

# Logic of computational semi-effects and categorical gluing for equivariant functors

Yuichi Nishiwaki

University of Tokyo, Japan  
nyuichi@is.s.u-tokyo.ac.jp

Toshiya Asai

University of Tokyo, Japan  
anori@is.s.u-tokyo.ac.jp

---

## Abstract

---

In this paper, we revisit Moggi’s celebrated calculus of computational effects from the perspective of logic of monoidal action (actegory). Our development takes the following steps. Firstly, we perform proof-theoretic reconstruction of Moggi’s computational metalanguage and obtain a type theory with a modal type  $\triangleright$  as a refinement. Through the proposition-as-type paradigm, its logic can be seen as a decomposition of lax logic via Benton’s adjoint calculus. This calculus models as a programming language a weaker version of effects, which we call *semi-effects*. Secondly, we give its semantics using actegories and equivariant functors. Compared to previous studies of effects and actegories, our approach is more general in that models are directly given by equivariant functors, which include Freyd categories (hence strong monads) as a special case. Thirdly, we show that categorical gluing along equivariant functors is possible and derive logical predicates for  $\triangleright$ -modality. We also show that this gluing, under a natural assumption, gives rise to logical predicates that coincide with those derived by Katsumata’s categorical  $\top\top$ -lifting for Moggi’s metalanguage.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Categorical semantics

**Keywords and phrases** computational effects, actegories, logical relations, categorical gluing, proof theoretic semantics, modal logic, Curry-Howard correspondence, fibrations

## 1 Introduction

It has passed about three decades since the deep connection between the notion of computation and monads in category theory was revealed by Moggi [29, 30]. Moggi’s papers have been not only affecting the design of most modern programming languages but also standing as a very foundation of the semantic analysis of the notion of computation. His insight that computation can be modeled by monads is now widely accepted and sometimes even considered as “general knowledge” in the community. In this paper, we revisit this prevalent slogan “computation as monads” with a somewhat critical eye, and attempt to propose our new alternative: “computation as monoidal actions”.

One important contribution in Moggi’s papers was the suggestion of a formal system of equational logic (called *the metalanguage*) that can uniformly represent various kinds of computation where the type of involved effect is given as a parameter. Once the papers’ importance was recognized, the metalanguage began to get analyzed with the help of logic. Lax logic [7] (or CL logic [2]) is a modal logic with a single modality  $\diamond$  representing intuitionistic possibility. Benton, Bierman, and de Paiva [2] showed that proof-term assignment to lax logic directly gives the Curry-Howard correspondence to the metalanguage. Benton and Wadler [3] showed that adjoint calculus [1] can serve as a logical foundation of the metalanguage provided that the underlying monad is commutative. Enriched Effect Calculus (EEC) [6] pushed forward this direction. EEC removed the limitation of the class of monads from adjoint calculus by carefully choosing the set of legitimate logical connectives and forms of typing judgments. Along this series of works, this paper presents another reformulation of

the metalanguage. Our reformulation starts by analyzing the above logics from the proof-theoretic viewpoint, motivated by the fact that none of them enjoy *stability* à la Dummett, a proof-theoretic criterion for “nice” logics [5, 20]. We derive our logic (and its corresponding type theory) in the following steps. Firstly, we decompose lax logic into a logic with two adjoint modalities  $\triangleleft$  and  $\triangleright$  exploiting the technique used by adjoint calculus. Secondly, to maintain the non-commutativity of the metalanguage, we restrict the number of possible variables in the realm of computation to at most one. At the final step, we throw away the  $\triangleleft$  modality. The resulted calculus is a rather weak system due to the lack of an adjoint modality. Nevertheless, it enjoys nice proof-theoretic properties, maintains the essence of computation, and has a simple categorical model. We call this calculus *semi-effect calculus (SEC)*, after its operational behavior.

Proposal of strong monads as a categorical semantics of computation is also a significant contribution of Moggi’s papers. Along this line, many further studies have been done so far [33, 9, 37]. For example, Møgelberg and Staton [28] used copowers in enriched categories to interpret elaborated connections between values and computations, and Levy [21] used Freyd categories. Although strong monad, Freyd category, and copower have a strong connection with *monoidal action (actegory)*, actegory itself had not been treated as a “first-class citizen” as a model of effects. Interestingly, because the expressive power is restricted enough, SEC can be directly modeled by equivariant functors (morphisms between actegories). While only the soundness holds in our semantics (i.e. the completeness result presumably fails due to its “lax” nature), this semantics allows us to perform the term model construction.

Categorical gluing (also called *sconing* or *Freyd cover* in the literature) is a method to create a structure satisfying a certain categorical notion from a morphism preserving structures to which the notion is related. Although categorical gluing is not always possible for every type of categorical structure, it is well known that cartesian closed structure (the structure of simply-typed lambda calculus) admits such gluing construction, with which one can prove some syntactic properties (e.g. conservativity) of the lambda calculus [4, 27]. Moreover, categorical gluing is closely related to logical predicates and logical relations [27, 11]. As a corollary of the present work, we prove that gluing along equivariant functors (*actegorical gluing*) is indeed possible, and derive logical predicates for SEC. We also show that in some typical cases, actegorical gluing can derive the same logical predicates constructed by  $\mathbb{T}\mathbb{T}$ -lifting for the metalanguage [22, 16].

Our contributions summarize as follows:

- We present a proof-theoretic reformulation of intuitionistic possibility modality.
- We propose a new calculus (SEC) capturing a weaker notion of computation (semi-effects).
- We show that SEC is modeled by actegories and equivariant functors.
- We prove that categorical gluing along equivariant functors is possible, and show that the  $\mathbb{T}\mathbb{T}$ -lifting of strong monads is reducible to this gluing in some typical cases.

## Construction of the paper

Section 2 presents related work. In Section 3, we recall Moggi’s metalanguage and introduce our calculus SEC. Their syntactic definitions and logical properties are discussed. In Section 4, we present some basics of actegories and equivariant functors, and give categorical models for SEC. Section 5 describes categorical gluing along models of SEC and its connection with fibrations. As an application, we present certain flavors of logical predicates for SEC. Comparison with  $\mathbb{T}\mathbb{T}$ -lifting is also presented here. Section 6 concludes the paper and discusses future work.

## 2 Related work

**Proof-theoretic reconstruction of modal logic.** It has been a long-standing issue to find a nice proof-theoretic account of intuitionistic modal logics. Pfenning and Davies proposed a reformulation of intuitionistic modalities of both necessity  $\Box$  and possibility  $\Diamond$  [32]. In the presence of both two modalities, a clean categorical account by an  $\mathcal{L}$ -strong monad is possible [18]. For the necessity-only fragment, a more refined calculus based on stratification of the modality and its semantics based on iterated enrichment of categories are presented in [31, 15]. A brief survey of this field is found in [17].

**Categorical gluing.** Mitchell and Scedrov [27] pointed out that the classical fundamental lemma for logical predicates is obtained as the uniqueness of the morphism from the classifying category of simply-typed  $\lambda$ -calculus to a cartesian closed category constructed by gluing (or sconing). In [4], a conservativity proof of  $\lambda\times$ -calculus over equational logic of algebras is presented. The proof uses the gluing technique and exploits the universal property of naïve translation from an algebraic theory to a  $\lambda\times$ -theory. In [8] and [38], a normalization proof of simply-typed  $\lambda$ -calculus by gluing is presented. Whether these techniques can be adapted to our results remains unclear.

**Logical predicates.** Logical predicates (and logical relations) have been used to prove syntactic results for many calculi (e.g. [36]). Examples include the computational adequacy result of PCF [39]. Hermida [12] generalized the logical predicates for simply-typed  $\lambda$ -calculus to cartesian closed categories using the internal logic and (Grothendieck) fibrations. We extensively use the results from this work. In [11], Hasegawa showed that logical predicates for certain fragments of linear logic can be described by *subgluing* via fibrational arguments, from which some semantic results (i.e. the ability to obtain new models) for gluing and subgluing are derived. Our present work is similar to Hasegawa’s work. There are some categorical formulations of logical predicate for monadic computation. Our construction is closely related to [16] (see Subsection 5.4). On the other hand, the relationship between [10] and the present work remains unknown.

## 3 Semi-effect calculus

In this section, we introduce *Semi-Effect Calculus (SEC)*, which will be studied throughout the paper. SEC is obtained by careful analysis of Moggi’s metalanguage.

### 3.1 Preliminaries on logical harmony and stability

We recall some basic notions from proof-theoretic semantics (PTS). The materials in this subsection will be necessary to understand the construction in the next subsection.

PTS is an approach to investigate the meaning of a logical constant (connective) by means of the structural nature of the natural deduction system associated to the logic. Unlike traditional Tarski-style semantics, PTS is considered a rather informal, philosophically-motivated semantics. Nonetheless, PTS is supposed to help more conceptual understanding of logics and provide a criterion for designing a well-behaved natural deduction system and hence the corresponding term calculus.

*Logical harmony* (a la Dummett) in PTS is such a property that (it is expected that) every “meaningful” logical connective shall enjoy. We consider Prior’s *tonk* [34], which is an imaginary logical connective having the introduction rule (I-rule) of disjunction and the

elimination rule (E-rule) of conjunction.

$$\frac{\Gamma \vdash A_i}{\Gamma \vdash A_1 \text{ tonk } A_2} \text{I-tonk}_i \quad \frac{\Gamma \vdash A_1 \text{ tonk } A_2}{\Gamma \vdash A_i} \text{E-tonk}_i$$

Clearly, having tonk makes the logic syntactically inconsistent (i.e. proves everything). Some criteria have been proposed to answer why tonk is nonsense (and others are not). Prawitz’ *inversion principle* (e.g. [35]), a (candidate of) formulation of logical harmony, claims that an E-rule should not be “weaker” than the I-rule, in the sense that using the E-rule immediately after the I-rule should only prove propositions that are already in the premises of the I-rule. tonk does not satisfy this property and hence is rejected. There is also a converse criterion, called *stability* [5], which states that an E-rule should not be “too strong” compared to the I-rule. All meaningful connectives (including connectives in ordinary intuitionistic logic) are considered to enjoy both the inversion principle and stability.

### 3.2 Lax logic

Lax logic is an intuitionistic modal logic with one possibility modality operator  $\diamond$ . It features the following rules for  $\diamond$  along with the usual rules for intuitionistic propositional logic.

$$\frac{\Gamma \vdash_{\text{lax}} A}{\Gamma \vdash_{\text{lax}} \diamond A} \text{I-}\diamond \quad \frac{\Gamma \vdash_{\text{lax}} \diamond A \quad \Gamma, A \vdash_{\text{lax}} \diamond B}{\Gamma \vdash_{\text{lax}} \diamond B} \text{E-}\diamond$$

Lax logic’s significance is the Curry-Howard correspondence with Moggi’s metalanguage. Term assignment to I- $\diamond$  and E- $\diamond$  yields terms **return**  $M$  and **let**  $x := M_1$  **in**  $M_2$  in the metalanguage in an evident way. It is also shown that operational aspects of the metalanguage are easily adapted to well-known proof-theoretic notions (e.g. proof normalization) [2, 7].

According to PTS, however, this formalism of lax logic is unsatisfactory, for that  $\diamond$  is unstable. Because I- $\diamond$  proves  $\diamond A$  from any  $A$ ,  $\diamond A$  is considered to have precisely the same information as  $A$ . To be as strong as I- $\diamond$ , E- $\diamond$  then must be such a rule that directly extracts  $A$  from any  $\diamond A$ , or dually, turns any sequent  $\Gamma, A \vdash_{\text{lax}} B$  with premise  $A$  into  $\Gamma \vdash_{\text{lax}} B$  given  $\Gamma \vdash_{\text{lax}} \diamond A$ . Clearly, the actual E- $\diamond$  rule has an extra restriction on the form of conclusion, namely  $\diamond B$ , by which stability fails. (See [35] for more details.)

### 3.3 Simple adjoint calculus

Adjoint calculus [1] is a calculus for linear logic that incorporates two styles of judgments, one for linear reasoning and the other for non-linear (classical) reasoning. Exploiting the idea of adjoint calculus, we decompose the modality  $\diamond$  into a composite of two modalities  $\triangleleft \circ \triangleright$ . To this end, we restrict our focus to a fragment of lax logic where every judgment has precisely one premise. This fragment Curry-Howard-corresponds to what is called the *simple metalanguage* in Moggi’s original paper [30], where every term has precisely one free variable. In fact, the decomposition presented in the sequel is the same as restriction of adjoint calculus to the single variable fragments. After this fact, we call the decomposed calculus *simple adjoint calculus (SAC)*.

Figure 1 presents the complete list of typing rules of SAC. In the figure,  $\boxed{\Gamma \mid \Delta}$  denotes either  $\boxed{x : \tau \mid \cdot}$  or  $\boxed{\cdot \mid v : A}$ . Therefore, a judgment in SAC is in one of the following forms of  $\boxed{x : \tau \vdash^v M : \tau'}$ ,  $\boxed{x : \tau \mid \cdot \vdash^c N : A}$ , or  $\boxed{\cdot \mid v : A \vdash^c N : A'}$ . Note that every judgment has exactly one free variable.

To convey the intuition, we start by explaining the semantics first rather than syntactic details. SAC’s semantics is simply given by any adjunction between any categories:

$$\begin{array}{c}
\frac{x: \tau \vdash^v M: \tau_1}{x: \tau \vdash^v f(M): \tau_2} f: \tau_1 \rightarrow \tau_2 \quad \frac{\Gamma \mid \Delta \vdash^c N: A_1}{\Gamma \mid \Delta \vdash^c g(N): A_2} g: A_1 \rightarrow A_2 \\
\frac{x: \tau \vdash^v M: \tau'}{x: \tau \mid \cdot \vdash^c h(M): A} h: \tau' \rightarrow A \quad \frac{x: \tau \vdash^v x: \tau}{\cdot \mid v: A \vdash^c v: A} \\
\frac{x: \tau \mid \cdot \vdash^c N: A}{x: \tau \vdash^v \mathbf{reify} N: \triangleleft A} \quad \frac{x: \tau \vdash^v M: \triangleleft A}{x: \tau \mid \cdot \vdash^c \mathbf{reflect} M: A} \quad \frac{x: \tau \vdash^v M: \tau'}{x: \tau \mid \cdot \vdash^c \mathbf{val} M: \triangleright \tau'} \\
\frac{\Gamma \mid \Delta \vdash^c N_1: \triangleright \tau \quad x: \tau \mid \cdot \vdash^c N_2: A}{\Gamma \mid \Delta \vdash^c \mathbf{let val} x := N_1 \mathbf{in} N_2: A}
\end{array}$$

■ **Figure 1** Typing rules of SAC

$$\begin{array}{c}
\mathcal{C} \xrightarrow{\triangleright} \mathcal{D} \\
\mathcal{C} \xleftarrow{\triangleleft} \mathcal{D} \\
\mathcal{C} \xleftrightarrow{\perp} \mathcal{D}
\end{array}$$

As the symbols suggest, we identify the type operators  $\triangleright$  and  $\triangleleft$  with the left and right adjoint functors in the model. Namely, we identify a judgment  $\vdash^v$  with a morphism in  $\mathcal{C}$  and  $\vdash^c$  with  $\mathcal{D}$ . By identifying context  $\boxed{x: \tau \mid \cdot}$  with  $\boxed{\cdot \mid v: \triangleright \tau}$ , one may think of  $\mathbf{val}$  as the functor  $\triangleright$ 's action on morphisms  $\triangleright_{\tau, \tau'}: \mathcal{C}(\tau, \tau') \rightarrow \mathcal{D}(\triangleright \tau, \triangleright \tau')$ . Similarly,  $\mathbf{reify}$  and  $\mathbf{reflect}$  are identified with functions sending a morphism to its transpose. While we have not yet introduced enough syntactic notions, the intention of the following statement should now be clear. That is, SAC serves as an internal language of adjunctions.

► **Theorem 1.** *There is a sound and complete interpretation of SAC in an adjunction.*

Let us explain the syntax of SAC in detail. All types in SAC are classified into two classes, which we call *value* types and *computation* types. Note that these terminologies are arbitrary. The model may no longer be a Kleisli adjunction, hence with no flavor of computation. We call terms of value type (resp. computation type) *value terms* (resp. *computation terms*). We use metavariables  $M, M', \dots$  for value terms,  $N, N', \dots$  for computation terms, and  $L, L', \dots$  for any terms. A *signature of SAC* consists of a set  $V$  of base value types, a set  $C$  of base computation types, and a set  $F$  of function symbols. Fixing a signature defines the sets of all value and computation types, which are freely generated by the base types and type operators  $\triangleright$  and  $\triangleleft$ .

Equations are given to typed terms as in the metalanguage. We only consider equations between terms with the same type under the same context (i.e. equations-in-context). The definitional equality (postulated equations) of SAC is given by the following rules. The rules for congruence, reflexivity, symmetry, transitivity, and substitution are omitted for brevity.

$$\begin{array}{ll}
\mathbf{reflect}(\mathbf{reify} N) =_A N & (\beta_{\triangleleft}) \\
M =_{\triangleleft A} \mathbf{reify}(\mathbf{reflect} M) & (\eta_{\triangleleft}) \\
(\mathbf{let val} x := \mathbf{val} M \mathbf{in} N) =_A N[M/x] & (\beta_{\triangleright}) \\
N =_{\triangleright \tau} (\mathbf{let val} x := N \mathbf{in val} x) & (\eta_{\triangleright}) \\
(\mathbf{let val} x_1 := C[N_1] \mathbf{in} N_2) =_A C[\mathbf{let val} x_1 := N_1 \mathbf{in} N_2] & (\text{comm. conv.})
\end{array}$$

Here,  $C[-]$  denotes any typed context. Given a signature, a *theory of SAC* is a set of equations-in-context in the signature.

Note that we can easily redefine  $\mathbf{return}$  and  $\mathbf{let}$  in the simple metalanguage in SAC:

$$\begin{array}{l}
\mathbf{return} M := \mathbf{reify}(\mathbf{val} M) \\
(\mathbf{let} x := M_1 \mathbf{in} M_2) := \mathbf{reify}(\mathbf{let val} x := \mathbf{reflect} M_1 \mathbf{in} M_2)
\end{array}$$

$$\begin{array}{c}
\frac{\Gamma \vdash^v M_1 : \tau_1 \quad \cdots \quad \Gamma \vdash^v M_n : \tau_n}{\Gamma \vdash^v f(M_1, \dots, M_n) : \tau} f : \vec{\tau}_i \rightarrow \tau \\
\frac{\Gamma \vdash^v M_1 : \tau_1 \quad \cdots \quad \Gamma \vdash^v M_n : \tau_n \quad \Gamma \mid \Delta \vdash^c N : A}{\Gamma \mid \Delta \vdash^c g(M_1, \dots, M_n, N) : A'} g : \vec{\tau}_i, A \rightarrow A' \\
\frac{\Gamma \vdash^v M_1 : \tau_1 \quad \cdots \quad \Gamma \vdash^v M_n : \tau_n}{\Gamma \mid \cdot \vdash^c h(M_1, \dots, M_n) : A} h : \vec{\tau}_i \rightarrow A \quad \frac{}{\Gamma \vdash^v x : \tau} (x : \tau) \in \Gamma \\
\frac{}{\Gamma \mid v : A \vdash^c v : A} \quad \frac{\Gamma \vdash^v M : \tau}{\Gamma \mid \cdot \vdash^c \text{val } M : \triangleright \tau} \quad \frac{\Gamma \mid \Delta \vdash^c N_1 : \triangleright \tau \quad x : \tau, \Gamma \mid \cdot \vdash^c N_2 : A}{\Gamma \mid \Delta \vdash^c \text{let val } x := N_1 \text{ in } N_2 : A}
\end{array}$$

■ **Figure 2** Typing rules of SEC (rules for finite product types are omitted)

However, the converse is not possible for that the class of models is widened from any monads to any adjunctions.

Unlike lax logic, the modalities in SAC are considered stable. The strength of the introduction and elimination rules of  $\triangleright$  is equalized (at least in its succedents) in the sense that **val** creates  $\triangleright \tau$  from any  $\tau$  and **let val** destructs  $\triangleright \tau$  into any proof term with a hole of type  $\tau$ . This is also confirmed by checking the associativity rule of the metalanguage is rephrased with a commutative conversion rule with typed context  $C[-]$  in SAC.

### 3.4 Semi-effect calculus

Now that we have accomplished our proof-theoretic reconstruction, we further derive another calculus that is interesting as a programming language. In SAC, we could freely switch back and forth between the realms of values (terms under  $\vdash^v$ ) and computations (terms under  $\vdash^c$ ). By removing the rules for **reify** and **reflect** from the calculus and allowing multiple variables in the value context, we obtain a new calculus, which we dub *semi-effect calculus* (*SEC*). In SEC, the realms of values and computations are no longer treated dually. Instead values can only “act” on computations in a way we later justify via semantic arguments. Still, the calculus has a flavor of computation as it incorporates **val** and **let val**. We call this phenomenon *semi-effectful*.

As in SAC, the set of types in SEC is given by a set of value types and a set of computation types, denoted by  $\tau$  and  $A$  respectively:

$$\begin{aligned}
\tau &::= \sigma \mid \tau \times \tau \mid 1 \\
A &::= b \mid \triangleright \tau
\end{aligned}$$

where  $\sigma$  is any base value type and  $b$  is any base computation type. Notice that we also assume finite product types in values. Since the right adjoint modality  $\triangleleft$  is dropped, nested computation types such as  $\triangleleft \triangleright \triangleleft \triangleright \tau$  are no longer valid. Each function symbol has one of three sorts:  $\vec{\tau}_i \rightarrow \tau$ ,  $\vec{\tau}_i, A \rightarrow A'$ , and  $\vec{\tau}_i \rightarrow A$ , where  $\vec{\tau}_i$  denotes  $\tau_1, \dots, \tau_n$  for some  $n \in \mathbb{N}$ .

Figure 2 lists the typing rules of SEC. A judgment in SEC has either of forms  $\boxed{\Gamma \vdash^v M : \tau'}$  or  $\boxed{\Gamma \mid \Delta \vdash^c N : A}$ . Here,  $\Gamma$  is a context of zero or more value variables and  $\Delta$  is a context of zero or one computation variable. While contexts in SEC have unusual forms, the usual properties of typing judgment hold without difficulty.

► **Lemma 2.** *The uniqueness of typing holds for both  $\vdash^v$  and  $\vdash^c$ . The weakening, contraction, and exchange rules hold for the value context. The structural rule of substitution holds for both the value and computation contexts.*

SEC inherits the equation-in-context rules from SAC. Using concepts up to here, we can introduce the theory of SEC.

► **Definition 3.** A signature of SEC is given by sets  $V$  and  $C$  of value/computation base types and a set  $F$  of function symbols. A theory of SEC consists of a signature  $\Sigma$  and a set  $Ax$  of axioms, well-formed equations under  $\Sigma$ .

By seeing `let val` as `let` and `val` as `return`, we can easily transport examples of Moggi’s metalanguage (e.g. stateful, nondeterministic, and so on) to SEC. Moreover, SEC can express a term that is not “effectful” but “semi-effectful”. Here we demonstrate this by showing an example using Haskell’s `Applicative` [26], which is a generalization of `Monad`.

Recall that a functor  $f$  in Haskell is `Applicative` if it is endowed with two operators

```
pure: a → f a
<*>: f (a → b) → f a → f b
```

satisfying some laws. A leading example that is not `Monad` but `Applicative` is `ZipList`. `ZipList a` is a type of finite or infinite sequence of type  $a$ . Its associated `pure` is given by `pure x := (x)i<∞` and `<*>` is given by `(fi)i<n <*> (xi)i<m := (fixi)i<min{n,m}`. Because `ZipList` is not `Monad`, we cannot use Moggi’s metalanguage to reason about it. On the other hand, in SEC, such reasoning is possible. We define a theory for `ZipList`  $\mathcal{T}_{\text{ZipList}}$  as the internal language of the Freyd category associated to the lax monoidal functor of `ZipList`, where we defer the technical details to Example 9 and Corollary 10. Here we only point out that  $\vdash^c$  corresponds to the applicative context, whereas  $\vdash^v$  is the pure context. Inside this theory, terms of `ZipList` can be defined in a style very much like applicative-do [25]:

```
let val x := [1, 2, 3] in let val y := [4, 5] in val (x + y).
```

This term roughly corresponds to the following expression in applicative-do:

```
do { x <- [1,2,3]; y <- [4,5]; pure (x + y) } (1)
```

which is desugared to `pure (\x y -> x + y) <*> [1,2,3] <*> [4,5]` and results in `[5,7]`. For the sake of soundness, applicative-do disallows a term at the position of `[4,5]` in (1) to use  $x$ . However, there is no such limitation in SEC, and thus the following is perfectly valid:

```
let val x := [1, 2, 3] in let val y := val (x + 1) in val (x + y).
```

In this way, we obtain a logic of `ZipList` for free, in which we can reason e.g. as follows:

$$\begin{aligned} \Gamma \mid \Delta \vdash^c (\text{let val } x := [1, 2, 3] \text{ in let val } y := [4, 5] \text{ in val } (x + y)) \\ = (\text{let val } x := [4, 5] \text{ in let val } y := [1, 2, 3] \text{ in val } (x + y)). \end{aligned}$$

Note that SEC admits more models beyond `Applicative`, as we will see in Section 4.

## 4 Categorical models for SEC

In this section, we introduce a categorical semantics of SEC. Our semantics is built upon monoidal actions. We fix a monoidal category  $(\mathcal{M}, \otimes, I, r, l, a)$ .

► **Definition 4** (monoidal action, actegory, e.g. [14]). Let  $\mathcal{C}$  be a category. A bifunctor  $(-)\cdot(-): \mathcal{M} \times \mathcal{C} \rightarrow \mathcal{C}$  is an  $\mathcal{M}$ -action on  $\mathcal{C}$  if there are natural isomorphisms  $\eta_c: I \cdot c \rightarrow c$  and  $\mu_{m_1, m_2, c}: (m_1 \otimes m_2) \cdot c \rightarrow m_1 \cdot (m_2 \cdot c)$  making the following diagrams commute.

$$\begin{array}{ccc} (m_1 \otimes m_2 \otimes m_3) \cdot c & \xrightarrow{\mu_{m_1 \otimes m_2, m_3, c}} & (m_1 \otimes m_2) \cdot (m_3 \cdot c) \\ \mu_{m_1, m_2 \otimes m_3, c} \downarrow & & \downarrow \mu_{m_1, m_2, m_3 \cdot c} \\ m_1 \cdot ((m_2 \otimes m_3) \cdot c) & \xrightarrow{\text{id} \cdot \mu_{m_2, m_3, c}} & m_1 \cdot (m_2 \cdot (m_3 \cdot c)) \end{array} \quad \begin{array}{ccc} (m \otimes I) \cdot c & \xrightarrow{\mu_{m, I, c}} & m \cdot (I \cdot c) \\ & \searrow r_m \cdot \text{id} & \swarrow \text{id} \cdot \eta_c \\ & m \cdot c & \end{array}$$

The left diagram implicitly uses the associativity  $a$ . An  $\mathcal{M}$ -actegory is a category with a fixed  $\mathcal{M}$ -action on it.

We often omit the prefix  $\mathcal{M}$ - from  $\mathcal{M}$ -action if it is inferrable from the context.

- **Example 5. 1.** Any monoidal category  $\mathcal{M}$  is automatically an  $\mathcal{M}$ -actegory, where the action  $(-)\cdot(-): \mathcal{M} \times \mathcal{M} \rightarrow \mathcal{M}$  is given by the tensor product  $\otimes$ .
- 2. Monoidal action subsumes the classical notion of monoid action. Any set is identified with a (small) discrete category and any monoid  $(M, *, e)$  is identified with a monoidal category whose underlying category is discrete and whose tensor is given by  $*$ . Under this identification, a set  $A$  is an  $M$ -actegory if and only if  $A$  has a monoid action of  $M$ .

Morphisms of actegories are defined in the following sense.

- **Definition 6.** Let  $\mathcal{C}, \mathcal{D}$  be  $\mathcal{M}$ -actegories. A functor  $F: \mathcal{C} \rightarrow \mathcal{D}$  is (lax)  $\mathcal{M}$ -equivariant (resp. strong  $\mathcal{M}$ -equivariant) if there is a coherent natural transformation (resp. isomorphism)  $\phi_{m,c}^F: m \cdot F(c) \rightarrow F(m \cdot c)$ . We mean by coherence that the diagrams below commute.

$$\begin{array}{ccc}
 (m \otimes m') \cdot Fc \xrightarrow{\mu_{m,m',Fc}} m \cdot (m' \cdot Fc) \xrightarrow{\text{id} \cdot \phi_{m',c}^F} m \cdot F(m' \cdot c) & & \begin{array}{ccc} & I \cdot Fc & \\ \phi_{I,c}^F \swarrow & & \searrow \eta_{Fc} \\ F(I \cdot c) & \xrightarrow{F\eta_c} & Fc \end{array} \\
 \phi_{m \otimes m', c}^F \downarrow & & \downarrow \phi_{m, m' \cdot c}^F \\
 F((m \otimes m') \cdot c) \xrightarrow{F\mu_{m,m',c}} F(m \cdot (m' \cdot c)) & & 
 \end{array}$$

We will omit the superscript  $F$  for  $\phi^F$  when this does not make confusion. *Strict  $\mathcal{M}$ -equivariant functor* is also defined in the same manner.

- **Example 7. 1.** Consider the canonical  $\mathcal{M}$ -action on  $\mathcal{M}$  (see Example 5). An equivariant functor  $F: \mathcal{M} \rightarrow \mathcal{M}$  is precisely a *strong* functor [19]  $F$  on  $\mathcal{M}$ , where the strength  $t_{A,B}: A \otimes FB \rightarrow F(A \otimes B)$  is  $\phi_{A,B}^F$ .
- 2. Freyd category or value/producer structure [21] is a special case of strong equivariant functor. A Freyd category is an identity-on-objects functor  $J: \mathcal{V} \rightarrow \mathcal{C}$  such that (1)  $\mathcal{V}$  has finite products, (2)  $\mathcal{C}$  has a  $\mathcal{V}$ -action, and (3) The  $v \times (-)$  can be extended to the  $\mathcal{V}$ -action on  $\mathcal{C}$  along  $J$  for any  $v \in \mathcal{V}$ . These conditions say that  $J$  is strict  $\mathcal{V}$ -equivariant.

Given a strong  $\mathcal{V}$ -equivariant functor  $\triangleright: \mathcal{V} \rightarrow \mathcal{C}$  where  $\mathcal{V}$  has finite products and  $\mathcal{C}$  has a  $\mathcal{V}$ -action w.r.t. the cartesian structure of  $\mathcal{V}$ , we can interpret theories of SEC. The interpretation follows the traditional category-of-contexts paradigm. It is defined inductively once we fix data for base types and function symbols. We will use  $\mathcal{V}$  to interpret types and terms in the realm of values, and use  $\mathcal{C}$  for the realm of computations.

Types and contexts in the realm of values are interpreted in  $\mathcal{V}$  as usual:  $\llbracket \tau_1 \times \tau_2 \rrbracket := \llbracket \tau_1 \rrbracket \times \llbracket \tau_2 \rrbracket$  and  $\llbracket \Gamma \rrbracket := \prod_{(x_i: \tau_i) \in \Gamma} \llbracket \tau_i \rrbracket$ . We use  $\Gamma$  and  $\tau$  almost interchangeably by this identification. Computation types of the form  $\triangleright\tau$  are interpreted using the functor  $\triangleright$  by  $\llbracket \triangleright\tau \rrbracket := \triangleright\llbracket \tau \rrbracket$ . The two kinds of computation context have different interpretations:  $(\Gamma \mid v: A)$  is interpreted by the action  $\llbracket \Gamma \mid v: A \rrbracket := \llbracket \Gamma \rrbracket \cdot \llbracket A \rrbracket$ , and  $(\Gamma \mid \cdot)$  is interpreted by application of the equivariant functor  $\llbracket \Gamma \mid \cdot \rrbracket := \triangleright\llbracket \Gamma \rrbracket$ .

As to (well-typed) terms, we only show the case of computation terms (the case of value terms is rather obvious). The interpretation of computation variables just discards value variables:  $\llbracket \Gamma \mid v: A \vdash^c v: A \rrbracket := \llbracket \Gamma \rrbracket \cdot \llbracket A \rrbracket \xrightarrow{! \cdot \text{id}} 1 \cdot \llbracket A \rrbracket \xrightarrow{\eta} \llbracket A \rrbracket$ . **val** sends a value term to a computation term with the functor:  $\llbracket \Gamma \mid \cdot \vdash^c \text{val } M: \triangleright\tau \rrbracket := \triangleright\llbracket \Gamma \vdash^v M: \tau \rrbracket$ . The most involved case is **let val**.  $\llbracket \Gamma \mid \cdot \vdash^c \text{let val } x := N_1 \text{ in } N_2: A \rrbracket$  is given by:

$$\triangleright\llbracket \Gamma \rrbracket \xrightarrow{\triangleright\delta} \triangleright(\llbracket \Gamma \rrbracket \times \llbracket \Gamma \rrbracket) \xrightarrow{\phi^{-1}} \llbracket \Gamma \rrbracket \cdot \triangleright\llbracket \Gamma \rrbracket \xrightarrow{\text{id} \cdot \llbracket N_1 \rrbracket} \llbracket \Gamma \rrbracket \cdot \triangleright\llbracket \tau \rrbracket \xrightarrow{\phi} \triangleright(\llbracket \Gamma \times \tau \rrbracket) \xrightarrow{\llbracket N_2 \rrbracket} \llbracket A \rrbracket.$$



In the same vein, we can interpret the case when `let val x := N1 in N2` has a free computation variable.

Given an interpretation  $\llbracket - \rrbracket$  of  $\mathcal{T}$ , an equation-in-context is defined to be *valid w.r.t.*  $\llbracket - \rrbracket$  if the two terms are externally equal i.e. interpreted by the same morphism. It then follows that this interpretation is indeed sound.

► **Theorem 8.** *Let  $\mathcal{T}$  be any theory of SEC and  $\llbracket - \rrbracket$  be an interpretation of  $\mathcal{T}$ . Assume that all axioms of  $\mathcal{T}$  are satisfied by  $\llbracket - \rrbracket$ . Then all equations derivable in  $\mathcal{T}$  are satisfied by  $\llbracket - \rrbracket$ .*

**Proof.** The proof is tedious but routine. One needs to be careful of whether  $\Delta$  is empty or non-empty. (See Appendix for the detailed proof.) ◀

We will call such a strong  $\mathcal{V}$ -equivariant functor  $\triangleright: \mathcal{V} \rightarrow \mathcal{C}$  a *model of SEC*.

- **Example 9. 1.** Given a strong monad on a cartesian category  $\mathcal{C}$ , we get a model of SEC by the Kleisli construction (see Example 7). By the definition of Kleisli category, a term  $\boxed{\Gamma \mid v: \triangleright \tau \vdash^c N: \triangleright \tau'}$  is interpreted by a morphism  $\llbracket \Gamma \rrbracket \times \llbracket \tau \rrbracket \rightarrow T \llbracket \tau' \rrbracket$  in  $\mathcal{C}$ . Furthermore,  $\llbracket \text{val } M \rrbracket = \eta \circ \llbracket M \rrbracket$  and  $\llbracket \text{let val } x := N_1 \text{ in } N_2 \rrbracket = \llbracket N_2 \rrbracket^\# \circ t \circ \langle \text{id}, \llbracket N_1 \rrbracket \rangle$  hold where  $\llbracket N_2 \rrbracket^\#$  is the Kleisli lifting of  $\llbracket N_2 \rrbracket$  and  $N_2$  has no free computation variables. These interpretations agree with those of  $\llbracket \text{return } M \rrbracket$  and  $\llbracket \text{let } x := N_1 \text{ in } N_2 \rrbracket$  in [30].
2. There is a model of SEC that is not a Freyd category. The simplest is the inclusion  $\iota_1: 1 \rightarrow 1 + 1$  where 1 is the terminal category and the action  $* \cdot (-)$  is the identity.
  3. It is folklore that a lax monoidal functor  $F$  on a CCC  $\mathcal{C}$  induces a Freyd category  $J: \mathcal{C} \rightarrow \mathcal{D}$  [13, 23]. A morphism  $f: X \rightarrow Y$  in  $\mathcal{D}$  is given by a morphism  $f: 1 \rightarrow F(Y^X)$  in  $\mathcal{C}$ . (A similar construction is also found in the semantics of multi-staged computation [31].) Because a lax monoidal functor models **Applicative** in Haskell, this serves as a model of the example presented at the end of Section 3. In this sense we consider SEC is semi-effectful, admitting more models than what were not supported by traditional models of effects, namely monads.

Every model of SEC gives rise to its *internal language*, a theory of SEC such that all objects and morphisms of the model are base types and function symbols and  $Ax$  contains all such equations-in-context  $L_1 = L_2$  that  $\llbracket L_1 \rrbracket = \llbracket L_2 \rrbracket$ .

- **Corollary 10.** *Let  $\mathcal{T}_F$  be the internal language of model  $F$ . The following are equivalent.*
- $\mathcal{T}_F \vdash L_1 = L_2$  (i.e., equation-in-context  $L_1 = L_2$  is derivable in  $\mathcal{T}_F$ )
  - $\llbracket L_1 \rrbracket = \llbracket L_2 \rrbracket$  in  $F$ .

Conversely, we can construct a strong equivariant functor from any theory of SEC.

► **Theorem 11.** *Any theory of SEC induces a strong equivariant functor.*

**Proof.** We perform the term model construction as follows. The value category  $\mathcal{V}$  is constructed as usual from value terms (see e.g. [4]). The construction of the computation category  $\mathcal{C}$  is somewhat tricky; it is defined by case distinction of computation context:

$$\begin{aligned} \mathcal{C}((\Gamma \mid A), (\tau \mid A')) &:= \{(\llbracket \Gamma \vdash^v M: \tau \rrbracket, \llbracket \Gamma \mid A \vdash^c N: A' \rrbracket)\} \\ \mathcal{C}((\Gamma \mid \cdot), (\Gamma' \mid \cdot)) &:= \mathcal{V}(\Gamma, \Gamma') \\ \mathcal{C}((\Gamma \mid A), (\Gamma' \mid \cdot)) &:= \emptyset \end{aligned}$$

where  $[\dots]$  denotes the equivalence class of judgments up to the definitional equality. The  $\mathcal{V}$ -action on  $\mathcal{C}$  is then given by  $\tau \cdot (\Gamma \mid \Delta) := (\tau, \Gamma \mid \Delta)$ . The equivariant functor  $\triangleright$  sends  $\tau$  to  $(\tau \mid \cdot)$ . See Appendix for the detailed construction. ◀

► **Remark 12.** The crucial point of our term model construction is that the syntactic functor  $\triangleright$  is defined to be  $\triangleright(\tau) := (\tau \mid \cdot)$  instead of  $(\cdot \mid \triangleright\tau)$ . In fact, setting  $\triangleright(\tau) := (\cdot \mid \triangleright\tau)$  only gives us a lax equivariant functor. However, by setting  $\triangleright(\tau) := (\tau \mid \cdot)$ , it in turn no longer holds that the term model interprets a term by itself. For example,  $\llbracket \text{val } M \rrbracket$  is given by  $M$  instead of  $\text{val } M$ . As a result, Theorem 11 does not imply completeness of our semantics.

Before proceeding, we introduce the notion of morphism of models of SEC.

► **Definition 13** (equivariant natural transformation). *Let  $\mathcal{C}, \mathcal{D}$  be  $\mathcal{M}$ -actegories and  $F, G : \mathcal{C} \rightarrow \mathcal{D}$  be lax equivariant functors. A natural transformation  $\theta : F \rightarrow G$  is equivariant if  $\theta_{m \cdot c} \circ \phi_{m,c}^F = \phi_{m,c}^G \circ (m \cdot \theta_c)$  holds for all  $m \in \mathcal{M}$  and  $c \in \mathcal{C}$ .*

$\mathcal{M}$ -actegories, lax equivariant functors, and equivariant natural transformations form a 2-category  $\mathcal{M}\text{-Act}$ . Replacing “lax equivariant functors” with “strong equivariant functors” yields another 2-category. Any (strong) monoidal functor  $F : \mathcal{M} \rightarrow \mathcal{M}'$  induces the change-of-base 2-functor  $F^* : \mathcal{M}'\text{-Act} \rightarrow \mathcal{M}\text{-Act}$ .

- **Example 14.** 1. For any 2-categorical notion  $X$ , we call an  $X$  internal to  $\mathcal{M}\text{-Act}$  an  $\mathcal{M}$ -equivariant  $X$ . A strong monad  $T$  on  $\mathcal{M}$  is just an  $\mathcal{M}$ -equivariant monad. Also, the Kleisli resolution  $J \dashv K$  of  $T$  is an  $\mathcal{M}$ -equivariant adjunction. Note that every equivariant left adjoint is strong equivariant.  $J$  is a model of SEC in this way.
2. Change of base along a strong monoidal functor  $F : \mathcal{M} \rightarrow \mathcal{M}'$  makes  $F$  a strong  $\mathcal{M}$ -equivariant functor, since  $m \cdot_{\mathcal{M}'} F(m') = F(m) \otimes_{\mathcal{M}} F(m') \cong F(m \otimes_{\mathcal{M}} m') = F(m \cdot_{\mathcal{M}} m')$ .

A morphism of models of SEC is defined in the language of actegories.

► **Definition 15** (morphism of models of SEC). *Let  $\mathcal{V}$  be a category with finite products and  $F : \mathcal{V} \rightarrow \mathcal{C}, F' : \mathcal{V} \rightarrow \mathcal{C}'$  be models of SEC. A morphism of models of SEC is given by a lax equivariant functor  $H : \mathcal{C} \rightarrow \mathcal{C}'$  and an equivariant natural transformation  $\theta : F' \rightarrow HF$ .*

$$\begin{array}{ccc} & \mathcal{V} & \\ F \swarrow & \theta & \searrow F' \\ \mathcal{C} & \xrightarrow{H} & \mathcal{C}' \end{array}$$

Definition of 2-cells of models of SEC is omitted. Note that every morphism of models is a morphism in  $\int \mathcal{V}\text{-Act}^{\text{co}}(\mathcal{V}, -)$ , where  $\int$  is the Grothendieck construction.

► **Remark 16.** For reasons similar to Remark 12, the Curry-Howard-Lambek correspondence fails in our semantics. Specifically, we do not have a (canonical) equivalence of models of SEC:  $F \xrightarrow{\cong} \text{Syn}(\mathcal{T}_F)$ , where  $\text{Syn}(\mathcal{T})$  is the term model of theory  $\mathcal{T}$ .

## 5 Categorical gluing for (lax) equivariant functors

### 5.1 Categorical gluing

*Categorical gluing* (also known as *sconing*) is a technique to obtain a new model from a morphism of models. It is a special case of the comma construction (see e.g. [24]).

► **Definition 17** (categorical gluing). *Given a functor  $\Gamma : \mathcal{C} \rightarrow \mathcal{D}$ , the gluing category of  $\mathcal{C}$  to  $\mathcal{D}$  along  $\Gamma$  is obtained as a comma category  $\mathcal{D} \downarrow \Gamma$ . The gluing category is equipped with the projection functor  $\pi : (\mathcal{D} \downarrow \Gamma) \rightarrow \mathcal{C}$ .*

One important and interesting fact about gluing is that the gluing category  $\mathcal{D} \downarrow \Gamma$  often inherits the involved structures. In other words, when  $\mathcal{C}$  and  $\mathcal{D}$  have a certain kind of categorical structure and  $\Gamma: \mathcal{C} \rightarrow \mathcal{D}$  preserves it, the gluing category  $\mathcal{D} \downarrow \Gamma$  often has the structure and the projection functor  $\pi$  preserves it.

► **Example 18.** Let  $\mathcal{C}$  and  $\mathcal{D}$  be categories with (chosen) finite products and  $\Gamma: \mathcal{C} \rightarrow \mathcal{D}$  be a functor preserving them (up to isomorphism). Then the gluing category  $\mathcal{D} \downarrow \Gamma$  has finite products and the projection functor  $\pi$  (strictly) preserves them. Specifically, the terminal object is given by  $(1, 1, \gamma)$ , and the binary product of  $(d, c, f)$  and  $(d', c', f')$  is given by to by

$$(d \times d', c \times c', d \times d' \xrightarrow{f \times f'} \Gamma c \times \Gamma c' \xrightarrow{\gamma_{c,c'}} \Gamma(c \times c'))$$

where  $\gamma$  and  $\gamma_{c,c'}$  are the associated isomorphisms.

An important variance of gluing is *subgluing* [11].

► **Definition 19 (subgluing).** Suppose a functor  $\Gamma: \mathcal{C} \rightarrow \mathcal{D}$  is given. By restricting the objects in  $\mathcal{D} \downarrow \Gamma$  to subobjects, we get the full subcategory  $(\mathcal{D} \downarrow \Gamma)_s$  of  $\mathcal{D} \downarrow \Gamma$ . In other words,  $(D, C, f: D \rightarrow \Gamma C)$  is an object in  $(\mathcal{D} \downarrow \Gamma)_s$  if and only if  $f$  is a subobject. This category  $(\mathcal{D} \downarrow \Gamma)_s$  is called the *subgluing* of  $\mathcal{C}$  to  $\mathcal{D}$  along  $\Gamma$ .

The gluing category and subgluing category for  $\Gamma: \mathcal{C} \rightarrow \mathcal{D}$  are obtained as a pullback [11].

$$\begin{array}{ccc} (\mathcal{D} \downarrow \Gamma) & \longrightarrow & \mathcal{D} \rightarrow \\ \downarrow \lrcorner & & \downarrow \text{cod} \\ \mathcal{C} & \xrightarrow{\Gamma} & \mathcal{D} \end{array} \quad \begin{array}{ccc} (\mathcal{D} \downarrow \Gamma)_s & \longrightarrow & \text{Sub}(\mathcal{D}) \\ \downarrow \lrcorner & & \downarrow \text{Sub} \\ \mathcal{C} & \xrightarrow{\Gamma} & \mathcal{D} \end{array}$$

## 5.2 Actegorical gluing

We are able to present categorical gluing for actegories and lax equivariant functors. First we show that gluing along lax equivariant functors yields an  $\mathcal{M}$ -actegory.

► **Proposition 20 (actegorical gluing).** Let  $\mathcal{C}, \mathcal{D}$  be  $\mathcal{M}$ -actegories and  $\Gamma: \mathcal{C} \rightarrow \mathcal{D}$  a lax equivariant functor. The gluing category  $\mathcal{D} \downarrow \Gamma$  is an  $\mathcal{M}$ -actegories and the projection functor  $\pi: \mathcal{D} \downarrow \Gamma \rightarrow \mathcal{C}$  is strict equivariant.

Proposition 20 can be generalized in terms of fibration.

► **Proposition 21.** Let  $\mathcal{B}, \mathcal{C}, \mathcal{E}$  be  $\mathcal{M}$ -actegories and  $\Gamma: \mathcal{C} \rightarrow \mathcal{B}$  a lax equivariant functor. In addition, let  $p: \mathcal{E} \rightarrow \mathcal{B}$  be a strict equivariant functor which is also an opfibration, and suppose the condition (\*) holds.

(\*) For any object  $m$  in  $\mathcal{M}$ , the functor  $m \cdot (-): \mathcal{E} \rightarrow \mathcal{E}$  preserves opcartesian morphisms. Consider the pullback diagram in **Cat** below.

$$\begin{array}{ccc} \mathcal{G} & \longrightarrow & \mathcal{E} \\ q \downarrow \lrcorner & & \downarrow p \\ \mathcal{C} & \xrightarrow{\Gamma} & \mathcal{B} \end{array}$$

In this diagram,  $\mathcal{G}$  has an  $\mathcal{M}$ -action and the functor  $q: \mathcal{G} \rightarrow \mathcal{C}$  is strict equivariant.

**Proof.** The  $\mathcal{M}$ -action on  $\mathcal{G}$  is defined using the universality of opcartesian lifting of  $\phi^\Gamma: m \cdot \Gamma C \rightarrow \Gamma(m \cdot C)$ . Notice  $m \cdot \Gamma C = m \cdot pX = p(m \cdot X)$  for any  $C \in \mathcal{C}$  and  $X \in \mathcal{E}$  satisfying  $\Gamma C = pX$ . See Appendix for the detailed proof.  $\blacktriangleleft$

Notice that Proposition 20 is just an instance of Proposition 21 when  $p$  is the codomain opfibration  $\text{cod}: \mathcal{D}^\rightarrow \rightarrow \mathcal{D}$ .

► **Remark 22.** The ‘‘opfibration’’ in the statement of Proposition 21 cannot be simply replaced by ‘‘fibration’’ because the coherent natural transformation  $\phi$  for  $\Gamma$  is the form of  $\phi_{m,C}: m \cdot \Gamma C \rightarrow \Gamma(m \cdot C)$  and the cartesian lifting of  $\phi_{m,C}$  cannot be considered in contrast to the opcartesian lifting.

Although a pullback along a lax equivariant functor does not inherit the action as stated in Remark 22, if we restrict  $\Gamma$  to a strong equivariant one, we can get a similar proposition to Proposition 23. Moreover, the condition  $(*)$  can be dropped.

► **Proposition 23.** *Let  $\mathcal{B}, \mathcal{C}, \mathcal{E}$  be  $\mathcal{M}$ -actegories and  $\Gamma: \mathcal{C} \rightarrow \mathcal{B}$  a strong equivariant functor. In addition, let  $p: \mathcal{E} \rightarrow \mathcal{B}$  be a fibration that is strict equivariant. Consider the diagram in Proposition 21. In the diagram,  $\mathcal{G}$  has an  $\mathcal{M}$ -action and the functor  $q: \mathcal{G} \rightarrow \mathcal{C}$  is strict equivariant.*

Note that, when  $p$  is a bifibration in the situation of Proposition 23, there are two ways to define an  $\mathcal{M}$ -action on  $\mathcal{G}$  by Proposition 21 and 23. These coincide in the sense that they are isomorphic in  $\mathcal{M}\text{-Act}$ . We can also consider subgluing.

► **Proposition 24.** *Consider the assumption of Proposition 20. Assume moreover that functors  $v \cdot (-)$  preserve monos for all  $v \in \mathcal{V}$ . If either of the following holds,  $(\mathcal{D} \downarrow \Gamma)_s$  has an  $\mathcal{M}$ -action and  $\pi: (\mathcal{D} \downarrow \Gamma)_s \rightarrow \mathcal{C}$  is strict equivariant.*

1.  $\mathcal{D}$  admits epi-mono factorization.
2.  $\phi^\Gamma$  is (componentwise) monic.

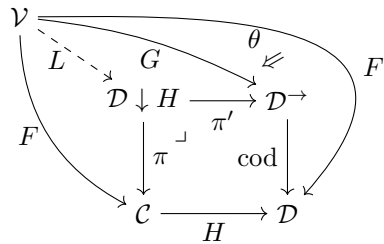
**Proof.** Condition 1 makes the subobject functor  $\text{Sub}(\mathcal{D}) \rightarrow \mathcal{D}$  an opfibration, so  $(\mathcal{D} \downarrow \Gamma)_s$  inherits the action by the preceding propositions. Under Condition 2 an action is directly defined by extension with  $\phi^\Gamma$ .  $\blacktriangleleft$

► **Proposition 25.** *A morphism  $(H: \mathcal{C} \rightarrow \mathcal{D}, \theta: F' \rightarrow HF)$  of models of SEC induces another model of SEC, i.e. a strong equivariant functor  $L: \mathcal{V} \rightarrow (\mathcal{D} \downarrow H)$  by the following construction.*

$$Lv := (F'v, Fv, \theta_v: F'v \rightarrow HFv)$$

$$L(a: v \rightarrow u) := (F'a, Fa): \theta_v \rightarrow \theta_u$$

In fact,  $L$  in Proposition 25 can also be described more conceptually from the point of view of the codomain fibration. Assume the situation in Proposition 25. We define a functor  $G: \mathcal{V} \rightarrow \mathcal{D}^\rightarrow$  by  $Gv := (\theta_v: F'v \rightarrow HFv)$ . This is well-defined by the naturality of  $\theta$  and functoriality of  $F$  and  $F'$ . By definition of  $G$ ,  $\text{cod} \circ G = HF$  holds where  $\text{cod}$  is the codomain functor  $\mathcal{D}^\rightarrow \rightarrow \mathcal{D}$ . Since  $\mathcal{D} \downarrow H$  is a pullback of  $\text{cod}$  along  $H$ , we obtain by universality a mediating functor  $L: \mathcal{V} \rightarrow (\mathcal{D} \downarrow H)$ , which indeed coincides with  $L$  in Proposition 25.



Combining Proposition 24 and 25, we obtain the subgluing version of Proposition 25.

► **Proposition 26.** *Let  $(H: \mathcal{C} \rightarrow \mathcal{D}, \theta: F' \rightarrow HF)$  be a morphism of models of SEC such that  $\theta$  and  $\phi^H$  are (componentwise) monic and the functors  $v \cdot (-)$  preserve monos for all  $v \in \mathcal{V}$ . Then we obtain a  $\mathcal{V}$ -actegory  $(\mathcal{D} \downarrow H)_s$  by Proposition 24. Moreover, we can get a model of SEC i.e. a strong equivariant functor  $L: \mathcal{V} \rightarrow (\mathcal{D} \downarrow H)_s$ .*

**Proof.** We use Condition 2 from Proposition 24. ◀

### 5.3 Flavor of logical predicates

As a toy example of the tools developed up to here, we derive logical predicates for SEC. We fix a theory  $\mathcal{T}$  of SEC.

Let  $F: \mathcal{V} \rightarrow \mathcal{C}$  be the term model of  $\mathcal{T}$  (see Theorem 11). For some object  $(\Gamma \mid \cdot)$  in  $\mathcal{C}$ , consider the functor  $H = \mathcal{C}((\Gamma \mid \cdot), -): \mathcal{C} \rightarrow \mathbf{Sets}$ . By the definition of the term model, this functor can be made a morphism of models as follows. To make  $\mathbf{Sets}$  a  $\mathcal{V}$ -actegory, we define its action by the functor  $\mathcal{V}(\Gamma, -) \times \text{id}_{\mathbf{Sets}}: \mathcal{V} \times \mathbf{Sets} \rightarrow \mathbf{Sets}$ . This bifunctor is indeed a  $\mathcal{V}$ -action on  $\mathbf{Sets}$  by the universality of products. A model of SEC over  $\mathbf{Sets}$  (i.e. a strong equivariant functor  $\mathcal{V} \rightarrow \mathbf{Sets}$ ) is given by  $\mathcal{V}(\Gamma, -)$ . Again, it is easy to see that this is indeed a model. Finally, by giving  $\theta_\tau([\Gamma \vdash^v M: \tau]) = [\Gamma \vdash^v M: \tau]$ ,  $H$  becomes a morphism of models from  $F$  to  $\mathcal{V}(\Gamma, -)$ . In this situation, we obtain the subgluing model  $L: \mathcal{V} \rightarrow (\mathbf{Sets} \downarrow H)_s$  by Proposition 26. Let us give an explicit definition of  $L$ . For an object  $\tau$  in  $\mathcal{V}$ ,  $L(\tau)$  is given by:

$$L(\tau) := \mathcal{V}(\Gamma, \tau) \subseteq \mathcal{C}((\Gamma \mid \cdot), (\tau \mid \cdot)).$$

To see what the  $\mathcal{V}$ -action on  $(\mathbf{Sets} \downarrow H)_s$  does, take any objects  $\tau \in \mathcal{V}$  and  $P \in (\mathbf{Sets} \downarrow H)_s$  where  $P \subseteq \mathcal{C}((\Gamma \mid \cdot), (\tau' \mid \Delta))$  for some  $(\tau' \mid \Delta) \in \mathcal{C}$ . When  $\Delta$  is not empty, say  $\Delta = A$ ,  $\tau \cdot P$  is given by

$$\left\{ ([\Gamma \vdash^v \langle M, M' \rangle: \tau \times \tau'], [\Gamma \mid \cdot \vdash^c N: A]) \mid ([\Gamma \vdash^v M': \tau'], [\Gamma \mid \cdot \vdash^c N: A]) \in P \right\}.$$

When  $\Delta$  is empty,  $\tau \cdot P$  is defined in the same way by using only the first component.

To summarize, we obtain the definition of logical predicates for SEC as follows.

► **Definition 27.** *Let  $\Gamma$  be a value context. We define a set  $I$  by*

$$I = \text{ob } \mathcal{C}_{\mathcal{T}} \cup \{A \mid A \text{ is a computation type}\}.$$

An  $I$ -indexed family of predicates  $\{P_i\}_{i \in I}$  is a logical predicate for  $\mathcal{T}$  if:

- $P_b \subseteq \left\{ [\Gamma \mid \cdot \vdash^c N: b] \right\}$
- $P_{\triangleright \tau} = P_{(\tau \mid \cdot)} = \left\{ [\Gamma \vdash^v M: \tau] \right\}$
- $P_{(\tau \mid b)} = \left\{ ([\Gamma \vdash^v M: \tau], [\Gamma \mid \cdot \vdash^c N: A]) \mid [\Gamma \mid \cdot \vdash^c N: b] \in P_b \right\}$
- $P_{(\tau \mid \triangleright \tau')} = \left\{ ([\Gamma \vdash^v M: \tau], [\Gamma \vdash^v M': \tau']) \mid [\Gamma \vdash^v M': \tau'] \in P_{\tau'} \right\}$
- A function  $\llbracket g \rrbracket: P_{(\prod_{\tau_i \mid \Delta})} \rightarrow P_A$  for each function symbol  $g: \vec{\tau}_i, \Delta \rightarrow A$

Note that  $P_{(\tau \mid A)}$  is equal to  $\mathcal{V}(\Gamma, \tau) \times P_A$ .

► **Lemma 28** (fundamental lemma). *Any logical predicate  $\{P_i\}_{i \in I}$  for  $\mathcal{T}$  straightforwardly defines a (set-theoretic) interpretation  $\llbracket - \rrbracket$ . In particular, a derivable judgement  $x: \tau' \mid \Delta \vdash^c N: A$  is interpreted as a function of the following form.*

$$\llbracket x: \tau' \mid \Delta \vdash^c N: A \rrbracket: P_{(\tau' \mid \Delta)} \rightarrow P_A$$

As a corollary, it follows that for any closed computation term  $N: A$ ,  $P_A(N)$  holds.

► **Remark 29.** We remark that  $P_{\triangleright\tau}$  is not a set of terms of type  $\triangleright\tau$  for the same reason of Remark 12. Namely, requiring  $P_{\triangleright\tau}$  to be a set of terms of type  $\triangleright\tau$  does not give rise to a strong equivariant functor. Due to this fact, we were unable to derive interesting syntactic results using this logical predicate. There are two ways to fix this: changing syntax or semantics. We expect whichever direction is hopeful, though we leave further investigation for our future work.

## 5.4 Actegorical gluing and $\top\top$ -lifting

The (categorical)  $\top\top$ -lifting [16] is a technique to derive lifting of strong monads along a fibration. It was originally introduced as a categorical formulation of logical predicates for the metalanguage [22]. The basic idea of the  $\top\top$ -lifting comes with the following lemma.

► **Lemma 30.** *For a fibration  $p: \mathcal{E} \rightarrow \mathcal{B}$ , the projection functor  $\pi: \mathbf{Mnd}(p) \rightarrow \mathbf{Mnd}(\mathcal{B})$  is also a fibration, where  $\mathbf{Mnd}(p)$  is the category of fibred monads over  $p$ .*

To accomodate the continuation monad  $S^{S^{(-)}}$  on  $\mathcal{E}$ , however, the lemma is insufficient because  $S^{S^{(-)}}$  is not fibred even if  $p$  strictly preserves the CCC structure. The crucial ingredient of the  $\top\top$ -lifting was that it generalized Lemma 30 by replacing  $\mathbf{Mnd}(p)$  with  $\mathbf{Mnd}'(p)$  the category of not-necessarily-fibred monads over  $p$ . This is further generalized to (non-fibred) strong monads. Consequently, given a preorder bifibration  $p: \mathcal{E} \rightarrow \mathcal{B}$  preserving the CCC structure, a strong monad  $T$  over  $\mathcal{B}$ , and some objects  $S, R$  such that  $S = TR$ , the  $\top\top$ -lifting constructs a strong monad over  $\mathcal{E}$  by the cartesian lifting of the canonical  $\sigma$ :

$$\begin{array}{ccc} T^{\top\top} & \xrightarrow{\bar{\sigma}} & S^{S^{(-)}} & \mathbf{Mnd}'_{\text{strong}}(p) \\ & & & \downarrow \pi \\ T & \xrightarrow{\sigma} & TR^{TR^{(-)}} & \mathbf{Mnd}_{\text{strong}}(\mathcal{B}) \end{array}$$

We relate the  $\top\top$ -lifting to actegorical gluing in the following sense.

► **Proposition 31.** *Let  $(T, \tilde{T})$  be a fibred monad over a fibration  $p: \mathcal{E} \rightarrow \mathcal{B}$ . Its Kleisli resolution gives us another fibration  $p_T$  and a pullback diagram in  $\mathbf{Cat}$ .*

$$\begin{array}{ccc} \tilde{T} \hookrightarrow \mathcal{E} & \xrightarrow{\tilde{J}} & \mathcal{E}_{\tilde{T}} \\ p \downarrow & \lrcorner & \downarrow p_T \\ T \hookrightarrow \mathcal{B} & \xrightarrow{J} & \mathcal{B}_T \end{array}$$

*If  $\mathcal{E}$  and  $\mathcal{B}$  are monoidal and  $p$  is strict monoidal, and if  $T$  and  $\tilde{T}$  are both strong monads such that  $\tilde{T}$ 's strength  $\tilde{t}$  is above  $T$ 's strength  $t$ , then both  $\mathcal{B}_T$  and  $\mathcal{E}_{\tilde{T}}$  have an  $\mathcal{E}$ -action and  $p_T$  strictly preserves it.*

Note that the  $\mathcal{E}$ -actions of  $\mathcal{B}$  and  $\mathcal{B}_T$  are given by change of base along  $p$ . Therefore, if  $p$  has a monoidal reflection (the prototypical example is the subobject fibration  $\mathbf{Sub}(\mathbf{Sets}) \rightarrow \mathbf{Sets}$ ),

we can perform change of base along the reflection on the whole diagram above, which in turn allows us to give  $\mathcal{B}$ -actions to  $\mathcal{E}$  and  $\mathcal{E}_T$  and recover the original  $\mathcal{B}$ -actions for  $\mathcal{B}$  and  $\mathcal{B}_T$ .

Proposition 31 states that if  $T^{\top\top}$  obtained by the  $\top\top$ -lifting is fibred, there exists a fibration that is strict equivariant, which yields  $T^{\top\top}$  by gluing along  $J$ . Although  $T^{\top\top}$  is not fibred in general, some  $T^{\top\top}$ s that naturally arise in the semantics are indeed fibred.

For the subobject fibration  $\text{Sub}(\mathbf{Sets}) \rightarrow \mathbf{Sets}$ , we can calculate its  $\top\top$ -lifting as follows:

$$T^{\top\top}(P \subseteq X) := \left( \left\{ m \in TX \mid \forall c \in TR^X. (\forall x. P(x) \rightarrow S(c(x))) \rightarrow S(c^\#(m)) \right\} \subseteq TX \right)$$

where  $c^\#$  is the Kleisli lifting of  $c$ . The following configurations yield fibred monads.

1. *Exception.* When  $T(X) := X \uplus E$  and  $(S \subseteq TR) := (\{*\} \subseteq T(\{*\}))$  for some  $E$ ,  $T^{\top\top}$  is computed by  $T^{\top\top}(P \subseteq X) = (P \subseteq X \uplus E)$ .
2. *Partiality.* This case is subsumed by the exception monad where  $E = \{\perp\}$ .
3. *Nondeterminism.* When  $T(X) := \mathcal{P}_{\text{fin}}(X)$  and  $(S \subseteq TR) := (\{\emptyset\} \subseteq T(\emptyset))$ ,  $T^{\top\top}$  is computed by  $T^{\top\top}(P \subseteq X) = (\{m \in \mathcal{P}_{\text{fin}}(X) \mid \exists x \in m. P(x)\} \subseteq \mathcal{P}_{\text{fin}}(X))$ .

Nonexamples include the state monad and the continuation monad. Proposition 31 gives us another view of logical predicates of the metalanguage. Such a view is sometimes more direct. In the case of the exception monad,  $\text{Sub}(\mathbf{Sets})_{T^{\top\top}}$  has the following structure.

$$\frac{(P \subseteq X) \xrightarrow{f} (Q \subseteq Y) \text{ in } \text{Sub}(\mathbf{Sets})_{T^{\top\top}}}{X \xrightarrow{f} Y \text{ in } \mathbf{Sets} \text{ such that } P(x) \text{ implies } \begin{cases} Q(f(x)) & (f(x) \in Y) \\ \text{false} & (f(x) \in E) \end{cases}}$$

Instantiating this to the case when  $E = \{\perp\}$ , the condition at the bottom may be viewed as the partial correctness of Hoare logic, by identifying  $X$  and  $Y$  as state (sub)spaces.

$$\frac{f: X \rightarrow Y \text{ in } \text{Sub}(\mathbf{Sets})_{T^{\top\top}}(P, Q)}{\vdash_{\text{partial}} \{P\} f \{Q\}}$$

## 6 Concluding remarks

This paper presented a new calculus of semi-effects (SEC) and its categorical models. As an application of our semantics, we introduced actegorical gluing and derived logical predicates for the calculus. A brief comparison with the  $\top\top$ -lifting is also presented. SEC incorporates a more general notion of effects, as exemplified with **Applicative**. Unlike related work, our semantics is purely defined in terms of actegories and equivariant functors.

As pointed out in Remark 12, 16, and 29, we do not consider our semantics is fully satisfactory. We expect that the true semantics of SEC is in the middle of lax equivariant and strong equivariant. However, we do not know whether as clean an account as the present work is possible in this direction. Another direction worth studying would be higher-order extensions. All our development took place in a first-order setting. While SEC's value side can be easily extended to a higher-order language, solely extending the computation side with higher-order functionals is not justified by the semantics, as Kleisli categories usually do not inherit a closed structure. We would also need to find examples that are not supported by **Applicative** but useful in practice. Such examples might help to introduce to existing functional programming languages a new general framework for structuring programs.

## References

- 1 N. Benton. A mixed linear and non-linear logic: Proofs, terms and models (extended abstract). In *Proceedings of Computer Science Logic (CSL)*, pages 121–135, 1994.
- 2 N. Benton, G. M. Bierman, and V. de Paiva. Computational types from a logical perspective. *Journal of Functional Programming*, 8(2):177–193, 1998.
- 3 N. Benton and P. Wadler. Linear logic, monads and the lambda calculus. In *Proceedings of Logic in Computer Science (LICS)*, pages 420–431. IEEE Computer Society, 1996.
- 4 R. L. Crole. *Categories for Types*. Cambridge mathematical textbooks. Cambridge University Press, 1993.
- 5 M. Dummett. *Logical Basis of Metaphysics*. Harvard University Press, 1991.
- 6 J. Egger, R. E. Møgelberg, and A. Simpson. The enriched effect calculus: syntax and semantics. *Journal of Logic and Computation*, 24(3):615–654, 2014.
- 7 M. Fairtlough and M. Mendler. Propositional lax logic. *Information and Computation*, 137(1):1–33, 1997.
- 8 M. P. Fiore. Semantic analysis of normalisation by evaluation for typed lambda calculus. In *Proceedings of Principles and Practice of Declarative Programming (PPDP)*, pages 26–37, 2002.
- 9 C. Führmann. Direct models for the computational lambda calculus. In *Proceedings of Mathematical Foundations of Programming Semantics (MFPS)*, volume 20 of *Electronic Notes in Theoretical Computer Science*, pages 245–292. Elsevier, 1999.
- 10 J. G.-Larrecq, S. Lasota, and D. Nowak. Logical relations for monadic types. *Mathematical Structures in Computer Science*, 18(6):1169–1217, 2008.
- 11 M. Hasegawa. Logical predicates for intuitionistic linear type theories. In *Proceedings of Typed Lambda Calculi and Applications (TLCA)*, volume 1581 of *Lecture Notes in Computer Science*, pages 198–212. Springer, 1999.
- 12 C. Hermida. *Fibrations, logical predicates and indeterminates*. PhD thesis, University of Edinburgh, UK, 1993.
- 13 C. Heunen and B. Jacobs. Arrows, like monads, are monoids. In *Proceedings of Mathematical Foundations of Programming Semantics (MFPS)*, volume 158 of *Electronic Notes in Theoretical Computer Science*, pages 219–236. Elsevier, 2006.
- 14 G. Janelidze and G. M. Kelly. A note on actions of a monoidal category. *Theory and Applications of Categories*, 9(4):61–91, 2001.
- 15 Y. Kakutani, Y. Murase, and Y. Nishiwaki. Dual-context modal logic as left adjoint of Fitch-style modal logic. *Journal of Information Processing*, 27:77–86, 2019.
- 16 S. Katsumata. A semantic formulation of  $\top\top$ -lifting and logical predicates for computational metalanguage. In *Proceedings of Computer Science Logic (CSL)*, pages 87–102, 2005.
- 17 G. A. Kavvos. The many worlds of modal  $\lambda$ -calculi: I. Curry-Howard for necessity, possibility and time. *CoRR*, abs/1605.08106, 2016.
- 18 S. Kobayashi. Monad as modality. *Theoretical Computer Science*, 175(1):29–74, 1997.
- 19 A. Kock. Strong functors and monoidal monads. *Archiv der Mathematik*, 23:113–120, 1972.
- 20 N. Kürbis. Proof-theoretic semantics, a problem with negation and prospects for modality. *Journal of Philosophical Logic*, 44(6):713–727, 2015.
- 21 P. B. Levy. *Call-by-push-value*. PhD thesis, Queen Mary University of London, UK, 2001.
- 22 S. Lindley and I. Stark. Reducibility and  $\top\top$ -lifting for computation types. In *Proceedings of Typed Lambda Calculi and Applications (TLCA)*, volume 3461 of *Lecture Notes in Computer Science*, pages 262–277. Springer, 2005.
- 23 S. Lindley, P. Wadler, and J. Yallop. Idioms are oblivious, arrows are meticulous, monads are promiscuous. *Electronic Notes in Theoretical Computer Science*, 229(5):97–117, 2011.
- 24 S. MacLane. *Categories for the Working Mathematician*. Springer-Verlag, New York, 1971.
- 25 S. Marlow, S. P. Jones, E. Kmett, and A. Mokhov. Desugaring haskell’s do-notation into applicative operations. In *Proceedings of Haskell Symposium*, pages 92–104. ACM, 2016.



- 26 C. McBride and R. Paterson. Applicative programming with effects. *Journal of Functional Programming*, 18(1):1–13, 2008.
- 27 J. C. Mitchell and A. Scedrov. Notes on scoping and relators. In *Proceedings of Computer Science Logic (CSL)*, pages 352–378, 1992.
- 28 R. E. Møgelberg and S. Staton. Linear usage of state. *Logical Methods in Computer Science*, 10(1), 2014.
- 29 E. Moggi. Computational lambda-calculus and monads. In *Proceedings of Logic in Computer Science (LICS)*, pages 14–23, 1989.
- 30 E. Moggi. Notions of computation and monads. *Information and Computation*, 93(1):55–92, 1991.
- 31 Y. Nishiwaki, Y. Kakutani, and Y. Murase. Modality via iterated enrichment. In *Proceedings of Mathematical Foundations of Programming Semantics (MFPS)*, volume 341 of *Electronic Notes in Theoretical Computer Science*, pages 297–320. Elsevier, 2018.
- 32 F. Pfenning and R. Davies. A judgmental reconstruction of modal logic. *Mathematical Structures in Computer Science*, 11(4):511–540, 2001.
- 33 J. Power and H. Thielecke. Closed Freyd- and kappa-categories. In *Proceedings of Automata, Languages and Programming (ICALP)*, volume 1644 of *Lecture Notes in Computer Science*, pages 625–634. Springer, 1999.
- 34 A. N. Prior. The runabout inference-ticket. *Analysis*, 21(2):38–39, 1960.
- 35 S. Read. General-elimination harmony and higher-level rules. In *Dag Prawitz on Proofs and Meaning*, pages 293–312. Springer, Cham, 2015.
- 36 R. Statman. Logical relations and the typed lambda-calculus. *Information and Control*, 65(2/3):85–97, 1985.
- 37 S. Staton. Freyd categories are enriched lawvere theories. *Electronic Notes in Theoretical Computer Science*, 303:197–206, 2014.
- 38 J. Sterling and B. Spitters. Normalization by gluing for free  $\lambda$ -theories. *CoRR*, abs/1809.08646, 2018.
- 39 T. Streicher. *Domain-theoretic foundations of functional programming*. World Scientific, 2006.

## A

 Omitted definitions

► **Definition 32** (strong monad). *Let  $\mathcal{C}$  be a category with finite products. A strong monad over  $\mathcal{C}$  is a quadruple  $(T, \eta, \mu, t)$  of a functor  $T: \mathcal{C} \rightarrow \mathcal{C}$  and natural transformations  $\eta: \text{id}_{\mathcal{C}} \rightarrow T$ ,  $\mu: T^2 \rightarrow T$  and  $t: (-) \times T(-) \rightarrow T(- \times -)$  such that  $(T, \eta, \mu)$  is a monad over  $\mathcal{C}$  and the following diagrams commute.*

$$\begin{array}{c}
 \begin{array}{ccc}
 & 1 \times TA & \\
 t_{1,A} \swarrow & & \searrow r_{TA} \\
 T(1 \times A) & \xrightarrow{Tr_A} & TA
 \end{array} \\
 \\
 \begin{array}{ccc}
 (A \times B) \times TC & \xrightarrow{\alpha_{A,B,TC}} & A \times (B \times TC) \\
 t_{A \times B, C} \downarrow & & \downarrow \text{id}_A \times t_{B,C} \\
 T((A \times B) \times C) & & A \times T(B \times C) \\
 T\alpha_{A,B,C} \searrow & & \swarrow t_{A, B \times C} \\
 & T(A \times (B \times C)) &
 \end{array} \\
 \\
 \begin{array}{ccc}
 & A \times B & \\
 \text{id}_A \times \eta_B \swarrow & & \searrow \eta_{A \times B} \\
 A \times TB & \xrightarrow{t_{A,B}} & T(A \times B) \\
 \text{id}_A \times \mu_B \uparrow & & \uparrow \mu_{A,B} \\
 A \times T^2 B & \xrightarrow{t_{A, TB}} T(A \times TB) \xrightarrow{Tt_{A,B}} T^2(A \times B) &
 \end{array}
 \end{array}$$

It is straightforward to generalize the above definition to any monoidal category.

► **Definition 33.** *The full typing rules for SEC including finite product types.*

$$\begin{array}{c}
 \frac{}{\Gamma \vdash^v x: \tau} (x: \tau) \in \Gamma \quad \frac{\Gamma \vdash^v M_1: \tau_1 \quad \cdots \quad \Gamma \vdash^v M_n: \tau_n}{\Gamma \vdash^v f(M_1, \dots, M_n): \tau} f: \vec{\tau}_i \rightarrow \tau \\
 \frac{\Gamma \vdash^v M: \tau \quad \Gamma \vdash^v M': \tau'}{\Gamma \vdash^v \langle M, M' \rangle: \tau \times \tau'} \quad \frac{\Gamma \vdash^v M: \tau \times \tau'}{\Gamma \vdash^v \pi_1(M): \tau} \quad \frac{\Gamma \vdash^v M: \tau \times \tau'}{\Gamma \vdash^v \pi_2(M): \tau'} \quad \frac{}{\Gamma \vdash^v \langle \rangle: 1} \\
 \\
 \frac{\Gamma \vdash^v M_1: \tau_1 \quad \cdots \quad \Gamma \vdash^v M_n: \tau_n \quad \Gamma \mid \Delta \vdash^c N: A}{\Gamma \mid \Delta \vdash^c g(M_1, \dots, M_n, N): A'} g: \vec{\tau}_i, A \rightarrow A' \\
 \frac{\Gamma \vdash^v M_1: \tau_1 \quad \cdots \quad \Gamma \vdash^v M_n: \tau_n}{\Gamma \mid \Delta \vdash^c h(M_1, \dots, M_n): A'} h: \vec{\tau}_i \rightarrow A' \quad \frac{\Gamma \vdash^v M: \tau}{\Gamma \mid \cdot \vdash^c \text{val } M: \triangleright \tau} \\
 \frac{\Gamma \mid \Delta \vdash^c N_1: \triangleright \tau \quad x: \tau, \Gamma \mid \cdot \vdash^c N_2: A}{\Gamma \mid \Delta \vdash^c \text{let val } x := N_1 \text{ in } N_2: A}
 \end{array}$$

► **Definition 34.** *Substitution of terms by a variable.*

$$\begin{aligned}
x_j[\vec{M}_i/\vec{x}_i] &:= M_j \\
y[\vec{M}_i/\vec{x}_i] &:= y \quad (x_j \neq y \text{ for any } j) \\
(f(M_1, \dots, M_n))[M/x] &:= f(M_1[M/x], \dots, M_n[M/x]) \\
\langle M'_1, M'_2 \rangle[\vec{M}_i/\vec{x}_i] &:= \langle M'_1[\vec{M}_i/\vec{x}_i], M'_2[\vec{M}_i/\vec{x}_i] \rangle \\
\pi_1(M')[\vec{M}_i/\vec{x}_i] &:= \pi_1(M'[\vec{M}_i/\vec{x}_i]) \\
\pi_2(M')[\vec{M}_i/\vec{x}_i] &:= \pi_2(M'[\vec{M}_i/\vec{x}_i]) \\
\langle \rangle[\vec{M}_i/\vec{x}_i] &:= \langle \rangle \\
(\text{val } M')[\vec{M}_i/\vec{x}_i] &:= \text{val } (M'[\vec{M}_i/\vec{x}_i]) \\
(\text{let val } y := N_1 \text{ in } N_2)[\vec{M}_i/\vec{x}_i] &:= \text{let val } y := N_1[\vec{M}_i/\vec{x}_i] \text{ in } N_2[\vec{M}_i/\vec{x}_i] \\
g(M'_1, \dots, M'_n, N)[\vec{M}_i/\vec{x}_i] &:= g(M'_1[\vec{M}_i/\vec{x}_i], \dots, M'_n[\vec{M}_i/\vec{x}_i], N[\vec{M}_i/\vec{x}_i]) \\
h(M'_1, \dots, M'_n)[\vec{M}_i/\vec{x}_i] &:= h(M'_1[\vec{M}_i/\vec{x}_i], \dots, M'_n[\vec{M}_i/\vec{x}_i])
\end{aligned}$$

► **Definition 35.** *The inference rules for equations-in-context. The rules for congruence, reflectivity, symmetry, transitivity, and substitution are omitted.*

$$\frac{\frac{(\Gamma \vdash^v M_1 =_\tau M_2) \in Ax}{\Gamma \vdash^v M_1 =_\tau M_2} \quad \frac{(\Gamma \mid \Delta \vdash^c N_1 =_A N_2) \in Ax}{\Gamma \mid \Delta \vdash^c N_1 =_A N_2}}{\Gamma \mid \Delta \vdash^c N_1 : \triangleright \tau \quad x : \tau, \Gamma \mid \cdot \vdash^c N_2 : A} \text{ (comm. conv.)} \\
\frac{\Gamma \vdash^v M : \tau \quad x : \tau, \Gamma \mid \cdot \vdash^c N : A}{\Gamma \mid \cdot \vdash^c (\text{let val } x := \text{val } M \text{ in } N) =_A N[M/x]} (\beta) \\
\frac{\Gamma \mid \Delta \vdash^c N : \triangleright \tau}{\Gamma \mid \Delta \vdash^c (\text{let val } x := N \text{ in val } x) =_{\triangleright \tau} N} (\eta)$$

► **Definition 36** (internal language). *Let  $F: \mathcal{V} \rightarrow \mathcal{C}$  be a model of SEC that is “small”, i.e.  $\mathcal{V}$  and  $\mathcal{C}$  are small. The internal language of  $F$  is given by the following data.*

Signature. Let  $\Sigma_F$  be a signature of SEC such that

1. (base types.) Base types are given by the sets of objects. The set of base value types is  $\text{ob } \mathcal{V}$  and the set of base computation type is  $\text{ob } \mathcal{C}$ . We write  $\ulcorner \tau \urcorner$  for the type corresponding to  $\tau \in \mathcal{V}$ , and  $\ulcorner A \urcorner$  for the type of  $A \in \mathcal{C}$ .

Note that fixing these two defines the interpretation of all types and contexts.

2. (function symbols.) Functions symbols are given by the sets of morphisms. Explicitly, we use the following set as the set of function symbols.

$$\{\ulcorner \tau \urcorner \xrightarrow{\ulcorner f \urcorner} \tau \mid \ulcorner \vec{\tau} \urcorner \xrightarrow{f} \ulcorner \tau \urcorner\} \cup \{\ulcorner \vec{\tau}, A \urcorner \xrightarrow{\ulcorner g \urcorner} A' \mid \ulcorner \vec{\tau} \urcorner \cdot \ulcorner A \urcorner \xrightarrow{g} \ulcorner A' \urcorner\} \cup \{\ulcorner \vec{\tau} \urcorner \xrightarrow{\ulcorner h \urcorner} A \mid F[\ulcorner \vec{\tau} \urcorner] \xrightarrow{h} \ulcorner A \urcorner\}$$

$\Sigma_F$  has a canonical interpretation  $\llbracket \cdot \rrbracket$  in  $F$ .

Theory. The internal language  $\mathcal{T}_F$  is a theory over  $\Sigma_F$  given by the following set of axioms:

$$(L_1 = L_2) \in Ax \iff \llbracket L_1 \rrbracket = \llbracket L_2 \rrbracket \text{ in } F$$

► **Definition 37.** *Let  $F: \mathcal{V} \rightarrow \mathcal{C}, F': \mathcal{V} \rightarrow \mathcal{C}'$  be models of SEC sharing the domain  $\mathcal{V}$ . Let  $(H, \theta), (H', \theta')$  be morphisms from  $F$  to  $F'$ . A 2-cell  $\alpha: (H, \theta) \rightarrow (H', \theta')$  is a 2-cell in  $\mathcal{V}\text{-Act}$*

from  $H$  to  $H'$  subject to the following equation.

$$\begin{array}{ccc}
 \mathcal{V} & & \mathcal{V} \\
 \swarrow F & \theta \Downarrow & \searrow F' \\
 \mathcal{C} & \xrightarrow{H} & \mathcal{C}' \\
 \searrow \alpha & \Downarrow & \swarrow H' \\
 & H' & 
 \end{array}
 =
 \begin{array}{ccc}
 \mathcal{V} & & \mathcal{V} \\
 \swarrow F & \theta' \Downarrow & \searrow F' \\
 \mathcal{C} & \xrightarrow{H'} & \mathcal{C}'
 \end{array}$$

## B Omitted proofs

**Proof of Theorem 8.** The nontrivial point is to check that  $\llbracket - \rrbracket$  is sound with respect to  $(\beta)$ ,  $(\eta)$ , and  $(\text{comm. conv.})$ . We check only  $(\beta)$  and  $(\eta)$ . For  $(\beta)$ , see the following diagram.

$$\begin{array}{ccccc}
 \triangleright \llbracket \Gamma \rrbracket & \xrightarrow{\triangleright \delta} & \triangleright (\llbracket \Gamma \rrbracket \times \llbracket \Gamma \rrbracket) & \xrightarrow{\phi^{-1}} & \llbracket \Gamma \rrbracket \cdot \triangleright \llbracket \Gamma \rrbracket \\
 & \searrow \triangleright \langle \text{id}, \llbracket M \rrbracket \rangle & \downarrow \triangleright (\text{id} \times \llbracket M \rrbracket) & & \downarrow \text{id} \cdot \triangleright \llbracket M \rrbracket \\
 & & \triangleright (\llbracket \Gamma \rrbracket \times \llbracket \tau \rrbracket) & \xrightarrow{\phi^{-1}} & \llbracket \Gamma \rrbracket \cdot \triangleright \llbracket \tau \rrbracket \\
 & & & \searrow & \downarrow \phi \\
 & & & & \triangleright (\llbracket \Gamma \rrbracket \times \llbracket \tau \rrbracket) \\
 & & & & \downarrow \llbracket N \rrbracket \\
 & & & & \llbracket A \rrbracket
 \end{array}$$

In the above diagram, the small diagram labeled (a) commutes by the universality of the product and the one labeled (b) does by the naturality of  $\phi$ .

For  $(\eta)$ , we show only the case where  $\Delta$  is nonempty because the case where  $\Delta$  is empty can be shown in essentially the same way. See the following diagram.

$$\begin{array}{ccc}
[[\Gamma]] \cdot [[\Delta]] & \xrightarrow{\delta \cdot \text{id}} & ([[ \Gamma ] \times [ \Gamma ] ) \cdot [[ \Delta ] \\
\downarrow & \swarrow^{l^{-1} \cdot \text{id}} \text{ (a)} & \downarrow \mu \\
& (1 \times [[ \Gamma ]]) \cdot [[ \Delta ] & \swarrow^{(! \times \text{id}) \cdot \text{id}} \\
& \downarrow \mu \text{ (b)} & \downarrow \mu \\
1 \cdot ([[ \Gamma ] \cdot [[ \Delta ]]) & \xleftarrow{! \cdot \text{id}} \text{ (c)} & [[ \Gamma ] \cdot ([[ \Gamma ] \cdot [[ \Delta ]]) \\
\downarrow \text{id} \cdot [[ M ] & \downarrow \text{id} \cdot [[ M ] & \downarrow \text{id} \cdot [[ M ] \\
1 \cdot \triangleright [[ \tau ] & \xleftarrow{! \cdot \text{id}} \text{ (e)} & [[ \Gamma ] \cdot \triangleright [[ \tau ] \\
\downarrow \eta & \swarrow^{\alpha} & \downarrow \alpha \\
& \triangleright (1 \times [[ \tau ]]) & \xleftarrow{! \cdot \text{id}} \text{ (f)} \triangleright ([[ \Gamma ] \times [[ \tau ]]) \\
& \downarrow \eta \text{ (g)} & \downarrow \triangleright \pi' \\
& \triangleright \pi' & \xleftarrow{! \cdot \text{id}} \text{ (h)} \triangleright [[ \tau ] \\
[[ \Gamma ] \cdot [[ \Delta ] & \xrightarrow{[[ M ]} & \triangleright [[ \tau ]
\end{array}$$

In the above diagram, each small diagram commutes because of:

- (a) the functoriality of the monoidal action and the fact that  $l \circ (! \times \text{id}) \circ \delta = \text{id}$ ,
- (b) the coherence for monoidal actions,
- (c) the naturality of  $\mu$  and the functoriality of the monoidal action,
- (d) the naturality of  $\eta$ ,
- (e) the functoriality of the monoidal action,
- (f) the naturality of  $\alpha$ ,
- (g) the coherence for  $\alpha$  and
- (h) the fact that  $\pi' \circ (! \times \text{id}) = \pi'$  and the functoriality of  $\triangleright$ .

The calculation for (comm. conv.) is more complicated and the involved diagram gets bigger than those for the above two, but they still are straightforward, and we omit them. ◀

**Proof of Example 9.3.** Let  $\mathcal{C}$  be a cartesian closed category and  $F: \mathcal{C} \rightarrow \mathcal{C}$  a lax monoidal functor. Let  $\mathcal{D}$  be the category defined by the following data:

$$\begin{aligned}
\text{ob } \mathcal{D} &:= \text{ob } \mathcal{C} \\
\mathcal{D}(X, Y) &:= \mathcal{C}(1, F(X \Rightarrow Y)) \\
(X \xrightarrow{\text{id}_X} X) &:= \left( 1 \xrightarrow{\iota} F1 \xrightarrow{F(\overline{\text{id}_X})} F(X \Rightarrow X) \right) \\
(Y \xrightarrow{g} Z) \circ (X \xrightarrow{f} Y) &:= 1 \cong 1 \times 1 \\
&\xrightarrow{f \times g} F(X \Rightarrow Y) \times F(Y \Rightarrow Z) \\
&\xrightarrow{\mu} F((X \Rightarrow Y) \times (Y \Rightarrow Z)) \\
&\xrightarrow{F(\text{comp})} F(X \Rightarrow Z)
\end{aligned}$$

where  $\iota$  and  $\mu$  are the morphisms required by the lax monoidality of  $F$ ,  $X \Rightarrow Y$  is the exponent in  $\mathcal{C}$ , and  $\text{comp}$  is the morphism  $\lambda(f, g) \cdot \lambda x. g(f(x))$ . In an abstract view,  $\mathcal{D}$  is described in terms of enriched categories. Given a (symmetric) monoidal closed category  $\mathcal{M}$ , there exists a 2-category  $\mathcal{M}\text{-Cat}$  of  $\mathcal{M}$ -enriched categories. Similarly, given (symmetric)

monoidal closed categories  $\mathcal{M}, \mathcal{M}'$  and a lax monoidal functor  $K: \mathcal{M} \rightarrow \mathcal{M}'$ , there exists a 2-functor  $K_*: \mathcal{M}\text{-Cat} \rightarrow \mathcal{M}'\text{-Cat}$  defined by change-of-base along  $K$ . We can then define  $\mathcal{D}$  by two successive applications of change-of-base  $\mathcal{C}(1, -)_* F_* \mathcal{C}$ , where  $\mathcal{C}(1, -): \mathcal{C} \rightarrow \mathbf{Sets}$  is the global section functor.

$\mathcal{D}$  has an  $\mathcal{C}$ -action defined by finite products:

$$\begin{aligned} X \cdot Y &:= X \times Y \\ (X \xrightarrow{f} X') \cdot (Y \xrightarrow{g} Y') &:= 1 \cong 1 \times 1 \\ &\xrightarrow{F\bar{f} \times g} F(X \Rightarrow X') \times F(Y \Rightarrow Y') \\ &\xrightarrow{\mu} F((X \Rightarrow X') \times (Y \Rightarrow Y')) \\ &\xrightarrow{F(\text{prod})} F(X \times Y \Rightarrow X' \times Y') \end{aligned}$$

where  $\text{prod}$  is the morphism  $\lambda\langle f, g \rangle. \lambda\langle x, y \rangle. \langle fx, gy \rangle$ .

Then there exists an identity-on-object functor  $J: \mathcal{C} \rightarrow \mathcal{D}$ , whose action on morphisms is given by:

$$J(X \xrightarrow{f} Y) := \left( 1 \xrightarrow{\iota} F1 \xrightarrow{F\bar{f}} F(X \Rightarrow Y) \right).$$

We show that  $J$  is strict equivariant, for which  $\phi_{X,Y}^J: X \cdot J(Y) \rightarrow J(X \times Y)$  is given by the identity. Here we only check that  $\phi^J$  is natural in both variables, by chasing the following diagrams.

$$\begin{array}{ccccc} 1 & \xrightarrow{\quad \iota \quad} & & & F1 \\ \cong \downarrow & & & & \downarrow F(\overline{f \times g}) \\ 1 \times 1 & \xrightarrow{\text{id} \times \iota} & 1 \times F1 & \xleftarrow{\cong} & \\ \iota \times \iota \downarrow & \swarrow \iota \times \text{id} & \downarrow \mu & \swarrow \cong & \\ F1 \times F1 & \xrightarrow{\quad \mu \quad} & F(1 \times 1) & \xleftarrow{\cong} & \\ F\bar{f} \times F\bar{g} \downarrow & & \downarrow F(\bar{f} \times \bar{g}) & & \\ F(X \Rightarrow X') \times F(Y \Rightarrow Y') & \xrightarrow{\quad \mu \quad} & F((X \Rightarrow X') \times (Y \Rightarrow Y')) & \xrightarrow{\text{prod}} & F(X \times Y \Rightarrow X' \times Y') \end{array}$$

In the above diagram, the bottom left composite is  $f \cdot Jg$  and the upper right is  $J(f \times g)$ . ◀

**Proof of Theorem 11.** Let  $\mathcal{T}$  be any theory of SEC. We define the monoidal category  $\mathcal{V}_{\mathcal{T}}$ , the category  $\mathcal{C}_{\mathcal{T}}$  with a  $\mathcal{V}_{\mathcal{T}}$ -action and the strong equivariant functor  $F: \mathcal{V}_{\mathcal{T}} \rightarrow \mathcal{C}_{\mathcal{T}}$ .

Firstly we define the cartesian category  $\mathcal{V}_{\mathcal{T}}$ .  $\mathcal{V}_{\mathcal{T}}$  has value types as its object and as its morphisms  $\tau_1 \rightarrow \tau_2$  equivalence classes of value terms  $[x: \tau_1 \vdash^{\vee} M: \tau_2]_{\mathcal{T}}$  derived under  $\mathcal{T}$ . We define the equivalence class by

$$\begin{aligned} [x: \tau_1 \vdash^{\vee} M: \tau_2]_{\mathcal{T}} &= [x': \tau_1 \vdash^{\vee} M': \tau_2]_{\mathcal{T}} \\ \iff x: \tau_1 \vdash^{\vee} M &=_{\tau_2} M'[x/x'] \text{ is derived.} \end{aligned}$$

The identity morphism  $\text{id}_{\tau}$  is  $[x: \tau \vdash^{\vee} x: \tau]_{\mathcal{T}}$ . The composition  $[x_2: \tau_2 \vdash^{\vee} M_3: \tau_3]_{\mathcal{T}} \circ [x_1: \tau_1 \vdash^{\vee} M_2: \tau_2]_{\mathcal{T}}$  is  $[x_1: \tau_1 \vdash^{\vee} M_3[M_2/x_2]: \tau_3]_{\mathcal{T}}$ . We can show that  $\mathcal{V}_{\mathcal{T}}$  has binary products and the terminal object. The binary products of  $\tau_1$  and  $\tau_2$  is  $\tau_1 \times \tau_2$  and the terminal object is 1.

We define  $\mathcal{C}_{\mathcal{T}}$  next.  $\mathcal{C}_{\mathcal{T}}$  has two kinds of objects:

- pairs  $(\tau \mid A)$  of a value type  $\tau$  and a computation type  $A$ , and
- value types  $\tau$ , which we write  $(\tau \mid \cdot)$ .

For morphisms, we have to be careful. At first, we define the equivalence class of typed computation terms as we do to define the morphisms in  $\mathcal{V}_{\mathcal{T}}$ .

$$\begin{aligned} [x: \tau \mid v: A \vdash^c N: A']_{\mathcal{T}} &= [x': \tau \mid v': A \vdash^v N': A']_{\mathcal{T}} \\ \iff x: \tau \mid v: A \vdash^c N &=_{A'} N'[v/v', x/x'] \text{ is derived} \end{aligned}$$

$$\begin{aligned} [x: \tau \mid \cdot \vdash^c N: A']_{\mathcal{T}} &= [x': \tau \mid \cdot \vdash^v N': A']_{\mathcal{T}} \\ \iff x: \tau \mid \cdot \vdash^c N &=_{A'} N'[x/x'] \text{ is derived} \end{aligned}$$

In the sequel, the subscript  $\mathcal{T}$  for the equivalence classes will be omitted for simplicity.

To define what is a morphism  $(\tau \mid \Delta) \rightarrow (\tau' \mid \Delta')$ , we have to be careful of whether  $\Delta$  and  $\Delta'$  are empty or not.

- When  $\Delta'$  is not empty, say  $\Delta' = A'$ , a morphism  $(\tau \mid \Delta) \rightarrow (\tau' \mid A')$  is a pair of equivalence classes of a typed term that has the following form.

$$([x: \tau \vdash^v M': \tau'], [x: \tau \mid \Delta \vdash^c N: A'])$$

- When both  $\Delta$  and  $\Delta'$  are empty, a morphism  $(\tau \mid \cdot) \rightarrow (\tau' \mid \cdot)$  is an equivalence class of typed value terms which has the following form.

$$[x: \tau \vdash^v M': \tau']$$

- When  $\Delta$  isn't empty and  $\Delta'$  is empty,  $\mathcal{C}_{\mathcal{T}}((\tau \mid \Delta), (\tau' \mid \Delta'))$  is empty.

The identity morphism  $\text{id}: (\tau \mid \cdot) \rightarrow (\tau \mid \cdot)$  is  $[x: \tau \vdash^v x: \tau]_{\mathcal{T}}$  and  $\text{id}: (\tau \mid A) \rightarrow (\tau \mid A)$  is  $([x: \tau \vdash^v x: \tau]_{\mathcal{T}}, [x: \tau \mid v: A \vdash^c v: A])$ . Next, we consider the composition of the morphism  $f: (\tau \mid \Delta) \rightarrow (\tau' \mid \Delta')$  and  $g: (\tau' \mid \Delta') \rightarrow (\tau'' \mid \Delta'')$ . Note that there are three cases to consider.

- When  $\Delta$ ,  $\Delta'$  and  $\Delta''$  are empty. Let

$$\begin{aligned} f &= [x: \tau \vdash^v M: \tau']_{\mathcal{T}} \\ g &= [x': \tau' \vdash^v M': \tau'']_{\mathcal{T}}, \end{aligned}$$

then we define  $g \circ f = [x: \tau \vdash^v M'[M/x']: \tau'']_{\mathcal{T}}$ .

- When  $\Delta$  and  $\Delta'$  is empty and  $\Delta''$  is not empty. Let

$$\begin{aligned} f &= [x: \tau \vdash^v M': \tau'] \\ g &= ([x': \tau' \vdash^v M'': \tau''], [x': \tau' \mid \cdot \vdash^c N'': A'']), \end{aligned}$$

then we define

$$g \circ f = ([x: \tau \vdash^v M''[M'/x']: \tau''], [x: \tau \mid \cdot \vdash^c N''[M'/x']: A'']).$$

- When  $\Delta$  is empty and  $\Delta'$  and  $\Delta''$  is not empty. Let

$$\begin{aligned} f &= ([x: \tau \vdash^v M': \tau'], [x: \tau \mid \Delta \vdash^c N': A']) \\ &= ([x': \tau' \vdash^v M'': \tau''], [x': \tau' \mid v': A' \vdash^c N'': A'']), \end{aligned}$$

then we define

$$g \circ f = ([x: \tau \vdash^v M''[M'/x']: \tau''], [x: \tau \mid \Delta \vdash^c N''[M'/x', N'/v']: A''])$$

Up to here, we define two categories  $\mathcal{V}_{\mathcal{T}}$  and  $\mathcal{C}_{\mathcal{T}}$ . Next, we define a monoidal action  $(-) \cdot (-): \mathcal{V}_{\mathcal{T}} \times \mathcal{C}_{\mathcal{T}} \rightarrow \mathcal{C}_{\mathcal{T}}$ . For objects  $\tau$  in  $\mathcal{V}_{\mathcal{T}}$  and  $(\tau' \mid \Delta)$  in  $\mathcal{C}_{\mathcal{T}}$ , we define  $\tau \cdot (\tau' \mid \Delta) = (\tau \times \tau' \mid \Delta)$ . For morphisms  $[x: \tau_1 \vdash^{\vee} M: \tau_2]_{\mathcal{T}}: \tau_1 \rightarrow \tau_2$  in  $\mathcal{V}_{\mathcal{T}}$  and  $([x': \tau_3 \vdash^{\vee} M': \tau_4]_{\mathcal{T}}, [y: \tau_3 \mid \Delta \vdash^{\text{c}} N: A]_{\mathcal{T}}): (\tau_3 \mid \Delta) \rightarrow (\tau_4 \mid A)$ , we define

$$\begin{aligned} & [x: \tau_1 \vdash^{\vee} M: \tau_2]_{\mathcal{T}} \cdot ([x': \tau_3 \vdash^{\vee} M': \tau_4]_{\mathcal{T}}, [y: \tau_3 \mid \Delta \vdash^{\text{c}} N: A]_{\mathcal{T}}) \\ &= ([z: \tau_1 \times \tau_3 \vdash^{\vee} \langle M[\pi_1(z)/x], M'[\pi_2(z)/x'] \rangle: \tau_2 \times \tau_4]_{\mathcal{T}}, \\ & \quad [w: \tau_1 \times \tau_3 \mid \Delta \vdash^{\text{c}} N/[\pi_2(w)/y]: A]_{\mathcal{T}}). \end{aligned}$$

For morphisms  $[x: \tau_1 \vdash^{\vee} M: \tau_2]_{\mathcal{T}}: \tau_1 \rightarrow \tau_2$  in  $\mathcal{V}_{\mathcal{T}}$  and  $[x': \tau_3 \vdash^{\vee} M': \tau_4]_{\mathcal{T}}: (\tau_3 \mid \cdot) \rightarrow (\tau_4 \mid \cdot)$ , the action is defined in a similar way. It is straightforward to show that  $(-) \cdot (-)$  preserves identities and compositions.

From the above, we can get a functor  $(-) \cdot (-): \mathcal{V}_{\mathcal{T}} \times \mathcal{C}_{\mathcal{T}} \rightarrow \mathcal{C}_{\mathcal{T}}$ . In order to make this functor a  $\mathcal{V}_{\mathcal{T}}$ -action, we have to define the coherence natural isomorphism  $\eta_{(\tau \mid \Delta)}: 1 \cdot (\tau \mid \Delta) \rightarrow (\tau \mid \Delta)$  and  $\mu_{\tau, \tau', (\tau'' \mid \Delta)}: (\tau \times \tau') \cdot (\tau'' \mid \Delta) \rightarrow \tau \cdot (\tau' \cdot (\tau'' \mid \Delta))$  for this monoidal action.

As to  $\eta_{(\tau \mid \Delta)}$ ,

- if  $\Delta$  is empty, we define  $\eta_{(\tau \mid \cdot)}$  to be  $[x: 1 \times \tau \vdash^{\vee} \pi_2(x): \tau]_{\mathcal{T}}$ , and
- if  $\Delta$  isn't empty, say  $\Delta = A$ , we define  $\eta_{(\tau \mid A)}$  to be

$$([x: 1 \times \tau \vdash^{\vee} \pi_2(x): \tau]_{\mathcal{T}}, [x: 1 \times \tau \mid v: A \vdash^{\text{c}} v: A]).$$

As to  $\mu_{(\tau \mid \Delta)}$ ,

- if  $\Delta$  is empty, we define  $\mu_{\tau, \tau', (\tau'' \mid \cdot)}$  to be

$$[x: (\tau \times \tau') \times \tau'' \vdash^{\vee} \langle \pi_1(\pi_1(x)), \langle \pi_2(\pi_1(x)), \pi_2(x) \rangle \rangle: \tau \times (\tau' \times \tau'')]_{\mathcal{T}},$$

and

- if  $\Delta$  isn't empty, say  $\Delta = A$ , we define  $\mu_{\tau, \tau', (\tau'' \mid A)}$  to be

$$\begin{aligned} & ([x: (\tau \times \tau') \times \tau'' \vdash^{\vee} \langle \pi_1(\pi_1(x)), \langle \pi_2(\pi_1(x)), \pi_2(x) \rangle \rangle: \tau \times (\tau' \times \tau'')]_{\mathcal{T}}, \\ & [x: (\tau \times \tau') \times \tau'' \mid v: A \vdash^{\text{c}} v: A]). \end{aligned}$$

The coherence condition for these natural isomorphism reduces to the monoidality of  $\mathcal{V}_{\mathcal{T}}$ .

Next, We define an equivariant functor  $F: \mathcal{V}_{\mathcal{T}} \rightarrow \mathcal{C}_{\mathcal{T}}$  and its coherent natural transformation  $\alpha: (-) \cdot F(-) \rightarrow F((-) \cdot (-))$ . For object  $\tau$  in  $\mathcal{V}_{\mathcal{T}}$ , we define  $F\tau = (\tau \mid \cdot)$  and  $F$  is identity on morphisms.  $F$  is clearly a functor. Moreover,  $F$  is equivariant with an identity natural transformation.  $\blacktriangleleft$

**Proof of Example 14.** We show that an equivariant left adjoint is always strong equivariant, which we believe is folklore. Notice the similarity with a fact about monoidal functors: a monoidal left adjoint is always strong monoidal.

Let  $F \dashv G$  an  $\mathcal{M}$ -equivariant adjunction. We define  $\psi^F: F(m \cdot x) \rightarrow m \cdot F(x)$  to be the mate of  $m \cdot x \xrightarrow{m \cdot \eta} m \cdot GFx \xrightarrow{\phi^G} G(m \cdot Fx)$ . Check the following diagrams to see that  $\psi^F$  is



an inverse of  $\phi^F$ .

$$\begin{array}{ccccc}
 & & \psi^F & & \\
 & \searrow & \curvearrowright & \searrow & \\
 F(m \cdot x) & \xrightarrow{F(m \cdot \eta)} & F(m \cdot GFx) & \xrightarrow{F\phi^G} & FG(m \cdot Fx) & \xrightarrow{\epsilon} & m \cdot Fx \\
 & \searrow & \downarrow F\phi^{GF} & \downarrow FG\phi^F & \downarrow \phi^F & & \\
 & & FGF(m \cdot x) & \xrightarrow{\epsilon} & F(m \cdot x) & & \\
 & \nearrow & \curvearrowleft & \nearrow & & & \\
 & & \text{id} & & & & 
 \end{array}$$

$$\begin{array}{ccccc}
 & & m \cdot \text{id} & & \\
 & \searrow & \curvearrowright & \searrow & \\
 m \cdot Fx & \xrightarrow{m \cdot F\eta} & m \cdot FGFx & \xrightarrow{m \cdot \epsilon} & m \cdot Fx \\
 \phi^F \downarrow & & \downarrow \phi^F & \downarrow \phi^{FG} & \\
 F(m \cdot x) & \xrightarrow{F(m \cdot \eta)} & F(m \cdot GFx) & \xrightarrow{F\phi^G} & FG(m \cdot Fx) & \xrightarrow{\epsilon} & m \cdot Fx \\
 & \searrow & \downarrow \phi^F & \downarrow \phi^{FG} & \downarrow \phi^F & & \\
 & & FGF(m \cdot x) & \xrightarrow{\epsilon} & F(m \cdot x) & & \\
 & \nearrow & \curvearrowleft & \nearrow & & & \\
 & & \psi^F & & & & 
 \end{array}$$

**Proof of Remark 16.** The Curry-Howard-Lambek correspondence usually refers to the following equivalence in a suitable 2-category.

$$\mathcal{M} \xrightarrow{\cong} \text{Syn}(\text{Lang}(\mathcal{M}))$$

where  $\mathcal{M}$  is a categorical model and  $\text{Syn}$  and  $\text{Lang}$  are operators giving the term model and the internal language.

To adapt this to models of SEC, we slightly modify the term model  $\text{Syn}(\mathcal{T})$  in Theorem 11 as follows:

$$\text{Syn}(\mathcal{T})_F: \mathcal{V} \xrightarrow{\cong} \text{Syn}(\mathcal{T})_{\mathcal{V}} \xrightarrow{(*)} \text{Syn}(\mathcal{T})_{\mathcal{C}}$$

where  $(*)$  is the term model presented in Theorem 11 and the first equivalence is the Curry-Howard-Lambek correspondence of algebraic theory (with finite products).

Then the question is reduced to existence of the following equivalence.

$$F \xrightarrow{\cong} \text{Syn}(\mathcal{T}_F)$$

We want to make the following  $(H, \theta)$  from  $F$  to  $\text{Syn}(\mathcal{T}_F)$  the witness of the above equivalence.

$$\begin{aligned}
 H(A) &:= (\cdot \mid \ulcorner A \urcorner) \\
 \theta_{\tau}: (\ulcorner \tau \urcorner \mid \cdot) &\rightarrow (\cdot \mid \ulcorner F\tau \urcorner) := [x: \ulcorner \tau \urcorner \mid \cdot \vdash^c \ulcorner \text{id}_{F\tau} \urcorner(x): \ulcorner F\tau \urcorner]
 \end{aligned}$$

Then the morphism in the reverse direction  $(H', \theta')$  will be

$$\begin{aligned}
 H'((\Gamma \mid \Delta)) &:= \llbracket \Gamma \mid \Delta \rrbracket_F \\
 \theta'_{\tau}: F\tau &\rightarrow F\tau := \text{id}_{F\tau}.
 \end{aligned}$$

If these form an equivalence, there should be a 2-cell  $\alpha : H \circ H' \rightarrow 1$ . However, this is impossible. For example, the  $(\tau | \cdot)$  component of  $\alpha$  has the following type:

$$\alpha_{(\tau | \cdot)} : (\cdot | \ulcorner F \llbracket \tau \rrbracket \urcorner) \rightarrow (\tau | \cdot).$$

By the definition of the term model, there is no such morphism.  $\blacktriangleleft$

**Proof of Proposition 20.** Define  $m \cdot (D, C, f)$  to be  $(m \cdot D, m \cdot C, \phi \circ (m \cdot f))$  for objects  $m$  in  $\mathcal{M}$  and  $(D, C, f : D \rightarrow \Gamma C)$  in  $\mathcal{D} \downarrow \Gamma$ , and  $a \cdot (d, c)$  to be  $(a \cdot d, a \cdot c)$  for morphisms  $a : m \rightarrow m'$  in  $\mathcal{M}$  and  $(d, c) : (D, C, f) \rightarrow (D', C', f')$  in  $\mathcal{D} \downarrow \Gamma$ . It follows that  $(a \cdot d, a \cdot c)$  is a morphism  $(m \cdot D, m \cdot C, \phi \circ (m \cdot f)) \rightarrow (m' \cdot D', m' \cdot C', \phi \circ (m' \cdot f'))$  in  $\mathcal{D} \downarrow \Gamma$  from the diagram below.

$$\begin{array}{ccc} m \cdot D & \xrightarrow{a \cdot d} & m' \cdot D' \\ m \cdot f \downarrow & & \downarrow m' \cdot f' \\ m \cdot \Gamma C & \xrightarrow{a \cdot \Gamma c} & m' \cdot \Gamma C' \\ \phi \downarrow & & \downarrow \phi \\ \Gamma(m \cdot C) & \xrightarrow{\Gamma(a \cdot c)} & \Gamma(m' \cdot C') \end{array}$$

The upper rectangle commutes by the fact that  $(d, c)$  is a morphism in  $\mathcal{D} \downarrow \Gamma$  and the (bi)factoriality of  $(-) \cdot (-)$ , and the lower one commutes by the naturality of  $\phi$ . It is straightforward to see  $(-) \cdot (-) : \mathcal{M} \times (\mathcal{D} \downarrow \Gamma) \rightarrow \mathcal{D} \downarrow \Gamma$  is indeed an  $\mathcal{M}$ -action on  $\mathcal{D} \downarrow \Gamma$ .

It is also straightforward to see that the projection functor  $\pi : \mathcal{D} \downarrow \Gamma \rightarrow \mathcal{C}$  is strict equivariant.  $\blacktriangleleft$

**Proof of Proposition 21.** The upper left category  $\mathcal{G}$  has pairs  $(C, X)$  of objects in  $\mathcal{C}$  and  $\mathcal{E}$  such that  $\Gamma C = pX$  for its objects, and pairs  $(f, x)$  of morphisms in  $\mathcal{C}$  and  $\mathcal{E}$  such that  $\Gamma f = px$  for its morphisms.

We can define an  $\mathcal{M}$ -action on  $\mathcal{G}$  by using universalities of opcartesian morphisms as follows. Let  $m$  be any object in  $\mathcal{M}$  and  $(C, X)$  be any object in  $\mathcal{G}$ . We define  $m \cdot (C, X)$  to be  $(m \cdot C, (\phi_{m,C})!(m \cdot X))$ . This definition is well-defined i.e.  $\Gamma(m \cdot C) = p((\phi_{m,C})!(m \cdot X))$  holds. This follows from the fact that  $\Gamma C = pX$  holds,  $m \cdot pX = p(m \cdot X)$  holds because  $p$  is strict equivariant and  $\phi_{m,C}$  has  $m \cdot \Gamma C$  as its domain.

$$\begin{array}{ccc} \phi_{m,c} : p(m \cdot X) & \rightarrow & \Gamma(m \cdot C) \\ \underline{\phi_{m,c}}(m \cdot X) : m \cdot X & \rightarrow & (\phi_{m,C})!(m \cdot X) \end{array}$$

Let  $a : m \rightarrow m'$  be any morphism of  $\mathcal{M}$  and  $(f, x) : (C, X) \rightarrow (C', X')$  be any morphism of  $\mathcal{G}$ . We define  $a \cdot (f, x)$  by means of the universality of  $\underline{\phi_{m,C}}(m \cdot X)$ : we define  $a \cdot (f, x)$  to be  $(a \cdot f, u)$  where  $u$  is the unique morphism which makes the left diagram commute and satisfies  $pu = \Gamma(a \cdot f)$ . Note that the lower right diagram commutes by the naturality of  $\phi$ , and so the upper right one does.

$$\begin{array}{ccc}
 m \cdot X \xrightarrow{\underline{\phi}_{m,C}(m \cdot X)} (\phi_{m,C})!(m \cdot X) & & p(m \cdot X) \xrightarrow{\phi_{m,C}} \Gamma(m \cdot C) \\
 a \cdot x \downarrow & \xrightarrow{p} & \downarrow p(a \cdot x) \\
 m' \cdot X' \xrightarrow{\underline{\phi}_{m',C'}(m' \cdot X')} (\phi_{m',C'})!(m' \cdot X') & & p(m' \cdot X') \xrightarrow{p(\underline{\phi}_{m',C'}(m' \cdot X'))} \Gamma(m' \cdot C') \\
 & & \downarrow \Gamma(a \cdot f) \\
 & & \Gamma(m' \cdot C') \\
 & & \parallel \\
 & & m \cdot \Gamma C \xrightarrow{\phi_{m,C}} \Gamma(m \cdot C) \\
 & & a \cdot \Gamma f \downarrow \\
 & & m' \cdot \Gamma C' \xrightarrow{\phi_{m',C'}} \Gamma(m' \cdot C')
 \end{array}$$

The functoriality of this monoidal action follows from the universalities of  $\underline{\phi}_{m,C}$ s.

At last, we define the coherent natural isomorphisms for this monoidal action. Let  $(C, X)$  be any object in  $\mathcal{G}$ . Consider the following diagram.

$$\begin{array}{ccc}
 1 \cdot \Gamma C & \xrightarrow{\phi_{1,C}} & \Gamma(1 \cdot C) \\
 \eta_{\Gamma C} \searrow & & \swarrow \Gamma \eta_C \\
 & \Gamma C &
 \end{array}$$

This diagram commutes because it is one of those for coherence for  $\phi$ . In addition,  $\eta_{\Gamma C} = p\eta_X$  holds because  $\Gamma C = pX$  holds and  $p$  is strict equivariant. With the universality of  $\underline{\phi}_{1,C}$ , the unique morphism  $u$  which makes the following diagram commute and satisfies  $pu = \overline{\Gamma} \eta_C$  is obtained.

$$\begin{array}{ccc}
 1 \cdot X & \xrightarrow{\underline{\phi}_{1,C}(1 \cdot X)} & (\phi_{1,C})!(1 \cdot X) \\
 \eta_X \searrow & & \swarrow u \\
 & X &
 \end{array}$$

We define  $\eta_{(C,X)}$  to be  $(\eta_C, u)$ .

In the same vein,  $\mu_{m,m',(C,X)}$  is defined to be  $(\mu_{m,m',C}, u)$  in which  $u$  is the unique morphism which makes the following diagram commute and satisfies  $pu = \Gamma \mu_{m,m',C}$ .

$$\begin{array}{ccc}
(m \otimes m') \cdot X & \xrightarrow{\underline{\phi_{m \otimes m', C}}((m \otimes m') \cdot X)} & (\phi_{m \otimes m', C})!((m \otimes m') \cdot X) \\
\mu_{m, m', X} \downarrow & & \downarrow u \\
m \cdot (m' \cdot X) & & \\
\downarrow m \cdot \underline{\phi_{m', C}}(m' \cdot X) & & \\
m \cdot (\phi_{m', C})!(m' \cdot X) & \xrightarrow{\underline{\phi_{m, m' \cdot C}}(m \cdot (\phi_{m', C})!(m' \cdot X))} & (\phi_{m, m' \cdot C})!(m \cdot (\phi_{m', C})!(m' \cdot X))
\end{array}$$

Note that the coherence diagrams for  $\phi$  and  $\mu$  is obtained by applying  $p$  to this whole diagram.

The coherent natural transformations  $\eta$  and  $\mu$  defined above are isomorphisms, using (\*), by several properties of opcartesian morphisms. The coherence conditions are reduced to those for the action on  $\mathcal{E}$  by using the universalities of  $\underline{\phi}$ s.  $\blacktriangleleft$

**Proof of Proposition 23.** Let  $\phi$  be the coherent natural transformation associated to  $\Gamma$ . This proposition can be proved in a similar way in Proposition 21. The  $\mathcal{M}$ -action on  $\mathcal{G}$  is defined by using the opcartesian lifting of  $\phi^{\Gamma^{-1}}: \Gamma(m \cdot C) \rightarrow m \cdot \Gamma C$ .

We present only the definition of the  $\mathcal{M}$ -action on  $\mathcal{G}$ . For any object  $m$  in  $\mathcal{M}$  and  $(C, X)$  in  $\mathcal{G}$ ,  $m \cdot (C, X)$  is defined to be  $(m \cdot C, (\phi_{m, C}^{-1})^*(m \cdot X))$ . The following figure states that this definition is well-defined.

$$\begin{array}{ccc}
(\phi_{m, C}^{-1})^*(m \cdot X) & \xrightarrow{\overline{(\phi_{m, C}^{-1})}(m \cdot X)} & m \cdot X \\
& & \uparrow \text{cartesian} \\
& & \uparrow \text{lifting} \\
\Gamma(m \cdot C) & \xrightarrow{\phi_{m, C}^{-1}} & m \cdot \Gamma C \\
& & \parallel \\
& & m \cdot pX \\
& & \parallel \\
& & p(m \cdot X)
\end{array}$$

Because  $\phi_{m, C}$  is an isomorphism,  $\overline{\phi_{m, C}^{-1}}$  is also an isomorphism, and thus  $m' \cdot \overline{\phi_{m, C}^{-1}}$  is also an isomorphism. Therefore,  $m' \cdot \overline{\phi_{m, C}^{-1}}$  is also cartesian because any isomorphism is cartesian. This is why the condition (\*) can be omitted.  $\blacktriangleleft$

**Proof of Proposition 25.** By proposition 20. We define  $L$  as follows.

$$\begin{aligned}
Lv &= (F'v, Fv, \theta_v: F'v \rightarrow HFv) \\
L(a: v \rightarrow u) &= (F'a, Fa): \theta_v \rightarrow \theta_u
\end{aligned}$$

where  $v$  is any object in  $\mathcal{V}$  and  $a: v \rightarrow u$  is any morphism in  $\mathcal{V}$ . It follows from the naturality of  $\theta$  that  $L\alpha$  above is a morphism in  $\mathcal{C}' \downarrow H$ .

$$\begin{array}{ccc}
F'v & \xrightarrow{F'\alpha} & F'u \\
\theta_v \downarrow & & \downarrow \theta_u \\
HFv & \xrightarrow{HF\alpha} & HFu
\end{array}$$

Then, we define  $\phi^L$  to be  $(\phi^{F'}, \phi^F)$ . The component  $\phi_{v,u}^L = (\phi_{v,u}^{F'}, \phi_{v,u}^F)$  is indeed a morphism  $v \cdot \theta_u \rightarrow \theta_v \cdot u$  in  $\mathcal{C}' \downarrow H$  by the definition 15.

The coherent natural transformation  $\phi^L$  is an isomorphism and satisfies the coherence since so  $m_F$  and  $m_{F'}$  are and do, and the composition in  $\mathcal{C}' \downarrow H$  is defined using that in  $\mathcal{C}'$  in a componentwise way.  $\blacktriangleleft$

**Proof of Proposition 31.** We first define the functor  $p_T$  by

$$\begin{aligned}
p_T(X) &:= pX \\
p_T(f: X \rightarrow \tilde{T}Y) &:= pf: pX \rightarrow TpY.
\end{aligned}$$

Notice  $Tp = p\tilde{T}$  holds because  $(T, \tilde{T})$  is a monad over  $p$ . The functoriality of  $p_T$  follows from  $p\tilde{\mu} = \mu_p$  and  $p\tilde{\eta} = \eta_p$ ; for example,  $p_T$  preserves identities by the latter equation. For the  $p_T$ -cartesian lifting of  $u: I \rightarrow TpY$ , we can take  $p$ -cartesian lifting  $\bar{u}: u^*(\tilde{T}Y) \rightarrow \tilde{T}Y$  of  $u$ . This is indeed  $p_T$ -cartesian by that fact that  $\tilde{T}$  is fibred and  $\tilde{\mu}$  is  $p$ -cartesian.

Next, consider  $\mathcal{B} \times_{\mathcal{B}_T} \mathcal{E}_{\tilde{T}}$ . Its object is a pair  $(K, X)$  satisfying  $K = pX$  and its morphism is a pair  $(u, f)$  satisfying  $\eta_p \circ u = pf$ . Because  $\eta_p = p\tilde{\eta}$  holds and  $\tilde{\eta}$  is cartesian, for each  $f$  there exists a unique  $h$  satisfying  $\tilde{\eta} \circ h = f$  and  $ph = u$ . Using these facts, the functor  $F: \mathcal{B} \times_{\mathcal{B}_T} \mathcal{E}_{\tilde{T}} \rightarrow \mathcal{E}$  defined as follows gives the isomorphism  $\mathcal{B} \times_{\mathcal{B}_T} \mathcal{E}_{\tilde{T}} \cong \mathcal{E}$  holds;

$$\begin{aligned}
F(X) &= (pX, X) \\
F(f) &= (pf, \tilde{\eta} \circ f).
\end{aligned}$$

In addition,  $F$  makes the following triangles commute.

$$\begin{array}{ccc}
& \mathcal{E} & \\
p \swarrow & \downarrow F \cong & \searrow J \\
\mathcal{B} & \mathcal{B} \times_{\mathcal{B}_T} \mathcal{E}_{\tilde{T}} & \mathcal{E}_{\tilde{T}}
\end{array}$$

Finally, we move on to the latter part of the proposition. At first, notice that the tensor in  $\mathcal{E}$  can be extended to the  $\mathcal{E}$ -action on  $\mathcal{E}_{\tilde{T}}$  by using the strength  $\tilde{t}$  of  $\tilde{T}$ . In a similar way,  $\mathcal{B}_T$  has an  $\mathcal{E}$ -action by  $X \cdot K = pX \otimes_{\mathcal{B}} K$  and

$$(f: X \rightarrow Y) \cdot (g: K \rightarrow TL) = ((pX \otimes_{\mathcal{B}} K) \xrightarrow{pf \otimes_{\mathcal{B}} g} pY \otimes_{\mathcal{B}} TK \xrightarrow{\tilde{t}} T(pX \otimes_{\mathcal{B}} L)).$$

It is straightforward to show that  $p_T$  preserves this action strictly.  $\blacktriangleleft$

The following proposition is claimed in the text right after Proposition 23.

**► Proposition 38** (Constructions of Proposition 21 and 23 coincide). *Suppose  $p$  is an opfibration in addition to the assumption in Proposition 23. The category  $\mathcal{G}$  for pullback has two kinds of  $\mathcal{M}$ -actions by Proposition 21 and 23. To distinguish these, we write  $\mathcal{G}$  and  $\mathcal{G}'$  for them.*

There are equivariant functors  $H: \mathcal{G} \rightarrow \mathcal{G}'$  and  $L: \mathcal{G}' \rightarrow \mathcal{G}$  such that  $FG$  and  $GF$  are identity functors as equivariant functors. In other words,  $FG$  and  $GF$  are identity functors in the (2-)category of  $\mathcal{M}$ -actegories and equivariant functors.

**Proof.** Notice that  $\mathcal{G}$  and  $\mathcal{G}'$  are the same as categories but different as  $\mathcal{M}$ -actegories. Therefore,  $H$  and  $L$  may be identity functors. The coherent natural isomorphisms for  $H$  and  $L$  are defined straightforwardly. ◀

**Proof of non/fibredness of  $T^{\top\top}$ .** At first, recall that a morphism in  $\text{Sub}(\mathbf{Sets})$  is cartesian if and only if the corresponding square is pullback.

■ fibredness of exception

When  $TX = X \uplus E$  and  $(S \subseteq TR)$  is  $(1 \subseteq 1 \uplus E)$ ,  $T^{\top\top}(P \subseteq X) = P \subseteq X \uplus E$  holds for any  $(P \subseteq X)$ . Consider the pullback square on the left-hand side. The upper left corner  $f^*(Q)$  is the inverse image of  $Q$  by  $f$ . The square on the right-hand is also pullback because  $(f \uplus E)^*(Q) = f^*(Q)$  holds. Therefore (the underlying functor of)  $T^{\top\top}$  is fibred.

$$\begin{array}{ccc} f^*(Q) & \longrightarrow & Q \\ \downarrow \lrcorner & & \downarrow \\ X & \xrightarrow{f} & Y \end{array} \xrightarrow{T^{\top\top}} \begin{array}{ccc} f^*(Q) & \longrightarrow & Q \\ \downarrow \lrcorner & & \downarrow \\ X \uplus E & \xrightarrow{f \uplus E} & Y \uplus E \end{array}$$

It is easy to see that the unit and multiplication are cartesian.

$$\begin{array}{ccc} P & \longrightarrow & P \\ \downarrow \lrcorner & & \downarrow \\ X & \xrightarrow{\eta} & X \uplus E \end{array} \quad \begin{array}{ccc} P & \longrightarrow & P \\ \downarrow \lrcorner & & \downarrow \\ X \uplus E \uplus E & \xrightarrow{\mu} & X \uplus E \end{array}$$

■ fibredness of nondeterminism

When  $(S \subseteq TR)$  is  $(1 \subseteq 2 = \mathcal{P}_{\text{fin}}(1))$ , we can calculate:  $T^{\top\top}(P \subseteq X) = \{m \in \mathcal{P}_{\text{fin}}(X) \mid \exists x \in m.P(x)\}$ . For  $X$  and  $Q \subseteq Y$ , consider the following pullback square.

$$\begin{array}{ccc} \mathbb{P} & \longrightarrow & T^{\top\top}(Q) \\ \downarrow \lrcorner & & \downarrow \\ \mathcal{P}_{\text{fin}}(X) & \xrightarrow{\mathcal{P}_{\text{fin}}(f)} & \mathcal{P}_{\text{fin}}(Y) \end{array}$$

We can confirm that  $\mathbb{P}$  is given by  $T^{\top\top}(f^*(Q))$  by the following calculation.

$$\begin{aligned} m \in \mathbb{P} &\iff \exists y \in \mathcal{P}_{\text{fin}}(f)(m).Q(y) \\ &\iff \exists x \in m.Q(f(x)) \\ &\iff \exists x \in m.x \in f^*(Q) \\ &\iff m \in T^{\top\top}(f^*(Q)) \end{aligned}$$

Therefore, the underlying functor of  $T^{\top\top}$  preserves cartesian morphisms. The unit is cartesian because  $\eta(x) = \{x\} \in T^{\top\top}(P) = \{m \in \mathcal{P}_{\text{fin}}(X) \mid \exists x \in m.P(x)\}$  if and only if  $x \in P$  holds. To see that the multiplication is cartesian, again consider a pullback diagram as follows.

$$\begin{array}{ccc} \mathbb{P} & \longrightarrow & T^{\top\top}(P) \\ \downarrow \lrcorner & & \downarrow \\ \mathcal{P}_{\text{fin}}(\mathcal{P}_{\text{fin}}(X)) & \xrightarrow{\mu} & \mathcal{P}_{\text{fin}}(X) \end{array}$$

Let us calculate  $\mathbb{P}$ :

$$\begin{aligned}
M \in \mathbb{P} &\iff \mu(M) \in T^{\top\top}(P) \\
&\iff \exists x \in \mu(M).P(x) \\
&\iff \exists x \in \{x \in X \mid \exists m \in M.x \in m\}.P(x) \\
&\iff \exists m \in M.\exists x \in m.P(x) \\
&\iff \exists m \in M.m \in T^{\top\top}(P) \\
&\iff M \in T^{\top\top}(T^{\top\top}(P)).
\end{aligned}$$

Therefore, the multiplication is cartesian.

■ non-fibredness of side-effect

For the side-effect monad  $T(X) := (A \times X)^A$  and  $(S \subseteq T(R)) := (B \rightrightarrows B \tilde{\times} 1 \subseteq T1)$  for some  $\emptyset \subsetneq B \subsetneq A$ ,  $T^{\top\top}(P)$  is given by  $B \rightrightarrows B \tilde{\times} P$ , where for  $(P \subseteq X)$  and  $(Q \subseteq Y)$ ,  $P \rightrightarrows Q := \{f: X \rightarrow Y \mid \forall x \in X.P(x) \Rightarrow Q(f(x))\}$  and  $P \tilde{\times} Q := \{(x, y) \in X \times Y \mid P(x) \wedge Q(y)\}$ . Let  $A = 2 = \{0, 1\}$  and  $B = 1 = \{0\}$ . Consider the pullback of the multiplication.

$$\begin{array}{ccc}
\mathbb{P} & \longrightarrow & 1 \rightrightarrows 1 \tilde{\times} P \\
\downarrow & \lrcorner & \downarrow \\
(2 \times (2 \times X)^2)^2 & \xrightarrow{\mu} & (2 \times X)^2
\end{array}$$

We will see that  $\mathbb{P}$  is not  $T^{\top\top}(T^{\top\top}(P))$ . We fix  $(P \subseteq X) := (1 \subseteq 1)$ . Let  $f$  be an element in  $(2 \times (2 \times X)^2)^2$  defined as follows.

$$f(s) := (1, \lambda s'.(0, 0))$$

$\mu$  sends this  $f$  to  $\lambda s.(0, 0)$ . Clearly,  $\mu(f)$  is in  $1 \rightrightarrows 1 \tilde{\times} P$ . Therefore,  $f$  is in  $\mathbb{P}$ . However  $f$  is not in  $T^{\top\top}(T^{\top\top}(P))$ , which concludes that  $T^{\top\top}(T^{\top\top}(P)) \neq \mathbb{P}$ . (Here we assume that  $\mathbb{P} \subseteq (2 \times (2 \times X)^2)^2$  w.l.o.g.)

■ non-fibredness of continuation

When  $TX = 2^{2^X}$ ,  $R = \emptyset$  and  $(S \subseteq 2^{2^0}) = (1 \subseteq 2)$ ,

$$T(P \subseteq X) = \{m \subseteq \mathcal{P}(X) \mid \forall P \subseteq C. C \in m\}$$

holds for any  $(P \subseteq X)$ .

In the sequel, we identify  $2^{2^{\mathbb{N}}}$  with the set of sets of real numbers in  $[0, 1]$ . Consider the left pullback square where  $0: \mathbb{N} \rightarrow \mathbb{N}$  is a constant map sending every number to 0. and apply  $T^{\top\top}$  to the diagram.

$$\begin{array}{ccc}
\emptyset \longrightarrow \emptyset & & \tilde{T}\emptyset \longrightarrow \tilde{T}\emptyset \\
\downarrow \lrcorner \downarrow & & \downarrow \quad \quad \downarrow \\
\mathbb{N} \xrightarrow{0} \mathbb{N} & \xrightarrow{T^{\top\top}} & \mathcal{P}([0, 1]) \xrightarrow{T0} \mathcal{P}([0, 1])
\end{array}$$

In the right square, the nodes and edges are determined as follows:

$$T(0)(U) = \begin{cases} [0, 1] & (0 \in U \text{ and } 1 \in U) \\ [0, 1/2) & (0 \in U \text{ and } 1 \notin U) \\ [1/2, 1] & (0 \notin U \text{ and } 1 \in U) \\ \emptyset & (0 \notin U \text{ and } 1 \notin U) \end{cases}$$

$$\begin{aligned} T(\emptyset \subseteq \mathbb{N}) &= \{m \subseteq [0, 1] \mid \forall r. r \in m\} \\ &= \{[0, 1]\}. \end{aligned}$$

Then, we get

$$T^{\top\top}(\emptyset) \times_{\mathcal{P}([0,1])} \mathcal{P}([0, 1]) = \{U \subseteq [0, 1] \mid 0 \in U \text{ and } 1 \in U\} \supseteq \{[0, 1]\} = T^{\top\top}(\emptyset)$$

and therefore  $T^{\top\top}$  is not fibred.

