$$(Y, d, y) \circ (X, c, x): A \to C$$

to be $(X \times Y, e, x \times y)$ where $e$ is a $\text{Set}_T$-morphism from $X \times Y \times A$ to $X \times Y \times C$ given by

$$(X \otimes d) \circ_T (X \otimes \sigma^*_{B,Y}) \circ_T (c \otimes Y) \circ_T (X \otimes \sigma^*_{Y,A}).$$

This is the sequential composition of machines:

$$(Y, d, y) \circ (X, c, x) = \begin{cases} C & \text{if } (Y, d, y) \circ (X, c, x) = B(X, c, x) \\ (X, c, x) & \text{if } (Y, d, y) \circ (X, c, x) = A \end{cases}$$

We note that the above diagram is an intuitive representation of the composition and is inept for rigorous reasoning. For example, the composition of $T$-transducers fails to be associative in the strict sense.

2) **Monoidal Product:** We define a “monoidal product”

$$(X, c, x) \boxplus (Y, d, y): A + C \to B + D$$

of $T$-transducers $(X, c, x): A \to B$ and $(Y, d, y): C \to D$ to be $(X \times Y, e, x \times y)$ where

$$e: X \times Y \times (A + C) \to_T X \times Y \times (B + D)$$
is a unique Set-$T$-morphism such that
e \circ_T (X \otimes Y \otimes \mathsf{inl}_{A,C}) = (\sigma_{Y,X} \otimes \mathsf{inl}_{B,D})
\circ_T (Y \otimes c) \circ_T (\sigma_{X,Y} \otimes A),
\mathcal{e} \circ_T (X \otimes Y \otimes \mathsf{inr}_{A,C}) = (X \otimes Y \otimes \mathsf{inr}_{B,D}) \circ_T (X \otimes d).

The “monoidal product” $oxplus$ is the parallel composition of machines:
$(X, c, x) \boxplus (Y, d, y) = \binom{(X,c,x)}{(Y,d,y)}$.

The $T$-transducers $(X, c, x)$ and $(Y, d, y)$ behave independently following their own internal states.

3) Trace: For a $T$-transducer $(X, c, x) : A + C \rightarrow B + C$, we define a $T$-transducer $\mathsf{Tr}_{A,B}^C(X, c, x) : A \rightarrow C$ to be

$\binom{X, \mathsf{tr}^X_{A,X,A,X,B}((\delta^n_{X,A,B,C})^{-1} \circ_T c \circ_T \delta^n_{X,A,C}), x}{(B \otimes C)}$.

The operator $\mathsf{Tr}$ is a “trace operator” with respect to the “symmetric monoidal structure” $(\boxplus, \emptyset)$. Checking that trace axioms are indeed “satisfied” is laborious but doable; see [22].

The “trace operator” introduces feedback:

\[
\begin{array}{ccc}
B & \mathsf{tr}^C_{A,B} & C \\
\mathsf{Tr}^C_{A,B}(X, c, x) & A & C
\end{array}
\]

4) GoI Situation: Let $\mathbb{N}$ be the set of natural numbers. We define maps

$\kappa_n : 1 \rightarrow \mathbb{N}$, \quad \varpi_{n,X} : X^n \rightarrow X \times X, \quad f^n : A^n \rightarrow B^n$
to be the constant map $\kappa_n(*) = n$, the permutation that picks the $n$-th element, and the $\mathbb{N}$-fold product of a map $f : A \rightarrow B$.

For a set $A$, we define a set $FA$ to be $\mathbb{N} \times A$, and for a $T$-transducer $(X, c, x) : A \rightarrow B$, we define a $T$-transducer $F(X, c, x) : FA \rightarrow FB$
to be $(X^n, c', x^n)$ whose transition map

$c' : X^n \times \mathbb{N} \times A \rightarrow X \times X \times B$
is a unique Set-$T$-morphism such that

$c' \circ_T (X^n \otimes \kappa^n_B \otimes A) =
((\varpi_{n,X})^{-1} \otimes \kappa^n_B \otimes A) \circ_T (X^n \otimes c) \circ_T (\varpi_{n,X} \otimes A)$
for all natural numbers $n$. The construction $F$ introduces a parallel composition of countably infinite copies:

$F(X, c, x) = \binom{1B}{1A} \binom{1B}{1A} \binom{1B}{1A} \cdots$.

Each $(X, c, x)$ in $F(X, c, x)$ behaves independently.

We choose bijections $\phi : \mathbb{N} + \mathbb{N} \cong \mathbb{N}$; $\psi$ and $u : FN \cong \mathbb{N}$; $v$ in Set, which induce the following “retractions”

$J_0 \phi : \mathbb{N} + \mathbb{N} \cong \mathbb{N} : J_0 \psi, \quad J_0 u : FN \cong \mathbb{N} : J_0 v$.

The list $(\mathbb{N}, F, J_0 \phi, J_0 \psi, J_0 u, J_0 v)$ forms a “GoI situation.” In fact, we have the following “retractions”

$J_0 (\kappa_1 \times A) : A \otimes FA : J_0 (\top_0 \times A)$ (dereliction)
$J_0 (u \times A) : FFA \cong FA : J_0 (\top_0 \times A) \circ_{\top_0} B$ (digging)
$J_0 (\phi \times A) : FA + FA \cong FA : J_0 (\psi \times A)$ (contraction)

where we omit several obvious “isomorphisms.”

We illustrate how these “retractions” act on $T$-transducers. We note that a pair of $T$-transducers

$(Y, d, y) : A' \cong A : (Y', d', y')$
induces a translation of $T$-transducers:

$(X, c, x) : A \rightarrow A \rightarrow (Y', d', y') \circ (X, c, x) \circ (Y, d, y) : A' \rightarrow A'$.

For a $T$-transducer $(X, c, x) : A \rightarrow A$, dereliction pulls out the first $(X, c, x)$ in $F(X, c, x)$, and digging sorts $F(X, c, x)$ into a bunch of bunches of $(X, c, x)$’s:

Weakening discards $F(X, c, x)$ completely, and contraction sorts $F(X, c, x)$ into a pair of $F(X, c, x)$’s:

For $T$-transducers $(X, c, x), (Y, d, y) : \mathbb{N} \rightarrow \mathbb{N}$, we define a $T$-transducer $(X, c, x) \bullet (Y, d, y) : \mathbb{N} \rightarrow \mathbb{N}$ to be

$\mathsf{Tr}_{\mathbb{N}, \mathbb{N}}^T((\mathbb{N} \otimes (J_0 u \circ F(Y, d, y) \circ J_0 v)) \circ J_0 \psi \circ (X, c, x) \circ J_0 \phi)$.

Since $(\mathbb{N}, F, J_0 \phi, J_0 \psi, J_0 u, J_0 v)$ is a “GoI situation”, the set of $T$-transducers with the binary application $\bullet$ and $\otimes$ forms an “SK-algebra” by Proposition II.4. The binary application consists of parallel composition plus hiding:

$(X, c, x) \bullet (Y, d, y) = \binom{J_0 \psi}{J_0 \phi} \binom{J_0 u}{J_0 v} \binom{J_0 \psi}{J_0 \phi}$

Hiding means that we can not observe interaction between $J_0 u \circ F(Y, d, y) \circ J_0 v$ and $J_0 \psi \circ (X, c, x) \circ J_0 \phi$ from outside.

5) Algebraic Operation: We extend algebraic operations on $T$ to operations on $T$-transducers. We first recall the definition of algebraic operation, which is a mathematical interface to computational effects.

**Definition IV.5** ([16]). Let $T$ be a strong monad on a cartesian closed category $(C, 1, \times, \Rightarrow)$ with countable products, and let
I be a countable set. An I-ary algebraic operation on T is a family of C-morphisms
\[ \{ \alpha_{A,B}: (A \Rightarrow TB)^I \rightarrow (A \Rightarrow TB) \}_{A,B \in C} \]
such that
\[ \alpha_{A',B} \circ \mathcal{C} \circ \Delta = \mathcal{C} \circ ((B \Rightarrow TB') \times \alpha_{A,B} \times (A' \Rightarrow A)) \]
where
\[ \mathcal{C}: (B \Rightarrow TB') \times (A \Rightarrow TB) \times (A' \Rightarrow A) \rightarrow (A' \Rightarrow TB') \]
is the (Kleisli) composition, and
\[ \Delta: (B \Rightarrow TB') \times (A \Rightarrow TB)^I \times (A' \Rightarrow A) \rightarrow \]
\[ ((B \Rightarrow TB') \times (A \Rightarrow TB) \times (A' \Rightarrow A))^I \]
is the \(C\)-morphism that is diagonal in the first argument and the third argument. We write \(\approx (\alpha)\) for \(I\).

We define AlgOp\(_T\) to be the category of algebraic operations on \(T\): an object is a countable set, and a morphism from \(I\) to \(I'\) is a family of \(C\)-morphisms
\[ \{ \alpha_{A,B}: (A \Rightarrow TB)^I \rightarrow (A \Rightarrow TB)^{I'} \}_{A,B \in C} \]
such that the family \(\{ \pi_{j,A,B} \circ \alpha_{A,B} \}_{A,B \in C}\) is an \(I\)-ary algebraic operation for all \(j \in I'\) where \(\pi_{j,A,B}\) is the \(j\)-th projection from \((A \Rightarrow TB)^{I'}\) to \(A \Rightarrow TB\). The category AlgOp\(_T\) has countable products given by the disjoint union.

**Example IV.6.** We give examples of algebraic operations.
- A binary algebraic operation \(\varnothing\) on \(P\) given by
  \[ (f \varnothing_{A,B} g)(a) = f(a) \cup g(a) \]
  where we use an infix notation. We will use the \(\varnothing\) to interpret the \(\square\) construct in the introduction.
- A binary algebraic operation \(\varnothing^p\) on \(D\) given by
  \[ (f \varnothing^p_{A,B} g)(a) = p \cdot f(a) + (1 - p) \cdot g(a) \]
  where \(p\) is a real number in the unit interval \([0,1]\).
- A state monad \(\mathcal{S}X = (1 + X \times L^T Y)^V\) for a countable set of locations \(L\) and a countable set of values \(V\) has the following algebraic operations
  \[ \text{lookup}_{\ell,A,B}: \mathcal{S}\text{Set}_S(A,B)^V \rightarrow \mathcal{S}\text{Set}_S(A,B), \]
  \[ \text{update}_{\ell,A,B}: \mathcal{S}\text{Set}_S(A,B) \rightarrow \mathcal{S}\text{Set}_S(A,B) \]
  for each \(\ell \in L\) and \(v \in V\) given by
  \[ ((\text{lookup}_{\ell,A,B}(f))(a))(s) = (f_s(\ell))(a)(s), \]
  \[ ((\text{update}_{\ell,A,B}(f))(a))(s) = (fa)[s[v/\ell]] \]
  where the state \(s[v/\ell]\) is the same as \(s\) everywhere except at \(\ell\) where \(s[v/\ell](\ell) = v\).

For other examples of algebraic operations, see [16].

For a monad \(T\) on Set and an \(I\)-ary algebraic operation \(\alpha\) on \(T\) and a family of \(T\)-transducers \(\{(X_i,c_i,x_i): A \Rightarrow B\}_{i \in I}\), we define
\[ \bar{\alpha}_{A,B}\{(X_i,c_i,x_i)\}_{i \in I}: A \Rightarrow B \]
to be a \(T\)-transducer \((1 + Y, d, \text{in}1, Y)\) consisting of a coproduct
\[ Y = \bigsqcup_{i \in I} X_i \text{ in}1 \rightarrow X_i \]
and a unique \(\text{Set}T\)-morphism \(d\) from \((1 + Y) \times A \rightarrow (1 + Y) \times B\) satisfying
\[ d \circ_T (\text{in}1_{i,Y} \otimes A) = \alpha_{A,(1,Y)\times B}\{c_i'\}_{i \in I} \]
\[ d \circ_T ((\text{in}1_{i,Y} \otimes \text{inj}_{i,Y}) \otimes A) = ((\text{inj}_{i,Y} \otimes \text{in}1_i) \otimes B) \circ_T c_i \]
where \(c_i'\) is a \(\text{Set}T\)-morphism from \(A\) to \((1 + Y) \times B\) given by \((\text{inj}_{i,Y} \otimes B) \circ_T (\text{inj}_{i,Y} \otimes B) \circ_T c_i\).

Intuitively, the construction \(\pi\) introduces branching at the (fresh) initial state. A \(T\)-transducer \(\bar{\alpha}_{A,B}\{(X_i,c_i,x_i)\}_{i \in I}\) memorizes the first branching information using its internal states, and after the first branching, the behavior of \(\bar{\alpha}_{A,B}\{(X_i,c_i,x_i)\}_{i \in I}\) in the \(i\)-th branching follows the \(i\)-th \(T\)-transducer \((X_i,c_i,x_i)\) for each \(i \in I\). For example, the nondeterministic Mealy machine in (3) is the same as the following \(P\)-transducer
\[ \{(x_0,c_0,x_0) \overline{\mathcal{P}}(q),\{0,1\}\{x_1,c_1,x_1\}: \{q\} \rightarrow \{0,1\} \]
where \(\{(x_1,c_1,x_1)\}: \{q\} \rightarrow \{0,1\}\) are \(P\)-transducers given by \(c_i(x_i,q) = (x_i,i)\) for \(i = 0,1\).

**V. Behavioral Equivalence**

We have presented “Gol situation” on a “traced symmetric monoidal category” of sets and \(T\)-transducers. Precisely speaking, they are not so in a strict sense: in order to satisfy the equational axioms of traced symmetric monoidal category and Gol situation, \(T\)-transducers must be suitably quotiented. For example, \((X,c,x) \circ (1,\eta_A,\text{id})\) is not equal to \((X,c,x)\), and we need to identify them. In this paper, we use behavioral equivalence, a notion common in coalgebra [27]. Intuitively, two (pointed) \(T\)-transducers are behaviorally equivalent if the initial states are connected by a zigzag of homomorphisms.

**Definition V.1.** Let \((X,c,x)\) and \((Y,d,y)\) be \(T\)-transducers from \(A\) to \(B\). A homomorphism from \((X,c,x)\) to \((Y,d,y)\) is a map \(h: X \rightarrow Y\) such that \((h^* \otimes B) \circ_T c = d \circ_T (h^* \otimes A)\) and \(h \circ y = y\).

**Definition V.2.** For \(T\)-transducers \((X,c,x)\) and \((Y,d,y)\) from \(A\) to \(B\), we say that \((X,c,x)\) is behaviorally equivalent to \((Y,d,y)\) if there is a \(T\)-transducer \((Z,e,z)\) in \(T\) and homomorphisms from \((X,c,x)\) to \((Z,e,z)\) and from \((Y,d,y)\) to \((Z,e,z)\). When \((X,c,x)\) is behaviorally equivalent to \((Y,d,y)\), we write \((X,c,x) \simeq_{T,A,B} (Y,d,y)\).

Up to the behavioral equivalence, we can drop the quotation marks in Section IV-B. It is easy to check that constructions \(\circ, \oplus, \text{Tr}, F, \bullet\) and \(\pi\) are compatible with the behavioral equivalence. Below we abuse notations: we use \(\oplus, \text{Tr}, F\) and \(\bullet\) for operators on \(T\)-transducers as well as those on equivalence classes of \(T\)-transducers.

We define a category Res\((T)\) by
- Objects are sets.
- Morphisms from \(A\) to \(B\) are \(\simeq_{T,A,B}\)-equivalence classes of \(T\)-transducers from \(A\) to \(B\).

For a \(T\)-transducer \((X,c,x)\) : \(A \rightarrow B\), we write \([[(X,c,x)]]\) for the Res\((T)\)-morphism from \(A\) to \(B\) represented by \((X,c,x)\).
The identity on $A$ is $\{1, \eta_A, id_1\}$, and the composition of a Res($T$)-morphism $[(X, c, x)]$ from $A$ to $B$ and a Res($T$)-morphism $[(Y, d, y)]$ from $B$ to $C$ is $[(Y, d, y) \circ (X, c, x)]$.

The category Res($T$) with $\llbracket, \emptyset, Tr \rrbracket$ is a traced symmetric monoidal category. The coherence isomorphisms of the symmetric monoidal category $(\text{Set}, +, \emptyset)$ induce coherence isomorphisms of the symmetric monoidal category (Res($T$), $\llbracket, \emptyset \rrbracket$). The following list

$$(\text{Res}(T), N, F, [J_0\phi], [J_0\psi], [J_0u], [J_0v])$$

is a GoI situation, and (Res($T$)($N, N, \bullet$)) is an SK-algebra.

**Theorem V.3** (Theorem A.5). Let $\alpha$ be an $I$-ary algebraic operation on $T$. The operation $\alpha$ is natural, and $\alpha$ is distributive over $\tau$ modulo the behavioral equivalence:

- For each $T$-transducer $(X, c, x) : B \rightarrow B'$ for each family of $T$-transducers $\{Y_i, d_i, y_i\} : A \rightarrow B$ for each map $h : A' \rightarrow A$, it holds that

$$(X, c, x) \circ \alpha_{A,B} \{Y_i, d_i, y_i\}_{i \in I} \equiv J_0 h$$

- For each family of $T$-transducers $\{(X_i, c_i, x_i)\}_{i \in I}$ from $A + C$ to $B + C$, it holds that

$$\alpha_{A+C,B+C} \{X_i, c_i, x_i\}_{i \in I} \sim_{A,B}^T \alpha_{A,B}^T \{\alpha_{A,B} \{X_i, c_i, x_i\}_{i \in I}\}.$$  

(6)

Let $\alpha$ be an $I$-ary algebraic operation on $T$. The following behavioral equivalence is a consequence of Theorem V.3:

$$\alpha_{N,N} \{X_i, c_i, x_i\}_{i \in I} \bullet (Y, d, y) \sim_{N,N}^T \alpha_{N,N} \{X_i, c_i, x_i\}_{i \in I} \bullet (Y, d, y)$$

(7)

where $\{(X_i, c_i, x_i)\}_{i \in I}$ and $\{Y, d, y\}$ are $T$-transducers from $N$ to $N$. In fact, we have

$$\alpha_{N,N} \{X_i, c_i, x_i\}_{i \in I} \bullet (Y, d, y)$$

$$= \text{Tr}_{N,N}^T \{Z, e, z\} \circ J_0 \psi \circ \alpha_{N,N} \{X_i, c_i, x_i\}_{i \in I} \circ J_0 \phi$$

$$\sim_{N,N}^T \text{Tr}_{N,N}^T \{Z, e, z\} \circ J_0 \psi \circ \alpha_{N,N} \{X_i, c_i, x_i\}_{i \in I} \circ J_0 \phi$$

$$\sim_{N,N}^T \text{Tr}_{N,N}^T \{Z, e, z\} \circ J_0 \psi \circ \alpha_{N,N} \{X_i, c_i, x_i\}_{i \in I}$$

(8)

where we write $\{Z, e, z\}$ for $N \llbracket J_0 u \circ F(Y, d, y) \circ J_0 \phi \rrbracket$. The first equivalence follows from naturality of $\alpha$, and the second equivalence follows from distributivity of $\text{Tr}$ over $\alpha$. The behavioral equivalences (6) and (7) indicate that our construction resolves the obstacles in categorical GoI interpretation of algebraic effects (c.f. (4) and (5)).

VI. REALIZABILITY AND CATEGORICAL MODELS

In the next section, we exemplify GoI interpretation of algebraic effects. The purpose of this section is to sketch how to derive them: we use realizability technique. For details of arguments and proofs of theorems in this section, see Section C in the appendix.

From the SK-algebra $\text{Res}(T)(N, N, \bullet)$, we can utilize the realizability construction and constructs a cartesian closed category $\text{Per}(T)$ consisting of partial equivalence relations on $\text{Res}(T)(N, N)$ and realizable maps. See [9] for a precise definition of $\text{Per}(T)$. Since the category $\text{Per}(T)$ has countable products (Proposition A.10), we can consider algebraic operations on monads on $\text{Per}(T)$.

The next theorem is our main theorem, from which soundness of GoI interpretation that we are going to give follows.

**Theorem VI.1.** The cartesian closed category $\text{Per}(T)$ has a strong monad $\Phi$ and an identity-on-object countable-product-preserving faithful functor $(-)\uparrow : \text{AlgOp}_T \rightarrow \text{AlgOp}_{\Phi}$. We only give a definition of $\Phi R$ for $R \in \text{Per}(T)$. Let $h$ be a map from $N$ to $N$ given by

$$\phi \circ (\phi + N) \circ (\psi + N) \circ \psi \circ (\phi + N) \circ (N + \psi) \circ (\psi + N) \circ \psi$$

where the map $\phi : N + N \rightarrow N + N$ is the swapping. We derived $h$ using combinatorial completeness: the map $h$ represents a term $\lambda x. \lambda k (\lambda x)$ of the untyped linear lambda calculus [28]. We say that an object $R$ in $\text{Per}(T)$ is closed when $\{\alpha_{N,N} \{a_i\}_{i \in \text{ary}(a)}\}_{(\alpha, a') \in \text{ary}(a)}$ is in $R$ for each

$$\{\{a_i\}_{i \in \text{ary}(a)}\} \in R \text{ and for each algebraic operation } a \text{ on } T. \text{ We define } \Phi R \text{ by }$$

$$\Phi R = \{S \in \text{Per}(T) \mid R' \subseteq S \text{ is closed}\}$$

(9)

where $R' = \{\{a_0 \bullet a, \{a_0 \bullet a\}' \mid (a, a') \in R\}$.

By Theorem VI.1, the Kleisli category $\text{Per}(T)_{\Phi}$ is a categorical model of the computational lambda calculus, i.e., there is a canonical interpretation of the computational lambda calculus in $\text{Per}(T)_{\Phi}$. The interpretation, which we call categorical interpretation, is sound with respect to the standard equational theory of the computational lambda calculus [15]. We can extend the categorical interpretation by interpreting algebraic effects using algebraic operations on $\Phi$ induced by algebraic operations on $T$ via $(\cdot)^\dagger$. For example, when we need nondeterminism, we can start from the powerset monad; when we need global states, we can start from a global state monad.

We sketch extraction of GoI interpretation—i.e. extraction of concrete $T$-transducers as realizers—from the categorical interpretation of the computational lambda calculus in $\text{Per}(T)_{\Phi}$. For simplicity, we only consider closed terms.

1) We choose a monad $T$ on Set that satisfies Requirement IV.2.

2) We interpret the computational lambda calculus in the Kleisli category $\text{Per}(T)_{\Phi}$ as in [15], [16] where we interpret algebraic effects by algebraic operations on $\Phi$ derived from algebraic operations on $T$ via $(\cdot)^\dagger$.

3) The categorical interpretation of a closed term $t$ of a type $\tau$ bijectively corresponds to an equivalence class of a partial equivalence relation $\Phi[\tau]$ where $[\tau]$ is the categorical interpretation of the type $\tau$. We choose a Res($T$)-morphism on $N$ that represents the equivalence class, and then, we extract a $T$-transducer $(\psi) : N \rightarrow N$ that represents the Res($T$)-morphism on $N$. 
We call the \( T \)-transducer \((t)\) \textit{GoI interpretation} of a term \( t \).

Let \( \mathcal{L} \) be an extension of the computational lambda calculus to algebraic effects and a base type \( \text{nat} \) of natural numbers. See Section C in the appendix for the syntax and equational theory of \( \mathcal{L} \). For closed terms \( t \) and \( s \) of type \( \tau \) in \( \mathcal{L} \), we write \( t \approx s \) when the equation holds in \( \mathcal{L} \). For example, we have

\[
\forall (3 \sqcup 5) \equiv \forall 3 \sqcup \forall 5, \quad 3 \sqcup 5 \sqcup 3 \approx 3 \sqcup 5 \approx 5 \sqcup 3
\]

for any value \( v \) when \( \mathcal{L} \) has nondeterminism. We extracted GoI interpretation so that the next theorem holds.

**Theorem VI.2 (Soundness).** For closed terms \( t \) and \( s \) of type \( \tau \) in \( \mathcal{L} \),

- If \( t \approx s \), then \( \langle \langle t \rangle \rangle, \langle \langle s \rangle \rangle \rangle \in \Theta \mathcal{L} \).
- If \( t \approx s \) and \( \tau \) is the base type \( \text{nat} \), then \( \langle \langle t \rangle \rangle \cong^{\text{Res}(T)} \langle \langle s \rangle \rangle \).

where \( \langle \langle t \rangle \rangle \) is the \textit{Res}(\( T \))-morphism represented by \( \langle t \rangle \).

**VII. GoI Interpretation of Algebraic Effects**

### A. Memoryless GoI Interpretation

For comparison, we first present (memoryless) GoI interpretation of the following programs:

\[
(\lambda xy : \text{nat}. x + y) \ 5 \ 3 \quad (\lambda x : \text{nat}. x + x) \ 3.
\]

We write \( g \) for \( \phi \circ \text{inr}_{N,N}, \) \( d \) for \( \phi \circ \text{inl}_{N,N} \) (gauche and droit like \([12]\)) and \((n, m)\) for \( u(n, m)\). For \( i \in \mathbb{N} \), we define a map \( k_i : N \rightarrow N \) by

\[
k_i(m, n) = (m, i),
\]

and we define maps \( \text{sum}, \ cpy : N + N + N \rightarrow N + N + N \) by

- \( \text{sum}(\text{inj}_1(n)) = \text{inj}_2(n) \)
- \( \text{sum}(\text{inj}_2(n)) = \text{inj}_3(n, 0) \)
- \( \text{sum}(\text{inj}_3((n, m), l)) = \text{inj}_3(n, m + l) \)
- \( \text{cpy}(\text{inj}_1(n, m)) = \text{inj}_3(gn, m) \)
- \( \text{cpy}(\text{inj}_2(n, m)) = \text{inj}_3(df, m) \)
- \( \text{cpy}(\text{inj}_3(gn, m)) = \text{inj}_3(n, m) \)
- \( \text{cpy}(\text{inj}_3(df, m)) = \text{inj}_3(n, m) \)

where \( \text{inj}_i : N \rightarrow N + N + N \) is the \( i \)-th injection. The map \( \text{cpy} \) is from contraction in the GoI situation.

In (memoryless) GoI interpretation, we interpret a closed term as a partial map from \( \mathbb{N} \) to \( \mathbb{N} \). The following diagrams present GoI interpretation of programs:

\[
\langle n \rangle = k_n : N \rightarrow N,
\]

\[
\langle(\lambda xy : \text{nat}. x + y) \ 5 \ 3 \rangle = \text{sum} \begin{bmatrix} k_3 & k_5 \\ \text{cpy} \end{bmatrix} : N \rightarrow N,
\]

\[
\langle(\lambda x : \text{nat}. x + x) \ 3 \rangle = \text{sum} \begin{bmatrix} k_3 \\ \text{cpy} \end{bmatrix} : N \rightarrow N.
\]

If we input \( \langle n, m \rangle \) to \( \langle(\lambda xy : \text{nat}. x + y) \ 5 \ 3 \rangle \), then we get an output \( \langle n, 8 \rangle \) as a result of the following interactive computation between \( \text{sum}, \ k_3 \) and \( k_5 \).

1. \( \text{sum} \) receives \( \langle n, m \rangle \) from the leftmost port and outputs \( \langle n, m \rangle \) from the middle port to ask a value of \( x \).
2. \( k_5 \) answers \( \langle n, 5 \rangle \) to \( \text{sum} \).
3. \( \text{sum} \) receives \( \langle n, 5 \rangle \) from the middle port and outputs \( \langle(\langle n, 5 \rangle), 0 \rangle \) from the rightmost port to ask a value of \( y \).
4. \( k_3 \) answers \( \langle(\langle n, 5 \rangle), 3 \rangle \) to \( \text{sum} \).
5. \( \text{sum} \) outputs \( \langle n, 8 \rangle \) from the leftmost port.

As a whole, GoI interpretation is sound with respect to \( \beta \)-equality: \( \langle(\lambda xy : \text{nat}. x + y) \ 5 \ 3 \rangle \) is equal to \( k_6 \). Similarly, we can check that the GoI interpretation \( \langle(\lambda x : \text{nat}. x + x) \ 3 \rangle \) is equal to \( k_6 \). The interactive computation illustrates how \( \text{sum} \) and \( \text{cpy} \) work: \( \text{sum} \) computes sum, and \( \text{cpy} \) copies data.

### B. Memoryful GoI Interpretation of Nondeterminism

GoI interpretation of the computational lambda calculus (i.e. call-by-value calculus) in Section VII-B and Section VII-C follows from the general scheme that we developed in this paper. In this section, we consider the following program:

\[
P = (\lambda x : \text{nat}. x + x) (3 \sqcup 5) : \text{nat}.
\]

The subterm \( 3 \sqcup 5 \) means nondeterministic choice between 3 and 5. We have the following equations:

\[
P = ((\lambda x : \text{nat}. x + x) \ 3) \sqcup ((\lambda x : \text{nat}. x + x) \ 5)
\]

\[
= (3 + 3) \sqcup (5 + 5)
\]

\[
= 6 \sqcup 10.
\]

We extract GoI interpretation from the categorical interpretation in \( \text{Per}(\mathcal{P})_{\Phi} \) where \( \Phi \) is the monad in Theorem VI.1 for \( T = \mathcal{P} \). We interpret \( \sqcup \) by the algebraic operation \( \oplus \) on \( \Phi \).

Below, we confuse a map \( f : N \rightarrow N \) with a \( T \)-transducer \( J_0 f : N \rightarrow N \). In memoryful GoI interpretation, we interpret closed terms by \( \mathcal{P} \)-transducers from \( N \) to \( N \). Memoryful GoI interpretation of a natural number is bit complicated: we interpret a natural number \( n \) as follows:

\[
\langle n \rangle = h \bullet k_n = \begin{bmatrix} k_3 & k_5 \\ \text{cpy} \end{bmatrix} : N \rightarrow N
\]

where we write a map \( h : N + N \rightarrow N + N \) for \( \psi \circ h \circ \phi \). Here \( h \) is from Section VI. The combinator \( h \) corresponds to the unit of the monad \( \Phi \).

We interpret the nondeterministic choice \( 3 \sqcup 5 \) by a \( \mathcal{P} \)-transducer \((3)_{\mathcal{P},N,N}(5) = \{(x_{3L5}, x_3, x_3), c, x_{3L5}\}\) given by

\[
c(x_{3L5}, n) = \{(x_{3}, (3)(n)), (x_5, (5)(n))\}
\]

\[
c(x_3, n) = \{(x_{3}, (3)(n))\}
\]

\[
c(x_5, n) = \{(x_{3}, (5)(n))\}.
\]

The \( \mathcal{P} \)-transducer \((3)_{\mathcal{P},N,N}(5) \) behaves like the nondeterministic Mealy machine (3): initially, the \( \mathcal{P} \)-transducer \((3)_{\mathcal{P},N,N}(5) \) nondeterministically chooses (3) or (5): thereafter \((3)_{\mathcal{P},N,N}(5) \) sticks to the same choice referring to its internal state.

GoI interpretation of the program \( P \) by

\[
P = \begin{bmatrix} 3 \oplus \text{cpy} \end{bmatrix} : N \rightarrow N
\]
The above diagram is modularly constructed in our generic framework; we also simplified it by reducing retraction pairs in the GoI situation \(\{\text{Res}(P), N, F, [J_0\phi], [J_0\psi], [J_0\eta], [J_0\nu]\}\). For example, since \(\phi\) is the inverse of \(\psi\), we can apply the following reduction to GoI interpretation:

\[
\sum \in \psi \vdash \sum \in \psi
\]

These diagrams represent the identity on \(\mathbb{N} + \mathbb{N}\), and the reduction does not affects the GoI interpretation.

The \(\mathcal{P}\)-transducer \(\langle P \rangle\) behaves as follows:

1. We input \(dd(n, m)\).
2. \(\langle 3 \rangle \mathcal{P}^{\langle 5 \rangle} N\) receives an input \(gdd(n, m)\), and the internal state nondeterministically changes to \(x_3\) or \(x_5\). We assume that \(\langle 3 \rangle \mathcal{P}^{\langle 5 \rangle} N\) chooses \(x_3\). Then \(\langle 3 \rangle \mathcal{P}^{\langle 5 \rangle} N\) outputs \(dgdd(n, m)\).
3. \(\sum \) receives \(\langle n, m \rangle\) from the leftmost input port and outputs \(\langle n, m \rangle\) from the middle port to ask a value of the right occurrence of \(x\) in \(x + x\).
4. \(\text{cpy}\) receives \(\langle n, m \rangle\) from the leftmost port and outputs \(\langle gn, m \rangle\) from the rightmost port to get \(x\).
5. \(\langle 3 \rangle \mathcal{P}^{\langle 5 \rangle} N\) receives \(dd(gn, m)\). Since the internal state is \(x_3\), it answers \(dd(gn, 3)\).
6. \(\text{cpy}\) receives \(\langle gn, 3 \rangle\) from the rightmost port and answers \(\langle n, 3 \rangle\) to \(\sum\) via the leftmost port.
7. \(\sum\) receives \(\langle n, 3 \rangle\) from the middle port and asks a value of the left occurrence of \(x\) in \(x + x\) by outputting \(\langle (n, 3), 0 \rangle\) from the rightmost port.
8. \(\text{cpy}\) receives \(\langle (n, 3), 0 \rangle\) from the middle port and outputs \(\langle d(n, 3), 0 \rangle\) from the rightmost port to get \(x\).
9. \(\langle 3 \rangle \mathcal{P}^{\langle 5 \rangle} N\) receives \(dd(d(n, 3), 0)\). Since the internal state is \(x_3\), it answers \(dd(d(n, 3), 3)\).
10. \(\text{cpy}\) receives \(\langle d(n, 3), 3 \rangle\) and answers \(\langle (n, 3), 3 \rangle\) to \(\sum\) via the middle port.
11. \(\sum\) receives \(\langle (n, 3), 3 \rangle\) from the rightmost port and outputs \(\langle n, 6 \rangle\) from the leftmost port.
12. We get an output \(dd(n, 6)\).

In the computation, the map \(h\) controls interaction between \(\langle 3 \rangle \mathcal{P}^{\langle 5 \rangle} N\) and the \(\text{sum-cpy}\) fragment.

Similarly, if \(\langle 3 \rangle \mathcal{P}^{\langle 5 \rangle} N\) chooses \(x_5\) at the first step, then we get \(dd(n, 10)\) as an output. As a whole, we have the following behavioral equivalence:

\[
\langle (\lambda x: \text{nat}. x + x) \ (3 \uplus 5) \rangle \approx^{\mathcal{P}}_{\mathcal{P}, N} \langle 6 \rangle \mathcal{P}^{\langle 5 \rangle} N(10) = (6 \uplus 10).
\]

C. Memoryful GoI Interpretation of Global State

Next, we present GoI interpretation of the computational lambda calculus extended with global states. We have a countably infinite set \(\text{Loc}\) of location names, and each location name stores a natural number. Existence of global states enables us to fetch a natural number stored at a location \(\ell \in \text{Loc}\):

\[
!\ell : \text{nat},
\]

and we can update a value stored at a location \(\ell\):

\[
\ell := 3 : \text{unit}.
\]

We extract GoI interpretation of global states from the categorical interpretation in \(\text{Per}(S)_\Phi\) where \(\Phi\) is the monad in Theorem VI.1 for \(T = S\) given by

\[
SA = (1 + A \times \mathbb{N}^{\text{Loc}})^{\mathbb{N}^{\text{Loc}}}.\]

Let \(\text{Per}(T)\)-objects \(L\) and \(N\) be countably infinite co-products of terminal object 1. The algebraic operations in Example IV.6 induce algebraic operations on \(\Phi\), which induce the following \(\text{Per}(S)_\Phi\) morphisms:

\[
drf: L \to \Phi N, \quad \text{asg}: L \times N \to \Phi 1
\]
called generic effects in [16]. We interpret dereferencing \(!\ell\) by \(\text{drf}\) and assignment \(\ell := n\) by \(\text{asg}\) respectively.

For simplicity, we give GoI interpretation of \(!\ell\) and \(\ell := n\) for a fixed location \(\ell\) and a fixed value \(n\).

- We interpret \(!\ell\) by an \(S\)-transducer

\[
drf_\ell = \{(x_\ell, x_1, x_2, \ldots), c, x_\ell\}: N \to N
\]

where the \(\text{Set}\)-morphism \(c\) from \(\{x_\ell, x_1, x_2, \ldots\} \times N\) to \(\{x_\ell, x_1, x_2, \ldots\} \times N\) is given by

\[
(c(x_\ell, n))(s) = (s(x_\ell), (s(\ell))(n), s),
\]

\[
(c(x_m, n))(s) = (x_m, (n0)(n), s).
\]

Initially, the \(S\)-transducer \(\text{drf}_\ell\) looks up the global state \(s\) and behaves as an \(S\)-transducer \((s(\ell))\). At the same time, the \(S\)-transducer \(\text{drf}_\ell\) stores the value \(s(\ell)\) locally using its internal state. Thereafter, \(\text{drf}_\ell\) looks up its internal state: if an internal state is \(x_m\), then \(\text{drf}_\ell\) behaves following \(\langle n \rangle\) without referring to global states.

- We interpret \(\ell := n\) by an \(S\)-transducer

\[
\text{asg}_{\ell, n} = (\{x_\run, x_{\done}\}, c', x_\run): N \to N
\]

where the \(\text{Set}\)-morphism \(c'\) from \(\{x_\run, x_{\done}\} \times N\) to \(\{x_\run, x_{\done}\} \times N\) is given by

\[
(c'(x_\run, m))(s) = (x_{\done}, (1)(m), s[n/\ell]),
\]

\[
(c'(x_{\done}, m))(s) = (x_{\run}, (1)(m), s).
\]

Initially, \(\{x_\run, x_{\done}\}, c', x_\run\) updates a global state, and thereafter, \(\{x_\run, x_{\done}\}, c', x_\run\) does nothing. We note that \(1\) in the right hand side can be any \(S\) interpretation of a constant.

We interpret the following program

\[
Q = (\lambda x: \text{nat} . x + (\ell := 3); x)(!\ell): \text{nat}
\]

where \(t; s\) is an abbreviation of \(\lambda x: \text{unit}. s t\) by

\[
\langle Q \rangle = \langle h \rangle \triangleleft \langle \text{sum} \rangle \triangleright \langle \text{cpy} \rangle \triangleleft \langle \text{asg} \rangle \triangleright \langle \text{drf} \rangle.
\]
We simplified the above diagram by reducing retraction pairs in the GoI situation \( (\text{Res}(S), \mathbb{N}, F, [J_0 \emptyset], [J_0 \emptyset], [J_0 u], [J_0 v]) \). For an input \( dd(n,m) \) and a global state \( s \) such that \( s(\ell) = 2 \), the \( S \)-transducer behaves as follows:

1. \( \text{drf}_\ell \) refers to the global state \( s \) and memorizes the value \( s(\ell) = 2 \) by means of its internal state.
2. \( \text{asg}_{\ell,3} \) assigns \( 3 \) to \( \ell \) changing its internal state to \( x_{\text{done}} \).
3. \( \text{sum} \) asks a value of the right occurrence of \( x \) in \( x + x \).
4. \( \text{cpy} \) passes the query from \( \text{sum} \) to \( \text{drf}_\ell \).
5. \( \text{drf}_\ell \) answers 2 to the query following its internal state.

In this step, \( \text{drf}_\ell \) does not refer to the global state \( s[3/\ell] \).
6. \( \text{cpy} \) passes the answer from \( \text{drf}_\ell \) to \( \text{sum} \).
7. \( \text{sum} \) asks a value of the left occurrence of \( x \) in \( x + x \).
8. \( \text{cpy} \) passes the query from \( \text{sum} \) to \( \text{drf}_\ell \).
9. \( \text{drf}_\ell \) answers 2 to the query following its internal state.

In this step, \( \text{drf}_\ell \) does not refer to the global state \( s[3/\ell] \).
10. \( \text{cpy} \) passes the answer from \( \text{drf}_\ell \) to \( \text{sum} \).
11. \( \text{sum} \) outputs \( 4 = 2 + 2 \).

We note that without internal states, \( \text{drf}_\ell \) can not but refer to a global state at 5) and 9), which results in a wrong output. We only sketched computation process for lack of space. For example, we omit some access to \( \text{asg}_{\ell,3} \).

As a whole, the \( S \)-transducer

\[
((\lambda x : \text{nat}. \, x + (\ell := 3); x) \, \text{ !} \, \ell) : \mathbb{N} \rightarrow \mathbb{N}
\]

is behaviorally equivalent to an \( S \)-transducer

\[
\{ (x_\ell, x_1, x_2, \ldots), d, x_3) : \mathbb{N} \rightarrow \mathbb{N}
\]

where the \( \text{Set}_{S} \)-morphism \( d \) from \( \{ x_\ell, x_1, x_2, \ldots \} \times \mathbb{N} \) to \( \{ x_\ell, x_1, x_2, \ldots \} \times \mathbb{N} \) is given by

\[
(d(x_\ell, n))(s) = (x_{s(\ell)}, s(\ell) + s(\ell)(n), s[3/\ell]),
\]

\[
(d(x_m, n))(s) = (x_m, (m + m)(n), s).
\]

We can observe that memoryful GoI interpretation of the following program

\[
((\lambda x : \text{nat}. \, (\ell := 3); (x + x)) \, \text{ !} \, \ell) : \text{nat}
\]

is also behaviorally equivalent to \( \{ (x_\ell, x_1, x_2, \ldots), d, x_3) \).

VIII. CONCLUSION

We gave a general GoI/realizability workflow that interprets the computational lambda calculus with algebraic effects as concrete state machines. In other words, our framework equips token machines with internal memories and it allows to handle generic algebraic effects. Parametrized by monads on \( \text{Set} \), our construction gives rise to sound GoI interpretation of the computational lambda calculus with various algebraic effects. It seems straightforward to extend our results to polymorphic languages with recursion and recursive types.

Our result provides a systematic approach to categorical GoI for algebraic effects. It would be interesting to apply our results to compiler construction, GoI for additives and quantum lambda calculi. At this point, we do not know how to extend our framework to GoI interpretation of computational effects that can not be captured by algebraic operations like exception handler and call/cc.

REFERENCES


A. A Proof of Lemma IV.3

A Ccppo-enriched cartesian category \( C \) is a Ccppo-enriched category \( C \) such that the underlying category has finite products and the product functor \( f \times g : A \times C \to B \times D \) is continuous on \( f : A \to B \) and \( g : C \to D \). For \( C \)-morphisms \( \{ f_i : A \to B_i \}_{i=1,\ldots,n} \), we write \( \{ f_1, \ldots, f_n \} : A \to B_1 \times \cdots \times B_n \) for the tupling of \( \{ f_i \}_{i=1,\ldots,n} \). We write \( \pi_{A,B} : A \times B \to A \) for the first projection, and we write \( \pi_{A,B} : A \times B \to B \) for the second projection. When \( C \) is a Ccppo-enriched cartesian category, we write \( \perp_{A,B} : A \to B \) for the bottom element in \( C(A,B) \).

**Lemma A.1.** If \( (C, \times, 1) \) is a Ccppo-enriched cartesian category and the bottom morphisms \( \perp_{A,B} : A \to B \) satisfy the following condition:

- \( \perp_{A,B} \circ f = \perp_{A',B} \) for all \( C \)-morphisms \( f : A' \to A \) then \( (C, \times, 1) \) has a Conway operator.

**Proof:** For a \( C \)-morphism \( f : A \times B \to B \), we define a \( C \)-morphism \( \fix_{A,B}(f) : A \to B \) by

\[
\fix_{A,B}(f) = \bigvee_{n \geq 1} \fix_{A,B}^{(n)}(f)
\]

where

\[
\fix_{A,B}^{(n)}(f) = \perp_{A,B} \land \fix_{A,B}^{(n+1)}(f) = \fix_{A,B}(f) \circ \langle \id_A, \fix_{A,B}(f) \rangle.
\]

The operator \( \fix \) is a fixed point operator:

\[
f \circ \langle \id_A, \fix_{A,B}(f) \rangle = \bigvee_{n \geq 1} f \circ \langle \id_A, \fix_{A,B}^{(n)}(f) \rangle
\]

By induction on \( n \). Hence, \( \fix \) is natural:

\[
\fix_{A,B}(f) \circ g = \fix_{A',B}(f \circ (g \times B)).
\]

For \( C \)-morphisms \( f : A \times B \to C \) and \( g : A \times C \to C \), we have

\[
f \circ \langle \id_A, \fix_{A,B}(g \circ \langle \pi_{A,B}, f \rangle) \rangle
\]

\[
= \bigvee_{n \geq 1} f \circ \langle \id_A, \fix_{A,B}^{(n)}(g \circ \langle \pi_{A,B}, f \rangle) \rangle
\]

\[
= \bigvee_{n \geq 1} f \circ \langle \pi_{A,C}, g \rangle \circ \langle \pi_{A,B}, f \rangle \circ \langle \id_A, \fix_{A,B}^{(n)}(g \circ \langle \pi_{A,B}, f \rangle) \rangle
\]

\[
= \bigvee_{n \geq 1} f \circ \langle \pi_{A,C}, g \rangle \circ \langle \id_A, f \circ \langle \id_A, \fix_{A,B}^{(n)}(g \circ \langle \pi_{A,B}, f \rangle) \rangle \rangle
\]

Therefore,

\[
f \circ \langle \id_A, \fix_{A,B}(g \circ \langle \pi_{A,B}, f \rangle) \rangle \geq \fix_{A,C}(f \circ \langle \pi_{A,C}, g \rangle).
\]

On the other hand, we can show that

\[
f \circ \langle \id_A, \fix_{A,B}^{(n)}(g \circ \langle \pi_{A,B}, f \rangle) \rangle \leq \fix_{A,C}^{(n+1)}(f \circ \langle \pi_{A,C}, g \rangle)
\]

by induction on \( n \). Hence, the operator \( \fix \) is dinatural:

\[
f \circ \langle \id_A, \fix_{A,B}(g \circ \langle \pi_{A,B}, f \rangle) \rangle = \fix_{A,C}(f \circ \langle \pi_{A,C}, g \rangle).
\]

For a \( C \)-morphism \( f : A \times B \to B \),

\[
f \circ \langle \pi_{A,B}, \pi_{A',B}, \pi_{A,B}' \rangle \circ \langle \id_A, \fix_{A,B}(g \circ \langle \pi_{A,B}, f \rangle) \rangle = f \circ \langle \id_A, \fix_{A,B}(g \circ \langle \pi_{A,B}, f \rangle) \rangle,
\]

\[
\fix_{A,B}(g \circ \langle \pi_{A,B}, f \rangle) \circ \langle \id_A, \fix_{A,B}(g \circ \langle \pi_{A,B}, f \rangle) \rangle
\]

\[
= \fix_{A,B}(g \circ \langle \pi_{A,B}, f \rangle).
\]

Therefore,

\[
\fix_{A,B}(f \circ \langle \pi_{A,B}, \pi_{A',B}, \pi_{A,B}' \rangle) \leq \fix_{A,B}(f \circ \langle \pi_{A,B}, f \rangle).
\]

On the other hand, we can show that

\[
\fix_{A,B}^{(n)}(f \circ \langle \pi_{A,B}, f \rangle) \circ \langle \id_A, \fix_{A,B}(g \circ \langle \pi_{A,B}, f \rangle) \rangle \leq \fix_{A,B}(f \circ \langle \pi_{A,B}, f \rangle)
\]

by induction on \( n \). Hence, \( \fix \) satisfies diagonal property:

\[
\fix_{A,B}(f \circ \langle \pi_{A,B}, \pi_{A',B}, \pi_{A,B}' \rangle) = \fix_{A,B}(f \circ \langle \pi_{A,B}, f \rangle).
\]

We used left strictness in naturality and diagonal property.

**Lemma A.2 (Lemma IV.3).** If the Kleisli category \( \text{Set}_T \) is a Ccppo-enriched cocartesian category and the bottom morphisms \( \perp_{A,B} : A \to B \) satisfy the following conditions:

- \( f \circ \perp_{A,B} = \perp_{A',B} \) for all \( f : B \to B' \)
- \( \perp_{A,B} \circ g^* = \perp_{A',B} \) for all \( g^* : A' \to A \)

then \( \text{Set}_T \) satisfies Requirement IV.2.

**Proof:** By the dual of Lemma A.1 and the bijective correspondence between Conway operators and trace operators, we see that \( \text{Set}_T \) has a trace operator: for \( f : A + C \to B + C \), we define \( \gamma_{C,B}(f) : A \to B \) by

\[
\gamma_{C,B}(f) = \bigvee_{n \geq 1} \gamma_{C,B}^{(n)}(f)
\]

where \( \gamma_{C,B}(f) : C \to B \) is given by

\[
\gamma_{C,B}^{(n)}(f) = \perp_{C,B}
\]

where \( \perp_{C,B} : C \to B \) is given by

\[
\gamma_{C,B}(f) = \bigvee_{n \geq 1} \gamma_{C,B}^{(n)}(f)
\]

\[
\gamma_{C,B}^{(n+1)}(f) = \gamma_{B} \circ (B + \text{iter}_{C,B}^{(n)}(f)) \circ f \circ \gamma_{A,C}.
\]

Therefore,

\[
\gamma_{C,B}(f) = \gamma_{A,C}^* \circ f \circ \gamma_{A,C}^*.
\]

For \( f : A + C \to B + C, g : A + D \to B + D \) and \( h : C \to D \), if \( (B + h^*) \circ f = g \circ T (A + h^*) \), then we can show

\[
\text{iter}_{D,B}(g) \circ \gamma_{A,C}^* = \text{iter}_{C,B}(f)
\]
by induction on \( n \). Then
\[
\begin{align*}
\text{tr}_{A,B}^C(f) &= \gamma_B^* \circ (B + \text{iter}_{C,B}(f)) \circ T f \circ \text{tr}^A \circ \text{inl}^*_A,
= \gamma_B^* \circ (B + (\text{iter}_{D,B}(g) \circ h^*)) \circ T f \circ \text{tr}^A \circ \text{inl}^*_A,
= \text{iter}_{D,B}(g) \circ T g \circ \text{tr}^A \circ \text{inl}^*_A,
= \text{tr}_{D,B}^A(g).
\end{align*}
\]
Hence, \((\text{Set}_T, +, \emptyset)\) satisfies Requirement IV.2.

**B. Proofs in Section V**

**Lemma A.3.** For a \(\text{Set}_T\)-morphism \( f : A + C \rightarrow B + C \) and a set \( D \), it holds that
\[
\text{tr}_{D \times A \times D \times B}(g) = (D \otimes \text{tr}_{A,B}^C(f)) \circ (d \times A).
\]
where \( g = (\delta_{D,B,C})^{-1} \circ T (D \otimes f) \circ \delta_{D,A,C} \).

**Proof:** For all maps \( d : 1 \rightarrow D \), we have
\[
g \circ (d \times A + d \times C) = T(d \times B + d \times C) \circ f.
\]
By uniformity, we obtain
\[
\text{tr}_{D \times A \times D \times B}(g) \circ (d \times A) = (D \otimes \text{tr}_{A,B}^C(f)) \circ (d \times A).
\]
Since \(\text{Set}\) is well-pointed, \(\text{tr}_{D \times A \times D \times B}(g)\) is equal to \(D \otimes \text{tr}_{A,B}^C(f)\).

**Proposition A.4.** The traced symmetric monoidal category \((\text{Res}(T), \boxplus, \emptyset, \text{Tr})\) and the GoI situation
\[
\text{(Res}(T), \mathbb{N}, F, J_\emptyset \phi, J_\emptyset \psi, J_\emptyset u, J_\emptyset v)
\]
are well-defined.

**Proof:** It is straightforward to check that the composition \(\circ\), the monoidal product \(\boxplus\), the functor \(F\) are compatible with the behavioral equivalence. We can show that \(\text{Tr}\) is compatible with the behavioral equivalence by uniformity of the trace operator \(\text{tr}\). It is easy to check that \(\text{Res}(T)\) forms a symmetric monoidal category. Yanking, exchange and superposing of \(\text{Tr}\) follow from these of \(\text{tr}\). Tightening follows from tightening of \(\text{tr}\) and Lemma A.3. It is straightforward to check naturality of retractions in Section IV-B4. It follows from Lemma A.3 and uniformity of \(\text{tr}\) that \(F\) is a traced symmetric monoidal functor.

**Theorem A.5** (Theorem V.3). Let \(\alpha\) be an \(I\)-ary algebraic operation on \(T\). The operator \(\overline{\alpha}\) is natural, and \(\text{tr}\) distributes over \(\overline{\alpha}\) modulo the behavioral equivalence:

- For all \(T\)-transducers \(\{X, c, x\} : A \rightarrow B\) for all families of \(T\)-transducers \(\{(Y, d, y) : A \rightarrow B\}_{i \in I}\) and for all maps \(h : A' \rightarrow A\), it holds that
\[
\overline{\alpha}_{A'B'}(\{Y, d, y\}_i, h)_i \circ J_0 h = \overline{\text{tr}}_{A'B'}(\overline{\alpha}_{A'B'}(\{X, c, x\} \circ (Y, d, y)) \circ J_0 h)_i.
\]
- For all families of \(T\)-transducers \(\{(X_i, c_i, x_i) : A' \rightarrow B'\}_{i \in I}\) from \(A + C\) to \(B + C\), it holds that
\[
\overline{\text{tr}}_{A'B'}(\overline{\alpha}_{A' + C'B'}(X_i, c_i, x_i) \circ J_0 h)_i \circ J_0 h = \overline{\text{tr}}_{A'B'}(\overline{\alpha}_{A'B'}(X_i, c_i, x_i) \circ J_0 h)_i.
\]

**Proof:** Let \(\alpha\) be an \(I\)-ary algebraic operation on \(T\). It is easy to check that a map
\[
x + X \times \prod_{i \in I} Y_i : 1 + X \times \prod_{i \in I} Y_i \rightarrow X \times \left(1 + \prod_{i \in I} Y_i\right)
\]
is a homomorphism from \(\overline{\alpha}_{A'B'}(\{X, c, x\} \circ (Y, d, y)) \circ J_0 h\).

We prove the second equivalence using string diagrams in the traced symmetric monoidal category \((\text{Set}_T, +, \emptyset, \text{tr})\). For a family of \(T\)-transducers \(\{(X_i, c_i, x_i) : A \rightarrow B + C\}_{i \in I}\), the \(T\)-transducer
\[
\text{Tr}_{A'B'}(\overline{\alpha}_{A' + C'B'}(X_i, c_i, x_i) \circ J_0 h)_i
\]
is given by
\[
(1 + Y, d, \text{inl}_i, Y) : A \rightarrow B
\]
where \(Y\) is the coproduct
\[
Y = \prod_{i \in I} X_i \cdot \text{inj}_i \cdot X_i
\]
and \(d\) is given by the following string diagram:

![String Diagram](image)

where
- The links of the form \(\triangleright\) are the codiagonal morphisms.
- The black circles are unique \(\text{Set}_T\)-morphisms from the empty set.
- \(\xi : A + C \rightarrow T Y \times B + Y \times C\)
\[
(\delta_{Y,B,C})^{-1} \circ T \alpha_{A + C, Y \times (B + C)}
\]
\[
\{(\text{inj}^*_i \circ (B + C)) \circ c_i \circ T x_i \otimes (A + C))_{i \in I}\}.
\]
- \(\chi : Y \times A + Y \times C \rightarrow T Y \times B + Y \times C\)
\[
\theta_{B,C} \circ T \left(\prod_{1 \leq i \leq n} c_i\right)
\]
\[
\otimes (\text{inj}^*_i \circ T x_i) \otimes (A + C).
\]

where \(\theta_{A,B}\) is the following canonical isomorphism
\[
\theta_{A,B} : \prod_{i \in I} (X_i \times (A + B)) \cong_{\text{tr}} T Y \times A + Y \times B.
\]

By the axioms of trace operator, \(d\) is equal to

![String Diagram](image)

By naturality of the algebraic operation \(\alpha\), this is equal to
Let \( Y \times A \) be a formal expression obtained by replacing all \( x_i \) in \( e \) by \( a_i \). We write \( \lambda x. e \) for \( \lambda x_1 \ldots x_n. e[x_1/\lambda x_1, \ldots, x_n/\lambda x_n] \). The following lemmas are basic in realizability arguments. For proofs of Lemma A.6 and Lemma A.8, see [8] and [28].

**Lemma A.6 (Combinatory Completenss).** For every formal expression \( e(x, x_1, x_2, \ldots, x_n) \) generated by

- variables \( x, x_1, x_2, \ldots, x_n \)
- elements in \( \text{Res}(T)(N, N) \)
- binary application symbols \((-) \cdot (-)\) such that

\[
((\lambda x. e(x, x_1, \ldots, x_n))[a_1/x_1, \ldots, a_n/x_n]) \cdot a
\]

for all \( a, a_1, \ldots, a_n \in \text{Res}(T)(N, N) \) where \( e(a, a_1, \ldots, a_n) \) is an element in \( \text{Res}(T)(N, N) \) obtained by replacing all \( x_i \) in \( e \) by \( a_i \). We write \( \lambda x. y \cdot z. e \) for \( \lambda y_1 \ldots y_n. \lambda z_1 \ldots z_n. e[x_1/\lambda y_1, \ldots, x_n/\lambda y_n, z_1/\lambda z_1, \ldots, z_n/\lambda z_n] \).

**Definition A.7.** Let \( e(x, x_1, x_2, \ldots, x_n) \) be a formal expression generated by

- variables \( x, x_1, x_2, \ldots, x_n \)
- elements in \( \text{Res}(T)(N, N) \)

we say that \( x \) appears **linearly** in \( e \) when \( x \) occurs exactly once in \( e \) and \( e \) is not of the form \((\ldots \bullet \ldots \cdot \ldots)\).

**Lemma A.8 (Linear Combinatory Completeness).** For every formal expression \( e(x, x_1, x_2, \ldots, x_n) \) generated by

- variables \( x, x_1, x_2, \ldots, x_n \)
- elements in \( \text{Res}(T)(N, N) \)
- binary application symbols \((-) \bullet (-) \) and \((-) \cdot (-)\) such that

\[
((\lambda x. e(x, x_1, \ldots, x_n))[a_1/x_1, \ldots, a_n/x_n]) \cdot a
\]

for all \( a, a_1, \ldots, a_n \in \text{Res}(T)(N, N) \) where \( e(a, a_1, \ldots, a_n) \) is an element in \( \text{Res}(T)(N, N) \) obtained by replacing all \( x_i \) in \( e \) by \( a_i \).

We define a category \( \text{Per}(T) \) by:

- An object is a partial equivalence relation (per) on \( \text{Res}(T)(N, N) \).
- A \( \text{Per}(T) \)-morphism from \( R \) to \( S \) is an equivalence class of the following pair:

\[
R \Rightarrow S = \{ (a, a') \mid \forall (b, b') \in R. (a \bullet b, a' \bullet b') \in S \}.
\]

For a \( \text{Per}(T) \)-morphism \( f : R \rightarrow S \), a **realizer** of \( f \) is a representative of \( f \). If \( r \) is a realizer of \( f \), then we say that \( r \) **realizes** \( f \). When \( r \) realizes a \( \text{Per}(T) \)-morphism from \( R \) to \( S \), we write \( [r] : R \rightarrow S \) for the morphism from \( R \) to \( S \) realized by \( r \).

**Proposition A.9 ([9]).** \( \text{Per}(T) \) is a bicartesian closed category with finite limits.

We describe the bicartesian closed structure of \( \text{Per}(T) \):

- The terminal object is \( \{(\lambda x. x, \lambda x. x)\} \).
- The initial object is the empty set.
- The cartesian product \( R \times S \) is

\[
\{(a, a') \mid (a \bullet T, a' \bullet T) \in R \wedge (b \bullet F, b' \bullet F) \in S \}
\]

where \( T = \lambda x y. z. F \), and \( F = \lambda x y. z \).
- The exponential from \( R \) to \( S \) is \( R \Rightarrow S \).
- The coproduct \( R + S \) is

\[
\{(a, a') \mid a \bullet T = a' \bullet T \wedge (a \bullet F, a' \bullet F) \in R \}
\]

\[
\cup \{(b, b') \mid b \bullet T = b' \bullet T \in F \wedge (b \bullet F, b' \bullet F) \in S \}.
\]

- The equalizer of \( \text{Per}(T) \)-morphisms \( f, f' : R \rightarrow S \) is

\[
m : \{(a, a') \in R \mid (r \bullet a, r' \bullet a') \in S \} \rightarrow S
\]

where \( r \) is a realizer of \( f \) and \( r' \) is a realizer of \( f' \). The \( \text{Per}(T) \)-morphism \( m \) is realized by \( \lambda x. x \).
**Proposition A.10.** \(\text{Per}(T)\) has countable products and countable coproducts.

**Proof:** Since \(\text{Per}(T)\) has finite limits and colimits, we give countably infinite products and coproducts. Let 
\[
\{(X_i, c_i, x_i)\}_{i \in \mathbb{N}}
\]
be a family of \(T\)-transducers from \(\mathbb{N}\) to \(\mathbb{N}\). For \(n \in \mathbb{N}\), we write \(\xi_n\) for the following canonical bijection 
\[
\xi_n : \prod_{i \in \mathbb{N}} X_n \rightarrow \left( \prod_{i \in \mathbb{N}\setminus\{n\}} X_i \right) \times X_n.
\]
We define a \(T\)-transducer \((X_i, c_i, x_i)_{i \in \mathbb{N}} : \mathbb{N} \rightarrow \mathbb{N}\) to be 
\[
\left( \prod_{i \in \mathbb{N}} X_i, d, \prod_{i \in \mathbb{N}} x_i \right)
\]
where \(d : \left(\prod_{i \in \mathbb{N}} X_i\right) \times \mathbb{N} \rightarrow T \left(\prod_{i \in \mathbb{N}} X_i\right) \times \mathbb{N}\) is a unique morphism such that \(d \circ_T \left(\prod_{i \in \mathbb{N}\setminus\{n\}} X_i \otimes \kappa^*_{(m,n)}\right)\) is equal to 
\[
\left(\xi_n, ((\xi_n^*)^{-1} \otimes (a^* \circ_T (n^* \otimes \mathbb{N}))) \circ_T \left(\prod_{i \in \mathbb{N}\setminus\{m\}} X_i \otimes c_m\right) \otimes (\xi_m^* \otimes \kappa^*_{(m,n)})\right).
\]
The construction 
\[
\{(X_i, c_i, x_i) : \mathbb{N} \rightarrow \mathbb{N}\}_{i \in \mathbb{N}} \mapsto (X_i, c_i, x_i)_{i \in \mathbb{N}} : \mathbb{N} \rightarrow \mathbb{N}
\]
is compatible with the behavioral equivalence. We define 
\[
\mathcal{M} : \text{Res}(T)(\mathbb{N}, \mathbb{N})^\mathbb{N} \rightarrow \text{Res}(T)(\mathbb{N}, \mathbb{N})
\]
to be the induced map. For \(n \in \mathbb{N}\), we define a map \(p_n : \mathbb{N} + \mathbb{N} \rightarrow \mathbb{N} + \mathbb{N}\) by 
\[
p_n(\text{inl}N,n(m)) = \text{inl}N,N(0, \langle n, m \rangle))
\]
\[
p_n(\text{inr}N,N((k, \langle l, m \rangle))) = \text{inl}N,N(m).
\]
We define \(P_n \in \text{Res}(T)(\mathbb{N}, \mathbb{N})\) to be the equivalence class of \(J_0(\phi \circ p_n \circ \psi)\). It is straightforward to check that \(P_n\) satisfies 
\[
P_n \circ M(a_i)_{i \in \mathbb{N}} = a_n.
\]
For a family of \(\text{Per}(T)\)-objects \(\{R_i\}_{i \in \mathbb{N}}\), we define a \(\text{Per}(T)\)-object \(\prod_{i \in \mathbb{N}} R_i\) to be 
\[
\{(a, a') | \forall i \in \mathbb{N}, (a \circ P_i, a' \circ P_i) \in R_i\}.
\]
The object \(\prod_{i \in \mathbb{N}} R_i\) together with projections 
\[
\left\{\lambda x : x \circ P_n : \prod_{i \in \mathbb{N}} R_i \rightarrow R_n \right\}_{n \in \mathbb{N}}
\]
forms a product of the family \(\{R_i\}_{i \in \mathbb{N}}\). Given a family of \(\text{Per}(T)\)-morphisms \(\{f_i : S \rightarrow R_i\}_{i \in \mathbb{N}}\), the tupling \(g : S \rightarrow \prod_{i \in \mathbb{N}} R_i\) is the morphism realized by 
\[
\lambda x : k \cdot (M(r_i)_{i \in \mathbb{N}}) \cdot x
\]
where \(r_i\) is a realizer of \(f_i\). We define a \(\text{Per}(T)\)-object \(\prod_{i \in \mathbb{N}} R_i\) to be 
\[
\bigcup_{i \in \mathbb{N}} \{(a, a') | a \circ T = a' \circ T = P_i \land (a \circ F, a' \circ F) \in R_i\}.
\]
The object \(\prod_{i \in \mathbb{N}} R_i\) together with injections 
\[
\{[\lambda x : k \cdot P_n \cdot x] : R_n \rightarrow \prod_{i \in \mathbb{N}} R_i \}_{n \in \mathbb{N}}
\]
forms a coproduct of the family \(\{R_i\}_{i \in \mathbb{N}}\). Let \(\{f_i : R_i \rightarrow S\}_{i \in \mathbb{N}}\) be a family of \(\text{Per}(T)\)-morphisms. The tupling \(g : \prod_{i \in \mathbb{N}} R_i \rightarrow S\) is the morphism realized by 
\[
\lambda x : T \cdot (M(r_i)_{i \in \mathbb{N}}) \cdot (x \circ F)
\]
where \(r_i\) is a realizer of \(f_i\).

Since \(\text{Per}(T)\) has countable coproducts, \(\text{Per}(T)\) has a natural number object. We explicitly describe a natural number object in \(\text{Per}(T)\). For a natural number \(n\), we define \(K_n \in \text{Res}(T)(\mathbb{N}, \mathbb{N})\) to be the equivalence class of \(J_0K_n\) where the map \(k_n : \mathbb{N} \rightarrow \mathbb{N}\) is given by \(k_n(m) = n\).

**Lemma A.11.** A \(\text{Per}(T)\)-object \(N = \{(K_n, K_n) | n \in \mathbb{N}\}\) is a natural number object.

**Proof:** As shown in [9], the following \(\text{Per}(T)\)-object is a natural number object: 
\[
N' = \{(K_n', K_n') | n \in \mathbb{N}\}
\]
where 
\[
K_n' = \lambda x : n \cdot \overline{\underbrace{\cdot \cdot \cdot x}_n}.
\]
We define \(s : \mathbb{N} + \mathbb{N} \rightarrow \mathbb{N} + \mathbb{N}\) by 
\[
s(\text{inl}N,N(n)) = \text{inr}N,N(0, n)
\]
\[
s(\text{inr}N,N(n, m)) = \text{inl}N,N(m + 1).
\]
Let \(\text{suc} \in \text{Res}(T)(\mathbb{N}, \mathbb{N})\) be the equivalence class of \(J_0(\phi \circ \psi)\). Then \(K_n' \circ \text{suc}\circ K_0\) is equal to \(K_n'\). Hence, \(\lambda n : \text{suc} \circ K_0\) realizes a \(\text{Per}(T)\)-morphism from \(N'\) to \(N\). Let 
\[
(X_n, c_n, x_n) : \mathbb{N} \rightarrow \mathbb{N}
\]
be a \(T\)-transducer that represents \(K_n'\). By the proof of combinatory completeness and by the definition of the GoI situation 
\[
(\text{Res}(T), \mathbb{N}, F, [J_0\phi], [J_0\psi], [J_0u], [J_0v]),
\]
we can assume that \(X_n = \{\ast\}\) and \(x_n = \ast\). Let \(c_n' : \mathbb{N} \rightarrow \mathbb{N}\) be a map such that \((\ast, c_n'(m)) = c_n(\ast, m)\). We define a map 
\[
d: \mathbb{N} \rightarrow \mathbb{N}
\]
by 
\[
d(0) = d(n, 0)
\]
\[
d(n + 1) = d(n, m).
\]
Then \(J_0d\) realizes a \(\text{Per}(T)\)-morphism from \(N\) to \(N'\). It is easy to check that \(J_0d\) is the inverse of the \(\text{Per}(T)\)-morphism realized by \(\lambda n : \text{suc} \circ K_0\).
2) **Strong Monad on \(\text{Per}(T)\):**

**Definition A.12.** We say that a \(\text{Per}(T)\)-object \(R\) is closed when the following statement

\[
\forall \alpha : \text{algebraic operation on } T, \forall \{(a_i, a'_i) \in R\}_{i \in \text{ary}(\alpha)}, \quad \{\pi_{\text{R,N}}(a_i), \pi_{\text{R,N}}(a'_i)\}_{i \in \text{ary}(\alpha)} \in R
\]

is true.

Let \(R\) be a closed \(\text{Per}(T)\)-object, and let \(\alpha\) be an \(I\)-ary algebraic operation on \(T\). Then \(\pi_{\text{R,N}}( \{q_{i,t} \}_{i \in I} )\) realizes a \(\text{Per}(T)\)-morphism

\[
[\pi_{\text{R,N}}( q_{i,t} )]_{i \in I} : R^I \to R
\]

where \(q_{i,t}\) is a realizer of the \(i\)-th projection from the \(I\)-fold product \(R^I\) to \(R\). The \(\text{Per}(T)\)-morphism \([\pi_{\text{R,N}}( q_{i,t} )]_{i \in I}\) is independent of our choice of realizers \(q_{i,t}\).

**Definition A.13.** For closed \(\text{Per}(T)\)-objects \(R\) and \(S\), a \(\text{Per}(T)\)-morphism \(f: R \to S\) is linear when the following diagram commutes for all algebraic operations \(\alpha\) on \(T\):

\[
\begin{array}{ccc}
R^I & \xrightarrow{f^I} & S^I \\
\downarrow{\pi_{\text{R,N}}( q_{i,t} )}_{i \in I} & & \downarrow{\pi_{\text{R,N}}( q_{i,t} )}_{i \in I} \\
R & \xrightarrow{f} & S
\end{array}
\]

where \(I\) is the arity of \(\alpha\).

We define \(\text{CPer}(T)\) to be the \(\text{Per}(T)\)-enriched category consisting of closed \(\text{Per}(T)\)-objects and hom-objects

\[
R \to S = \{(a,b) \in R \Rightarrow S \mid [\alpha](a) : R \to S \text{ is linear}\}.
\]

**Proposition A.14.** The inclusion functor \(U : \text{CPer}(T) \to \text{Per}(T)\) is a \(\text{Per}(T)\)-enriched right adjunction.

**Proof:** We define \(L \in \text{Res}(T)[\text{N,N}]\) to be \(\lambda x. \lambda^k.k \cdot x\). It is easy to see that the set of \(\text{CPer}(T)\)-objects is closed under small intersection. For a \(\text{Per}(T)\)-object \(R\), we define a \(\text{CPer}(T)\)-object \(\Phi R\) to be the least \(\text{CPer}(T)\)-object that includes the following \(\text{Per}(T)\)-object:

\[
R' = \{(L \cdot a, L \cdot a') \mid (a, a') \in R\}.
\]

Since \(\Phi R\) includes \(R'\), the combinator \(L\) realizes a \(\text{Per}(T)\)-morphism from \(R\) to \(\Phi R\). We show that \([L] : R \Rightarrow \Phi R\) is a unit of the right adjoint functor \(U\). Let \(\{r\} : R \to S\) be a \(\text{Per}(T)\)-morphism from \(R\) to a \(\text{CPer}(T)\)-object \(S\). By the right distributivity of \(\cdot\) over \(\pi_{\text{R,N}}\) for all algebraic operations \(\alpha\), the following \(\text{Per}(T)\)-object

\[
\{(a, a') \mid (a, r, a' \cdot r) \in S\}
\]

is closed and includes \(R'\). Therefore, by the definition of \(\Phi R\), we see that \(\lambda x.x \cdot r\) realizes a \(\text{Per}(T)\)-morphism from \(\Phi R\) to \(S\). Linearity of \(\lambda x.x \cdot r\) : \(\Phi R \to S\) follows from the right distributivity of \(\cdot\) over \(\pi_{\text{R,N}}\) for all algebraic operations \(\alpha\). Let \(f, g : \Phi R \to S\) be \(\text{CPer}(T)\)-morphisms such that

\[
\xrightarrow{[L]} : \Phi R \xrightarrow{f} S = \xrightarrow{R} \Phi R \xrightarrow{g} S.
\]

Let \(e : R'' \to \Phi R\) be an equalizer of \(f\) and \(g\). We can assume that \(e\) is realized by \(\lambda x.x\) and that \(R''\) is a subset of \(\Phi R\). For any \(I\)-ary algebraic operation \(\alpha\) on \(T\), we have the following diagram:

\[
\begin{array}{ccc}
R'' & \xrightarrow{e} & (\Phi R)^I \\
\downarrow{\pi_{\text{R,N}}( q_{i,t} )}_{i \in I} & & \downarrow{\pi_{\text{R,N}}( q_{i,t} )}_{i \in I} \\
R & \xrightarrow{f} & S
\end{array}
\]

Since \([\pi_{\text{R,N}}( q_{i,t} )]_{i \in I}\) is realized by \(\pi_{\text{R,N}}( q_{i,t} )\), there exists a unique \(\text{Per}(T)\)-morphism \(h : R'' \to R''\) realized by \(\pi_{\text{R,N}}( q_{i,t} )\). Hence, \(e\) is linear. By the definition of \(\Phi R\), we see that \(R'' = \Phi R\). Since \(e\) equals \(f\) and \(g\), we obtain \(f = g\). We have shown that the \(\text{Per}(T)\)-object \(R \Rightarrow S\) is isomorphic to \(\Phi R \Rightarrow S\), and the isomorphism is natural in \(S\). Hence, \(U\) is \(\text{Per}(T)\)-enriched right adjunction.

Since \(U\) is a \(\text{Per}(T)\)-enriched right adjoint functor on \(\text{Per}(T)\), the adjunction induces a strong monad \(\Phi\) on \(\text{Per}(T)\).

**Lemma A.15.** Let \(\alpha\) be an \(I\)-ary algebraic operation on \(T\). The family of maps

\[
\alpha_{L,R}: (S \Rightarrow \Phi R)^I \to \Phi R
\]

realized by \(\pi_{\text{R,N}}( q_{i,t} )\) is an algebraic operation on \(\Phi\). We note that \(S \Rightarrow \Phi R\) is closed.

**Proof:** For any \(\text{Per}(T)\)-morphisms \(f : S' \to S\) and \(g : R \Rightarrow \Phi R'\), the following \(\text{Per}(T)\)-morphism

\[
f \Rightarrow g' : \Phi R \Rightarrow S' \Rightarrow \Phi R'
\]

is linear where \(g' : \Phi R \Rightarrow \Phi R'\) is the Kleisli lifting of \(g\). Since \(\text{Per}(T)\) is well-pointed, \(\alpha_{L,R}\) is an \(I\)-ary algebraic operation on \(\Phi\).

**Lemma A.16.** Let \(\text{proj}_{i,AB} : (A \Rightarrow TB)^I \to (A \Rightarrow TB)\) be the \(i\)-th projection. Then \(\text{proj}_{i,R,S}\) is the \(i\)-th injection.

**Proof:** Let \(\{(X_i, c_i, x_i) : A \Rightarrow B\}_{i \in I}\) be a family of \(T\)-transducers. We write \(\text{inj}_i : X_i \to \coprod_{i \in I} X_i\) for the \(i\)-th injection. The injection

\[
1 + \text{inj}_i : 1 + X_i \to 1 + \coprod_{i \in I} X_i
\]

is a homomorphism from \((1 + X_i, c'_i, \text{inl}_i, X_i)\) to

\[
\text{proj}_{i,AB}(\{(X_i, c_i, x_i)\}_{i \in I})
\]

where \(c'_i\) is a unique \(\text{Set}_{T}\)-morphism from \(1 + X_i\) to \((1 + X_i) \times B\) such that

\[
\begin{align*}
\text{proj}_{i} (\text{inl}_i, X_i) \otimes A &= (\text{inr}_i, X_i) \otimes B) \otimes c_i \otimes (x' \otimes A) \\
\text{proj}_{i} (\text{inl}_i, X_i) \otimes A &= (\text{inr}_i, X_i) \otimes B) \otimes c_i.
\end{align*}
\]

On the other hand, since the cotuple \(\{x_i, \text{id}_{1X_i}\} : 1 + X_i \to X_i\) is a homomorphism from \((1 + X_i, c'_i, \text{inl}_i, X_i)\) to \((X_i, c_i, x_i)\), the \(T\)-transducer

\[
\text{proj}_{i,AB}(\{(X_i, c_i, x_i)\}_{i \in I})
\]
is behaviorally equivalent to \((X_i, c_i, x_i)\). Hence, \(\text{proj}^\dagger_{R,S}\) is the morphism realized by \(Q_{i,I}\), which realizes the \(i\)-th projection. ■

Lemma A.17. For an \(I\)-ary algebraic operation \(\alpha\) and a family of \(J\)-ary algebraic operations \(\{\beta_i\}_{i \in I}\) on \(T\), we define a \(J\)-ary algebraic operation \(\gamma\) on \(T\) by

\[
\gamma\{f_j\}_{j \in J} = \alpha_{A,B}\{\beta_{iA,B}\{f_j\}_{j \in J}\}_{i \in I}
\]

where \(f_j\) are \(\text{Set}_T\)-morphisms from \(A\) to \(TB\). Then it holds that

\[
\gamma^\dagger_{R,S} = \alpha^\dagger_{R,S} \circ (\beta^\dagger_{iR,S})_{i \in I}
\]

where \((\beta^\dagger_{iR,S})_{i \in I}: (R \Rightarrow \Phi S)^J \rightarrow (R \Rightarrow \Phi S)^I\) is a unique \(\text{Per}(T)\)-morphism whose \(i\)-th projection is equal to \(\beta^\dagger_{iR,S}\).

Proof: Let \(\{(X_j, c_j, x_j)\}_{j \in J}\) be a family of \(T\)-transducers from \(A\) to \(B\). For \(i \in I\), we define a \(T\)-transducer

\[
(1 + Y, d_i, \text{inl}_{1,Y}): A \rightarrow B
\]

to be \(\overline{\gamma}_{iA,B}\{(X_j, c_j, x_j)\}_{j \in J}\) where

\[
Y = \prod_{j \in J} X_j.
\]

We define a \(T\)-transducer

\[
(1 + Z, e, \text{inl}_{1,Z}): A \rightarrow B
\]

to be \(\overline{\gamma}_{A,B}\{(1 + Y, d_i, \text{inl}_{1,Y})\}_{i \in I}\) where

\[
Z = \prod_{i \in I} (1 + Y) = \prod_{i \in I} \left(1 + \prod_{j \in J} X_j\right).
\]

We define a \(T\)-transducer

\[
(1 + W, f, \text{inl}_{1,W}): A \rightarrow B
\]

to be \(\overline{\gamma}_{A,B}\{(X_j, c_j, x_j)\}_{j \in J}\) where

\[
W = \prod_{j \in J} X_j.
\]

We define a \(T\)-transducer

\[
(1 + V, g, \text{inl}_{1,V}): A \rightarrow B
\]

by

\[
V = \prod_{i \in I} \prod_{j \in J} X_j \xleftarrow{\text{inj}_{i,j}} X_j
\]

where \(g: (1 + V) \otimes A \rightarrow_T (1 + V) \otimes B\) is a unique \(\text{Set}_T\)-morphism such that

\[
g \circ_T (\text{inl}_{1,V}^* \otimes A) = \alpha_{A, (1 + V) \otimes B}\{\text{inl}_{1,V}^* \otimes B\} \circ_T b_i \}_{i \in I}
\]

\[
g \circ_T ((\text{inl}_{1,V}^* \otimes \text{inj}_{i,j}^*) \otimes A) = ((\text{inl}_{1,V}^* \otimes \text{inj}_{i,j}^*) \otimes A) \circ_T c_j
\]

where \(b_i: A \rightarrow_T V \otimes B\) is

\[
\beta_{iA,V \otimes B}\{(\text{inj}_{i,j}^* \otimes B) \circ_T c_j \circ_T (x_i^* \otimes A)\}_{j \in J}.
\]

The injection

\[
1 + \prod_{i \in I} \text{inr}_{1,V} : 1 + V \rightarrow 1 + Z
\]

is a homomorphism from \((1 + V, g, \text{inl}_{1,V})\) to \((1 + Z, e, \text{inl}_{1,Z})\), and the map \(h: 1 + V \rightarrow 1 + W\) that is codiagonal on the right summand \(\prod_{j=1}^m X_j\) is a homomorphism from \((1 + V, g, \text{inl}_{1,V})\) to \((1 + W, f, \text{inl}_{1,W})\). Hence, we obtain

\[
\overline{\gamma}_{A,B}\{(X_j, c_j, x_j)\}_{j \in J} \xrightarrow{\beta^\dagger_{A,B}} \overline{\gamma}_{A,B}\{(X_j, c_j, x_j)\}_{j \in J}
\]

\[
\approx \overline{\gamma}_{A,B}\{(X_j, c_j, x_j)\}_{j \in J}
\]

\[
\Rightarrow \overline{\gamma}_{A,B}\{(X_j, c_j, x_j)\}_{j \in J} = \overline{\gamma}_{A,B}\{(X_j, c_j, x_j)\}_{j \in J}
\]

Then it is easy to check \(\gamma^\dagger_{R,S} \circ \overline{\gamma}_{A,B}\{(X_j, c_j, x_j)\}_{j \in J} = \overline{\gamma}_{A,B}\{(X_j, c_j, x_j)\}_{j \in J}\) for all \(\overline{\gamma}\)-morphism \(x: 1 \rightarrow (R \Rightarrow \Phi S)^I\). Since \(\overline{\gamma}\) is well-pointed, we obtain the statement. ■

Theorem A.18. The operation \((-)^\dagger\) is an identity-on-object countable-product-preserving faithful functor from \(\text{AlgOp}_T\) to \(\text{AlgOp}_P\).

Proof: By Lemma A.16 and Lemma A.17, we can extend \((-)^\dagger\) to an identity on object countable product preserving functor from \(\text{AlgOp}_T\) to \(\text{AlgOp}_P\). For \(I\)-ary algebraic operations \(\alpha\) and \(\beta\) on \(T\), we suppose that \(\alpha^\dagger\) is equal to \(\beta^\dagger\). Let \(\Delta\) be a \(\overline{\gamma}\)-object given by

\[
\Delta = \{(a, a) \mid a \in \text{Res}(T)(N, N)\},
\]

which is closed. For any family \(\{a_i \in \text{Res}(T)(N, N)\}_{i \in I}\),

\[
1, \prod_{i \in I} \text{inl}_{\text{Res}(T)(N, N), a_i} \rightarrow_T (\Phi \Delta)^I \xrightarrow{\alpha^\dagger_{I,\Delta}} \Phi \Delta \xrightarrow{[\lambda y. (\lambda x. y)]} \Delta
\]

is equal to

\[
1, \prod_{i \in I} \text{inl}_{\text{Res}(T)(N, N), a_i} \rightarrow_T (\Phi \Delta)^I \xrightarrow{\beta^\dagger_{I,\Delta}} \Phi \Delta \xrightarrow{[\lambda y. (\lambda x. y)]} \Delta.
\]

This equality means that

\[
\alpha_{N,N}\{(X_i, c_i, x_i)\}_{i \in I} \approx \beta_{N,N}\{(X_i, c_i, x_i)\}_{i \in I}
\]

for each family of \(T\)-transducers \(\{(X_i, c_i, x_i) : N \rightarrow N\}_{i \in I}\). In particular, for each family of \(\text{Set}_T\)-morphisms \(\{f_i : N \rightarrow_T N\}_{i \in I}\), we have

\[
\overline{\gamma}_{N,N}\{f_i\}_{i \in I} \approx \overline{\gamma}_{N,N}\{f_i\}_{i \in I}.
\]

Therefore, there exists a \(T\)-transducer \((Y, d, y) : N \rightarrow N\) and homomorphisms

\[
h : \overline{\gamma}_{N,N}\{f_i\}_{i \in I} \rightarrow (Y, d, y),
\]

\[
k : \beta_{N,N}\{f_i\}_{i \in I} \rightarrow (Y, d, y).
\]

We write \((Z, e, z)\) for \(\overline{\gamma}_{N,N}\{f_i\}_{i \in I}\) and \((Z', e', z')\) for \(\beta_{N,N}\{f_i\}_{i \in I}\). By the definition of \(\overline{\gamma}\), \(Z\) and \(Z'\) are equal to 1 + \(I\), and \(z\) and \(z'\) are the first injections. The \(\text{Set}_T\)-morphisms \(e\) and \(e'\) satisfy the following equations:

\[
\overline{T}_1^\dagger \otimes N \rightarrow_T (z^* \otimes N) = \alpha_{N,N}\{f_i\},
\]

\[
\overline{T}_1^\dagger \otimes N \rightarrow_T (z^* \otimes N) = \beta_{N,N}\{f_i\}.
\]
Since
\[ (T_1^1 + T \otimes N) \circ e \circ T \left( z^* \otimes N \right) = (T_1^1 + T \otimes N) \circ (h^* \otimes N) \circ e \circ (T \circ z^* \otimes N) = (T_1^1 + T \otimes N) \circ d \circ (h^* \otimes N) \circ (T \circ z^* \otimes N) = (T_1^1 + T \otimes N) \circ d \circ (y^* \otimes N) = (T_1^1 + T \otimes N) \circ d \circ (k^* \otimes N) \circ (T \circ z^* \otimes N) = (T_1^1 + T \otimes N) \circ e' \circ (T \circ z^* \otimes N), \]
we obtain \( \alpha_{MN}(\{f\}) = \beta_{MN}(\{f\}) \). By naturality of \( \alpha \) and \( \beta \), we have \( \alpha_{1,\text{ary}(\alpha)} = \beta_{1,\text{ary}(\beta)} \). Hence, \( \alpha = \beta \).

3) A Proof of Theorem VI.2: We first give the syntax and equational theory of an extension \( \mathcal{L} \) of the computational lambda calculus to algebraic effects and a base type \( \text{nat} \) of natural numbers. For treatment of algebraic effects, we employ generic effects.

We define types by the following BNF:

\[
\tau := \text{unit} | \text{nat} | \tau + \tau | \tau \times \tau | \tau \Rightarrow \tau.
\]

We define \( A \) for the subclass of types given by the following BNF:

\[
\beta := \text{unit} | \text{nat} | \beta + \beta | \beta \times \beta.
\]

We call types in \( A \) arity types.

A signature \( \Sigma \) is a set of triples

\[
(\text{gen}, \beta, \beta'),
\]
consisting of an operation symbol \( \text{gen} \) and arity types \( \beta \) and \( \beta' \). We define terms \( t \), values \( v \), effect terms \( e \) and evaluation contexts \( E \) by the following BNF:

\[
t := x \in \text{Var} | \ast | \text{c}_n \ (n \in \text{N}) | t + t | t t | \lambda x : \tau. t
\| \text{fst}(t) | \text{snd}(t) | (t, t) | \text{inl}_{\tau, \sigma} t | \text{inr}_{\tau, \sigma} t
\| \text{case}(t, t, t, y. t) | (\text{gen}(t))
\]

\[
e := x \in \text{Var} | \ast | \text{c}_n \ (n \in \text{N}) | e + e
\| \text{fst}(e) | \text{snd}(e) | (e, e) | \text{inl}_{\tau, \sigma} e | \text{inr}_{\tau, \sigma} e
\| \text{case}(e, e, e, y. e) | (\text{gen}(e))
\]

\[
v := x \in \text{Var} | \ast | \text{c}_n \ (n \in \text{N}) | v + v | \text{fst}(v) | \text{snd}(v)
\| (v, v) | \lambda x : \tau. t | \text{inl}_{\tau, \sigma} v | \text{inr}_{\tau, \sigma} v
\| \text{case}(v, v, v, v. v)
\]

\[
E := [\ ] | E + t | E + v | E \text{ v} | Et | \text{fst}(E) | \text{snd}(E)
\| (E, t) | (E, v) | \text{inl}_{\tau, \sigma} E | \text{inr}_{\tau, \sigma} E | \text{case}(E, t, t, t)
\| (\text{gen}(E))
\]

where \text{Var} is a (countable) set of variables. We will use effect terms to define the equational theory of \( \mathcal{L} \). We write \( \text{Fv}(t) \) for the set of free variables in \( t \) and \( \text{Fv}(E) \) for the set of free variables in \( E \) where the lambda abstraction and case introduce variable bindings. As usual, we identify terms modulo \( \alpha \)-equivalence, and for terms \( t \) and \( s \), we write \( t[s/x] \) for the term substitution in a capture-avoiding manner. We use let-notation as a syntactic sugar:

\[
\text{let } x : \tau \text{ be } t \text{ in } s = (\lambda x : \tau. s) t.
\]

Typing rules for the core fragment are standard. For \((\text{gen}, \beta, \beta') \in \Sigma\), we have the following typing rule:

\[
\Gamma \vdash t : \beta \quad \Gamma \vdash \text{gen}(t) : \beta'
\]

where \( \Gamma \) is a term environment; \( \Gamma \) is a finite sequence consisting of pairs of a variable and a type.

A theory \( T \) is a set of lists \((\Gamma, t, s, \beta)\) consisting of effect terms \( \Gamma \vdash e : \beta \) and \( \Gamma \vdash e' : \beta \) such that types that appear in \( \Gamma \), \( e \) and \( e' \) are arity types. When \((\Gamma, e, e')\) is in \( T \), we write \( \Gamma \vdash e \leadsto_{T} e' \) and \( \tau \). In Figure 1, we give axioms of the equational theory where we implicitly assume that all terms are well-typed under the term environments \( \Gamma \). The equational theory of \( \mathcal{L} \) consists of the axioms in Figure 1 and rules stating that \( \approx \) is a congruence.

Remark A.19. As observed in [16], we can regard the following term construction

\[
\Gamma \vdash t : \beta \quad \Gamma, x : \beta' \vdash s : \tau
\]

\[
\Gamma \vdash \text{op}(t, x : \beta'. s) : \tau
\]

as a syntactic sugar of \( \text{let } x : \beta' = \text{gen} t \text{ in } s \). On the other hand, the term construction \( \text{gen}(t) \) can be encoded as follows:

\[
\text{op}(t, x : \beta'. x).
\]

With respect to this correspondence, the syntax of \( \mathcal{L} \) is the same as the one given in [16].

Next, we give a categorical interpretation of \( \mathcal{L} \). We inductively define a set \( \beta \) for an arity type \( \beta \) by

\[
\begin{align*}
\text{unit} & = \{ \ast \} \\
\text{nat} & = \mathbb{N} \\
\beta + \beta' & = \beta + \beta' \\
\beta \times \beta' & = \beta \times \beta'.
\end{align*}
\]

We assume that \( T \) is a monad on \( \text{Set} \) such that
For each

\[ \text{We interpret the core fragment following [15].} \]

We interpret each \( t \) to effect terms: for an effect term

\[ x_1 : \beta_1 \cdots x_n : \beta_n \vdash e : \beta, \]

we define a \( \text{Set}_T \)-morphism

\[ e : \beta_1 \times \cdots \times \beta_n \to T \beta' \]

by the categorical interpretation of effect terms in \( \text{Set}_T \).

- For effect terms \( \Delta \vdash e : \beta \) and \( \Delta \vdash e' : \beta' \), if \( \Delta \vdash e \sim_T e' \), then \( e = e' \).

Let \( \Phi \) be the strong monad in Theorem VI.1. For a term

\[ x_1 : \tau_1, \ldots, x_n : \tau_n \vdash t : \tau \]

in \( \mathcal{L} \), we define a \( \text{Per}(T)_{\Phi} \)-morphism

\[ \llbracket t \rrbracket : \llbracket \tau_1 \rrbracket \times \cdots \times \llbracket \tau_n \rrbracket \to \llbracket \tau \rrbracket \]

to be the categorical interpretation of the term \( t \) in \( \text{Per}(T)_{\Phi} \) given as follows:

- We interpret the core fragment following [15].
- We interpret \( \text{nat} \) by the natural number object \( N \) of \( \text{Set}(T) \) (Lemma A.11).
- For \( (\text{gen}, \beta, \beta') \in \Sigma \), we interpret

\[ \Gamma \vdash \text{gen}(t) : \beta' \]

by

\[ \llbracket \Gamma \rrbracket \llbracket t \rrbracket \xymatrix{ \Phi[\llbracket \beta \rrbracket] \ar[r]^\alpha & \Phi[\llbracket \beta' \rrbracket] } \]

where the \( \text{Per}(T) \)-morphism \( \alpha : \Phi[\llbracket \beta \rrbracket] \to \Phi[\llbracket \beta' \rrbracket] \) is the Klei6si lifting of the \( \text{Per}(T)_{\Phi} \)-morphism from \( \llbracket \beta \rrbracket \) to \( \llbracket \beta' \rrbracket \) induced by \( \text{gen} : \beta \to T \beta' \).

Note that \( \text{AlgOp}(T) \) is isomorphic to the full subcategory of \( \text{Set}_T \) consisting of countable sets, and \( \text{AlgOp}(\Phi) \) is equivalent to the full subcategory of \( \text{Per}(T)_{\Phi} \) generated by \( 1 \) and countable products. The functor \( (-)^\dagger \) and the equivalences induce a countable-coproducts-preserving faithful functor from the subcategory of \( \text{Set}_T \) to \( \text{Per}(T)_{\Phi} \) that maps \( \beta \) to \( \llbracket \beta \rrbracket \).

Soundness of the categorical interpretation of \( \mathcal{L} \) follows from Theorem VI.1.

**Theorem A.20.** If \( \Gamma \vdash t \approx s : \tau \), then \( \llbracket t \rrbracket = \llbracket s \rrbracket \).

For a closed term \( t \) of type \( \tau \), the interpretation of the term \( t \) is a \( \text{Per}(T) \)-morphism from \( 1 \) to \( \Phi[\llbracket \tau \rrbracket] \), and therefore, the interpretation of the term \( t \) bijectively corresponds to an equivalence class of \( \Phi[\llbracket \tau \rrbracket] \). We define GoI interpretation \( \llbracket t \rrbracket \) so that the \( \text{Res}(T) \)-morphism represented by \( \llbracket t \rrbracket \) represents the equivalence class of \( \Phi[\llbracket \tau \rrbracket] \) that bijectively corresponds to \( \llbracket t \rrbracket \).

**Theorem A.21** (Theorem VI.2). For closed terms \( t \) and \( s \) of type \( \tau \) in \( \mathcal{L} \),

- If \( t \approx s \), then \( \llbracket (t) \rrbracket, \llbracket (s) \rrbracket \in \Theta[\llbracket \tau \rrbracket] \).
- If \( t \approx s \) and \( \tau \) is the base type \text{nat}, then \( \llbracket t \rrbracket \sim_{\text{Res}(T)} \llbracket s \rrbracket \).

Proof: The first clause follows from soundness of the categorical interpretation. Since the natural number object \( N \) of \( \text{Per}(T) \) is a subset of the following closed per:

\[ \Delta = \{ (a, a) \mid a \in \text{Res}(T)(N, N) \} \]

the per \( \Phi N \) is a subset of \( \Delta \). Hence, all equivalence classes of \( \Phi N \) are singletons, and if \( t \approx s \) and \( \tau \) is the base type \text{nat}, then \( \llbracket t \rrbracket \sim_{\text{Res}(T)} \llbracket s \rrbracket \).