Big Steps in Higher-Order Mathematical Operational Semantics

ANONYMOUS AUTHOR(S)

Small-step and *big-step operational semantics* are two fundamental styles of structural operational semantics (SOS), extensively used in practice. The former one is more fine-grained and is usually regarded as primitive, as it only defines a one-step reduction relation between a given program and its direct descendant under an ambient *evaluation strategy*. The latter one implements, in a self-contained manner, such a strategy directly by relating a program to the net result of the evaluation process. The agreement between these two styles of semantics is one of the key pillars in operational reasoning on programs; however, such agreement is typically proven from scratch every time on a case-by-case basis. A general, abstract mathematical argument behind this agreement is up till now missing. We cope with this issue within the framework of *higher-order mathematical operational semantics* by providing an abstract categorical notion of big-step SOS, complementing the existing notion of abstract higher-order GSOS. Moreover, we introduce a general construction for deriving the former from the latter, and prove an abstract equivalence result between the two.

CCS Concepts: • **Theory of computation → Operational semantics**; **Categorical semantics**; **Abstraction**.

Additional Key Words and Phrases: Operational semantics, Functional semantics, Abstract higher-order GSOS, Coalgebra, Extended combinatory logic

ACM Reference Format:
Anonymous Author(s). 2025. Big Steps in Higher-Order Mathematical Operational Semantics. 1, 1 (June 2025), 37 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Operational semantics of programming languages comes in two major styles: the *small-step* and the *big-step*. In both cases we deal with a rule-based specification of program behaviour, however, the respective rules operate with two principally different judgement formats. In paradigmatic cases, such as the (call-by-name) λ-calculus, the small-step judgements have the form $t \to t'$, and the big-step judgements have the form $t \Downarrow v$. Here, t is a (closed) program, t' is its direct successor under the reduction relation of interest, and v is a final *value*, to which t evaluates. The desired connection between these two judgements is expressed by the fundamental equivalence:

$$t \Downarrow v \iff t \to^{\star} v \wedge v\downarrow \qquad (\star)$$

where $\to^{\star}$ is the transitive-reflexive closure of $\to$ and $v\downarrow$ means that $v \to v'$ for no v'. In the case of the call-by-name λ-calculus, the setup is particularly simple: there are precisely two small-step rules and two big-step rules – see Figure 1. Even in this simple case, proving the equivalence ($\star$) is non-trivial and requires some creativity.

The equivalence ($\star$) plays an important role in various settings where both small-step and big-step semantics are defined. These settings can vary widely, encompassing different evaluation strategies, language features, computational effects (such as partiality, nondeterminism, and state), and even extensions to quantitative semantics [29, 30]. While in many standard cases the proof of ($\star$) follows familiar and often routine patterns it can still become tedious and error-prone when applied to expressive, feature-rich languages. In the absence of a unified mathematical framework that abstracts these patterns, such proofs need to be re-established manually, which may obscure reuse and increase the risk of oversight, unless the proofs are fully formalized and machine-checked.

The main goal of our present work is to provide suitable abstractions for the notions of small-step and big-step operational semantics, enabling us to formulate and prove ($\star$) at a high level of generality, particularly by parametrizing over suitable notions of *syntax* and *behaviour*. To that

$$\frac{}{(\lambda x.\, t)s \longrightarrow t[s/x]} \qquad \frac{t \longrightarrow t'}{ts \longrightarrow t's} \qquad \frac{}{\lambda x.\, t \Downarrow \lambda x.\, t} \qquad \frac{t \Downarrow \lambda x.\, t' \quad t'[s/x] \Downarrow v}{ts \Downarrow v}$$

Fig. 1. Operational semantics of call-by-name λ-calculus (small-step and big-step).

end, we capitalize on recent advances in *higher-order mathematical operational semantics* [21], which is a higher-order extension of Turi and Plotkin's [37] (first-order) mathematical operational semantics. Thus, a general notion of (higher-order) small-step semantics in the form of an *abstract HO-GSOS law* is already available from previous work [21]. However, motivated by ($\star$), in this paper, we develop a refinement of this notion, called *separated abstract HO-GSOS law*. Such a refinement is necessary because (unsurprisingly) an unrestricted small-step semantics need not generally correspond to a meaningful big-step semantics. Essentially this expressivity surcharge of small-step rule formats has been previously acknowledged in the context of (failure of) congruence properties of weak bisimilarity [7, 20, 36, 38, 40]. In fact, proving ($\star$) first requires interpreting it meaningfully. The original notion of abstract HO-GSOS is too general for this purpose: it does not distinguish between values and non-values, nor does it generally support the definition of multi-step transitions $\to^\star$. The separation requirement is introduced precisely to cater for this. We then define an abstract notion of big-step semantics and establish an abstract formulation of equivalence ($\star$). Yet, in order to prove ($\star$), restricting to separated abstract HO-GSOS is not enough, which led us to a sufficient condition that we call *strong separability*. We elaborate on these matters by example in Section 2.

In our developments, we employ the versatile language of *category theory*, while our approach as a whole aligns closely with *functional semantics* [13, 39], as opposed to relational semantics. In functional semantics, we can view rules such as those in Figure 1 as functional transformations of premises to conclusions, and therefore, define the semantics of programs as certain fixpoints. The equivalence to the familiar relational semantics is achieved by interpreting relations as nondeterministic functions – specifically, as effectful functions w.r.t. the powerset monad.

In summary, we contribute as follows.

- We introduce the notion of separated abstract HO-GSOS law (Definition 3.1) for modelling small-step semantics, refining the existing abstract HO-GSOS laws [21];
- We introduce and argue for the strong separation conditions (Definition 3.6), meant to guarantee that a given small-step semantics can have a big-step counterpart;
- We introduce an abstract notion of big-step semantics (Definition 4.1);
- We provide an abstract translation from small-step to big-step and establish ($\star$) under the strong separation assumption;
- We elaborate various instances of our abstract framework and the equivalence ($\star$) (Section 6, Section 7).

We implemented our notions and constructions in Haskell, as well as those examples from Section 6 that are hosted in the category **Set** of sets and functions. Most of the proofs are placed in the appendix for space reasons.

Related Work. The abstract (categorical) perspective on the small-step and big-step semantics we develop here is enabled by recent advances in *higher-order mathematical operational semantics* [21], establishing a connection between sets of operational semantics rules and certain (di-)natural transformations. The first-order form of this connection goes back to the seminal work of Turi and Plotkin [37]. Without this leverage, the question of the general connection between small-step and big-step semantics was addressed in the literature in a syntax-driven manner. Ciobâcă [11],

$$\frac{}{S \xrightarrow{t} S'(t)} \qquad \frac{}{S'(t) \xrightarrow{s} S''(t,s)} \qquad \frac{}{S''(t,s) \xrightarrow{r} (tr)(sr)}$$

$$\frac{}{K \xrightarrow{t} K'(t)} \qquad \frac{}{K'(t) \xrightarrow{s} t} \qquad \frac{}{I \xrightarrow{t} t} \qquad \frac{t \to t'}{ts \to t's} \qquad \frac{t \xrightarrow{s} t'}{ts \to t'}$$

Fig. 2. Small-step operational semantics of **xCL**.

motivated similarly to us, proposed an automatic translation of small-step specifications to big-step specifications, and essentially proved (⋆) using purely syntactic methods, under a number of assumptions different from ours. Similarly, Bach Poulsen and Mosses [5] described a translation of small-step specifications to *pretty-big-step specifications*, which are fundamentally a certain form of big-step specifications. Our categorical abstraction of operational semantics and their transformations are related to the ideas of functional semantics [13, 35, 39]. A functional semantics essentially replaces (big-step) rules with functional transformations equipped with a clock to ensure totality. This clock can be integrated into a monad, such as the *delay monad* [8]. Our treatment requires an ω-continuous monad [23], as a parameter, instead, e.g. a partiality monad, which can be viewed as an extensional counterpart of the delay monad [2]. In this paper, we are not focusing on constructive aspects, and – mainly for the sake of simplicity – stick to the powerset monad as the main example of ω-continuous monad.

Special restricted rule formats for small-step semantics were proposed by Bloom [7] and van Glabbeek [40], to ensure congruence properties of behavioural equivalences. Similar restrictions later resurfaced in previous abstract treatments of small-step semantics [20, 36, 38]. Our condition of *strong separation* abstracts similar restrictions – it is analogous to claiming that the specification contains enough *patience rules* [7, 40].

2 Abstract Higher-Order Operational Semantics: Overture

As a simple motivating example we consider call-by-name *extended combinatory logic* (**xCL**) previously introduced for analogous purposes [21]. This language is a variant of the well-known **SKI** calculus [12] and being computationally equivalent to the λ-calculus avoids the technical overhead of name management. The terms (i.e. programs) of **xCL** are generated by the following grammar

$$t, s ::= S \mid K \mid I \mid S'(t) \mid K'(t) \mid S''(t,s) \mid ts. \tag{1}$$

Here, the binary application operator is as usual represented by juxtaposition, however, in the sequel, we will also use $\mathrm{app}(t,s)$ as a verbose synonym to ts. The letters S, K and I are the standard combinators, i.e. constants that represent corresponding closed λ-terms. Their variants S', S'' and K' capture partial applications of S and K.

The small-step semantics rules for **xCL** are displayed in Figure 2. For example, we can derive the standard reduction for the S-combinator: $S\,t\,s\,r \to^{\star} (t\,r)(s\,r)$ (modulo addition of intermediate unlabeled transitions). The auxiliary labeled transitions $t \xrightarrow{s} t'$ represent the fact that t reduces to t' by consuming an argument s. Such use of labeled transitions has previously been adopted by Abramsky [1] and Gordon [24].

2.1 Combinatory Logic in Haskell

To build intuition for the upcoming technical developments, we present a semi-formal exposition of **xCL**, including its small-step and big-step operational semantics, as well as their relationship, using Haskell. This exposition serves to motivate and clarify the original notion of higher-order

```
data Free s x  = Res x | Cont (s (Free s x))       -- Free monad over given functor
type Initial s = Free s Void                       -- Initial algebra of given functor

newtype Mrg s x = Mrg (s x x)
sigOp = Cont . Mrg

-- Type class for HO−GSOS, parametrized by signature and behaviour
class (Bifunctor s, MixFunctor b) ⇒ HOGSOS s b where
  -- Abstract representation of small−step rules
  rho :: s (x, b x y) x → b x (Free (Mrg s) (Either x y))

  -- Operational model: derivable abstract semantics function
  gamma :: Initial (Mrg s) → b (Initial (Mrg s)) (Initial (Mrg s))
  gamma (Cont (Mrg t)) = mx_second (≫= nabla) $ rho $ first (id &&& gamma) t
    where nabla = either id id

-- Instantiations for xCL:

data XCL′ x y = S | K | I | S′ x | K′ x | S′′ x x | Comp x y     -- Signature
type XCL      = Mrg XCL′

data Beh x y = Eval (x → y) | Red y                            -- Behaviour

instance HOGSOS XCL′ Beh where
  rho :: XCL′ (x, Beh x y) x → Beh x (Free XCL (Either x y))
  rho S = Eval $ sigOp . S′ . Res . Left
  rho K = Eval $ sigOp . K′ . Res . Left
  rho I = Eval $ Res . Left

  rho (S′ (s, _)) = Eval $ λt → sigOp $ S′′ (Res $ Left s) (Res $ Left t)
  rho (K′ (s, _)) = Eval $ λt → Res $ Left s

  rho (S′′ (s, _) (u, _)) =
    Eval $ λt → sigOp $ Comp (sigOp $ Comp (Res $ Left s) (Res $ Left t))
                             (sigOp $ Comp (Res $ Left u) (Res $ Left t))

  rho (Comp (_, Red s) u)  = Red $ sigOp $ Comp (Res $ Right s) (Res $ Left u)
  rho (Comp (_, Eval f) u) = Red $ Res (Right $ f u)
```

Fig. 3. *HOGSOS* type class and *XCL′* as its instance.

abstract GSOS laws [21] and their separated variant. A rigorous categorical formulation of these notions is provided in Section 2.2, to which readers primarily interested in the formal theory may skip directly.

Consider the (incomplete) Haskell code in Figure 3. Assuming that s models a signature, *Free s* and *Initial s* model terms over this signature with and without variables respectively. The central definition is that of *HOGSOS*, which is a type class, parameterized by a signature bifunctor s and a mixed variant behaviour functor b (more precisely, b is contra-variant in the first argument and co-variant in the second). We explain later why the signature functor has two arguments (i.e. is a bifunctor) – the standard notion of signature is recovered as *Mrg s*.

The *HOGSOS* class has one method *rho*, which encodes the rules of operational semantics. How exactly *rho* does that is illustrated by instantiating it to the case of **xCL** (the code stating that *XCL′* and *Beh* are a bifunctor and a mixed variance functor correspondingly is omitted). The argument of *rho* refers to a relevant signature symbol (via s), to its arguments that match the left bottom part of the corresponding rule (via x), and to the corresponding argument's behaviours that match the right top parts of the corresponding rule (via $b\ x\ y$). The *operational model gamma* for every choice of s, b and *rho* produces the semantics of a given closed term, which in the case of **xCL** corresponds to the transitions $p \longrightarrow q$ or $p \xrightarrow{t} q$ derivable with the rules in Figure 2. The recursive definition of *gamma* uses the fact that *Free s* is a monad and calls the operation *&&&* for pairing functions and functorial actions *first* , *mx_second* of s and b on the first and the second argument correspondingly. Recall that $\$$ reads as function composition and the inlined definition of *nabla* captures the universal map from the coproduct of x with itself to x yielding $\gg\!= nabla :: Free\ \ (Mrg\ s)\ \ (Either\ \ (Free\ \ (Mrg\ s))\ \ (Free\ \ (Mrg\ s)))\ \ \longrightarrow Free\ (Mrg\ s)$.

The behaviour functor *Beh* is a coproduct of two functors: the first one caters for labeled transitions (evaluations), the second one caters for unlabeled transitions (reductions). The following **instance** declaration captures the rules from Figure 2. The separation on evaluations and reductions is not present on the abstract level of *HOGSOS* and *rho*, and hence they are not suitable for defining *multi-step semantics* $\longrightarrow^\star$ involved in ($\star$). We thus introduce a refinement of abstract HO-GSOS in Figure 4 and call it *separated abstract HO-GSOS*. That is, we postulate a partitioning of both the signature functor *SepSig* and the behaviour functor *SepBeh* into a *“value part“* and a *“computation part“*, which is indicated by appending V and C correspondingly. The computation part of the signature functor for **xCL** is the application operator, and the computation part of the behaviour functor for **xCL** is the part of unlabeled transitions.

It follows from the above code that *SepHOGSOS* instantiates *HOGSOS*, which is shown by assembling a suitable abstract HO-GSOS rule *rho* from *rhoV* and *rhoC*. For technical reasons (to ensure unambiguous type checking) we use explicit type applications via *@* – those can be safely ignored in reading. We thus inherit the operational model *gamma*, which, in turn, yields its own computation part *gammaC*, thanks to the separability assumption. Using *gammaC*, we recursively define the abstract multi-step semantics *beta*. Here, the sake of unambiguous type checking, we use Haskell's mechanism of proxies, i.e., in this case, the dummy argument p, which is only needed to facilitate type checking.

Finally, in order to interpret the left-hand side of the equivalence ($\star$) and the equivalence itself, we need to define big-step semantics abstractly and to link it to the small-step semantics. We do this as shown in Figure 5. The notion of big-step semantics is formalized with the class *BSSOS* and its method *xi*. Again, for technical reasons we let *BSSOS* vacuously depend on d – this is needed for the following **instance** declaration to type check. The derivable map *zetahat* and its variant *zeta* abstractly define the evaluation relation $\Downarrow$. From an HO-GSOS specification we automatically obtain a big-step SOS specification in a slightly wordy, but essentially simple manner.

In the case of **xCL** we obtain the specification displayed in Figure 6. This specification can facilitate understanding the type of *xi*: unless the rule is an axiom of the form $v \Downarrow v$ where v is a term whose topmost operator is from *cv*, the conclusion of the rule has the form $pq \Downarrow v$ and the premise of the rule contains the judgement of the form $t \Downarrow v$ with t determined by q and by such w that $p \Downarrow w$ also occurs in the premise. The latter type of rules is thus determined by a choice of an operation from *sc*, by a choice of an operation from *sv* for every of its argument that corresponds to the first position of *sc* as a functor, and by the term t that can parametrically depend on the

```
-- Signature functor as a sum of value and computation parts
data SepSig′ sv sc x y = SigV (sv y) | SigC (sc x y)
type SepSig sv sc      = Mrg (SepSig′ sv sc)

type InitialV sv sc = sv (Initial (SepSig sv sc))
type InitialC sv sc = sc (Initial (SepSig sv sc)) (Initial (SepSig sv sc))

-- Form of the behaviour functor in the separable setting
data SepBeh d x y = BehV (d x y) | BehC y

-- Type class for separated HO-GSOS
class (MixFunctor d, Functor sv, Bifunctor sc) ⇒ SepHOGSOS sv sc d where
  rhoV :: sv x → d x (Free (SepSig sv sc) x)
  rhoC :: sc (x, SepBeh d x y) x → Free (SepSig sv sc) (Either x y)

  rhoCV :: sc (x, d x y) x → Free (SepSig sv sc) (Either x y)
  rhoCV = rhoC . first (second BehV)

  gammaC :: Proxy d → InitialC sv sc → Initial (SepSig sv sc)
  gammaC (p :: Proxy d) t = (rhoC @_ @_ @d $ first (id &&& gamma) t) ≫= nabla
    where nabla = either id id

  beta :: (Functor sv, Bifunctor sc, MixFunctor d, SepHOGSOS sv sc d)
    ⇒ Proxy d → InitialC sv sc → InitialV sv sc
  beta (p :: Proxy d) t = case gammaC p t of Cont (Mrg (SigV t)) → t;
                                             Cont (Mrg (SigC t)) → beta p t

instance (SepHOGSOS sv sc d) ⇒ HOGSOS (SepSig′ sv sc) (SepBeh d) where
  rho :: SepSig′ sv sc (x, SepBeh d x y) x → SepBeh d x (Free (SepSig sv sc) (Either x y))
  rho (SigV v) = BehV $ (right $ fmap Left) $ rhoV v
  rho (SigC c) = BehC $ rhoC c
```

Fig. 4. *SepHOGSOS* type class as a refinement of *HOGSOS*.

arguments of these operations. This is, in a nutshell, the information that *xi* carries. Note that the rules in Figure 6 marked by an asterisk can be simplified in the obvious way by removing premises of the form $w \Downarrow v$ and by replacing v with w in the conclusions.

The rules in Figure 6 also help one to see why the signature functor *sc* has two arguments. This concretely means partitioning the arguments of any operation in *sc* over two types: *strict* and *lazy*. In **xCL** *sc* contains precisely one operator – application, whose first argument is strict, and whose second argument is lazy. As the rules in Figure 6 illustrate, we allow evaluation of strict arguments only. Unlike small-step semantics, we cannot simply add judgements of the form $q \Downarrow w$ even if we do not use w. For example, given any diverging term Ω, we must have $KI\Omega \Downarrow I$, which would not be the case if we conditioned this on the existence of a derivation $\Omega \Downarrow w$.

We can now express ($\star$) as the equality

$$\mathit{beta}\ p\ t == \mathit{zeta}\ t \tag{2}$$

```
-- Type class for big-step SOS
class (Functor sv, Bifunctor sc) ⇒ BSSOS d sv sc where
  xi :: sc (sv x) x → Free (SepSig sv sc) x

  zetahat :: Initial (SepSig sv sc) → InitialV sv sc
  zetahat (Cont (Mrg (SigV v))) = v
  zetahat (Cont (Mrg (SigC c))) = zetahat @d $ join $ xi @d $ first (zetahat @d) c

  zeta :: InitialC sv sc → InitialV sv sc
  zeta = zetahat @d . sigOp . SigC

instance (SepHOGSOS sv sc d) ⇒ BSSOS d sv sc where
  xi :: sc (sv x) x → Free (SepSig sv sc) x
  xi t = rhoCV (bimap ((sigOp . SigV &&& mx_second @d join . rhoV) . fmap return) return t)
              ≫= nabla
  where nabla = either id id
```

Fig. 5. *BSSOS* type class, and *SepHOGSOS* as its instance.

for all p :: *Proxy d* and t :: *InitialC sv sc*.

For a final note, observe that (2), albeit desirable, need not always be true.

Example 2.1. Consider a language over the signature $\{f/1, g/1, \Omega/0\}$ where $/n$ indicates that the arity of the corresponding operation is n. Consider the following small-step specification:

$$\frac{}{g(x) \xrightarrow{y} f(y)} \qquad \frac{}{\Omega \to \Omega} \qquad \frac{x \to y}{f(x) \to g(y)} \qquad \frac{x \xrightarrow{x} y}{f(x) \to x}$$

which identifies g as a value former and f and Ω as computation formers. This specification yields a separated abstract HO-GSOS law, in which the only argument of f is necessarily strict, because the behaviour of $f(t)$ generally depends on the behaviour of t. The only way to define big-step rules would be as follows:

$$\frac{}{g(x) \Downarrow g(x)} \qquad \frac{x \Downarrow g(y) \qquad g(y) \Downarrow v}{f(x) \Downarrow v}$$

The equivalence ($\star$) now fails, because $f(f(g(\Omega))) \to g(g(\Omega))$, but $f(f(g(\Omega))) \Downarrow g(\Omega)$. ◇

2.2 Categorical Modeling

The reasoning of the previous section is not sufficiently precise in various respects. Haskell provides a very concrete type-theoretic ambient with general recursion and other features we might want or not want to include, but in any case we need to be conscious about them. This is particularly important for constructing proofs, an aspect, we completely omitted so far, but which we consider as the main contribution. The above treatment of **xCL** can be naturally formalized in the category of sets, using a suitable *partiality monad* [2, 9, 19] for modeling iteration and recursion. More generally, one would need other categories: multisorted sets for typed languages, categories of presheaves or nominal sets for the λ-calculus, the corresponding combinations and extensions thereof. It is also not necessary to restrict to partiality as the only effect – one can treat nondeterministic or even probabilistic semantics in a similar manner. We thus generally work with *strong ω-continuous*

Values: $v, w ::= I \mid K \mid S \mid K'(t) \mid S'(t) \mid S''(s,t)$

Terms: $s, t, r, q ::= v \mid st$

$$\frac{}{v \Downarrow v} \qquad \frac{s \Downarrow I \quad t \Downarrow v}{st \Downarrow v} \qquad \frac{s \Downarrow K \quad K'(t) \Downarrow v}{st \Downarrow v}\,{}_{*} \qquad \frac{s \Downarrow S \quad S'(t) \Downarrow v}{st \Downarrow v}\,{}_{*}$$

$$\frac{s \Downarrow K'(r) \quad r \Downarrow v}{st \Downarrow v} \qquad \frac{s \Downarrow S'(r) \quad S''(r,t) \Downarrow v}{st \Downarrow v}\,{}_{*} \qquad \frac{s \Downarrow S''(r,q) \quad (rt)(qt) \Downarrow v}{st \Downarrow v}$$

Fig. 6. Big-step operational semantics of **xCL**.

monads, which are arguably the largest semantically relevant class of monads that support iteration via least fixed points [23].

We then recall and/or fix relevant categorical notations and conventions in order. For the basics of category theory we refer to [4, 31]. In a category $\mathbf{C}$, $|\mathbf{C}|$ will denote the class of objects and $\mathbf{C}(X, Y)$ will denote the set of morphisms from X to Y. The judgement $f : X \to Y$ will be regarded as an equivalent to $f \in \mathbf{C}(X, Y)$ if $\mathbf{C}$ is clear from the context. We will denote by id_X, or simply id the identity morphism on X. In what follows, we generally work in an ambient distributive category $\mathbf{C}$ (as defined below). By $|\mathbf{C}|$ we refer to the objects of $\mathbf{C}$. We tend to omit indexes at natural transformations for better readability. *Dinatural transformations* generalize natural transformations to the case of mixed variance functors. More precisely, given two functors $F, G : \mathbf{C}^{\mathsf{op}} \times \mathbf{C} \to \mathbf{D}$, a family of morphisms $\alpha = (\alpha_{X,Y} : F(X,Y) \to G(X,Y))_{X,Y \in |\mathbf{C}|}$ is a dinatural transformation if the diagram

$$\begin{tikzcd}
 & F(X,X) \arrow[r, "\alpha_{X,X}"] & G(X,X) \arrow[dr, "G(\mathsf{id},f)"] & \\
F(Y,X) \arrow[ur, "F(f,\mathsf{id})"] \arrow[dr, "F(\mathsf{id},f)"'] & & & G(X,Y) \\
 & F(Y,Y) \arrow[r, "\alpha_{Y,Y}"] & G(Y,Y) \arrow[ur, "G(f,\mathsf{id})"'] &
\end{tikzcd}$$

commutes for any $f : X \to Y$. A prototypical example of a dinatural transformation is the evaluation transformation $Y^X \times X \to Y$.

Some further notions we will need are as follows.

Distributive categories. A *distributive category* is a category with finite products and coproducts, and such that every morphism $[\mathsf{id} \times \mathsf{inl}, \mathsf{id} \times \mathsf{inr}] : X \times Y + X \times Z \to X \times (Y + Z)$ is an isomorphism. Let $\nabla = [\mathsf{id}, \mathsf{id}] : X + X \to X$ and let χ be the functor $\mathbf{C} \to \mathbf{C} \times \mathbf{C}$, sending X to (X, X). Moreover, let $\Pi_1, \Pi_2 : \mathbf{C} \times \mathbf{C} \to \mathbf{C}$ be the obvious projections.

Functors and algebras. We involve three types of (endo-)functors: the usual unary covariant functors $F : \mathbf{C} \to \mathbf{C}$, bifunctors $F : \mathbf{C} \times \mathbf{C} \to \mathbf{C}$ and mixed variance functors $F : \mathbf{C}^{\mathsf{op}} \times \mathbf{C} \to \mathbf{C}$. Given a functor $F : \mathbf{C} \to \mathbf{C}$ in a distributive category $\mathbf{C}$, the *(pointwise) free monad* $F^\star : \mathbf{C} \to \mathbf{C}$ over F is characterized by a universal property: for any object X there are morphisms $\iota_X : F(F^\star X) \to F^\star X$, and $\eta_X : X \to F^\star X$, such that for any $f : FY \to Y$ and $g : X \to Y$ the diagram

$$\begin{tikzcd}
F(F^\star X) \arrow[r, "\iota_X"] \arrow[d, "F(\mathsf{init}[f,g])"'] & F^\star X \arrow[d, "\mathsf{init}[f,g]"] & X \arrow[l, "\eta_X"'] \arrow[dl, "g"] \\
FY \arrow[r, "f"] & Y &
\end{tikzcd}$$

commutes for precisely one morphism $\mathsf{init}[f,g]$. It follows that ι_X and η_X are natural in X. By generalities (Lambek's theorem), $[\eta,\iota]$ is an isomorphism. For every X, $F^\star X$ is also called a *free (F-)algebra* on X. We can often think of $F^\star X$ as an object of terms with variables from X and with operations from F, and of $\mu F = F^\star\emptyset$ as an object of closed terms thereof. This is specifically true for *polynomial functors*, i.e. functors of the form $FX = \coprod_{f\in Ops} X^{\mathsf{ar}(f)}$ where Ops is a set of operations f of corresponding arities $\mathsf{ar}(f)$. One example is the functor

$$FX = \underbrace{1+1+1}_{I,K,S} + \underbrace{X+X}_{K',S'} + \underbrace{X\times X}_{S''}$$

induced by the grammar of **xCL** (1). By definition, an element of the dependent sum $\coprod_{f\in Ops} X^{\mathsf{ar}(f)}$ has the form $(f \in Ops, (x_i \in X)_{i\in\mathsf{ar}(f)})$, which we will also write as $f(x_i)_{i\in\mathsf{ar}(f)}$, or even $f(x_1,\dots,x_{\mathsf{ar}(f)})$ when $\mathsf{ar}(f)\in\mathbb{N}$, if no confusion arises. In what follows we assume that all the involved free objects $F^\star X$ exist, without further mention.

Strong monads and distributive laws. A monad $\mathbf{T}$ on $\mathbf{C}$ is determined by a *Kleisli triple* $(T,\eta,(-)^\sharp)$, consisting of a map $T\colon|\mathbf{C}|\to|\mathbf{C}|$, a family of morphisms $(\eta_X\colon X\to TX)_{X\in|\mathbf{C}|}$ and *Kleisli lifting* sending each $f\colon X\to TY$ to $f^\sharp\colon TX\to TY$ and obeying *monad laws*:

$$\eta^\sharp = \mathsf{id}, \qquad f^\sharp\cdot\eta = f, \qquad (f^\sharp\cdot g)^\sharp = f^\sharp\cdot g^\sharp.$$

It follows that T extends to a functor, η extends to a natural transformation – *unit*, $\mu = \mu\colon TTX\to TX$ extends to a natural transformation – *multiplication*, and that (T,η,μ) is a monad in the standard sense [31]. Any free monad $F^\star$ is an example of a monad.

We will emphasize that a functor T is also a monad by writing it boldfaced (such as $\mathbf{T}$). Given $f\colon X\to TZ$ and $g\colon Y\to TW$, we abbreviate $f\boxplus g = [T\,\mathsf{inl}\cdot f, T\,\mathsf{inr}\cdot g]\colon X+Y\to T(Z+W)$.

A monad $\mathbf{T}$ is *strong* if it comes with a natural transformation $\tau_{X,Y}\colon X\times TY\to T(X\times Y)$ called *strength* and satisfying a number of coherence conditions [34]. A well-known fact due to Kock [28] is that in self-enriched categories (such as **Set**) strength is equivalent to enrichment. In particular, in **Set**, every functor T and every monad $\mathbf{T}$ are strong with the canonical strength $\tau_{X,Y} = \lambda(x,z).\,T(\lambda y.\,(x,y))(z)$.

By a *(Kleisli) distributive law* between a monad $\mathbf{T}$ and a functor F we mean a natural transformation $FT\to TF$ suitably interacting with unit and multiplication of the monad [25].

Order enrichment and fixpoints. Recall that *Kleene's fixpoint theorem* states that every continuous endomap f on a pointed ω-cpo has the least pre-fixpoint μf (which is also the least fixpoint), which is a least upper bound of the chain

$$\bot \sqsubseteq f(\bot) \sqsubseteq f(f(\bot)) \sqsubseteq \dots$$

An *ω-continuous* monad [23] is a monad $\mathbf{T}$ together with an enrichment of the Kleisli category $\mathbf{C}_{\mathbf{T}}$ of $\mathbf{T}$ over pointed ω-cpos and (nonstrict) ω-continuous maps, satisfying the following principles:

- strength is ω-continuous: $\tau(\mathsf{id}\times\bigsqcup_i f_i) = \bigsqcup_i \tau(\mathsf{id}\times f_i)$;
- copairing in $\mathbf{C}_{\mathbf{T}}$ is ω-continuous: $\left[\bigsqcup_i f_i, \bigsqcup_i g_i\right] = \bigsqcup_i [f_i, g_i]$;
- bottom elements are preserved by strength and by postcomposition in $\mathbf{C}_{\mathbf{T}}$: $\tau(\mathsf{id}\times\bot) = \bot$, $f^\sharp\cdot\bot = \bot$.

Every ω-continuous monad $\mathbf{T}$ is a (complete) Elgot monad, i.e. it supports an (Elgot) iteration operator that sends every $f\colon X\to T(Y+X)$ to $f^\dagger\colon X\to TY$, subject to several standard laws of iteration. Specifically, $f^\dagger = \mu g.\,[\eta,g]^\sharp\cdot f$. Standard classical examples of (strong) ω-continuous monads are the *maybe-monad* $TX = X+1$ and the *powerset monad* $TX = \mathcal{P}X$. Note that the identity monad $T = \mathsf{Id}$ is generally not ω-continuous, for the Kleisli hom-sets need not posses least elements. We call a distributive law $\chi\colon F\mathbf{T} \to \mathbf{T}F$ ω-continuous if all the correspondences $f \in \mathbf{C}(X, TY) \mapsto \chi_Y \cdot Ff \in \mathbf{C}(FX, TFY)$ are ω-continuous.

2.3 Abstract HO-GSOS

We proceed to recall and modify slightly the notion of abstract HO-GSOS law from previous work [21]. This is indeed the notion we already implemented in Section 2.1. Let us fix a bifunctor $\Sigma'\colon \mathbf{C} \times \mathbf{C} \to \mathbf{C}$ (signature) and a mixed variance functor $B\colon \mathbf{C}^{\mathrm{op}} \times \mathbf{C} \to \mathbf{C}$ (behaviour) on a distributive category $\mathbf{C}$. Let $\Sigma = \Sigma'\chi$ and let

$$\rho_{X,Y}\colon \Sigma'(X \times B(X,Y), X) \to B(X, \Sigma^\star(X+Y)), \tag{3}$$

be a family of morphisms natural in Y and dinatural in X. This is a slight generalization of the original notion, which is obtained making Σ' independent of the second argument. In that case $\Sigma'(X,Y) = \Sigma X$. We need the additional parameter for Σ' to identify those arguments of signature operations, whose behaviour is not inspected. The key notion that (3) generates is that of *operational model*, which is a morphism γ, defined by parametrized structural recursion as follows:

$$\begin{tikzcd}
\Sigma'(\mu\Sigma, \mu\Sigma) \arrow[rr, "\iota"] \arrow[d, "\Sigma'(\langle \mathrm{id},\gamma\rangle,\mathrm{id})"'] & & \mu\Sigma \arrow[d, "\gamma"] \\
\Sigma'(\mu\Sigma \times B(\mu\Sigma,\mu\Sigma), \mu\Sigma) \arrow[r, "\rho"] & B(\mu\Sigma, \Sigma^\star(\mu\Sigma+\mu\Sigma)) \arrow[r, "B(\mathrm{id},\nabla^\sharp)"] & B(\mu\Sigma,\mu\Sigma)
\end{tikzcd} \tag{4}$$

That is: there is precisely one $\gamma\colon \mu\Sigma \to B(\mu\Sigma,\mu\Sigma)$, such that (4) commutes.

In fact, given a law (3), we can introduce an abstract higher-order GSOS for Σ and B in the original sense as follows:

$$\begin{tikzcd}
\Sigma'(X \times B(X,Y), X \times B(X,Y)) \arrow[r, "\Sigma'(\mathrm{id}\times B(\mathrm{id},\mathrm{inr}),\mathrm{inl}\cdot\mathrm{fst})"] & \Sigma'(X \times B(X, X+Y), X+Y) \arrow[dl, "\rho"'] \\
B(X, \Sigma^\star(X+(X+Y))) \arrow[r, "B(\mathrm{id},\Sigma^\star[\mathrm{inl},\mathrm{id}])"] & B(X, \Sigma^\star(X+Y))
\end{tikzcd} \tag{5}$$

and it is shown below that the operational model for it coincides with the one specified by (4). In that sense our present generalization is indeed mild – it does not add more strength to the original notion, but it provides more flexibility for the analysis of the transformations (3).

Proposition 2.2. *Let $\rho'_{X,Y}\colon \Sigma(X \times B(X,Y)) \to B(X,\Sigma^\star(X+Y))$ be defined by (5). Then the operational model for ρ' is the unique such morphism that the diagram (4) commutes.*

3 Separable Abstract Higher-Order GSOS

Let us reintroduce the notion of separable abstract HO-GSOS from Section 2.1 in categorical terms.

Definition 3.1 (Separable Abstract Higher-Order GSOS). We say that the law (3) is *separable* if $B(X,Y) = D(X,Y) + TY$, $\Sigma' = \Sigma_{\mathsf{v}}\Pi_2 + \Sigma_{\mathsf{c}}$ and

$$\rho = D(\mathrm{id}, \Sigma^\star\,\mathrm{inl}) \cdot \rho^{\mathsf{v}} + \rho^{\mathsf{c}} \tag{6}$$

for some $D\colon \mathbf{C}^{\mathrm{op}} \times \mathbf{C} \to \mathbf{C}$, $\Sigma_{\mathsf{v}}\colon \mathbf{C} \to \mathbf{C}$, $\Sigma_{\mathsf{c}}\colon \mathbf{C} \times \mathbf{C} \to \mathbf{C}$, a strong monad $\mathbf{T}$, families of morphisms

$$\rho^{\mathsf{v}}_X\colon \Sigma_{\mathsf{v}}X \to D(X, \Sigma^\star X), \tag{7}$$

$$\rho^{\mathsf{c}}_{X,Y}\colon \Sigma_{\mathsf{c}}(X \times B(X,Y), X) \to T\Sigma^\star(X+Y), \tag{8}$$

dinatural in X and natural in Y, and a distributive law

$$\chi_{X,Y}\colon \Sigma_{\mathsf{c}}(TX, Y) \to T\Sigma_{\mathsf{c}}(X,Y). \tag{9}$$

between the monad $\mathbf{T}$ and the functors $\Sigma_{\mathsf{c}}(-, Y)$ naturally in Y. The triple $(\rho^{\mathsf{v}}, \rho^{\mathsf{c}}, \chi)$ is then a *separated abstract higher-order GSOS law*.

We call Σ_{v} *value formers* and Σ_{c} *computation formers*. Moreover, we call $\mu\Sigma_{\mathsf{v}} = \Sigma_{\mathsf{v}}(\mu\Sigma)$ the *object of values* and $\mu\Sigma_{\mathsf{c}} = \Sigma_{\mathsf{c}}(\mu\Sigma, \mu\Sigma)$ the *object of computations*. Let $\iota^{\mathsf{v}} = \iota \cdot \mathsf{inl}\colon \Sigma_{\mathsf{v}}\Sigma^{\star} \to \Sigma^{\star}$ and $\iota^{\mathsf{c}} = \iota \cdot \mathsf{inr}\colon \Sigma_{\mathsf{c}}\Delta\Sigma^{\star} \to \Sigma^{\star}$.

For the time being we are not making any assumptions about the monad $\mathbf{T}$. In the simplest (total, deterministic) case, $\mathbf{T}$ in Definition 3.1 is the identity monad and $\chi = \mathrm{id}$.

Since $\mu\Sigma \cong \mu\Sigma_{\mathsf{v}} + \mu\Sigma_{\mathsf{c}}$, the object of all closed Σ-terms $\mu\Sigma$ crisply decomposes into values and computations. That Σ_{c} is a binary functor is meant to capture a partitioning of the arguments of the computation formers into those that depend on the associated behaviour, and those that do not. We call the former type of arguments *strict* and the latter *lazy*.

For a separated abstract higher-order GSOS law $(\rho^{\mathsf{v}}, \rho^{\mathsf{c}}, \chi)$, we define a refinement $(\gamma^{\mathsf{v}}, \gamma^{\mathsf{c}})$ of the operational model (4). The morphism γ^{v} is the composition

$$\mu\Sigma_{\mathsf{v}} \xrightarrow{\rho^{\mathsf{v}}} D(\mu\Sigma, \Sigma^{\star}\mu\Sigma) \xrightarrow{D(\mathrm{id},\mu)} D(\mu\Sigma, \mu\Sigma) \tag{10}$$

and the morphism γ^{c} is characterized by the diagram

$$\begin{tikzcd}
\Sigma_{\mathsf{c}}(\mu\Sigma_{\mathsf{v}} + \mu\Sigma_{\mathsf{c}}, \mu\Sigma) \arrow[rr, "\Sigma_{\mathsf{c}}(\iota{,}\mathrm{id})"] \arrow[d, "\Sigma_{\mathsf{c}}(\langle\iota{,}\gamma^{\mathsf{v}}+\gamma^{\mathsf{c}}\rangle{,}\mathrm{id})"'] & & \mu\Sigma_{\mathsf{c}} \arrow[d, "\gamma^{\mathsf{c}}"] \\
\Sigma_{\mathsf{c}}(\mu\Sigma \times B(\mu\Sigma, \mu\Sigma), \mu\Sigma) \arrow[r, "\rho^{\mathsf{c}}"] & T\Sigma^{\star}(\mu\Sigma + \mu\Sigma) \arrow[r, "T\nabla^{\sharp}"] & T\mu\Sigma
\end{tikzcd} \tag{11}$$

in the following sense.

Proposition 3.2. *There is unique $\gamma^{\mathsf{c}}\colon \mu\Sigma_{\mathsf{c}} \to T(\mu\Sigma)$, for which* (11) *commutes. Moreover:*

$$\gamma^{\mathsf{v}} + \gamma^{\mathsf{c}} = \gamma \cdot \iota. \tag{12}$$

Let $\hat{\gamma}^{\mathsf{c}}$ be the morphism $[\eta \cdot \iota^{\mathsf{v}}, \gamma^{\mathsf{c}}] \cdot \iota^{-1}\colon \mu\Sigma \to T(\mu\Sigma)$. Intuitively, $\hat{\gamma}^{\mathsf{c}}$ acts on computations as γ^{c} and as η on values.

Example 3.3 (Extended Combinatory Logic). Recall the grammar (1) of the extended combinatory logic **xCL**. The corresponding signature functor consists of two parts:

$$\Sigma_{\mathsf{v}}X = \coprod\nolimits_{f \in \{S,K,I,K',S',S''\}} X^{\mathrm{ar}(f)} \qquad\qquad \Sigma_{\mathsf{c}}(X, Y) = \{\mathsf{app}\} \times (X \times Y).$$

Here $\mathrm{ar}(f)$ denotes the arity of f. Binary application operator app is the only computation former. The expression for Σ_{c} indicates that the one-step behaviour of it only depends on the behaviour of the first argument, but not on the second.

Let $D(X, Y) = Y^X$ and $T = \mathrm{Id}$. The small-step operational semantics rules in Figure 2 define ρ^{v} and ρ^{c}, and hence the law (3). Concretely (eliding the obvious isomorphisms and parentheses):

$$\begin{array}{lll}
\rho^{\mathsf{v}}(I)(r) = r & \rho^{\mathsf{v}}(K'(t))(r) = t & \rho^{\mathsf{c}}(\mathsf{app}((t, t'), s)) = \mathsf{app}(t', s) \\
\rho^{\mathsf{v}}(K)(r) = K'(r) & \rho^{\mathsf{v}}(S'(t))(r) = S''(t, r) & \rho^{\mathsf{c}}(\mathsf{app}((t, f), s)) = f(s) \\
\rho^{\mathsf{v}}(S)(r) = S'(r) & \rho^{\mathsf{v}}(S''(t, s))(r) = \mathsf{app}(\mathsf{app}(t, r), \mathsf{app}(s, r)) &
\end{array}$$

◇

From (8) we derive

$$\rho^{\mathsf{cv}}_{X,Y}\colon \Sigma_{\mathsf{c}}(X \times D(X, Y), X) \xrightarrow{\rho^{\mathsf{c}} \cdot \Sigma_{\mathsf{c}}(\mathrm{id} \times \mathsf{inl}, \mathrm{id})} T\Sigma^{\star}(X + Y). \tag{13}$$

In what follows, we globally make the following mild technical assumption.

Assumption 3.4. We assume that for all X, Y, Z, and for Σ_{c} figuring in Definition 3.1 the morphisms $\Sigma_{\mathsf{c}}(\mathsf{inl}, \mathsf{id})\colon \Sigma_{\mathsf{c}}(X, Z) \to \Sigma_{\mathsf{c}}(X + Y, Z)$ are *complemented*, and the complementation is natural in X, Y and Z. In other words, for some functor $\Theta\colon \mathbf{C} \times \mathbf{C} \times \mathbf{C} \to \mathbf{C}$ and some natural transformation $\theta_{X,Y,Z}\colon \Theta(X, Y, Z) \to \Sigma_{\mathsf{c}}(X + Y, Z)$, the cospan

$$\Sigma_{\mathsf{c}}(X, Z) \xrightarrow{\Sigma_{\mathsf{c}}(\mathsf{inl},\mathsf{id})} \Sigma_{\mathsf{c}}(X + Y, Z) \xleftarrow{\theta} \Theta(X, Y, Z)$$

is a coproduct naturally in X, Y and Z.

Remark 3.5. Note that if Σ_{c} is polynomial $\Sigma_{\mathsf{c}}(X, Y) = \coprod_{f \in Ops} X^{\mathsf{ar_s}(f)} \times Y^{\mathsf{ar_l}(f)}$ (where $\mathsf{ar_s}(f)$ is the number of strict arguments of f and $\mathsf{ar_l}(f)$ is the number of lazy ones) Assumption 3.4 is satisfied whenever it is satisfied for every summand $X^{\mathsf{ar_s}(f)} \times Y^{\mathsf{ar_l}(f)}$. Let us fix f, and let $n = \mathsf{ar_s}(f)$, $m = \mathsf{ar_l}(f)$. Assumption 3.4 is then satisfied, since (by using the binomial formula)

$$(X + Y)^n \times Z^m \cong X^n \times Z^m + \coprod_{k=1}^{n} C_n^k \times Y^k \times X^{n-k} \times Z^m,$$

and we can take $\Theta(X, Y, Z) = \coprod_{k=1}^{n} C_n^k \times Y^k \times X^{n-k} \times Z^m$.

We now introduce a well-behavedness condition on separated abstract HO-GSOS guaranteeing that a notion of big-step operational semantics can be sensibly derived.

Recall that we defined $f \boxplus g = [T\,\mathsf{inl} \cdot f, T\,\mathsf{inr} \cdot g]$.

Definition 3.6 (Strong Separation). A separated abstract HO-GSOS law $(\rho^{\mathsf{v}}, \rho^{\mathsf{c}}, \chi)$ is *strongly separated* if the following diagram commutes:

$$\begin{tikzcd}
 & \Theta(X \times D(X,Y), X \times TY, X) \arrow[dl, "\theta"'] \arrow[dr, "\theta"] & \\
\Sigma_{\mathsf{c}}(X \times D(X,Y) + X \times TY, X) \arrow[dd, "\cong"'] & & \Sigma_{\mathsf{c}}(X \times D(X,Y) + X \times TY, X) \arrow[d, "\Sigma_{\mathsf{c}}(\eta \cdot \mathsf{fst} \boxplus \mathsf{snd}, \mathsf{inl})"] \\
 & & \Sigma_{\mathsf{c}}(T(X+Y), X+Y) \arrow[d, "\chi"] \\
\Sigma_{\mathsf{c}}(X \times (D(X,Y) + TY), X) \arrow[dr, "\rho^{\mathsf{c}}"'] & & T\Sigma_{\mathsf{c}}(X+Y, X+Y) \arrow[dl, "T(\iota^{\mathsf{c}} \cdot \Sigma_{\mathsf{c}}(\eta,\eta))"] \\
 & T\Sigma^{\star}(X+Y) &
\end{tikzcd} \tag{14}$$

If Σ_{c} is a coproduct $\coprod_{i \in I} \Sigma_{\mathsf{c}}^i$, it suffices to verify (14) with $\Sigma_{\mathsf{c}} \coloneqq \Sigma_{\mathsf{c}}^i$ for every i. Intuitively, the left path of the diagram (from top to bottom) corresponds to the general form of a rule, as represented by ρ^{c}, while the right path specifies the required format. The commutativity of the diagram thus imposes a constraint on ρ^{c}. Precomposition with θ ensures that this constraint becomes effective only for rules with at least one premise from the computation part of the behaviour.

One could directly verify that strong separation holds for **xCL**. It is instructive though to spell out (14) for a larger class of examples, of which **xCL** is a member.

Remark 3.7. Let us spell out the strong separation condition in the total deterministic case, i.e. when $T = \mathrm{Id}$ and $\chi = \mathrm{id}$. The diagram (14) then simplifies as follows:

$$\begin{tikzcd}
 & \Theta(X \times D(X,Y), X \times Y, X) \arrow[dl, "\theta"'] \arrow[dr, "\theta"] & \\
\Sigma_{\mathsf{c}}(X \times D(X,Y) + X \times Y, X) \arrow[d, "\cong"'] & & \Sigma_{\mathsf{c}}(X \times D(X,Y) + X \times Y, X) \arrow[d, "\Sigma_{\mathsf{c}}(\mathsf{fst} + \mathsf{snd}, \mathsf{inl})"] \\
\Sigma_{\mathsf{c}}(X \times (D(X,Y) + Y), X) \arrow[dr, "\rho^{\mathsf{c}}"'] & & \Sigma_{\mathsf{c}}(X + Y, X + Y) \arrow[dl, "\iota^{\mathsf{c}} \cdot \Sigma_{\mathsf{c}}(\eta,\eta)"] \\
 & \Sigma^{\star}(X + Y) &
\end{tikzcd}$$

Furthermore, let us interpret this diagram in the category of sets. Suppose that $\Sigma_{\mathsf{c}}(X, Y)$ contains $F(X, Y) = X^n \times Y^m$ as a summand for some natural numbers n and m. That is, Σ_{c} contains a computation former, say f, whose n first arguments are strict and whose remaining m arguments are lazy. As in the case of **xCL** (Example 3.3), let $D(X, Y) = Y^X$. The restriction of ρ^{c} to F corresponds to the rules that describe the behaviour of terms of the form $f(x_1, \dots, x_n, y_1, \dots, y_m)$. The strong separation condition requires the following: if the rule has at least one premise of the form $x_k \to x'_k$ then the conclusion of the rule must be of the form

$$f(x_1, \dots, x_n, y_1, \dots, y_m) \to f(x'_1, \dots, x'_n, y_1, \dots, y_m)$$

where either $x_i \to x'_i$ occurs in the premise, or else, the premise contains a labeled transition for x_i, in which case $x'_i = x_i$. ◇

Remark 3.8. Our strong separation condition (14) is a reminiscent of cool GSOS formats by Bloom [7] and van Glabbeek [40] in the context of process algebra. These formats require that certain operations come with enough *patience rules*, which are rules for the form

$$\frac{x_k \to x'_k}{f(x_1, \dots, x_n) \to f(x_1, \dots, x_{k-1}, x'_k, x_{k+1}, \dots, x_n)}$$

This is needed to ensure that the semantics is sufficiently transparent to unlabeled transitions (or τ-transitions in op.cit.). Originally, this enabled congruence properties of weak notions of process equivalence (such as weak bisimilarity). In our context, similar condition is needed to ensure that a small-step semantics can have a sensible a big-step semantics counterpart.

Example 3.9. It is easy to see that **xCL** satisfies strong separation: in view of Remark 3.7, the only relevant operation is application and the only relevant rule is

$$\frac{t \to t'}{ts \to t's}$$

Clearly, it has the requisite form. ◇

Example 3.10. Revisiting Example 2.1, note that the strong separation condition is violated by the third rule. As we argued, for the present small-step semantics we cannot define big-step semantics satisfying ($\star$). Thus, in this case, strong separation effectively rules out an undesired example. ◇

As we see latter, the equivalence between the small-step and the big-step semantics requires an ω-continuous monad $\mathbf{T}$, while the identity monad is not ω-continuous (Kleisli hom-sets do not have least elements). We thus may need to adjoin a (separated) abstract higher-order GSOS law over a monad $\mathbf{T}$ to a given (separated) abstract higher-order GSOS law over the identity monad.

Proposition 3.11. *Given families of morphisms*

$$\begin{aligned}\rho^{\mathsf{v}}_X &\colon \Sigma_{\mathsf{v}} X \to D(X, \Sigma^\star X),\\ \rho^{\mathsf{c}}_{X,Y} &\colon \Sigma_{\mathsf{c}}(X \times (D(X,Y)+Y), X) \to \Sigma^\star(X+Y),\end{aligned}$$

natural in Y and dinatural in X, a strong monad $\mathbf{T}$ and a distributive law $\chi_{X,Y} \colon \Sigma_{\mathsf{c}}(TX,Y) \to T\Sigma_{\mathsf{c}}(X,Y)$, let $\hat{\rho}^{\mathsf{c}}$ be the family of morphisms, whose components are the compositions

$$\begin{tikzcd}
\Sigma_{\mathsf{c}}(X \times B(X,Y), X) \arrow[rr, "\Sigma_{\mathsf{c}}(\mathrm{id} \times (\eta \boxplus \mathrm{id}), \mathrm{id})"] & & \Sigma_{\mathsf{c}}(X \times T(D(X,Y)+Y), X) \arrow[dll, "\Sigma_{\mathsf{c}}(\tau, \mathrm{id})"'] \\
\Sigma_{\mathsf{c}}(T(X \times (D(X,Y)+Y)), X) \arrow[r, "\chi"] & T\Sigma_{\mathsf{c}}(X \times (D(X,Y)+Y), X) \arrow[r, "T\rho^{\mathsf{c}}"] & T\Sigma^\star(X+Y)
\end{tikzcd}$$

Suppose that the morphisms

$$\Theta(TX, TY, Z) \xrightarrow{\theta} \Sigma_{\mathsf{c}}(TX + TY, Z) \xrightarrow{\Sigma_{\mathsf{c}}(\mathrm{id} \boxplus \mathrm{id}, \mathrm{id})} \Sigma_{\mathsf{c}}(T(X+Y), Z) \xrightarrow{\chi} T\Sigma_{\mathsf{c}}(X+Y, Z)$$

factor through $T\theta \colon T\Theta(X,Y,Z) \to T\Sigma_{\mathsf{c}}(X+Y,Z)$. Then if $(\rho^{\mathsf{v}}, \rho^{\mathsf{c}}, \mathrm{id})$ is a strongly separated abstract higher-order GSOS law, then so is $(\rho^{\mathsf{v}}, \hat{\rho}^{\mathsf{c}}, \chi)$.

A typical case for using Proposition 3.11 is extending a deterministic law to a nondeterministic one, using the powerset monad.

Remark 3.12. As in Remark 3.5, consider $\Sigma_{\mathsf{c}}(X,Y) = X^n \times Y^m$, now on the category of sets. Let us choose $\Theta(X,Y,Z) \subseteq \Sigma_{\mathsf{c}}(X+Y,Z)$ to be $\{(\bar{v}, \bar{z}) \in (X+Y)^n \times Z^m \mid \exists i.\, \exists y \in Y.\, v_i = \mathsf{inr}\, y\}$ (so, θ is a set inclusion). Let $\mathbf{T}$ be the powerset functor $\mathcal{P}$, and assume the standard distributive law [25] $\chi_{X,Y}((A_i)_{i \in \{1,\dots,n\}}, \bar{y}) = \{(\bar{x}, \bar{y}) \mid x_i \in A_i\}$. Then, to show that $\chi \cdot \Sigma_{\mathsf{c}}(\mathrm{id} \boxplus \mathrm{id}, \mathrm{id}) \cdot \theta$ factors through $\mathcal{P}\theta$ is to show that for any $\bar{V} \in (\mathcal{P}X + \mathcal{P}Y)^n$ and $\bar{z} \in Z^m$ such that at least one V_i is of the form $\mathsf{inr}\, A$, if $(\bar{v}, \bar{z}') \in (\chi \cdot \Sigma_{\mathsf{c}}(\mathrm{id} \boxplus \mathrm{id}, \mathrm{id}))(\bar{V}, \bar{z})$ then also v_i is of the form $\mathsf{inr}\, y$. Let us show it by contradiction: suppose that $(\bar{v}, \bar{z}') \in (\chi \cdot \Sigma_{\mathsf{c}}(\mathrm{id} \boxplus \mathrm{id}, \mathrm{id}))(\bar{V}, \bar{z})$ but $v_i = \mathsf{inl}\, x$ for some $x \in X$. Then for some $(\bar{W}, \bar{z}') \in (\Sigma_{\mathsf{c}}(\mathrm{id} \boxplus \mathrm{id}, \mathrm{id}))(\bar{V}, \bar{z})$, W_i contains $\mathsf{inl}\, x$, contradicting to the assumption that $V_i = \mathsf{inr}\, A$.

4 Abstract Big-Step SOS

The natural transformations (7) and (8) model small-step semantics, and thus involve small-step transitions $\to$ and possibly others, which are binary relations between programs, i.e. closed Σ-terms. Contrastingly, the big-step operational semantics involves judgements of the form $t \Downarrow v$ where t is a program and v is a value. The expected big-step operational semantics for the extended combinatory logic (Example 3.3) is provided in Figure 6.

The combinatory logic example suggests three natural questions:

(1) How to define a general notion of big-step semantics?
(2) How to generally produce a big-step specification from a small-step specification?
(3) How to prove the equivalence of the big-step and the small-step semantics?

We deal with the first and the second question in this section, and with the third one in the next one.

Definition 4.1 (Abstract Big-Step SOS). Given two functors $\Sigma_{\mathsf{v}} \colon \mathbf{C} \to \mathbf{C}$, $\Sigma_{\mathsf{c}} \colon \mathbf{C} \times \mathbf{C} \to \mathbf{C}$, a strong monad $\mathbf{T}$ on $\mathbf{C}$, an *abstract big-step SOS* over these data is a natural transformation

$$\xi \colon \Sigma_{\mathsf{c}}(\Sigma_{\mathsf{v}} X, X) \to T(\Sigma^\star X) \tag{15}$$

where $\Sigma = \Sigma_{\mathsf{v}} + \Sigma_{\mathsf{c}}\Delta$.

Assuming that $T = \mathrm{Id}$, Definition 4.1 for one thing encodes the following concrete big-step operational semantics rules:

$$\frac{}{g(x_1,\ldots,x_n) \Downarrow g(x_1,\ldots,x_n)}$$

for all n-ary symbols g from Σ_{v}. These rules are fully determined by the notion of value, i.e. by the component Σ_{v} of the partitioning of Σ into Σ_{v} and Σ_{c}. The transformation (15) additionally encodes the rules of the form

$$\frac{x_1 \Downarrow g_1(x^1_1,\ldots,x^1_{n_1}) \quad \ldots \quad x_k \Downarrow g_k(x^k_1,\ldots,x^k_{n_k}) \qquad t \Downarrow v}{f(x_1,\ldots,x_n) \Downarrow v} \tag{16}$$

where f is a computation former whose strict arguments are precisely the first k, t is a term whose free variables are in $\{x^1_1,\ldots,x^1_{n_1},\ldots,x^k_1,\ldots,x^k_{n_k},x_{k+1},\ldots,x_n\}$, and v is a fresh variable referring to the result of evaluation of t. Specifications build from these two types of rules determine a natural transformation (15) with $T = \mathrm{Id}$, and hence also such transformations with arbitrary $\mathbf{T}$ by composition with the monad unit η.

Given an abstract higher-order separated GSOS law $(\rho^{\mathsf{v}}, \rho^{\mathsf{c}}, \chi)$, we define (15) via:

$$\begin{tikzcd}
\Sigma_{\mathsf{c}}(\Sigma_{\mathsf{v}}X, X) \arrow[rr, "\Sigma_{\mathsf{c}}(\Sigma_{\mathsf{v}}\eta{,}\eta)"] & & \Sigma_{\mathsf{c}}(\Sigma_{\mathsf{v}}(\Sigma^\star X), \Sigma^\star X) \arrow[dll, bend right=10, "\Sigma_{\mathsf{c}}(\langle\iota^{\mathsf{v}}{,}D(\mathrm{id}{,}\mu)\cdot\rho^{\mathsf{v}}\rangle{,}\mathrm{id})"'] \\
\Sigma_{\mathsf{c}}(\Sigma^\star X \times D(\Sigma^\star X, \Sigma^\star X), \Sigma^\star X) \arrow[rr, "T\nabla^\sharp\cdot\rho^{\mathsf{cv}}"] & & T(\Sigma^\star X)
\end{tikzcd} \tag{17}$$

This is indeed the way, the rules in Figure 6 are obtained from the rules in Figure 2.

Using (17) in derivations requires instantiating X with $\mu\Sigma$ and flattening $\Sigma^\star(\mu\Sigma)$ to $\mu\Sigma$. This results in a significant simplification.

Lemma 4.2. *The following diagram commutes:*

$$\begin{tikzcd}
\Sigma_{\mathsf{c}}(\mu\Sigma_{\mathsf{v}}, \mu\Sigma) \arrow[rr, "\xi"] \arrow[d, "\Sigma_{\mathsf{c}}(\langle\iota^{\mathsf{v}}{,}\gamma^{\mathsf{v}}\rangle{,}\mathrm{id})"'] & & T(\Sigma^\star\mu\Sigma) \arrow[d, "T\mu"] \\
\Sigma_{\mathsf{c}}(\mu\Sigma \times D(\mu\Sigma, \mu\Sigma), \mu\Sigma) \arrow[r, "\rho^{\mathsf{cv}}"] & T\Sigma^\star(\mu\Sigma + \mu\Sigma) \arrow[r, "T\nabla^\sharp"] & T(\mu\Sigma)
\end{tikzcd}$$

The proof of this lemma uses the (di-)naturality assumption on (8).

Example 4.3. Assuming that $\mathbf{T}$ is the identity monad, for the extended combinatory logic (Example 3.3) we obtain the following assignments:

$$\xi(\mathrm{app}(I,r)) = r \qquad \xi(\mathrm{app}(K,r)) = K'(r) \qquad \xi(\mathrm{app}(S,r)) = S'(r)$$
$$\xi(\mathrm{app}(K'(t),r)) = t \qquad \xi(\mathrm{app}(S'(t),r)) = S''(t,r)$$
$$\xi(\mathrm{app}(S''(s,t),r)) = \mathrm{app}(\mathrm{app}(s,r),\mathrm{app}(t,r))$$

◇

The power of abstract (higher-order) GSOS semantics is not only in that the derivation rules can be captured by a single (di-)natural transformation (3), but also in that this transformation allows one to execute this semantics via the ensuing notion of operational model γ, determined by (4) just as abstractly. Analogously, we need to define how big-step operational semantics is executed, i.e. how the process of deriving the judgements $t \Downarrow v$ is modelled, by providing a big-step counterpart of the HO-GSOS operational model. We seek to define a morphism

$$\hat{\zeta} \colon \mu\Sigma \to T(\mu\Sigma_{\mathsf{v}}), \tag{18}$$

which is meant to send a term t to a value v, possibly triggering a side-effect (in the simplest case, partiality). In general, $\mathbf{T}$ must be ω-continuous.

Let us explain briefly the role of the monad. The rule (16) involves premises that try to evaluate $x_1, \ldots, x_k$, which are all structurally smaller than $f(x_1, \ldots, x_n)$ from the conclusion. Hence, any derivation w.r.t. to these premises is necessarily well-founded. However, the premise $t \Downarrow v$ involves t, which need not be structurally smaller than $f(x_1, \ldots, x_n)$. For a given term of the form $f(t_1, \ldots, t_n)$, we thus cannot generally decide if $f(t_1, \ldots, t_n) \Downarrow v$ is derivable for some v. This phenomenon is not an artefact of the format (16), but an inherent feature of the big-step semantics. Consider for example the terms $\Omega_k = (I^k(SII))(I^k(SII))$ of **xCL** (where U^k refers to the term $U(\ldots U(UU)\ldots)$ with U repeated k times). We have

$$\Omega_k \to^{\star} (SII)(I^k(SII)) \to (S'(I)I)(I^k(SII)) \to (S''(I,I))(I^k(SII)) \to \Omega_{k+1}.$$

Even though every Ω_k has an outgoing unlabeled transition, no value can be reached from any of the Ω_k. In the big-step style this means that $\Omega_k \Downarrow v$ is not derivable for any v. Let us illustrate that with the following derivation fragment:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overline{S \Downarrow S}}{SI \Downarrow S'(I)}}{SII \Downarrow S''(I,I)}}{\vdots}}{I^k(SII) \Downarrow S''(I,I)} \qquad \dfrac{?}{\Omega_{k+1} \Downarrow v}}{\Omega_k \Downarrow v}$$

The rightmost premise of the rule would require a finite derivation whose size could not be smaller than that for the original goal $\Omega_k \Downarrow v$; hence, neither derivation can exist, i.e. $\Omega_k \Downarrow v$ is not derivable.

Assuming that $\mathbf{T}$ is ω-continuous and an ω-continues distributive law (9), we introduce (18) as the least fixpoint

$$\hat{\zeta} = \mu f.\, [\eta, f^\sharp \cdot T\mu \cdot \xi^\sharp \cdot \chi \cdot \Sigma_{\mathrm{c}}(f, \mathrm{id})] \cdot \iota^{-1} \tag{19}$$

where the composition $f^\sharp \cdot T\mu \cdot \xi^\sharp \cdot \chi \cdot \Sigma_{\mathrm{c}}(f, \mathrm{id})$ spells out as follows:

$$\begin{tikzcd}
\Sigma_{\mathrm{c}}(\mu\Sigma, \mu\Sigma) \arrow[r, "\Sigma_{\mathrm{c}}(f, \mathrm{id})"] & \Sigma_{\mathrm{c}}(T\mu\Sigma_{\mathrm{v}}, \mu\Sigma) \arrow[r, "\chi"] & T\Sigma_{\mathrm{c}}(\mu\Sigma_{\mathrm{v}}, \mu\Sigma) \arrow[dll, "\xi^\sharp"'] \\
T(\Sigma^\star\mu\Sigma) \arrow[r, "T\mu"] & T(\mu\Sigma) \arrow[r, "f^\sharp"] & T(\mu\Sigma_{\mathrm{v}})
\end{tikzcd}$$

Let $\zeta = \hat{\zeta} \cdot \iota^{\mathrm{c}} \colon \mu\Sigma_{\mathrm{c}} \to T(\mu\Sigma_{\mathrm{v}})$. By definition, $\hat{\zeta} = [\hat{\zeta} \cdot \iota^{\mathrm{v}}, \hat{\zeta} \cdot \iota^{\mathrm{c}}] \cdot \iota^{-1} = [\eta, \zeta] \cdot \iota^{-1}$.

5 Equivalence of Small-Step and Big-Step, Abstractly

In this section we assume that the monad $\mathbf{T}$ is ω-continuous. We then abstractly define multi-step transitions $t \to^{\star} v$ from computations to values as the least fixpoint

$$\beta = (T\iota^{-1} \cdot \gamma^{\mathrm{c}})^{\dagger} = \mu f.\, [\eta, f]^\sharp \cdot T\iota^{-1} \cdot \gamma^{\mathrm{c}} \colon \mu\Sigma_{\mathrm{c}} \to T(\mu\Sigma_{\mathrm{v}}). \tag{20}$$

In other words, β is the least solution of the equation $\beta = [\eta, \beta]^\sharp \cdot T\iota^{-1} \cdot \gamma^{\mathrm{c}}$. Let moreover $\hat{\beta} = [\eta, \beta] \cdot \iota^{-1} \colon \mu\Sigma \to T(\mu\Sigma_{\mathrm{v}})$.

Suppose that $D(X, Y) = Y^X$ and that Σ is some algebraic signature on **Set**. Let f be some $(n+m)$-ary computation former with precisely n strict first arguments. Suppose that $t_1 \to^{\star} v_1, \ldots, t_n \to^{\star} v_n$

and that $f(v_1,\dots,v_n,t_{n+1},\dots,t_m) \to t \to^\star v$ for some values $v_1,\dots,v_n,v$. We then expect that $f(t_1,\dots,t_{n+m}) \to^\star v$. The following lemma captures this fact abstractly.

Lemma 5.1. *Let $(\rho^{\mathrm{v}},\rho^{\mathrm{c}},\chi)$ be a strongly separated abstract higher-order GSOS law, such that the law χ is ω-continuous. Then*

$$\hat{\beta}^\sharp \cdot T\nabla^\sharp \cdot (\rho^{\mathrm{cv}})^\sharp \cdot \chi \cdot \Sigma_{\mathrm{c}}(T\langle \iota^{\mathrm{v}},\gamma^{\mathrm{v}}\rangle \cdot \hat{\beta},\mathrm{id}) \sqsubseteq \beta. \tag{21}$$

Proof Sketch. The key step is proving that the following diagram commutes:

$$\begin{tikzcd}
\mu\Sigma_{\mathrm{c}} \arrow[r, "\beta"] \arrow[d, "\Sigma_{\mathrm{c}}(\hat{\gamma}^{\mathrm{c}},\mathrm{id})"'] & T(\mu\Sigma_{\mathrm{v}}) \\
\Sigma_{\mathrm{c}}(T\mu\Sigma,\mu\Sigma) \arrow[r, "\chi"] & T(\mu\Sigma_{\mathrm{c}}) \arrow[u, "\beta^\sharp"']
\end{tikzcd} \tag{22}$$

To that end we use the fact that, by Assumption 3.4, $\mu\Sigma_{\mathrm{c}}$ is a coproduct of $\Sigma_{\mathrm{c}}(\mu\Sigma_{\mathrm{v}},\mu\Sigma)$ and $\Theta(\mu\Sigma_{\mathrm{v}},\mu\Sigma_{\mathrm{c}},\mu\Sigma)$. The diagram then falls into two equations:

$$\beta^\sharp \cdot \chi \cdot \Sigma_{\mathrm{c}}(\hat{\gamma}^{\mathrm{c}},\mathrm{id}) \cdot \Sigma_{\mathrm{c}}(\iota^{\mathrm{v}},\mathrm{id}) = \beta \cdot \Sigma_{\mathrm{c}}(\iota^{\mathrm{v}},\mathrm{id}),$$
$$\beta^\sharp \cdot \chi \cdot \Sigma_{\mathrm{c}}(\hat{\gamma}^{\mathrm{c}},\mathrm{id}) \cdot \Sigma_{\mathrm{c}}(\iota,\mathrm{id}) \cdot \theta = \beta \cdot \Sigma_{\mathrm{c}}(\iota,\mathrm{id}) \cdot \theta.$$

The first is obtained by unfolding definitions and using the fact that χ is a distributive law. The second one relies on (14). Using the diagram, the goal is obtained by induction, using the fact that (by Kleene's fixpoint theorem) $\hat{\beta} = \bigsqcup_n \hat{\beta}_n$ where $\hat{\beta}_0 = [\eta,\bot] \cdot \iota^{-1}$, $\hat{\beta}_{n+1} = \hat{\beta}_n^\sharp \cdot \hat{\gamma}^{\mathrm{c}}$. □

Another property that we need to abstract is that $t \to s$ together with $s \Downarrow v$ entails $t \Downarrow v$.

Lemma 5.2. *Let $(\rho^{\mathrm{v}},\rho^{\mathrm{c}},\chi)$ be an strongly separated abstract HO-GSOS with ω-continuous χ and let ξ be defined by* (17)*. Then $\hat{\zeta}^\sharp \cdot \gamma^{\mathrm{c}} \sqsubseteq \zeta$.*

Proof Sketch. The diagram (11) identifies γ^{c} as the unique fixpoint of the map $f \mapsto T\nabla^\sharp \cdot \rho^{\mathrm{c}} \cdot \Sigma_{\mathrm{c}}(\langle \mathrm{id},(\gamma^{\mathrm{v}}+f)\cdot\iota^{-1}\rangle,\mathrm{id})$, hence, also as the least fixpoint. Thus $\gamma^{\mathrm{c}} = \bigsqcup_n \gamma^{\mathrm{c}}_n$ where $\gamma^{\mathrm{c}}_0 = \bot$ and $\gamma^{\mathrm{c}}_{n+1} = T\nabla^\sharp \cdot \rho^{\mathrm{c}} \cdot \Sigma_{\mathrm{c}}(\langle \mathrm{id},(\gamma^{\mathrm{v}}+\gamma^{\mathrm{c}}_n)\cdot\iota^{-1}\rangle,\mathrm{id})$, and it suffices to show $\hat{\zeta}^\sharp \cdot \gamma^{\mathrm{c}}_n \sqsubseteq \zeta$ for all n. The induction base is obvious. The induction step, by Assumption 3.4, we reduce to two subgoals:

$$\hat{\zeta}^\sharp \cdot \gamma^{\mathrm{c}}_{n+1} \cdot \Sigma_{\mathrm{c}}(\iota^{\mathrm{v}},\mathrm{id}) \sqsubseteq \zeta \cdot \Sigma_{\mathrm{c}}(\iota^{\mathrm{v}},\mathrm{id}),$$
$$\hat{\zeta}^\sharp \cdot \gamma^{\mathrm{c}}_{n+1} \cdot \Sigma_{\mathrm{c}}(\iota,\mathrm{id}) \cdot \theta \sqsubseteq \zeta \cdot \Sigma_{\mathrm{c}}(\iota,\mathrm{id}) \cdot \theta.$$

The first one is easy to obtain by unfolding definitions. The second one is a result of a somewhat tedious calculation, using the induction hypothesis and the strong separation assumption. □

We proceed with formalizing and proving an abstract version of the equivalence (⋆) between the small-step and the big-step semantics for strongly separated abstract HO-GSOS laws. To that end, we first rewrite the formula (19) in the setting when ξ comes from (17).

Proposition 5.3. *Let $(\rho^{\mathrm{v}},\rho^{\mathrm{c}},\chi)$ be an abstract higher-order separated GSOS with ω-continuous χ and let ξ be defined by* (17)*. Then*

$$\hat{\zeta} = \mu f.\, [\eta, f^\sharp \cdot T\nabla^\sharp \cdot (\rho^{\mathrm{cv}})^\sharp \cdot \chi \cdot \Sigma_{\mathrm{c}}(T\langle \iota^{\mathrm{v}},\gamma^{\mathrm{v}}\rangle \cdot f,\mathrm{id})] \cdot \iota^{-1}. \tag{23}$$

Finally, we prove our main result.

Theorem 5.4. *Let $(\rho^{\mathrm{v}},\rho^{\mathrm{c}},\chi)$ be a strongly separated abstract higher-order GSOS with ω-continuous χ and let ξ be defined by* (17)*. Then $\hat{\zeta} = \hat{\beta}$.*

Proof. We proceed by proving mutual inequality.

$\hat{\zeta} \sqsubseteq \hat{\beta}$. By Proposition 5.3, $\hat{\zeta}$ is the least pre-fixpoint of (23). By Lemma 5.1, $\hat{\beta}$ is a pre-fixpoint of the same function. Hence, indeed, $\hat{\zeta} \sqsubseteq \hat{\beta}$.

$\hat{\beta} \sqsubseteq \hat{\zeta}$. The goal is equivalent to $[\eta, \beta] \cdot \iota^{-1} \sqsubseteq [\eta, \zeta] \cdot \iota^{-1}$. Proving this is equivalent to proving that $\beta \sqsubseteq \zeta$, which will then be our new goal. By definition, β is the least pre-fixpoint of the map $f \mapsto [\eta, f]^\sharp \cdot T\iota^{-1} \cdot \gamma^{\mathsf{c}}$. Hence, it suffices to prove that ζ is a pre-fixpoint of the same map, i.e. the inequality $[\eta, \zeta]^\sharp \cdot T\iota^{-1} \cdot \gamma^{\mathsf{c}} \sqsubseteq \zeta$, i.e. $\hat{\zeta}^\sharp \cdot \gamma^{\mathsf{c}} \sqsubseteq \zeta$, which we have by Lemma 5.2. □

6 Case Studies

We proceed to consider various aspects of programming languages from the perspective of our framework, and demonstrate in detail, how it can cope with them.

6.1 Types

In this section, we stick to a typed version of **xCL**, previously dubbed **xTCL** [22]. The transition from **xCL** to **xTCL** demonstrates the strength of our categorical modeling: from **Set** as the ambient category **C** we simply switch to the category of sorted sets $\mathbf{Set}^{\mathsf{Ty}}$, where the sorts Ty are indeed the conventional types of typed combinatory logic [26]. The remaining ingredients of the framework are then upgraded as expected, which we detail further below.

Assuming a postulated set of basic sorts $\mathcal{S}$, the set of types Ty is defined by the grammar:

$$\mathsf{Ty} ::= (\tau \in \mathcal{S}) \mid \mathsf{Ty} \rightarrowtriangle \mathsf{Ty}.$$

More concretely, an object A of **C** is a family of sets $(A_\tau)_{\tau \in \mathsf{Ty}}$ and a morphism $f \colon A \to B$ in **C** is a family of functions $(f_\tau \colon A_\tau \to B_\tau)_{\tau \in \mathsf{Ty}}$. Morphism composition $g \cdot f$ is defined as $(g_\tau \cdot f_\tau)_{\tau \in \mathsf{Ty}}$. Note that **C** is a presheaf topos over Ty as a discrete category, in particular, limits and colimits in **C** are computed pointwise. We introduce Σ_{v} and Σ_{c} for **xTCL** as follows:

$$\Sigma_{\mathsf{v}}(X)_\tau = \coprod\nolimits_{f \colon \tau_1, \dots, \tau_n \to \tau \in Ops} \prod\nolimits_{i=1}^{n} X_{\tau_i} \qquad \Sigma_{\mathsf{c}}(X, Y)_\tau = \{\mathsf{app}_{\tau', \tau}\} \times (X_{\tau' \rightarrowtriangle \tau} \times Y_{\tau'}),$$

where $Ops = \{I_{\tau_1}, K_{\tau_1,\tau_2}, K'_{\tau_1,\tau_2}, S_{\tau_1,\tau_2,\tau_3}, S'_{\tau_1,\tau_2,\tau_3}, S''_{\tau_1,\tau_2,\tau_3}\}$ is the set of operations, constituting Σ_{v}, and typed in the expected way. For example, the type of K'_{τ_1,τ_2} is $\tau_1 \to (\tau_2 \rightarrowtriangle \tau_1)$. Analogously, we view Σ_{c} as a signature of one binary operation $\mathsf{app}_{\tau_1,\tau_2}$ of type $(\tau_1 \rightarrowtriangle \tau_2), \tau_1 \to \tau_2$.

To define the behaviour functor B, let $\mathbf{T}$ be the identity monad, and let D be as follows:

$$D(X, Y)_\tau = \{\} \quad \text{for } \tau \in \mathcal{S}, \qquad D(X, Y)_{\tau_1 \rightarrowtriangle \tau_2} = Y_{\tau_2}^{X_{\tau_1}}$$

and then $B(X, Y)_\tau = \mathsf{fix}_\tau$ if $\tau \in \mathcal{S}$ and $B(X, Y)_{\tau_1 \rightarrowtriangle \tau_2} = Y_{\tau_1 \rightarrowtriangle \tau_2} + Y_{\tau_2}^{X_{\tau_1}}$. Next, we define ρ^{v} and ρ^{c} essentially like in Example 3.3, modulo adding the typing information. For example, the clause for S'' in the definition of $\rho^{\mathsf{v}}_\tau \colon \Sigma_{\mathsf{v}}(X)_\tau \to D(X, \Sigma^\star X)_\tau$ becomes

$$\rho^{\mathsf{v}}_\tau(S''_{\tau_1,\tau_2,\tau}(t, s))(r) = \mathsf{app}_{\tau_2,\tau}(\mathsf{app}_{\tau_1,\tau_2 \rightarrowtriangle \tau}(t, r), \mathsf{app}_{\tau_1,\tau_2}(s, r)).$$

The definition of $\rho^{\mathsf{c}}_\tau \colon \Sigma_{\mathsf{c}}(X \times B(X, Y), X)_\tau \to \Sigma^\star(X + Y)_\tau$ is as follows:

$$\begin{aligned} \rho^{\mathsf{c}}_\tau(\mathsf{app}_{\tau_1,\tau_2}((s, s'), t)) &= \mathsf{app}_{\tau_1,\tau_2}(s', t) && \text{if } s' \in Y_{\tau_1 \rightarrowtriangle \tau_2} \\ \rho^{\mathsf{c}}_\tau(\mathsf{app}_{\tau_1,\tau_2}((s, f), t)) &= f(t) && \text{if } f \in Y_{\tau_2}^{X_{\tau_1}} \end{aligned}$$

Finally, we obtain $\xi_\tau \colon \Sigma_{\mathsf{c}}(\Sigma_{\mathsf{v}} X, X)_\tau \to \Sigma^\star(X)_\tau$ by instantiating the general definition (17). For example, we have $\xi_\tau(\mathsf{app}_{\tau_1,\tau}(S''_{\tau_1,\tau_2,\tau}(t, s), r \colon \tau_1)) = \mathsf{app}_{\tau_2,\tau}(\mathsf{app}_{\tau_1,\tau_2 \rightarrowtriangle \tau}(t, r), \mathsf{app}_{\tau_1,\tau_2}(s, r))$.

6.2 Recursion and Conditionals

In this section, we complement **xTCL** with recursion and control in the style of PCF [33]. We assume the sort $\mathsf{bool} \in \mathcal{S}$ of Booleans. Then we add the following operators to the signature:

$$\mathsf{fix}_\tau \colon (\tau \rightarrowtriangle \tau) \rightarrowtriangle \tau \qquad\qquad \mathsf{if}_\tau \colon \mathsf{bool}, \tau, \tau \longrightarrow \tau \qquad\qquad \mathsf{true}, \mathsf{false} \colon \mathsf{bool}$$

More concretely, this amounts to defining Σ_v with the same formula from Ops, which is now the set $\{I_{\tau_1}, K_{\tau_1,\tau_2}, K'_{\tau_1,\tau_2}, S_{\tau_1,\tau_2,\tau_3}, S'_{\tau_1,\tau_2,\tau_3}, S''_{\tau_1,\tau_2,\tau_3}, \mathsf{fix}_{\tau_1}\}$, and to defining Σ_c as follows:

$$\Sigma_\mathsf{c}(X, Y)_\tau = \{\mathsf{app}_{\tau',\tau}\} \times (X_{\tau' \rightarrowtriangle \tau} \times Y_{\tau'}) \; + \; \{\mathsf{if}_\tau\} \times (X_\mathsf{bool} \times Y_\tau \times Y_\tau).$$

The left summand corresponds to the application operator $\mathsf{app}_{\tau_1,\tau_2}$, as before, while the right summand corresponds to the conditional statement if_τ. This indicates that the first argument of if_τ is strict and the others are lazy. The new operators are subject to the following small-step specification:

$$\frac{}{\mathsf{fix}_\tau \xrightarrow{t} t(\mathsf{fix}_\tau t)} \qquad \frac{}{\mathsf{true}{\downarrow_\mathsf{true}}} \qquad \frac{}{\mathsf{false}{\downarrow_\mathsf{false}}}$$

$$\frac{b \to b'}{\mathsf{if}_\tau(b, s, t) \to \mathsf{if}_\tau(b', s, t)} \qquad \frac{b{\downarrow_\mathsf{true}}}{\mathsf{if}_\tau(b, s, t) \to s} \qquad \frac{b{\downarrow_\mathsf{false}}}{\mathsf{if}_\tau(b, s, t) \to t}$$

where we use a new type of judgements $b{\downarrow_\mathsf{true}}$ and $b{\downarrow_\mathsf{false}}$, indicating that $b = \mathsf{true}$ and $b = \mathsf{false}$ correspondingly. Note that the if-then-else statement suffices to program the standard logical connectives: $\neg b = \mathsf{if}_\mathsf{bool}(b, \mathsf{false}, \mathsf{true})$, $b \wedge b' = \mathsf{if}_\mathsf{bool}(b, b', \mathsf{false})$, $b \vee b' = \mathsf{if}_\mathsf{bool}(b, \mathsf{true}, b')$. To address the above rules we modify D as follows:

$$D(X, Y)_{\tau_1 \rightarrowtriangle \tau_2} = Y_{\tau_2}^{X_{\tau_1}} \qquad\qquad D(X, Y)_\tau = \begin{cases} \{\mathsf{true}, \mathsf{false}\} & \text{if } \tau = \mathsf{bool} \\ \{\} & \text{if } \tau \in \mathcal{S} \setminus \{\mathsf{bool}\} \end{cases}$$

Finally, we complement the definitions of ρ^v and ρ^c with the clauses

$$\begin{aligned} \rho^\mathsf{v}_\tau(\mathsf{fix}_\tau)(t) &= \mathsf{app}_{\tau,\tau}(t, \mathsf{app}_{\tau \rightarrowtriangle \tau, \tau}(\mathsf{fix}_\tau, t)) \qquad & \rho^\mathsf{c}_\tau(\mathsf{if}_\tau((b, b'), s, t)) &= \mathsf{if}_\tau(b', s, t) \\ \rho^\mathsf{v}_\mathsf{bool}(\mathsf{true}) &= \mathsf{true} & \rho^\mathsf{c}_\tau(\mathsf{if}_\tau((b, \mathsf{true}), s, t)) &= s \\ \rho^\mathsf{v}_\mathsf{bool}(\mathsf{false}) &= \mathsf{false} & \rho^\mathsf{c}_\tau(\mathsf{if}_\tau((b, \mathsf{false}), s, t)) &= t \end{aligned}$$

It is easy to see by Remark 3.7 that $(\rho^\mathsf{v}, \rho^\mathsf{c})$ constitute a strongly separated abstract HO-GSOS law. By applying (17) we obtain the following new clauses for ξ:

$$\begin{aligned} \xi_\tau(\mathsf{app}_{\tau \rightarrowtriangle \tau, \tau}(\mathsf{fix}_\tau, t \colon \tau \rightarrowtriangle \tau)) &= \mathsf{app}_{\tau,\tau}(t, \mathsf{app}_{\tau \rightarrowtriangle \tau, \tau}(\mathsf{fix}_\tau, t)) \\ \xi_\tau(\mathsf{if}_\tau(\mathsf{true}, s, t)) &= s \\ \xi_\tau(\mathsf{if}_\tau(\mathsf{false}, s, t)) &= t \end{aligned}$$

Equivalently: the following big-step rules:

$$\frac{s \Downarrow \mathsf{fix}_\tau \quad t(\mathsf{fix}_\tau t) \Downarrow v}{st \Downarrow v} \qquad \frac{b \Downarrow \mathsf{true} \quad s \Downarrow v}{\mathsf{if}_\tau(b, s, t) \Downarrow v} \qquad \frac{b \Downarrow \mathsf{false} \quad t \Downarrow v}{\mathsf{if}_\tau(b, s, t) \Downarrow v}$$

6.3 Nondeterminism and Parallelism

We proceed to illustrate how our modeling can cope with nondeterminism and parallelism in higher-order setting by extending **xCL** suitably. The standard binary *erratic choice* operator $\oplus$ [14] can easily be specified with the small-step rules:

$$\frac{}{t \oplus s \to t} \qquad\qquad \frac{}{t \oplus s \to s} \tag{24}$$

This amounts to updating the separable abstract HO-GSOS law in Example 3.3 as follows: $\Sigma_{\mathsf{c}}(X,Y) = \{\mathsf{app}\} \times (X \times Y) + \{\oplus\} \times (Y \times Y)$, $\mathbf{T}$ is the powerset monad $\mathcal{P}$, and the clauses for ρ^{c} are as follows:

$$\begin{aligned} \rho^{\mathsf{c}}(\mathsf{app}((t,S),s)) &= \{\mathsf{app}(t',s) \mid t' \in S \cap Y\} \cup \{f(s) \mid f\colon X \to Y \in S\} \\ \rho^{\mathsf{c}}(t \oplus s) &= \{t,s\} \end{aligned}$$

The rules in (24) thus jointly correspond to a single clause for ρ^{c}, stating that $\{t,s\}$ is the set of direct successors of $t \oplus s$. The construction (17) produces an abstract big-step SOS, which can be represented with the rules:

$$\frac{t \Downarrow v}{t \oplus s \Downarrow v} \qquad\qquad \frac{s \Downarrow v}{t \oplus s \Downarrow v}$$

In order to specify the behaviour of a parallel composition operator we orient to *concurrent λ-calculus* [15]. This is an extension of the (call-by-name) λ-calculus with the erratic choice operator, as above, with a (fair) parallel composition operator $\|$ and with a certain version of call-by-value evaluation. The latter feature is independent of the others, and was added to express the well-known *parallel-or* operator. We will presently not include call-by-value evaluation, but consider it in dedicated Section 6.4 instead. Intuitively, in $s \parallel t$, s and t are reduced simultaneously if possible, and otherwise the one that can be reduced is reduced, while keeping the other one intact; eventually, neither s nor t can be reduced, meaning that $s \parallel t$ is morally a value. We capture the latter situation by arranging a reduction from $s \parallel t$ to $s \mathbin{\underline{\|}} t$ where $\underline{\|}$ is a new value former.

$$\frac{t \to t' \quad s \to s'}{t \parallel s \to t' \parallel s'} \qquad \frac{t \xrightarrow{r} t' \quad s \to s'}{t \parallel s \to t \parallel s'} \qquad \frac{t \to t' \quad s \xrightarrow{r} s'}{t \parallel s \to t' \parallel s}$$

$$\frac{t \xrightarrow{r} t' \quad s \xrightarrow{r} s'}{t \parallel s \to t \mathbin{\underline{\|}} s} \qquad \frac{}{t \mathbin{\underline{\|}} s \xrightarrow{r} tr \parallel sr}$$

These rules yield the following new clauses for ρ^{c}:

$$\begin{aligned} \rho^{\mathsf{c}}((t,T) \parallel (s,S)) &= \{t' \parallel s' \mid t' \in T, s' \in S\} & \rho^{\mathsf{c}}((t,f) \parallel (s,S)) &= \{t \parallel s' \mid s' \in S\} \\ \rho^{\mathsf{c}}((t,T) \parallel (s,g)) &= \{t' \parallel s \mid t' \in T\} & \rho^{\mathsf{c}}((t,f) \parallel (s,g)) &= \{t \mathbin{\underline{\|}} s\} \end{aligned}$$

and the following new clause for ρ^{v}:

$$\rho^{\mathsf{v}}(t \mathbin{\underline{\|}} s)(r) = (t \cdot r) \parallel (s \cdot r).$$

The new big-step rules are as follows:

$$\frac{s \Downarrow v \mathbin{\underline{\|}} u \qquad vt \parallel ut \Downarrow w}{st \Downarrow w} \qquad\qquad \frac{s \Downarrow v \qquad t \Downarrow u}{s \parallel t \Downarrow v \mathbin{\underline{\|}} u}$$

6.4 Call-by-Value

Rules in Figure 2 can be easily modified to capture the familiar call-by-value evaluation strategy:

$$\frac{}{S \xrightarrow{t} S'(t)} \qquad \frac{}{S'(t) \xrightarrow{s} S''(t,s)} \qquad \frac{}{S''(t,s) \xrightarrow{r} (tr)(sr)}$$

$$\frac{}{K \xrightarrow{t} K'(t)} \qquad \frac{}{K'(t) \xrightarrow{s} t} \qquad \frac{}{I \xrightarrow{t} t}$$

$$\frac{t \to t'}{ts \to t's}\ (a) \qquad \frac{t \xrightarrow{r} t' \quad s \to s'}{ts \to ts'}\ (b) \qquad \frac{t \xrightarrow{s} t' \quad s \xrightarrow{r} s'}{ts \to t'}\ (c)$$

In this specification, we reduce ts by first reducing t (a), unless it expects an input, i.e. t is a value, in which case we reduce s (b), unless also s is a value; in the remaining case we evaluate t on s (c).

To turn this specification into a separated abstract HO-GSOS, we need to decide which arguments of the application operator are lazy and which are strict. In general, the behaviour of ts depends both on the behaviour of t and of s, hence both arguments must be strict. However, as explained in Remark 3.7, for the above specification to be strongly separated, it must contain the rule

$$\frac{t \to t' \qquad s \to s'}{t\,s \to t'\,s'}\ (a1)$$

which is not the case. We can amend the original specification by replacing (a) by (a1) together with

$$\frac{t \to t' \qquad s \xrightarrow{r} s'}{t\,s \to t'\,s}\ (a2)$$

This results in a specification to which we can apply Theorem 5.4 and obtain the equivalence of the small-step semantics with the following big-step semantics:

$$\begin{gathered}
\frac{}{v \Downarrow v} \qquad \frac{s \Downarrow I \quad t \Downarrow v}{st \Downarrow v} \qquad \frac{s \Downarrow K \quad t \Downarrow w \quad K'(w) \Downarrow v}{st \Downarrow v} \\[1em]
\frac{s \Downarrow S \quad t \Downarrow w \quad S'(w) \Downarrow v}{st \Downarrow v} \qquad \frac{s \Downarrow K'(r) \quad t \Downarrow w \quad r \Downarrow v}{st \Downarrow v} \\[1em]
\frac{s \Downarrow S'(r) \quad t \Downarrow w \quad S''(r,w) \Downarrow v}{st \Downarrow v} \qquad \frac{s \Downarrow S''(r,q) \quad t \Downarrow w \quad (rw)(qw) \Downarrow v}{st \Downarrow v}
\end{gathered} \tag{25}$$

Although, the variant of the small-step specification with (a) replaced by (a1) and (a2) makes perfect sense, the resulting multi-step relation $\to^\star$ is not the standard one: e.g. originally $(I\,I)\,(I\,I) \to^\star I\,(II)$, but in the modified system $(I\,I)\,(I\,I) \to I\,I$, and hence $(I\,I)\,(I\,I) \not\to^\star I\,(II)$. To model the standard multi-step relation, we replace the rules (a)–(c) with the following ones:

$$\begin{gathered}
\frac{s \to s'}{s\,t \to s'\,t} \qquad \frac{s \xrightarrow{r} s'}{s\,t \to s \mathbin{\LEFTcircle} t} \qquad \frac{t \to t'}{s \mathbin{\LEFTcircle} t \to s \mathbin{\LEFTcircle} t'} \\[1em]
\frac{t \xrightarrow{r} t'}{s \mathbin{\LEFTcircle} t \to s \bullet t} \qquad \frac{s \to s'}{s \bullet t \to s' \bullet t} \qquad \frac{s \xrightarrow{t} s'}{s \bullet t \to s'}
\end{gathered}$$

where $\LEFTcircle$ and $\bullet$ are new auxiliary composition operators. The idea is to treat the original application operator (juxtaposition) and $\bullet$ as strict in the first argument and lazy in the second, and $\LEFTcircle$ as lazy in the first argument and strict in the second. The original multi-step behaviour is thus recovered, strong separation holds and the equivalent big-step specification takes the form:

$$\begin{gathered}
\frac{}{v \Downarrow v} \qquad \frac{s \Downarrow w \quad w \mathbin{\LEFTcircle} t \Downarrow v}{st \Downarrow v} \qquad \frac{t \Downarrow w \quad s \bullet w \Downarrow v}{s \mathbin{\LEFTcircle} t \Downarrow v} \qquad \frac{s \Downarrow I \quad t \Downarrow v}{s \bullet t \Downarrow v} \qquad \frac{s \Downarrow K \quad K'(t) \Downarrow v}{s \bullet t \Downarrow v} \\[1em]
\frac{s \Downarrow S \quad S'(t) \Downarrow v}{s \bullet t \Downarrow v} \qquad \frac{s \Downarrow K'(r) \quad r \Downarrow v}{s \bullet t \Downarrow v} \qquad \frac{s \Downarrow S'(r) \quad S''(r,t) \Downarrow v}{s \bullet t \Downarrow v} \qquad \frac{s \Downarrow S''(r,q) \quad (rt)(qt) \Downarrow v}{s \bullet t \Downarrow v}
\end{gathered}$$

Observe that the only way to obtain $st \Downarrow v$ is by using the following derivation

$$\dfrac{\dfrac{\vdots}{s \Downarrow w} \qquad \dfrac{\dfrac{}{t \Downarrow u} \qquad \dfrac{\vdots}{w \bullet u \Downarrow v}}{w \mathbin{\LEFTcircle} t \Downarrow v}}{st \Downarrow v}$$

where $\vdots$ refers to undischarged premises. By inspecting the six rules whose conclusions match $w \bullet u \Downarrow v$, it is easy to see that for terms in the original signature (without $\circ\!\!\!\!\bullet$ and $\bullet$) we obtain the same big-step semantics as in (25). In summary, (25) is equivalent to the two variants of the small-step semantics: the one with the rules (a), (b), (c) and the one with the rules (a1), (a2), (b), (c). Hence, they are equivalent to each other in the sense that $t \to^\star v$ in one system iff $t \to^\star v$ in the other system (for any value v).

Our use of the auxiliary operators $\circ\!\!\!\!\bullet$ and $\bullet$ tactically helped us to satisfy our strong separation condition. However, such operations emerge independently in the context of *pretty-big-step semantics* [10]. In fact, one can say that what we generally call abstract big-step SOS, specialized to the pretty-big-step semantics in this case.

7 Variable Binders and the λ-calculus

So far, we stuck to the extended combinatory logic as the core higher-order vehicle for language features of interest. Modeling languages with binders adds a certain technical overhead to the problem of modeling the small-step and the big-step semantics categorically. In fact, the type profile in (7) turns out to be too restrictive. In this section we show how to remedy this and to cover the λ-calculus. Similar issue arose recently in categorical treatment of logical predicates for languages with binders [22], and required a refinement of abstract HO-GSOS, called λ-laws.

7.1 Separable HO-GSOS for Languages with Binders

We begin by equipping our ambient category $\mathbf{C}$ with a closed monoidal structure $(\mathbf{C}, V, \bullet, -\!\!\bullet)$, meant to abstract from the internal mechanisms of variable management [16]. Monoidal closedness yields a natural isomorphism $(-)^\flat \colon \mathbf{C}(X \bullet Y, Z) \to \mathbf{C}(X, Y -\!\!\bullet Z)$, which we will need below.

In this setting, we need the following technical

Definition 7.1 (Pointed Strength [16, 27]). Let us denote by j the forgetful functor from $V/\mathbf{C}$ to $\mathbf{C}$, sending a morphism $V \to X$ to X. Objects of $V/\mathbf{C}$ are called *(V-)pointed objects* of $\mathbf{C}$.

A *(V-)pointed strength* on an endofunctor $F \colon \mathbf{C} \to \mathbf{C}$ is a family of morphisms $\mathsf{st}_{X,Y} \colon FX \bullet jY \to F(X \bullet jY)$, natural in $X \in \mathbf{C}$ and $Y \in V/\mathbf{C}$, such that the following diagrams commute:

$$\begin{tikzcd}
& FX \arrow[dl, phantom, "\cong" sloped] \arrow[dr, phantom, "\cong" sloped] & \\
FX \bullet V \arrow[rr, "\mathsf{st}_{X,V}"] & & F(X \bullet V)
\end{tikzcd}
\qquad
\begin{tikzcd}
(FX \bullet jY) \bullet jZ \arrow[r, "\mathsf{st}_{X,Y} \bullet jZ"] \arrow[d, phantom, "\cong" rotate=90] & F(X \bullet jY) \bullet jZ \arrow[r, "\mathsf{st}_{X \bullet jY, Z}"] & F((X \bullet jY) \bullet jZ) \arrow[d, phantom, "\cong" rotate=90] \\
FX \bullet (jY \bullet jZ) \arrow[rr, "\mathsf{st}_{X, Y \bullet Z}"] & & F(X \bullet (jY \bullet jZ))
\end{tikzcd}$$

(eliding the names of the canonical isomorphisms).

We then assume that Σ_{v} has the form $V + \Sigma'_{\mathsf{v}}$ and that the functor $\Sigma'_{\mathsf{v}} + \Sigma_{\mathsf{c}}\Delta$ is V-strong. This guarantees that the initial algebra $\mu\Sigma$ (if it exists) is a monoid [17, Theorem 4.1], whose multiplication we denote as $\mathsf{subst} \colon \mu\Sigma \bullet \mu\Sigma \to \mu\Sigma$.

We modify our framework of separable abstract HO-GSOS laws slightly by replacing (7) with

$$\rho^{\mathsf{v}}_{X,Y} \colon \Sigma_{\mathsf{v}}(jX \times (jX -\!\!\bullet Y)) \to D(jX, \Sigma^\star(jX + Y)), \tag{26}$$

dinatural in $X \in V/\mathbf{C}$ and natural in $Y \in |\mathbf{C}|$.

We then redefine γ^{v} as follows:

$$\mu\Sigma_{\mathsf{v}} \xrightarrow{\Sigma_{\mathsf{v}}\langle \mathsf{id}, \mathsf{subst}^\flat \rangle} \Sigma_{\mathsf{v}}(\mu\Sigma \times (\mu\Sigma -\!\!\bullet \mu\Sigma)) \xrightarrow{\rho^{\mathsf{v}}} D(\mu\Sigma, \Sigma^\star(\mu\Sigma + \mu\Sigma)) \xrightarrow{D(\mathsf{id}, \nabla^\sharp)} D(\mu\Sigma, \mu\Sigma)$$

and, as before, obtain γ^{c} from it as the unique such morphism that the diagram

$$\begin{tikzcd}
\Sigma_{\mathsf{c}}(\mu\Sigma_{\mathsf{v}} + \mu\Sigma_{\mathsf{c}}, \mu\Sigma) \arrow[rr, "\Sigma_{\mathsf{c}}(\iota,\mathrm{id})"] \arrow[d, "\Sigma_{\mathsf{c}}(\langle \iota, \gamma^{\mathsf{v}}+\gamma^{\mathsf{c}}\rangle,\mathrm{id})"'] & & \mu\Sigma_{\mathsf{c}} \arrow[d, "\gamma^{\mathsf{c}}"] \\
\Sigma_{\mathsf{c}}(\mu\Sigma \times B(\mu\Sigma,\mu\Sigma), \mu\Sigma) \arrow[r, "\rho^{\mathsf{c}}"] & T\Sigma^{\star}(\mu\Sigma + \mu\Sigma) \arrow[r, "T\nabla^{\sharp}"] & T\mu\Sigma
\end{tikzcd}$$

commutes. Using the new operational model $(\gamma^{\mathsf{v}}, \gamma^{\mathsf{c}})$, we obtain the multistep semantics β using the same formula (20).

7.2 Strongly Separable HO-GSOS for the λ-calculus

Let us spell out, how the above applies to the (call-by-name) λ-calculus.

Category. Following Fiore et al. [17], let $\mathbb{F}$ be the category of finite cardinals, the skeleton of the category of finite sets. The objects of $\mathbb{F}$ are the sets $n = \{0, \dots, n-1\}$ with $n \in \mathbb{N}$, and morphisms $n \to m$ are functions. Let $\mathbf{C}$ be the category of presheaves $\mathbf{Set}^{\mathbb{F}}$. Intuitively, the objects of $\mathbf{Set}^{\mathbb{F}}$ are families $(X(n))_{n\in\mathbb{N}}$ of sets where each $X(n)$ is meant to parametrically depend on $0, \dots, n-1$ as free variables.

The process of substituting terms for variables can be treated at the abstract level of presheaves as follows. For every presheaf $Y \in \mathbf{Set}^{\mathbb{F}}$, there is a functor $- \bullet Y \colon \mathbf{Set}^{\mathbb{F}} \to \mathbf{Set}^{\mathbb{F}}$ given by

$$(X \bullet Y)(m) = \int^{n\in\mathbb{F}} X(n) \times (Y(m))^n = \left(\coprod\nolimits_{n\in\mathbb{F}} X(n) \times (Y(m))^n\right)\Big/_{\approx} \tag{27}$$

where $\approx$ is the equivalence relation generated by all pairs

$$(x, y_0, \dots, y_{n-1}) \approx (x', y'_0, \dots, y'_{k-1})$$

such that $(x, y_0, \dots, y_{n-1}) \in X(n) \times (Y(m))^n$, $(x', y'_0, \dots, y'_{k-1}) \in X(k) \times Y(m)^k$ and there exists $r \colon n \to k$ satisfying $x' = X(r)(x)$ and $y_i = y'_{r(i)}$ for $i = 0, \dots, n-1$. An equivalence class in (27) can be thought of as a term $x \in X(n)$ with n free variables, together with n terms $y_0, \dots, y_{n-1} \in Y(m)$ to be substituted for them. The equivalence relation then captures the idea that the outcome of the substitution should be invariant under renamings that reflect equalities among $y_0, \dots, y_{n-1}$; for instance, if $y_i = y_j$ and $r \colon n \to n$ is the bijective renaming that swaps i and j, then substituting $y_0, \dots, y_{n-1}$ for $0, \dots, n-1$ in the term $X(r)(x)$ should produce the same outcome. Varying Y, one obtains the *substitution tensor*

$$- \bullet - \colon \mathbf{Set}^{\mathbb{F}} \times \mathbf{Set}^{\mathbb{F}} \to \mathbf{Set}^{\mathbb{F}},$$

which makes $\mathbf{Set}^{\mathbb{F}}$ into a (non-symmetric) monoidal category with the following *presheaf of variables V* as the unit object: $V(n) = \{0, \dots, n-1\}$.

Monoidal closedness of $\mathbf{Set}^{\mathbb{F}}$ is witnessed by the fact that for every $X \in |\mathbf{Set}^{\mathbb{F}}|$ the functor $- \bullet X \colon \mathbf{Set}^{\mathbb{F}} \to \mathbf{Set}^{\mathbb{F}}$ has a right adjoint given by

$$X \mathrel{-\!\!\bullet} - \colon \mathbf{Set}^{\mathbb{F}} \to \mathbf{Set}^{\mathbb{F}}, \qquad (X \mathrel{-\!\!\bullet} Y)(n) = \int_{m\in\mathbb{F}} [(X(m))^n, Y(m)] = \mathsf{Nat}(X^n, Y).$$

An element of $(X \mathrel{-\!\!\bullet} Y)(n)$, viz. a natural in $m \in |\mathbb{F}|$ family of maps $(X(m))^n \to Y(m)$, is thought of as describing the substitution of X-terms in m variables for the n variables of some fixed ambient term, resulting in a Y-term in m variables.

Syntax. The following *context extension* endofunctor $\delta \colon \mathbf{Set}^{\mathbb{F}} \to \mathbf{Set}^{\mathbb{F}}$ is used for including the lambda abstraction to the signature. On objects: $\delta X(n) = X(n+1)$ and $\delta X(h) = X(h + \mathrm{id}_1)$, on morphisms: $h \colon X \to Y$, $(\delta h)_n = (h_{n+1} \colon X(n+1) \to Y(n+1))$. Informally, the elements of $\delta X(n)$ are terms arising by binding the last variable is a term with $n+1$ free variables.

We define the value and the computation signatures as follows:

$$\Sigma_{\mathsf{v}} X = V \times \textstyle\coprod_{k\in\mathbb{N}} X^k + \delta X \qquad\qquad \Sigma_{\mathsf{c}}(X,Y) = X \times Y$$

As in the case of extended combinatory logic, Σ_{c} covers application, while Σ_{v} covers two types of values: expressions of the form $(\dots(xt_1)\dots)t_k$ which we write as $x * (t_1,\dots,t_k)$ with x being a variable, and λ-abstractions. The initial object $\mu\Sigma$ is known to model the terms of λ-calculus (with free variables) [17], although our case is slightly different in that we distinguish generic applications st from those that are representable as $x * (t_1,\dots,t_k)$. Modulo that, $(\mu\Sigma)(n)$ are the λ-terms over $0,\dots,n-1$ as free variables, under α-equivalence. Our requirement of V-pointed strength for $(V \times \coprod_{k\in\mathbb{N}\smallsetminus\{0\}} X^k + \delta X) + X \times X$ means essentially that substitution is definable by structural recursion, and the result that $\mu\Sigma$ is a monoid, means that λ-terms are closed under substitution. In terms of the presentation (27), for every $(t, s_0,\dots,s_{n-1}) \in \mu\Sigma(n) \times (\mu\Sigma(m))^n$, $\mathsf{subst}([(t,s_0,\dots,s_{n-1})]_{\approx})$ coherently computes the substitution $t[s_0/0,\dots,s_{n-1}/n-1]$.

Semantics. We define the behaviour functor D via $D(X,Y) = V \times \coprod_{k\in\mathbb{N}} Y^k + Y^X$. Let us spell out (26) and (8). The components of the first transformation are obtained by cotupling

$$\begin{tikzcd}
V \times \coprod_{k\in\mathbb{N}}(jX \times (jX \mathrel{-\!\!\bullet} Y))^k \arrow[d, "\mathsf{inl}\cdot(\mathsf{id}\times\coprod_{k\in\mathbb{N}}(\eta\cdot\mathsf{inl}\cdot\mathsf{fst})^k)"] \\
V \times \coprod_{k\in\mathbb{N}}(\Sigma^\star(jX+Y))^k + (\Sigma^\star(jX+Y))^{jX}
\end{tikzcd}$$

and

$$\begin{tikzcd}
\delta(jX \times (jX \mathrel{-\!\!\bullet} Y)) \arrow[rr, "\delta\,\mathsf{snd}"] & & \delta(jX \mathrel{-\!\!\bullet} Y) \arrow[dll, bend right=10, "\mathsf{curry}(\kappa_{X,Y})"'] \\
Y^{jX} \arrow[r, "\mathsf{inr}\cdot(\eta\cdot\mathsf{inr})^{jX}"] & V \times \coprod_{k\in\mathbb{N}}(\Sigma^\star(jX+Y))^k + (\Sigma^\star(jX+Y))^{jX} &
\end{tikzcd}$$

where $(\kappa_{jX,Y})_n \colon \mathsf{Nat}(jX^{n+1},Y) \times jX(n) \to Y(n)$ sends $\alpha \in \mathsf{Nat}(jX^{n+1},Y)$ and $t \in jX(n)$ to $\alpha_n(v_0,\dots,v_{n-1},t)$ and $v_i = X_n(i)$ (recall that $X \in V/\mathbf{C}$, i.e. is a natural transformation from V to jX). Thus, ρ^{v} encodes small-step rules:

$$\frac{}{x * (t_1,\dots,t_k)\downarrow x * (t_1,\dots,t_k)} \quad (x \in V(n), t_1,\dots,t_k \in X(n))$$

$$\frac{t[s/n] = t'}{\lambda n.\, t \xrightarrow{s} t'} \quad (t \in jX(n), s \in X(n-1))$$

The notation $\downarrow$ refers to the left summand of D, and the first rule associates this behaviour with variables and variable applications. The notation $\xrightarrow{s} t$ refers to the right summand of D, and the second rule describes the behaviour of λ-abstraction. Unlike the rule for values in **xCL** (Figure 2), this rule requires a premise, which motivates the inclusion of $jX \mathrel{-\!\!\bullet} Y$ in (26). This premise provides the information of how substitutions act on t, specifically how the substitution $t[s/n]$ is computed. The latter is, in fact, an abbreviation for $t[0/0,\dots,n-1/n-1,s/n]$, which makes it clear why X must be an object of $V/\mathbf{C}$ and not of $\mathbf{C}$. Indeed, when applying $[i/i]$ to t, only the right i is defined for $t \in X(n)$ with $X \in |\mathbf{C}|$, but the left one requires a transformation $V \to X$ reifying variables into X.

The transformation (8) instantiates as

$$\rho^{\mathsf{c}}_{X,Y} \colon (X \times ((V \times \textstyle\coprod_{k\in\mathbb{N}} Y^k + Y^X) + Y)) \times X \to \Sigma^\star(X+Y),$$

(assuming that the monad $\mathbf{T}$ is identity). Now that binding operators are involved, the intended transformation is the one that captures the following rules in a straightforward manner:

$$\frac{t \downarrow x * (t_1, \dots, t_k)}{ts \downarrow x * (t_1, \dots, t_k, t)} \qquad \frac{t \xrightarrow{s} t'}{ts \to t'} \qquad \frac{t \to t'}{ts \to t's}$$

By Remark 3.7, we obtain a strongly separated abstract HO-GSOS law. The simplest way to extend it to a law w.r.t. an ω-continuous monad $\mathbf{T}$ is to take T to be the pointwise powerset functor $\mathcal{P}_\star$, i.e. $\mathcal{P}_\star(X) = \mathcal{P} \cdot X$, and apply Proposition 3.11. The powerset monad is an ω-continuous monad on **Set**, with a standard ω-continuous distributive law over $(-) \times Y$. Since all relevant constructions are pointwise, we obtain that $\mathcal{P}_\star$ is ω-continuous, we obtain an ω-continuous distributive law (9), and also inherit the strong separation condition per Proposition 3.11: we have it for $\mathcal{P}$ and $(X, Y) \mapsto X{\times}Y$ in **Set** by Remark 3.12, and extend it to $\mathcal{P}_\star$ and Σ_{c} pointwise.

7.3 Big-Step SOS and the λ-calculus

The modification of the big-step SOS law (15) for our present setup is:

$$\xi\colon \Sigma_{\mathsf{c}}(\Sigma_{\mathsf{v}}(X \times (\Sigma^\star X \mathrel{-\!\bullet} Y)), X) \to T\Sigma^\star(X + Y).$$

The operational model (18) is computed using the following modification of (19):

$$\hat{\zeta} = \mu f.\, [\eta, f^\sharp \cdot T\nabla^\sharp \cdot \xi^\sharp \cdot T\Sigma_{\mathsf{c}}(\Sigma_{\mathsf{v}}\langle \mathsf{id}, (\mathsf{subst} \cdot (\mathsf{id} \times \mu))^\flat \rangle, \mathsf{id}) \cdot \chi \cdot \Sigma_{\mathsf{c}}(f, \mathsf{id})] \cdot \iota^{-1}.$$

The key change is the inclusion of the morphism $\langle \mathsf{id}, (\mathsf{subst} \cdot (\mathsf{id} \times \mu))^\flat \rangle\colon \mu\Sigma \to \mu\Sigma \times (\Sigma^\star \mu\Sigma \mathrel{-\!\bullet} \mu\Sigma)$, which creates the new expected part of the input for ξ – informally, given a term t from $\mu\Sigma(n)$, we render $\Sigma^\star\mu\Sigma \mathrel{-\!\bullet} \mu\Sigma$ as the space of substitution actions $t[-/0, \dots, -/n-1]$, awaiting terms from $\Sigma^\star\mu\Sigma$.

The updated translation (17) of a separated abstract HO-GSOS law to a big-step SOS law is as follows:

$$\begin{tikzcd}
\Sigma_{\mathsf{c}}(\Sigma_{\mathsf{v}}(X \times (\Sigma^\star X \mathrel{-\!\bullet} Y)), X) \arrow[r, "\Sigma_{\mathsf{c}}(\Sigma_{\mathsf{v}}(\eta \times \mathsf{id}), \eta)"] & \Sigma_{\mathsf{c}}(\Sigma_{\mathsf{v}}(\Sigma^\star X \times (\Sigma^\star X \mathrel{-\!\bullet} Y)), \Sigma^\star X) \arrow[dl, bend left=10, "\Sigma_{\mathsf{c}}(\langle \iota^{\mathsf{v}} \cdot \Sigma_{\mathsf{v}}\, \mathsf{fst}, \rho^{\mathsf{v}} \rangle, \mathsf{id})"'] \\
\Sigma_{\mathsf{c}}(\Sigma^\star X \times D(\Sigma^\star X, \Sigma^\star(\Sigma^\star X + Y)), \Sigma^\star X) \arrow[r, "T[\Sigma^\star \mathsf{inl}, [\Sigma^\star \mathsf{inl}, \eta \cdot \mathsf{inr}]^\sharp]^\sharp \cdot \rho^{\mathsf{cv}}"] & T\Sigma^\star(X + Y)
\end{tikzcd} \tag{28}$$

Here, we treat $\Sigma^\star X$ as an object of $V/\mathbf{C}$, which is justified since, by assumption, V is a coproduct summand of $\Sigma^\star X$. This allows us to invoke ρ^{v}.

Again, if ξ is obtained from ρ^{v} and ρ^{c} by (28), the above definition of $\hat{\zeta}$ via ξ reduces to a definition via ρ^{v} and ρ^{c}. The following is the analogue of Proposition 5.3.

Proposition 7.2. *Let ρ^{v} and ρ^{c}, χ be as in Section 7.1, and let χ be an ω-continuous distributive law. Let ξ be defined by* (28)*. Then $\hat{\zeta}$ satisfies the equation* (23)*.*

Our main result (Theorem 5.4) can now be reestablished (the proof relies on Proposition 7.2, but otherwise remains essentially unchanged, because it does not depend on how ρ^{v} is defined).

Theorem 7.3. *Let ρ^{v} and ρ^{c}, χ be as in Section 7.1, and let χ be an ω-continuous distributive law. Let ξ be defined by* (28)*. Then $\hat{\zeta} = \hat{\beta}$.*

Applying these results to the λ-calculus example, we obtain the equivalence of small-step semantics and the following big-step semantics:

$$\frac{}{\lambda n.\, t \Downarrow \lambda n.\, t}\ (t \in X(n)) \qquad \frac{}{x * (t_1, \dots, t_k) \Downarrow x * (t_1, \dots, t_k)}\ (x \in V(n), t_1, \dots, t_k \in X(n))$$

$$\frac{t \Downarrow x * (t_1, \dots, t_k) \qquad x * (t_1, \dots, t_k, s) \Downarrow v}{ts \Downarrow v} \quad (x \in V(n), t_1, \dots, t_k \in X(n))$$

$$\frac{t \Downarrow \lambda n.\, t' \quad t'[s/n] \Downarrow v}{ts \Downarrow v} \quad (t, s \in X(n-1), t' \in X(n))$$

8 Conclusions and Further Work

Building on recent advances in higher-order mathematical operational semantics, our present work addresses the well-known question of equivalence between small-step and big-step operational semantics. This equivalence arises in various settings, serving both as a sanity check and an essential tool for program analysis. Rather than developing syntax-driven recipes for rule transformations, our approach is rooted in semantic ideas going back to Turi and Plotkin [37]. Specifically, we represent the syntax and behaviour of programs as functors, with operational semantics rules modelled as (di-)natural transformations. This abstraction allowed us to systematically define small-step and big-step semantics and formulate a key condition, which we call *strong separation*, enabling us to prove the desired equivalence. We provided numerous examples demonstrating that our framework accommodates a wide range of features specified through operational semantics.

Our refinement of the general notion of abstract HO-GSOS is motivated by the goals outlined above. While other formats – including alternative refinements of abstract HO-GSOS – may also be viable, we believe our approach achieves a balanced trade-off between expressiveness and practical applicability. This is substantiated by the following points:

- Our translation produces big-step rules, which are arguably in accord with the common understanding of "big-step", as illustrated by case studies (for contrast, see Bernstein [6], who, motivated by the problem of congruence of program equivalence, abstracted big-step semantics in a way that departs from the established format of big-step rules and judgments).
- Our framework is grounded in several structural assumptions – monads, enrichment in ω-cpos, and strong separability – all of which are explicitly justified. For instance, enrichment is required to interpret recursive definitions, and strong separability is necessary to ensure equivalence properties.
- While more general rule formats than strongly separated HO-GSOS are conceivable, they often involve intricate constructions and non-elementary conditions (see e.g. Assumptions III.1–III.4 in [11]). Our design reflects a deliberate choice to favor conceptual simplicity and usability without compromising the applicability to natural examples.

Moving forward, we plan to extend our framework to cover other flavours of semantics, particularly stateful and quantitative, such as probabilistic semantics. With our approach, we hope to gain insights into challenging cases, such as McCarthy's amb operator [32], helping one to better understand its sophisticated behaviour. In particular, since the relevant small-step semantics is essentially quantitative (reductions are indexed by numbers to ensure fairness) this raises hopes that a carefully chosen monad could effectively capture this behaviour. Additionally, we plan to complement our current Haskell translations and examples with an implementation in Agda, accommodating not only constructions, but also formal proofs. Another interesting direction for future work is the abstract reverse translation of big-step semantics into small-step semantics, a topic already explored in the literature [3, 18].

References

[1] S. Abramsky. 1990. The lazy λ-calculus. In *Research topics in Functional Programming*. Addison Wesley, 65–117.

[2] Thorsten Altenkirch, Nils Danielsson, and Nicolai Kraus. 2017. Partiality, Revisited - The Partiality Monad as a Quotient Inductive-Inductive Type. In *Foundations of Software Science and Computation Structures, FOSSACS 2017 (LNCS, Vol. 10203)*, Javier Esparza and Andrzej Murawski (Eds.). 534–549.

[3] Guillaume Ambal, Alan Schmitt, and Sergueï Lenglet. 2020. *Automatic Transformation of a Big-Step Skeletal Semantics into Small-Step.* Ph. D. Dissertation. Inria Rennes-Bretagne Atlantique.

[4] Steve Awodey. 2010. *Category Theory* (2nd ed.). Oxford University Press, Inc., New York, NY, USA.

[5] Casper Bach Poulsen and Peter D. Mosses. 2014. Deriving Pretty-Big-Step Semantics from Small-Step Semantics. In *Proceedings of the 23rd European Symposium on Programming Languages and Systems - Volume 8410*. Springer-Verlag, Berlin, Heidelberg, 270–289. https://doi.org/10.1007/978-3-642-54833-8_15

[6] Karen L. Bernstein. 1998. A Congruence Theorem for Structured Operational Semantics of Higher-Order Languages. In *13th Annual IEEE Symposium on Logic in Computer Science, LICS'98*. IEEE Computer Society, 153–164. https://doi.org/10.1109/LICS.1998.705652

[7] Bard Bloom. 1995. Structural Operational Semantics for Weak Bisimulations. *Theor. Comput. Sci.* 146, 1&2 (1995), 25–68. https://doi.org/10.1016/0304-3975(94)00152-9

[8] Venanzio Capretta. 2005. General recursion via coinductive types. *Log. Meth. Comput. Sci.* 1, 2 (2005).

[9] James Chapman, Tarmo Uustalu, and Niccolò Veltri. 2015. Quotienting the Delay Monad by Weak Bisimilarity. In *Theoretical Aspects of Computing, ICTAC 2015 (LNCS, Vol. 9399)*. Springer, 110–125.

[10] Arthur Charguéraud. 2013. Pretty-Big-Step Semantics. In *Programming Languages and Systems*, Matthias Felleisen and Philippa Gardner (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 41–60.

[11] Ştefan Ciobâcă. 2013. From Small-Step Semantics to Big-Step Semantics, Automatically. In *Integrated Formal Methods*, Einar Broch Johnsen and Luigia Petre (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 347–361.

[12] H. B. Curry. 1930. Grundlagen der Kombinatorischen Logik. *Am. J. Math.* 52, 3 (1930), 509–536. http://www.jstor.org/stable/2370619

[13] Nils Anders Danielsson. 2012. Operational semantics using the partiality monad. In *Proceedings of the 17th ACM SIGPLAN International Conference on Functional Programming (ICFP 2012)*. Association for Computing Machinery, New York, NY, USA, 127–138. https://doi.org/10.1145/2364527.2364546

[14] U. Deliguoro and A. Piperno. 1995. Nondeterministic Extensions of Untyped λ-Calculus. *Information and Computation* 122, 2 (1995), 149–177. https://doi.org/10.1006/inco.1995.1145

[15] Mariangiola Dezani-Ciancaglini, Ugo de'Liguoro, and Adolfo Piperno. 1998. A Filter Model for Concurrent Lambda-Calculus. *SIAM J. Comput.* 27, 5 (1998), 1376–1419. https://doi.org/10.1137/S0097539794275860

[16] Marcelo P. Fiore. 2008. Second-Order and Dependently-Sorted Abstract Syntax. In *23d Annual IEEE Symposium on Logic in Computer Science (LICS 2008)*. IEEE Computer Society, 57–68. https://doi.org/10.1109/LICS.2008.38

[17] Marcelo P. Fiore, Gordon D. Plotkin, and Daniele Turi. 1999. Abstract Syntax and Variable Binding. In *14th Annual IEEE Symposium on Logic in Computer Science (LICS 1999)*. IEEE Computer Society, 193–202. https://doi.org/10.1109/LICS.1999.782615

[18] John P. Gallagher, Manuel Hermenegildo, Bishoksan Kafle, Maximiliano Klemen, Pedro López García, and José Morales. 2020. From big-step to small-step semantics and back with interpreter specialisation. In *Electronic Proceedings in Theoretical Computer Science, EPTCS*, Vol. 320. Open Publishing Association, 50–64. https://doi.org/10.4204/EPTCS.320.4

[19] Sergey Goncharov. 2021. Uniform Elgot Iteration in Foundations. In *48th International Colloquium on Automata, Languages, and Programming (ICALP 2021) (LIPIcs, Vol. 198)*, Nikhil Bansal, Emanuela Merelli, and James Worrell (Eds.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 131:1–131:16.

[20] Sergey Goncharov, Stefan Milius, Lutz Schröder, Stelios Tsampas, and Henning Urbat. 2022. Stateful Structural Operational Semantics. In *7th International Conference on Formal Structures for Computation and Deduction, FSCD'22 (LIPIcs, Vol. 228)*, Amy P. Felty (Ed.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 30:1–30:19. https://doi.org/10.4230/LIPIcs.FSCD.2022.30

[21] Sergey Goncharov, Stefan Milius, Lutz Schröder, Stelios Tsampas, and Henning Urbat. 2023. Towards a Higher-Order Mathematical Operational Semantics. *Proc. ACM Program. Lang.* 7, POPL (2023), 632–658. https://doi.org/10.1145/3571215

[22] Sergey Goncharov, Alessio Santamaria, Lutz Schröder, Stelios Tsampas, and Henning Urbat. 2024. Logical Predicates in Higher-Order Mathematical Operational Semantics. In *Foundations of Software Science and Computation Structures*, Naoki Kobayashi and James Worrell (Eds.). Springer Nature Switzerland, Cham, 47–69.

[23] Sergey Goncharov, Lutz Schröder, Christoph Rauch, and Julian Jakob. 2018. Unguarded Recursion on Coinductive Resumptions. *Log. Methods Comput. Sci.* 14, 3 (2018).

[24] Andrew D. Gordon. 1999. Bisimilarity as a Theory of Functional Programming. *Theor. Comput. Sci.* 228, 1-2 (1999), 5–47. https://doi.org/10.1016/S0304-3975(98)00353-3

[25] Ichiro Hasuo, Bart Jacobs, and Ana Sokolova. 2007. Generic Trace Semantics via Coinduction. *Logical Methods in Computer Science* 3, 4 (2007). https://doi.org/10.2168/LMCS-3(4:11)2007

[26] J. Roger Hindley and Jonathan P. Seldin. 2008. *Lambda-Calculus and Combinators: An Introduction* (2 ed.). Cambridge University Press. https://doi.org/10.1017/CBO9780511809835

[27] André Hirschowitz, Tom Hirschowitz, Ambroise Lafont, and Marco Maggesi. 2022. Variable binding and substitution for (nameless) dummies. In *25th International Conference on Foundations of Software Science and Computation Structures (FOSSACS 2022) (LNCS, Vol. 13242)*. Springer, 389–408. https://doi.org//10.1007/978-3-030-99253-8_20

[28] Anders Kock. 1972. Strong Functors and Monoidal Monads. *Archiv der Mathematik* 23, 1 (1972), 113–120.

[29] Ugo Dal Lago and Margherita Zorzi. 2012. Probabilistic operational semantics for the lambda calculus. *RAIRO - Theoretical Informatics and Applications - Informatique Théorique et Applications* 46, 3 (2012), 413–450. https://doi.org/10.1051/ita/2012012

[30] James Laird. 2016. Weighted Relational Models for Mobility. In *1st International Conference on Formal Structures for Computation and Deduction (FSCD 2016) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 52)*, Delia Kesner and Brigitte Pientka (Eds.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 24:1–24:15. https://doi.org/10.4230/LIPIcs.FSCD.2016.24

[31] S. Mac Lane. 1978. *Categories for the Working Mathematician* (2 ed.). Graduate Texts in Mathematics, Vol. 5. Springer. http://link.springer.com/10.1007/978-1-4757-4721-8

[32] John McCarthy. 1959. A Basis for a Mathematical Theory of Computation. In *Computer Programming and Formal Systems*, P. Braffort and D. Hirschberg (Eds.). Studies in Logic and the Foundations of Mathematics, Vol. 26. Elsevier, 33–70. https://doi.org/10.1016/S0049-237X(09)70099-0

[33] John C. Mitchell. 1996. *Foundations for programming languages.* MIT Press.

[34] Eugenio Moggi. 1991. Notions of Computation and Monads. *Inf. Comput.* 93, 1 (1991), 55–92. https://doi.org/10.1016/0890-5401(91)90052-4

[35] Scott Owens, Magnus O. Myreen, Ramana Kumar, and Yong Kiam Tan. 2016. Functional Big-Step Semantics. In *Proceedings of the 25th European Symposium on Programming Languages and Systems - Volume 9632*. Springer-Verlag, Berlin, Heidelberg, 589–615. https://doi.org/10.1007/978-3-662-49498-1_23

[36] Stelios Tsampas, Christian Williams, Andreas Nuyts, Dominique Devriese, and Frank Piessens. 2021. Abstract Congruence Criteria for Weak Bisimilarity. In *46th International Symposium on Mathematical Foundations of Computer Science, MFCS'21 (LIPIcs, Vol. 202)*, Filippo Bonchi and Simon J. Puglisi (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 88:1–88:23. https://doi.org/10.4230/LIPIcs.MFCS.2021.88

[37] Daniele Turi and Gordon D. Plotkin. 1997. Towards a Mathematical Operational Semantics. In *12th Annual IEEE Symposium on Logic in Computer Science (LICS 1997)*. 280–291. https://doi.org/10.1109/LICS.1997.614955

[38] Henning Urbat, Stelios Tsampas, Sergey Goncharov, Stefan Milius, and Lutz Schröder. 2023. Weak Similarity in Higher-Order Mathematical Operational Semantics. In *2023 38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. 1–13. https://doi.org/10.1109/LICS56636.2023.10175706

[39] Tarmo Uustalu. 2013. Coinductive big-step semantics for concurrency. *Electronic Proceedings in Theoretical Computer Science, EPTCS* 137 (2013), 63–78.

[40] Rob J. van Glabbeek. 2011. On cool congruence formats for weak bisimulations. *Theor. Comput. Sci.* 412, 28 (2011), 3283–3302. https://doi.org/10.1016/j.tcs.2011.02.036

A Omitted Proofs

A.1 Proof of Proposition 2.2

We have the following Σ-algebra structure on $\mu\Sigma \times B(\mu\Sigma, \mu\Sigma)$:

$$\Sigma(\mu\Sigma \times B(\mu\Sigma, \mu\Sigma)) \xrightarrow{\langle \iota \cdot \Sigma\,\mathsf{fst}, B(\mathsf{id}, \nabla^\sharp) \cdot \rho' \rangle} \mu\Sigma \times B(\mu\Sigma, \mu\Sigma).$$

This induces a universal Σ-algebra morphism $\mu\Sigma \to \mu\Sigma \times B(\mu\Sigma, \mu\Sigma)$, necessarily of the form $\langle \mathsf{id}, \gamma \rangle$ [21], where γ is the operational model for ρ', i.e. γ is the unique such morphism that the following diagram commutes:

$$\begin{tikzcd}
\Sigma(\mu\Sigma) \arrow[rr, "\iota"] \arrow[d, "\Sigma\langle \mathsf{id}, \gamma \rangle"'] & & \mu\Sigma \arrow[d, "\gamma"] \\
\Sigma(\mu\Sigma \times B(\mu\Sigma, \mu\Sigma)) \arrow[r, "\rho'"] & B(\mu\Sigma, \Sigma^\star(\mu\Sigma + \mu\Sigma)) \arrow[r, "B(\mathsf{id}, \nabla^\sharp)"] & B(\mu\Sigma, \mu\Sigma)
\end{tikzcd} \tag{29}$$

We will show that the last diagram can be transformed equivalently, so that the result coincides with (4) – this will immediately imply the claim. Let us first show that $\rho'_{\mu\Sigma,\mu\Sigma} = \rho_{\mu\Sigma,\mu\Sigma} \cdot \Sigma'(\mathsf{id}, \mathsf{fst})$ as follows:

$$\begin{aligned}
\rho' &= B(\mathsf{id}, \Sigma^\star[\mathsf{inl}, \mathsf{id}]) \cdot \rho_{\mu\Sigma,\mu\Sigma+\mu\Sigma} \cdot \Sigma'(\mathsf{id} \times B(\mathsf{id}, \mathsf{inr}), \mathsf{inl} \cdot \mathsf{fst}) \\
&= B(\mathsf{id}, \Sigma^\star[\mathsf{inl}, \mathsf{id}]) \cdot B(\mathsf{id}, \Sigma^\star\,\mathsf{inr}) \cdot B(\mathsf{id}, \Sigma^\star\nabla) \cdot \rho_{\mu\Sigma,\mu\Sigma+\mu\Sigma} \\
&\qquad \cdot \Sigma'(\mathsf{id} \times B(\mathsf{id}, \mathsf{inr}), \mathsf{inl} \cdot \mathsf{fst}) \\
&= B(\mathsf{id}, \Sigma^\star[\mathsf{inl}, \mathsf{id}]) \cdot B(\mathsf{id}, \Sigma^\star\,\mathsf{inr}) \cdot \rho_{\mu\Sigma,\mu\Sigma} \\
&\qquad \cdot \Sigma'(\mathsf{id} \times B(\mathsf{id}, \nabla), \nabla) \cdot \Sigma'(\mathsf{id} \times B(\mathsf{id}, \mathsf{inr}), \mathsf{inl} \cdot \mathsf{fst}) && \text{// naturality } \rho \\
&= \rho_{\mu\Sigma,\mu\Sigma} \cdot \Sigma'(\mathsf{id}, \mathsf{fst}).
\end{aligned}$$

Now, using the calculation

$$\begin{aligned}
B(\mathsf{id}, \nabla^\sharp) \cdot \rho' \cdot \Sigma\langle \mathsf{id}, \gamma \rangle &= B(\mathsf{id}, \nabla^\sharp) \cdot \rho_{\mu\Sigma,\mu\Sigma} \cdot \Sigma'(\mathsf{id}, \mathsf{fst}) \cdot \Sigma\langle \mathsf{id}, \gamma \rangle \\
&= B(\mathsf{id}, \nabla^\sharp) \cdot \rho_{\mu\Sigma,\mu\Sigma} \cdot \Sigma'(\mathsf{id}, \mathsf{fst}) \cdot \Sigma'(\langle \mathsf{id}, \gamma \rangle, \langle \mathsf{id}, \gamma \rangle) \\
&= B(\mathsf{id}, \nabla^\sharp) \cdot \rho_{\mu\Sigma,\mu\Sigma} \cdot \Sigma'(\langle \mathsf{id}, \gamma \rangle, \mathsf{id})
\end{aligned}$$

the diagram (29) can be transforms into (5), as desired. □

A.2 Proof of Proposition 3.2

Equation (12) gives an idea, how to define γ^{c} using (11). Let γ^{c} be the composition

$$\mu\Sigma_{\mathsf{c}} \xrightarrow{\Sigma_{\mathsf{c}}(\langle \mathsf{id}, \gamma \rangle, \mathsf{id})} \Sigma_{\mathsf{c}}(\mu\Sigma \times B(\mu\Sigma, \mu\Sigma), \mu\Sigma) \xrightarrow{\rho^{\mathsf{c}}} T\Sigma^\star(\mu\Sigma + \mu\Sigma) \xrightarrow{T\nabla^\sharp} T\mu\Sigma.$$

We then can prove (12):

$$\begin{aligned}
\gamma^{\mathsf{v}} + \gamma^{\mathsf{c}} &= [\mathsf{inl} \cdot D(\mathsf{id}, \mu) \cdot \rho^{\mathsf{v}}, \mathsf{inr} \cdot T\nabla^\sharp \cdot \rho^{\mathsf{c}} \cdot \Sigma_{\mathsf{c}}(\langle \mathsf{id}, \gamma \rangle, \mathsf{id})] \\
&= [\mathsf{inl} \cdot D(\mathsf{id}, \mu) \cdot \rho^{\mathsf{v}}, B(\mathsf{id}, \nabla^\sharp) \cdot \mathsf{inr} \cdot \rho^{\mathsf{c}} \cdot \Sigma_{\mathsf{c}}(\langle \mathsf{id}, \gamma \rangle, \mathsf{id})] \\
&= B(\mathsf{id}, \nabla^\sharp) \cdot [\mathsf{inl} \cdot D(\mathsf{id}, \Sigma^\star\,\mathsf{inl}) \cdot \rho^{\mathsf{v}}, \mathsf{inr} \cdot \rho^{\mathsf{c}} \cdot \Sigma_{\mathsf{c}}(\langle \mathsf{id}, \gamma \rangle, \mathsf{id})] \\
&= B(\mathsf{id}, \nabla^\sharp) \cdot (D(\mathsf{id}, \Sigma^\star\,\mathsf{inl}) \cdot \rho^{\mathsf{v}} + \rho^{\mathsf{c}}) \cdot \Sigma'(\langle \mathsf{id}, \gamma \rangle, \mathsf{id}) \\
&= B(\mathsf{id}, \nabla^\sharp) \cdot \rho \cdot \Sigma'(\langle \mathsf{id}, \gamma \rangle, \mathsf{id}) \\
&= \gamma \cdot \iota.
\end{aligned}$$

By applying (12) to (11), we obtain a diagram that commutes by the definition of γ^{c}. We are left to show that γ^{c} is the only morphism, for which (11) commutes. Let g be a morphism replacing γ^{c} for which (11) commutes. It then follows that the diagram

$$\begin{tikzcd}
\Sigma'(\mu\Sigma, \mu\Sigma) \arrow[rr, "\iota"] \arrow[d, "\Sigma'(\iota^{-1},\mathrm{id})"'] & & \mu\Sigma \arrow[d, "\iota^{-1}"] \\
\Sigma'(\mu\Sigma_{\mathsf{v}} + \mu\Sigma_{\mathsf{c}}, \mu\Sigma) \arrow[rr, "\Sigma'(\iota,\mathrm{id})"] \arrow[d, "\Sigma'(\langle\iota,\gamma^{\mathsf{v}}+g\rangle,\mathrm{id})"'] & & \Sigma'(\mu\Sigma, \mu\Sigma) \arrow[d, "\gamma^{\mathsf{v}}+g"] \\
\Sigma'(\mu\Sigma \times B(\mu\Sigma, \mu\Sigma), \mu\Sigma) \arrow[r, "\rho"] & B(\mu\Sigma, \Sigma^{\star}(\mu\Sigma + \mu\Sigma)) \arrow[r, "B(\mathrm{id},\nabla^{\sharp})"] & B(\mu\Sigma, \mu\Sigma)
\end{tikzcd}$$

commutes as well. Since $\Sigma'(\langle\iota, \gamma^{\mathsf{v}} + g\rangle, \mathrm{id}) \cdot \Sigma'(\iota^{-1}, \mathrm{id}) = \Sigma'(\langle\mathrm{id}, (\gamma^{\mathsf{v}} + g) \cdot \iota^{-1}\rangle, \mathrm{id})$, we conclude that $(\gamma^{\mathsf{v}} + g) \cdot \iota^{-1}$ satisfies the characteristic property (4) of γ. Therefore, $\gamma = (\gamma^{\mathsf{v}} + g) \cdot \iota^{-1}$. By combining it with (12), we obtain $\gamma^{\mathsf{v}} + g = \gamma^{\mathsf{v}} + \gamma^{\mathsf{c}}$, and therefore $g = T\nabla^{\sharp} \cdot \rho^{\mathsf{c}} \cdot \Sigma_{\mathsf{c}}(\langle\iota, \gamma^{\mathsf{v}} + \gamma^{\mathsf{c}}\rangle, \mathrm{id}) \cdot \Sigma_{\mathsf{c}}(\iota^{-1}, \mathrm{id})$, using the assumption about g. We obtain that any g, for which (11) commutes equals to an expression that does not depend on g. Hence, there is at most one such g. □

A.3 Proof of Proposition 3.11

The strong separation condition (14) follows from the assumptions, as the following diagram depicts:

$$\begin{tikzcd}
 & & \Theta(X \times D(X,Y), X \times TY, X) \arrow[dll, "\theta"'] \arrow[drr, "\theta"] \arrow[d, "\Theta(\eta,\tau,\mathrm{id})"] & & \\
\Sigma_{\mathsf{c}}(X \times D(X,Y) + X \times TY, X) \arrow[ddr, "\Sigma_{\mathsf{c}}(\eta+\tau,\mathrm{id})"] \arrow[ddd, "\cong"'] & & & & \Sigma_{\mathsf{c}}(X \times D(X,Y) + X \times TY, X) \arrow[ddl, "\Sigma_{\mathsf{c}}(\eta+\tau,\mathrm{id})"'] \arrow[dddddd, "\Sigma_{\mathsf{c}}(\eta\cdot\mathsf{fst}\ \boxplus \mathsf{snd},\mathsf{inl})"'] \\
 & & \Theta(T(X \times D(X,Y)), T(X \times Y), X) \arrow[dl, "\theta"'] \arrow[dr, "\theta"] \arrow[ddddd] & & \\
 & \Sigma_{\mathsf{c}}(T(X \times D(X,Y)) + T(X \times Y), X) \arrow[ddd, "\Sigma_{\mathsf{c}}(\mathrm{id}\boxplus\mathrm{id},\mathrm{id})"] & & \Sigma_{\mathsf{c}}(T(X \times D(X,Y)) + T(X \times Y), X) \arrow[ddd, "\Sigma_{\mathsf{c}}(\mathrm{id}\boxplus\mathrm{id},\mathrm{id})"'] & \\
\Sigma_{\mathsf{c}}(X \times B(X,Y), X) \arrow[d, "\Sigma_{\mathsf{c}}(\mathrm{id}\times(\mathrm{id}\boxplus\mathrm{id}),\mathrm{id})"] & & & & \\
\Sigma_{\mathsf{c}}(X \times T(D(X,Y) + Y), X) \arrow[dd, "\Sigma_{\mathsf{c}}(\tau,\mathrm{id})"'] & & & & \\
 & \Sigma_{\mathsf{c}}(T(X \times D(X,Y) + X \times Y), X) \arrow[dl, "\cong"'] \arrow[dd, "\chi"] & & \Sigma_{\mathsf{c}}(T(X \times D(X,Y) + X \times Y), X) \arrow[dr, "\Sigma_{\mathsf{c}}(T(\mathsf{fst}+\mathsf{snd}),\mathsf{inl})"] \arrow[dd, "\chi"'] & \\
\Sigma_{\mathsf{c}}(T(X \times (D(X,Y) + Y)), X) \arrow[ddr, bend right, "\chi"'] & & T\Theta(X \times D(X,Y), X \times Y, X) \arrow[dl, "T\theta"'] \arrow[dr, "T\theta"] & & \Sigma_{\mathsf{c}}(T(X + Y), X + Y) \arrow[ddl, bend left, "\chi"] \\
 & T\Sigma_{\mathsf{c}}(X \times D(X,Y) + X \times Y, X) \arrow[d, "\cong"] & & T\Sigma_{\mathsf{c}}(X \times D(X,Y) + X \times Y, X) \arrow[d, "T\Sigma_{\mathsf{c}}(\mathsf{fst}+\mathsf{snd},\mathsf{inl})"'] & \\
 & T\Sigma_{\mathsf{c}}(X \times (D(X,Y) + Y), X) \arrow[dr, "T\rho^{\mathsf{c}}"'] & & T\Sigma_{\mathsf{c}}(X + Y, X + Y) \arrow[dl, "T\iota^{\mathsf{c}}\cdot T\Sigma_{\mathsf{c}}(\eta,\eta)"] & \\
 & & T\Sigma^{\star}(X + Y) & &
\end{tikzcd}$$

□

A.4 Proof of Lemma 4.2

For the sake of succinctness, we abbreviate $\Sigma^\star(\mu\Sigma)$ as $\mu^2\Sigma$. The claim follows from commutativity of the diagram:

$$\begin{tikzcd}
\Sigma_{\mathsf{c}}(\mu\Sigma_{\mathsf{v}}, \mu\Sigma) \arrow[rr, "\Sigma_{\mathsf{c}}(\langle\iota^{\mathsf{v}}, \rho^{\mathsf{v}}\rangle, \mathsf{id})"] \arrow[d, "\Sigma_{\mathsf{c}}(\Sigma_{\mathsf{v}}\eta, \eta)"'] & & \Sigma_{\mathsf{c}}(\mu\Sigma \times D(\mu\Sigma, \mu^2\Sigma), \mu\Sigma) \arrow[dddddd, "\Sigma_{\mathsf{c}}(\mathsf{id}\times D(\mathsf{id},\mu), \mathsf{id})"] \\
\Sigma_{\mathsf{c}}(\Sigma_{\mathsf{v}}(\mu^2\Sigma), \mu^2\Sigma) \arrow[dr, bend right, "\Sigma_{\mathsf{c}}(\langle\iota^{\mathsf{v}}, \Sigma_{\mathsf{v}}\mu\rangle, \mathsf{id})"] \arrow[ddd, "\Sigma_{\mathsf{c}}(\langle\iota^{\mathsf{v}}, \rho^{\mathsf{v}}\rangle, \mathsf{id})"'] & & \\
 & \Sigma_{\mathsf{c}}(\mu^2\Sigma \times \Sigma_{\mathsf{v}}\mu\Sigma, \mu^2\Sigma) \arrow[d, "\Sigma_{\mathsf{c}}(\mathsf{id}\times\rho^{\mathsf{v}}, \mathsf{id})"'] & \\
 & \Sigma_{\mathsf{c}}(\mu^2\Sigma \times D(\mu\Sigma, \Sigma^\star(\mu^2\Sigma), \mu^2\Sigma) \arrow[uuur, bend right, "\Sigma_{\mathsf{c}}(\mu\times\mathsf{id}, \mu)"] \arrow[dd, "\Sigma_{\mathsf{c}}(\mathsf{id}\times D(\mu, \mathsf{id}), \mathsf{id})"] & \\
\Sigma_{\mathsf{c}}(\mu^2\Sigma \times D(\mu^2\Sigma, \Sigma^\star(\mu^2\Sigma)), \mu^2\Sigma) \arrow[dr, bend right, "\Sigma_{\mathsf{c}}(\mathsf{id}\times D(\mathsf{id}, \Sigma^\star\mu), \mathsf{id})"] \arrow[dd, "\Sigma_{\mathsf{c}}(\mathsf{id}\times D(\mathsf{id}, \mu), \mathsf{id})"'] & & \\
 & \Sigma_{\mathsf{c}}(\mu^2\Sigma \times D(\mu^2\Sigma, \mu^2\Sigma), \mu^2\Sigma) \arrow[dd, "\Sigma_{\mathsf{c}}(\mathsf{id}\times D(\mathsf{id}, \mu), \mathsf{id})"] & \\
\Sigma_{\mathsf{c}}(\mu^2\Sigma \times D(\mu^2\Sigma, \mu^2\Sigma), \mu^2\Sigma) \arrow[dr, bend right, "\Sigma_{\mathsf{c}}(\mathsf{id}\times D(\mathsf{id}, \mu), \mathsf{id})"] \arrow[dd, "\rho^{\mathsf{cv}}"'] & & \Sigma_{\mathsf{c}}(\mu\Sigma \times D(\mu\Sigma, \mu\Sigma), \mu\Sigma) \arrow[dd, "\rho^{\mathsf{cv}}"] \\
 & \Sigma_{\mathsf{c}}(\mu^2\Sigma \times D(\mu^2\Sigma, \mu\Sigma), \mu^2\Sigma) \arrow[dd, "\rho^{\mathsf{cv}}"] & \\
T\Sigma^\star(\mu^2\Sigma + \mu^2\Sigma) \arrow[dr, bend right, "T\Sigma^\star(\mathsf{id}+\mu)"] \arrow[dd, "T\nabla^\sharp"'] & & T\Sigma^\star(\mu\Sigma + \mu\Sigma) \arrow[dd, "T\nabla^\sharp"] \\
 & T\Sigma^\star(\mu^2\Sigma + \mu\Sigma) \arrow[ur, bend right, "T\Sigma^\star(\mu+\mathsf{id})"] & \\
T(\mu^2\Sigma) \arrow[rr, "T\mu"] & & T(\mu\Sigma)
\end{tikzcd}$$

The top cell commutes, because ι^{v} satisfies the equation $\iota^{\mathsf{v}} \cdot \Sigma_{\mathsf{v}}\mu = \mu \cdot \iota^{\mathsf{v}}$, by naturality of Σ_{v} and because $\mu \cdot \eta = \mathsf{id}$. The bottom cell commutes because $\mu \cdot \nabla^\sharp = \mu \cdot \mu \cdot \Sigma^\star\nabla = \mu \cdot \Sigma^\star\mu \cdot \Sigma^\star\nabla = \mu \cdot \Sigma^\star\nabla \cdot \Sigma^\star(\mu + \mu) = \nabla^\sharp \cdot \Sigma^\star(\mu + \mu)$. The remaining three cells on the left-hand side of the diagram commute by dinaturality of ρ^{v}, by definition of Kleisli lifting for $\Sigma^\star$ and by naturality of ρ^{cv} correspondingly. The remaining large cell on the right-hand side commutes by naturality and dinaturality of ρ^{cv}. □

A.5 Proof of Lemma 5.1

As a preliminary step, we show that the following diagram commutes:

$$\begin{tikzcd}
\mu\Sigma_{\mathsf{c}} \arrow[r, "\beta"] \arrow[d, "\Sigma_{\mathsf{c}}(\hat{\gamma}^{\mathsf{c}},\mathsf{id})"'] & T(\mu\Sigma_{\mathsf{v}}) \\
\Sigma_{\mathsf{c}}(T\mu\Sigma, \mu\Sigma) \arrow[r, "\chi"] & T(\mu\Sigma_{\mathsf{c}}) \arrow[u, "\beta^{\sharp}"']
\end{tikzcd} \tag{30}$$

To that end we use the fact that, by Assumption 3.4, $\mu\Sigma_{\mathsf{c}}$ is a coproduct of $\Sigma_{\mathsf{c}}(\mu\Sigma_{\mathsf{v}}, \mu\Sigma)$ and $\Theta(\mu\Sigma_{\mathsf{v}}, \mu\Sigma_{\mathsf{c}}, \mu\Sigma)$. The equation encoded by (30) thus falls into two equations:

$$\beta^{\sharp} \cdot \chi \cdot \Sigma_{\mathsf{c}}(\hat{\gamma}^{\mathsf{c}}, \mathsf{id}) \cdot \Sigma_{\mathsf{c}}(\iota^{\mathsf{v}}, \mathsf{id}) = \beta \cdot \Sigma_{\mathsf{c}}(\iota^{\mathsf{v}}, \mathsf{id}),$$
$$\beta^{\sharp} \cdot \chi \cdot \Sigma_{\mathsf{c}}(\hat{\gamma}^{\mathsf{c}}, \mathsf{id}) \cdot \Sigma_{\mathsf{c}}(\iota, \mathsf{id}) \cdot \theta = \beta \cdot \Sigma_{\mathsf{c}}(\iota, \mathsf{id}) \cdot \theta.$$

The first one is easy to obtain by unfolding definitions and using the fact that χ is a distributive law:

$$\beta^{\sharp} \cdot \chi \cdot \Sigma_{\mathsf{c}}(\hat{\gamma}^{\mathsf{c}}, \mathsf{id}) \cdot \Sigma_{\mathsf{c}}(\iota^{\mathsf{v}}, \mathsf{id}) \ = \beta^{\sharp} \cdot \chi \cdot \Sigma_{\mathsf{c}}(\eta \cdot \iota^{\mathsf{v}}, \mathsf{id}) = \beta \cdot \Sigma_{\mathsf{c}}(\iota^{\mathsf{v}}, \mathsf{id}).$$

Let us proceed with the second equation. Using the strong separation condition (14) we obtain that the following diagram commutes:

$$\begin{tikzcd}
\Theta(\mu\Sigma_{\mathsf{v}}, \mu\Sigma_{\mathsf{c}}, \mu\Sigma) \arrow[rrr, "\theta"] \arrow[drr, "\Theta(\eta\cdot\iota^{\mathsf{v}},\gamma^{\mathsf{c}},\mathsf{id})"] \arrow[ddr, "\Theta(\langle\iota^{\mathsf{v}},\gamma^{\mathsf{v}}\rangle,\langle\iota^{\mathsf{c}},\gamma^{\mathsf{c}}\rangle,\mathsf{id})"] \arrow[ddd, "\theta"'] & & & \Sigma_{\mathsf{c}}(\mu\Sigma_{\mathsf{v}} + \mu\Sigma_{\mathsf{c}}, \mu\Sigma) \arrow[ddd, "\Sigma_{\mathsf{c}}(\eta\cdot\iota^{\mathsf{v}}+\gamma^{\mathsf{c}},\mathsf{id})"] \\
& & \Theta(T\mu\Sigma, T\mu\Sigma, \mu\Sigma) \arrow[ddr, "\theta"] & \\
& \Theta(\mu\Sigma \times D(\mu\Sigma, \mu\Sigma), \mu\Sigma \times T\mu\Sigma, \mu\Sigma) \arrow[ur, "\Theta(\eta\cdot\mathsf{fst},\mathsf{snd},\mathsf{id})"] \arrow[ddr, "\theta"] \arrow[dddd, "\theta"'] & & \\
\Sigma_{\mathsf{c}}(\mu\Sigma_{\mathsf{v}} + \mu\Sigma_{\mathsf{c}}, \mu\Sigma) \arrow[dddr, "\Sigma_{\mathsf{c}}(\langle\iota^{\mathsf{v}},\gamma^{\mathsf{v}}\rangle+\langle\iota^{\mathsf{c}},\gamma^{\mathsf{c}}\rangle,\mathsf{id})"'] \arrow[dddd, "\Sigma_{\mathsf{c}}(\langle\iota,\gamma^{\mathsf{v}}+\gamma^{\mathsf{c}}\rangle,\mathsf{id})"'] & & & \Sigma_{\mathsf{c}}(T\mu\Sigma + T\mu\Sigma, \mu\Sigma) \arrow[dd, "\Sigma_{\mathsf{c}}(\mathsf{id}\boxplus\mathsf{id},\mathsf{inl})"] \\
& & \Sigma_{\mathsf{c}}(\mu\Sigma \times D(\mu\Sigma, \mu\Sigma) + \mu\Sigma \times T\mu\Sigma, \mu\Sigma) \arrow[dr, "\Sigma_{\mathsf{c}}(\eta\cdot\mathsf{fst}\ \boxplus\mathsf{snd},\mathsf{inl})"'] & \\
& & & \Sigma_{\mathsf{c}}(T(\mu\Sigma + \mu\Sigma), \mu\Sigma + \mu\Sigma) \arrow[d, "\chi"] \\
& \Sigma_{\mathsf{c}}(\mu\Sigma \times D(\mu\Sigma, \mu\Sigma) + \mu\Sigma \times T\mu\Sigma, \mu\Sigma) \arrow[dl, "\Sigma_{\mathsf{c}}([\mathsf{id}\times\mathsf{inl},\mathsf{id}\times\mathsf{inr}],\mathsf{id})"] & & T\Sigma_{\mathsf{c}}(\mu\Sigma + \mu\Sigma, \mu\Sigma + \mu\Sigma) \arrow[d, "T(\iota^{\mathsf{c}}\cdot\Sigma_{\mathsf{c}}(\eta,\eta))"] \\
\Sigma_{\mathsf{c}}(\mu\Sigma \times B(\mu\Sigma, \mu\Sigma), \mu\Sigma) \arrow[rrr, "\rho^{\mathsf{c}}"] & & & T\Sigma^{\star}(\mu\Sigma + \mu\Sigma)
\end{tikzcd}$$

Let us rewrite $\beta^{\sharp} \cdot \chi \cdot \Sigma_{\mathsf{c}}(\hat{\gamma}^{\mathsf{c}}, \mathsf{id}) \cdot \Sigma_{\mathsf{c}}(\iota, \mathsf{id}) \cdot \theta$ equivalently as follows:

$$\begin{aligned}
\beta^{\sharp} \cdot \chi\cdot\ & \Sigma_{\mathsf{c}}(\hat{\gamma}^{\mathsf{c}}, \mathsf{id}) \cdot \Sigma_{\mathsf{c}}(\iota, \mathsf{id}) \cdot \theta \\
&= \beta^{\sharp} \cdot \chi \cdot \Sigma_{\mathsf{c}}([\eta \cdot \iota^{\mathsf{v}}, \gamma^{\mathsf{c}}], \mathsf{id}) \cdot \theta \\
&= \beta^{\sharp} \cdot \chi \cdot \Sigma_{\mathsf{c}}(T\nabla, \nabla) \cdot \Sigma_{\mathsf{c}}(\eta \cdot \iota^{\mathsf{v}} \boxplus \gamma^{\mathsf{c}}, \mathsf{inl}) \cdot \theta \\
&= \beta^{\sharp} \cdot T\Sigma_{\mathsf{c}}(\nabla, \nabla) \cdot \chi \cdot \Sigma_{\mathsf{c}}(\eta \cdot \iota^{\mathsf{v}} \boxplus \gamma^{\mathsf{c}}, \mathsf{inl}) \cdot \theta
\end{aligned}$$

$$
\begin{aligned}
&= [\eta,\beta]^\sharp \cdot T\Sigma\nabla \cdot T\,\mathsf{inr} \cdot \chi \cdot \Sigma_{\mathsf{c}}(\eta \cdot \iota^{\mathsf{v}} \boxplus \gamma^{\mathsf{c}}, \mathsf{inl}) \cdot \theta \\
&= [\eta,\beta]^\sharp \cdot T\iota^{-1} \cdot T\nabla^\sharp \cdot T(\iota \cdot \Sigma\eta \cdot \mathsf{inr}) \cdot \chi \cdot \Sigma_{\mathsf{c}}(\eta \cdot \iota^{\mathsf{v}} \boxplus \gamma^{\mathsf{c}}, \mathsf{inl}) \cdot \theta \\
&= [\eta,\beta]^\sharp \cdot T\iota^{-1} \cdot T\nabla^\sharp \cdot T(\iota^{\mathsf{c}} \cdot \Sigma_{\mathsf{c}}(\eta,\eta)) \cdot \chi \cdot \Sigma_{\mathsf{c}}(\eta \cdot \iota^{\mathsf{v}} \boxplus \gamma^{\mathsf{c}}, \mathsf{inl}) \cdot \theta \\
&= [\eta,\beta]^\sharp \cdot T\iota^{-1} \cdot T\nabla^\sharp
\end{aligned}
$$

Now, we can apply the above diagram and obtain the goal as follows:

$$
\begin{aligned}
&= [\eta,\beta]^\sharp \cdot T\iota^{-1} \cdot T\nabla^\sharp \cdot \rho^{\mathsf{c}} \cdot \Sigma_{\mathsf{c}}(\langle \iota, \gamma^{\mathsf{v}} + \gamma^{\mathsf{c}} \rangle, \mathsf{id}) \cdot \theta \\
&= [\eta,\beta]^\sharp \cdot T\iota^{-1} \cdot \gamma^{\mathsf{c}} \cdot \Sigma_{\mathsf{c}}(\iota, \mathsf{id}) \cdot \theta \\
&= \beta \cdot \Sigma_{\mathsf{c}}(\iota, \mathsf{id}) \cdot \theta.
\end{aligned}
$$

This completes the proof of commutativity of (30). Let us proceed with the main goal. For every $n \in \mathbb{N}$ let $\hat\beta_n \colon \mu\Sigma \to T(\mu\Sigma_{\mathsf{v}})$ be recursively defined as follows: $\hat\beta_0 = [\eta,\bot] \cdot \iota^{-1}$, $\hat\beta_{n+1} = \hat\beta_n^\sharp \cdot \hat\gamma^{\mathsf{c}}$. Observe that $\hat\beta = \bigsqcup_n \hat\beta_n$. Indeed, by definition (Kleene's fixpoint theorem), $\beta = \bigsqcup_n \beta_n$ where $\beta_0 = \bot$, $\beta_{n+1} = [\eta,\beta_n]^\sharp \cdot T\iota^{-1} \cdot \gamma^{\mathsf{c}}$, and then, by induction, $\hat\beta_1 = \hat\beta_0^\sharp \cdot \gamma^{\mathsf{c}} = [\eta,\beta_0]^\sharp \cdot T\iota^{-1} \cdot \gamma^{\mathsf{c}} = \beta_1$ and $\hat\beta_{n+1} = \hat\beta_n^\sharp \cdot \gamma^{\mathsf{c}} = (\beta_n)^\sharp \cdot \gamma^{\mathsf{c}} = [\eta,\beta_n]^\sharp \cdot T\iota^{-1} \cdot \gamma^{\mathsf{c}} = \beta_{n+1}$. It thus suffices to prove

$$
\hat\beta^\sharp \cdot T\nabla^\sharp \cdot (\rho^{\mathsf{cv}})^\sharp \cdot \chi \cdot \Sigma_{\mathsf{c}}(T\langle \iota^{\mathsf{v}}, \gamma^{\mathsf{v}} \rangle \cdot \hat\beta_n, \mathsf{id}) \sqsubseteq \beta
$$

for every $n \in \mathbb{N}$. We proceed by induction on n.

Induction base: $n = 0$. The goal is obtained as follows:

$$
\begin{aligned}
&\hat\beta^\sharp \cdot T\nabla^\sharp \cdot (\rho^{\mathsf{cv}})^\sharp \cdot \chi \cdot \Sigma_{\mathsf{c}}(T\langle \iota^{\mathsf{v}}, \gamma^{\mathsf{v}} \rangle \cdot [\eta,\bot] \cdot \iota^{-1}, \mathsf{id}) \\
&\quad = \hat\beta^\sharp \cdot T\nabla^\sharp \cdot (\rho^{\mathsf{cv}})^\sharp \cdot \chi \cdot \Sigma_{\mathsf{c}}([\eta \cdot \langle \iota^{\mathsf{v}}, \gamma^{\mathsf{v}} \rangle, \bot] \cdot \iota^{-1}, \mathsf{id}) \\
&\quad = \hat\beta^\sharp \cdot T\nabla^\sharp \cdot \rho^{\mathsf{cv}} \cdot \Sigma_{\mathsf{c}}([\langle \iota^{\mathsf{v}}, \gamma^{\mathsf{v}} \rangle, \bot] \cdot \iota^{-1}, \mathsf{id}) \\
&\quad = \hat\beta^\sharp \cdot T\nabla^\sharp \cdot \rho^{\mathsf{c}} \cdot \Sigma_{\mathsf{c}}([\langle \iota^{\mathsf{v}}, \mathsf{inl} \cdot \gamma^{\mathsf{v}} \rangle, \bot] \cdot \iota^{-1}, \mathsf{id}) \\
&\quad \sqsubseteq \hat\beta^\sharp \cdot T\nabla^\sharp \cdot \rho^{\mathsf{c}} \cdot \Sigma_{\mathsf{c}}([\langle \iota^{\mathsf{v}}, \mathsf{inl} \cdot \gamma^{\mathsf{v}} \rangle, \langle \iota^{\mathsf{c}}, \mathsf{inr} \cdot \gamma^{\mathsf{c}} \rangle] \cdot \iota^{-1}, \mathsf{id}) \\
&\quad = \hat\beta^\sharp \cdot T\nabla^\sharp \cdot \rho^{\mathsf{c}} \cdot \Sigma_{\mathsf{c}}(\langle \mathsf{id}, [\mathsf{inl} \cdot \gamma^{\mathsf{v}}, \mathsf{inr} \cdot \gamma^{\mathsf{c}}] \cdot \iota^{-1} \rangle, \mathsf{id}) \\
&\quad = [\eta,\beta]^\sharp \cdot T\iota^{-1} \cdot \gamma^{\mathsf{c}} \\
&\quad = \beta.
\end{aligned}
$$

Induction step: $n > 0$. We have

$$
\begin{aligned}
&\hat\beta^\sharp \cdot T\nabla^\sharp \cdot (\rho^{\mathsf{cv}})^\sharp \cdot \chi \cdot \Sigma_{\mathsf{c}}(T\langle \iota^{\mathsf{v}}, \gamma^{\mathsf{v}} \rangle \cdot \hat\beta_n, \mathsf{id}) \\
&\quad = \hat\beta^\sharp \cdot T\nabla^\sharp \cdot (\rho^{\mathsf{cv}})^\sharp \cdot \chi \cdot \Sigma_{\mathsf{c}}(T\langle \iota^{\mathsf{v}}, \gamma^{\mathsf{v}} \rangle \cdot \hat\beta_{n-1}^\sharp \cdot \hat\gamma^{\mathsf{c}}, \mathsf{id}) \\
&\quad = \hat\beta^\sharp \cdot T\nabla^\sharp \cdot (\rho^{\mathsf{cv}})^\sharp \cdot \chi^\sharp \cdot \chi \cdot \Sigma_{\mathsf{c}}(TT\langle \iota^{\mathsf{v}}, \gamma^{\mathsf{v}} \rangle \cdot T\hat\beta_{n-1} \cdot \hat\gamma^{\mathsf{c}}, \mathsf{id}) \\
&\quad = \hat\beta^\sharp \cdot T\nabla^\sharp \cdot (\rho^{\mathsf{cv}})^\sharp \cdot \chi^\sharp \cdot T\Sigma_{\mathsf{c}}(T\langle \iota^{\mathsf{v}}, \gamma^{\mathsf{v}} \rangle \cdot \hat\beta_{n-1}, \mathsf{id}) \cdot \chi \cdot \Sigma_{\mathsf{c}}(\hat\gamma^{\mathsf{c}}, \mathsf{id}).
\end{aligned}
$$

By induction hypothesis, the latter is smaller or equal $\beta^\sharp \cdot \chi \cdot \Sigma_{\mathsf{c}}(\hat\gamma^{\mathsf{c}}, \mathsf{id})$, which is β by (30). □

A.6 Proof of Lemma 5.2

The diagram (11) identifies γ^{c} as the unique fixpoint of $f \mapsto T\nabla^\sharp \cdot \rho^{\mathsf{c}} \cdot \Sigma_{\mathsf{c}}(\langle \mathsf{id}, (\gamma^{\mathsf{v}} + f) \cdot \iota^{-1} \rangle, \mathsf{id})$, hence, also as the least one. Thus, $\gamma^{\mathsf{c}} = \bigsqcup_n \gamma^{\mathsf{c}}_n$ where $\gamma^{\mathsf{c}}_0 = \bot$ and $\gamma^{\mathsf{c}}_{n+1} = T\nabla^\sharp \cdot \rho^{\mathsf{c}} \cdot \Sigma_{\mathsf{c}}(\langle \mathsf{id}, (\gamma^{\mathsf{v}} + \gamma^{\mathsf{c}}_n) \cdot \iota^{-1} \rangle, \mathsf{id})$, and it suffices to show

$$
\hat\zeta^\sharp \cdot \gamma^{\mathsf{c}}_n \sqsubseteq \zeta \tag{31}
$$

for all n. The induction base is obvious. For the induction step, assume (31) and show $\hat{\zeta}^\sharp \cdot \gamma^{\mathsf{c}}_{n+1} \sqsubseteq \zeta$. By Assumption 3.4, $\mu\Sigma_{\mathsf{c}}$ is a coproduct of $\Sigma_{\mathsf{c}}(\mu\Sigma_{\mathsf{v}}, \mu\Sigma)$ and $\Theta(\mu\Sigma_{\mathsf{v}}, \mu\Sigma_{\mathsf{c}}, \mu\Sigma)$. We thus reduce to two inequations:

$$\begin{gathered}
\hat{\zeta}^\sharp \cdot \gamma^{\mathsf{c}}_{n+1} \cdot \Sigma_{\mathsf{c}}(\iota^{\mathsf{v}}, \mathsf{id}) \sqsubseteq \zeta \cdot \Sigma_{\mathsf{c}}(\iota^{\mathsf{v}}, \mathsf{id}),\\
\hat{\zeta}^\sharp \cdot \gamma^{\mathsf{c}}_{n+1} \cdot \Sigma_{\mathsf{c}}(\iota, \mathsf{id}) \cdot \theta \sqsubseteq \zeta \cdot \Sigma_{\mathsf{c}}(\iota, \mathsf{id}) \cdot \theta.
\end{gathered}$$

The first inequation is easy to obtain by unfolding definitions:

$$\begin{aligned}
&\hat{\zeta}^\sharp \cdot \gamma^{\mathsf{c}}_{n+1} \cdot \Sigma_{\mathsf{c}}(\iota^{\mathsf{v}}, \mathsf{id})\\
&\qquad = \hat{\zeta}^\sharp \cdot T\nabla^\sharp \cdot \rho^{\mathsf{c}} \cdot \Sigma_{\mathsf{c}}(\langle \iota, \gamma^{\mathsf{v}} + \gamma^{\mathsf{c}}_n \rangle, \mathsf{id}) \cdot \Sigma_{\mathsf{c}}(\mathsf{inl}, \mathsf{id})\\
&\qquad = \hat{\zeta}^\sharp \cdot T\nabla^\sharp \cdot \rho^{\mathsf{c}} \cdot \Sigma_{\mathsf{c}}(\langle \iota^{\mathsf{v}}, \mathsf{inl} \cdot \gamma^{\mathsf{v}} \rangle, \mathsf{id})\\
&\qquad = \hat{\zeta}^\sharp \cdot T\nabla^\sharp \cdot \rho^{\mathsf{cv}} \cdot \Sigma_{\mathsf{c}}(\langle \iota^{\mathsf{v}}, \gamma^{\mathsf{v}} \rangle, \mathsf{id})\\
&\qquad = \hat{\zeta}^\sharp \cdot T\nabla^\sharp \cdot (\rho^{\mathsf{cv}})^\sharp \cdot \chi \cdot \Sigma_{\mathsf{c}}(T\langle \iota^{\mathsf{v}}, \gamma^{\mathsf{v}} \rangle \cdot \eta, \mathsf{id})\\
&\qquad = \hat{\zeta}^\sharp \cdot T\nabla^\sharp \cdot (\rho^{\mathsf{cv}})^\sharp \cdot \chi \cdot \Sigma_{\mathsf{c}}(T\langle \iota^{\mathsf{v}}, \gamma^{\mathsf{v}} \rangle \cdot \hat{\zeta}, \mathsf{id}) \cdot \Sigma_{\mathsf{c}}(\iota^{\mathsf{v}}, \mathsf{id})\\
&\qquad = \zeta \cdot \Sigma_{\mathsf{c}}(\iota^{\mathsf{v}}, \mathsf{id}). && /\!/ \text{ Proposition 5.3}
\end{aligned}$$

For the second inequation, we need the following auxiliary property:

$$\Sigma_{\mathsf{c}}(\eta \cdot \mathsf{fst} \boxplus \mathsf{snd}, \mathsf{inl}) \cdot \theta \cdot \Theta(\langle \iota^{\mathsf{v}}, \gamma^{\mathsf{v}} \rangle, \langle \iota^{\mathsf{c}}, \gamma^{\mathsf{c}}_n \rangle, \mathsf{id}) = \Sigma_{\mathsf{c}}(\eta \cdot \iota^{\mathsf{v}} \boxplus \gamma^{\mathsf{c}}_n, \mathsf{inl}) \cdot \theta, \tag{32}$$

which entails the goal as follows:

$$\begin{aligned}
&\hat{\zeta}^\sharp \cdot \gamma^{\mathsf{c}}_{n+1} \cdot \Sigma_{\mathsf{c}}(\iota, \mathsf{id}) \cdot \theta\\
&\quad = \hat{\zeta}^\sharp \cdot T\nabla^\sharp \cdot \rho^{\mathsf{c}} \cdot \Sigma_{\mathsf{c}}(\langle \iota, \gamma^{\mathsf{v}} + \gamma^{\mathsf{c}}_n \rangle, \mathsf{id}) \cdot \theta\\
&\quad = \hat{\zeta}^\sharp \cdot T\nabla^\sharp \cdot \rho^{\mathsf{c}} \cdot \Sigma_{\mathsf{c}}([\mathsf{id} \times \mathsf{inl}, \mathsf{id} \times \mathsf{inr}], \mathsf{id}) \cdot \theta \cdot \Theta(\langle \iota^{\mathsf{v}}, \gamma^{\mathsf{v}} \rangle, \langle \iota^{\mathsf{c}}, \gamma^{\mathsf{c}}_n \rangle, \mathsf{id})\\
&\quad = \hat{\zeta}^\sharp \cdot T\nabla^\sharp \cdot T(\iota^{\mathsf{c}} \cdot \Sigma_{\mathsf{c}}(\eta, \eta)) \cdot \chi \cdot\\
&\qquad\qquad \Sigma_{\mathsf{c}}(\eta \cdot \mathsf{fst} \boxplus \mathsf{snd}, \mathsf{inl}) \cdot \theta \cdot \Theta(\langle \iota^{\mathsf{v}}, \gamma^{\mathsf{v}} \rangle, \langle \iota^{\mathsf{c}}, \gamma^{\mathsf{c}}_n \rangle, \mathsf{id}) && /\!/ \text{ (14)}\\
&\quad = \hat{\zeta}^\sharp \cdot T\nabla^\sharp \cdot T\iota^{\mathsf{c}} \cdot T\Sigma_{\mathsf{c}}(\eta, \eta) \cdot \chi \cdot \Sigma_{\mathsf{c}}(\eta \cdot \iota^{\mathsf{v}} \boxplus \gamma^{\mathsf{c}}_n, \mathsf{inl}) \cdot \theta && /\!/ \text{ (32)}\\
&\quad = \hat{\zeta}^\sharp \cdot T\Sigma_{\mathsf{c}}(\nabla^\sharp, \nabla^\sharp) \cdot T\Sigma_{\mathsf{c}}(\eta, \eta) \cdot \chi \cdot \Sigma_{\mathsf{c}}(\eta \cdot \iota^{\mathsf{v}} \boxplus \gamma^{\mathsf{c}}_n, \mathsf{inl}) \cdot \theta\\
&\quad = \hat{\zeta}^\sharp \cdot T\Sigma_{\mathsf{c}}(\nabla, \nabla) \cdot \chi \cdot \Sigma_{\mathsf{c}}(\eta \cdot \iota^{\mathsf{v}} \boxplus \gamma^{\mathsf{c}}_n, \mathsf{inl}) \cdot \theta\\
&\quad = \hat{\zeta}^\sharp \cdot \chi \cdot \Sigma_{\mathsf{c}}(T\nabla, \nabla) \cdot \Sigma_{\mathsf{c}}(\eta \cdot \iota^{\mathsf{v}} \boxplus \gamma^{\mathsf{c}}_n, \mathsf{inl}) \cdot \theta\\
&\quad = \hat{\zeta}^\sharp \cdot \chi \cdot \Sigma_{\mathsf{c}}([\eta \cdot \iota^{\mathsf{v}}, \gamma^{\mathsf{c}}_n], \mathsf{id}) \cdot \theta\\
&\quad = \hat{\zeta}^\sharp \cdot T\nabla^\sharp \cdot (\rho^{\mathsf{cv}})^\sharp \cdot \chi^\sharp \cdot T\Sigma_{\mathsf{c}}(T\langle \iota^{\mathsf{v}}, \gamma^{\mathsf{v}} \rangle \cdot \hat{\zeta}, \mathsf{id}) \cdot \chi \cdot \Sigma_{\mathsf{c}}([\eta \cdot \iota^{\mathsf{v}}, \gamma^{\mathsf{c}}_n], \mathsf{id}) \cdot \theta\\
&\quad = \hat{\zeta}^\sharp \cdot T\nabla^\sharp \cdot (\rho^{\mathsf{cv}})^\sharp \cdot \chi^\sharp \cdot \chi \cdot \Sigma_{\mathsf{c}}(TT\langle \iota^{\mathsf{v}}, \gamma^{\mathsf{v}} \rangle \cdot T\hat{\zeta}, \mathsf{id}) \cdot \Sigma_{\mathsf{c}}([\eta \cdot \iota^{\mathsf{v}}, \gamma^{\mathsf{c}}_n], \mathsf{id}) \cdot \theta\\
&\quad = \hat{\zeta}^\sharp \cdot T\nabla^\sharp \cdot (\rho^{\mathsf{cv}})^\sharp \cdot \chi \cdot \Sigma_{\mathsf{c}}(T\langle \iota^{\mathsf{v}}, \gamma^{\mathsf{v}} \rangle \cdot \hat{\zeta}^\sharp \cdot [\eta \cdot \iota^{\mathsf{v}}, \gamma^{\mathsf{c}}_n], \mathsf{id}) \cdot \theta\\
&\quad \sqsubseteq \hat{\zeta}^\sharp \cdot T\nabla^\sharp \cdot (\rho^{\mathsf{cv}})^\sharp \cdot \chi \cdot \Sigma_{\mathsf{c}}(T\langle \iota^{\mathsf{v}}, \gamma^{\mathsf{v}} \rangle \cdot \hat{\zeta}^\sharp \cdot [\eta \cdot \iota^{\mathsf{v}}, \eta \cdot \iota^{\mathsf{c}}], \mathsf{id}) \cdot \theta && /\!/ \text{ (31)}\\
&\quad = \hat{\zeta}^\sharp \cdot \chi \cdot \Sigma_{\mathsf{c}}([\eta \cdot \iota^{\mathsf{v}}, \eta \cdot \iota^{\mathsf{c}}], \mathsf{id}) \cdot \theta\\
&\quad = \zeta \cdot \Sigma_{\mathsf{c}}(\iota, \mathsf{id}) \cdot \theta.
\end{aligned}$$

The equation (32) is shown as follows:

$$\begin{aligned}
&\Sigma_{\mathsf{c}}(\eta \cdot \mathsf{fst} \;\boxplus \mathsf{snd}, \mathsf{inl}) \cdot \theta \cdot \Theta(\langle \iota^{\mathsf{v}}, \gamma^{\mathsf{v}} \rangle, \langle \iota^{\mathsf{c}}, \gamma_n^{\mathsf{c}} \rangle, \mathsf{id}) \\
&\qquad = \Sigma_{\mathsf{c}}(\mathsf{id} \boxplus \mathsf{id}, \mathsf{inl}) \cdot \Sigma_{\mathsf{c}}(\eta \cdot \mathsf{fst} + \mathsf{snd}, \mathsf{id}) \cdot \theta \cdot \Theta(\langle \iota^{\mathsf{v}}, \gamma^{\mathsf{v}} \rangle, \langle \iota^{\mathsf{c}}, \gamma_n^{\mathsf{c}} \rangle, \mathsf{id}) \\
&\qquad = \Sigma_{\mathsf{c}}(\mathsf{id} \boxplus \mathsf{id}, \mathsf{inl}) \cdot \theta \cdot \Theta(\eta \cdot \mathsf{fst}, \mathsf{snd}, \mathsf{id}) \cdot \Theta(\langle \iota^{\mathsf{v}}, \gamma^{\mathsf{v}} \rangle, \langle \iota^{\mathsf{c}}, \gamma_n^{\mathsf{c}} \rangle, \mathsf{id}) \\
&\qquad = \Sigma_{\mathsf{c}}(\mathsf{id} \boxplus \mathsf{id}, \mathsf{inl}) \cdot \theta \cdot \Theta(\eta, \mathsf{id}, \mathsf{id}) \cdot \Theta(\iota^{\mathsf{v}}, \gamma_n^{\mathsf{c}}, \mathsf{id}) \\
&\qquad = \Sigma_{\mathsf{c}}(\mathsf{id} \boxplus \mathsf{id}, \mathsf{inl}) \cdot \Sigma_{\mathsf{c}}(\eta + \mathsf{id}, \mathsf{id}) \cdot \theta \cdot \Theta(\iota^{\mathsf{v}}, \gamma_n^{\mathsf{c}}, \mathsf{id}) \\
&\qquad = \Sigma_{\mathsf{c}}(\eta \boxplus \mathsf{id}, \mathsf{inl}) \cdot \theta \cdot \Theta(\iota^{\mathsf{v}}, \gamma_n^{\mathsf{c}}, \mathsf{id}) \\
&\qquad = \Sigma_{\mathsf{c}}(\eta \cdot \iota^{\mathsf{v}} \boxplus \gamma_n^{\mathsf{c}}, \mathsf{inl}) \cdot \theta \qquad \square
\end{aligned}$$

A.7 Proof of Proposition 5.3

By rewriting the relevant subexpression of (19):

$$\begin{aligned}
&[\eta, f^{\sharp} \cdot T\mu \cdot \xi^{\sharp} \cdot \chi \cdot \Sigma_{\mathsf{c}}(f, \mathsf{id})] \cdot \iota^{-1} \\
&\quad = [\eta, f^{\sharp} \cdot T\nabla^{\sharp} \cdot (\rho^{\mathsf{cv}})^{\sharp} \cdot T\Sigma_{\mathsf{c}}(\langle \iota^{\mathsf{v}}, \gamma^{\mathsf{v}} \rangle, \mathsf{id}) \cdot \chi \cdot \Sigma_{\mathsf{c}}(f, \mathsf{id})] \cdot \iota^{-1} && \text{// Lemma 4.2} \\
&\quad = [\eta, f^{\sharp} \cdot T\nabla^{\sharp} \cdot (\rho^{\mathsf{cv}})^{\sharp} \cdot \chi \cdot \Sigma_{\mathsf{c}}(T\langle \iota^{\mathsf{v}}, \gamma^{\mathsf{v}} \rangle, \mathsf{id}) \cdot \Sigma_{\mathsf{c}}(f, \mathsf{id})] \cdot \iota^{-1} \\
&\quad = [\eta, f^{\sharp} \cdot T\nabla^{\sharp} \cdot (\rho^{\mathsf{cv}})^{\sharp} \cdot \chi \cdot \Sigma_{\mathsf{c}}(T\langle \iota^{\mathsf{v}}, \gamma^{\mathsf{v}} \rangle \cdot f, \mathsf{id})] \cdot \iota^{-1}. && \square
\end{aligned}$$

A.8 Proof of Proposition 7.2

Observe first, that the following diagram commutes:

$$\begin{tikzcd}
\Sigma_{\mathsf{c}}(\Sigma_{\mathsf{v}}(\mu\Sigma \times (\mu\Sigma \multimapdot \mu\Sigma)), \mu\Sigma) \arrow[rr, "\Sigma_{\mathsf{c}}(\langle \iota^{\mathsf{v}} \cdot \Sigma_{\mathsf{v}}\,\mathsf{fst},\, \rho^{\mathsf{v}} \rangle, \mathsf{id})"] \arrow[d, "\Sigma_{\mathsf{c}}(\Sigma_{\mathsf{v}}(\eta \times \mathsf{id}), \eta)"'] & & \Sigma_{\mathsf{c}}(\mu\Sigma \times D(\mu\Sigma, \Sigma^{\star}(\mu\Sigma + \mu\Sigma)), \mu\Sigma) \arrow[ddddddd, "\Sigma_{\mathsf{c}}(\mathsf{id} \times D(\mathsf{id}, \nabla^{\sharp}), \mathsf{id})"] \\
\Sigma_{\mathsf{c}}(\Sigma_{\mathsf{v}}(\mu^2\Sigma \times (\mu\Sigma \multimapdot \mu\Sigma)), \mu^2\Sigma) \arrow[d, "\Sigma_{\mathsf{c}}(\Sigma_{\mathsf{v}}(\mathsf{id} \times (\mu \multimapdot \mathsf{id})), \mathsf{id})"'] \arrow[ddr, bend left, "\Sigma_{\mathsf{c}}(\langle \iota^{\mathsf{v}} \cdot \Sigma_{\mathsf{v}}\,\mathsf{fst}, \Sigma_{\mathsf{v}}(\mu \times \mathsf{id}) \rangle, \mathsf{id})"] & & \\
\Sigma_{\mathsf{c}}(\Sigma_{\mathsf{v}}(\mu^2\Sigma \times (\mu^2\Sigma \multimapdot \mu\Sigma)), \mu^2\Sigma) \arrow[ddd, "\Sigma_{\mathsf{c}}(\langle \iota^{\mathsf{v}} \cdot \Sigma_{\mathsf{v}}\,\mathsf{fst},\, \rho^{\mathsf{v}} \rangle, \mathsf{id})"'] & & \\
& \Sigma_{\mathsf{c}}(\mu^2\Sigma \times \Sigma_{\mathsf{v}}(\mu\Sigma \times (\mu\Sigma \multimapdot \mu\Sigma)), \mu^2\Sigma) \arrow[d, "\Sigma_{\mathsf{c}}(\mathsf{id} \times \rho^{\mathsf{v}}, \mathsf{id})"'] & \\
& \Sigma_{\mathsf{c}}(\mu^2\Sigma \times D(\mu\Sigma, \Sigma^{\star}(\mu\Sigma + \mu\Sigma)), \mu^2\Sigma) \arrow[uuuur, bend right, "\Sigma_{\mathsf{c}}(\mu \times \mathsf{id}, \mu)"'] \arrow[dd, "\Sigma_{\mathsf{c}}(\mathsf{id} \times D(\mu, \mathsf{id}), \mathsf{id})"] & \\
\Sigma_{\mathsf{c}}(\mu^2\Sigma \times D(\mu^2\Sigma, \Sigma^{\star}(\mu^2\Sigma + \mu\Sigma)), \mu^2\Sigma) \arrow[dr, bend right, "\Sigma_{\mathsf{c}}(\mathsf{id} \times D(\mathsf{id}, \mu + \mathsf{id}), \mathsf{id})"] \arrow[dddr, bend right, "\Sigma_{\mathsf{c}}(\mathsf{id} \times D(\mathsf{id}, [\mu, \mathsf{id}]^{\sharp}), \mathsf{id})"] \arrow[dd, "\rho^{\mathsf{cv}}"'] & & \\
& \Sigma_{\mathsf{c}}(\mu^2\Sigma \times D(\mu^2\Sigma, \Sigma^{\star}(\mu\Sigma + \mu\Sigma)), \mu^2\Sigma) \arrow[dd, "\Sigma_{\mathsf{c}}(\mathsf{id} \times D(\mathsf{id}, \nabla^{\sharp}), \mathsf{id})"] & \\
T\Sigma^{\star}(\mu^2\Sigma + \Sigma^{\star}(\mu^2\Sigma + \mu\Sigma)) \arrow[dddr, bend right, "T\Sigma^{\star}(\mathsf{id} + [\mu, \mathsf{id}]^{\sharp})"] \arrow[dddd, "{T[\Sigma^{\star}\,\mathsf{inl}, [\Sigma^{\star}\,\mathsf{inl}, \eta \cdot \mathsf{inr}]^{\sharp}]^{\sharp}}"'] & & \Sigma_{\mathsf{c}}(\mu\Sigma \times D(\mu\Sigma, \mu\Sigma), \mu\Sigma) \arrow[dd, "\rho^{\mathsf{cv}}"] \\
& \Sigma_{\mathsf{c}}(\mu^2\Sigma \times D(\mu^2\Sigma, \mu\Sigma), \mu^2\Sigma) \arrow[dd, "\rho^{\mathsf{cv}}"] & \\
& & T\Sigma^{\star}(\mu\Sigma + \mu\Sigma) \arrow[dd, "T\nabla^{\sharp}"] \\
& T\Sigma^{\star}(\mu^2\Sigma + \mu\Sigma) \arrow[ur, bend right, "T\Sigma^{\star}(\mu + \mathsf{id})"] & \\
T\Sigma^{\star}(\mu\Sigma + \mu\Sigma) \arrow[rr, "T\nabla^{\sharp}"] & & T(\mu\Sigma)
\end{tikzcd}$$

We then prove the claim by modifying the argument from Proposition 5.3. It suffices to show that

$$T\nabla^{\sharp} \cdot \xi \cdot \Sigma_{\mathsf{c}}(\Sigma_{\mathsf{v}}\langle \mathsf{id}, (\mathsf{subst} \cdot (\mathsf{id} \times \mu))^{\flat} \rangle, \mathsf{id}) = T\nabla^{\sharp} \cdot \rho^{\mathsf{cv}} \cdot \Sigma_{\mathsf{c}}(\langle \iota^{\mathsf{v}}, \gamma^{\mathsf{v}} \rangle, \mathsf{id}).$$

Using the definition

$$\xi = T[\Sigma^{\star}\,\mathsf{inl}, [\Sigma^{\star}\,\mathsf{inl}, \eta \cdot \mathsf{inr}]^{\sharp}]^{\sharp} \cdot \rho^{\mathsf{cv}} \cdot \Sigma_{\mathsf{c}}(\langle \iota^{\mathsf{v}} \cdot \Sigma_{\mathsf{v}}\,\mathsf{fst},\ \rho^{\mathsf{v}} \rangle, \mathsf{id}) \cdot \Sigma_{\mathsf{c}}(\Sigma_{\mathsf{v}}(\eta \times \mathsf{id}), \eta)$$

and the above diagram,

$$\begin{aligned}
&T\nabla^\sharp \cdot \xi \cdot \Sigma_\mathsf{c}(\Sigma_\mathsf{v}\langle \mathsf{id}, (\mathsf{subst}\cdot(\mathsf{id}\times\mu))^\flat\rangle, \mathsf{id}) \\
&\qquad = T\nabla^\sharp \cdot T[\Sigma^\star\,\mathsf{inl}, [\Sigma^\star\,\mathsf{inl}, \eta\cdot\mathsf{inr}]^\sharp]^\sharp \cdot \rho^\mathsf{cv} \cdot \Sigma_\mathsf{c}(\langle \iota^\mathsf{v}\cdot\Sigma_\mathsf{v}\,\mathsf{fst},\ \rho^\mathsf{v}\rangle, \mathsf{id})\cdot \\
&\qquad\qquad \Sigma_\mathsf{c}(\Sigma_\mathsf{v}(\eta\times\mathsf{id}), \eta) \cdot \Sigma_\mathsf{c}(\Sigma_\mathsf{v}\langle \mathsf{id}, (\mathsf{subst}\cdot(\mathsf{id}\times\mu))^\flat\rangle, \mathsf{id}) \\
&\qquad = T\nabla^\sharp \cdot T[\Sigma^\star\,\mathsf{inl}, [\Sigma^\star\,\mathsf{inl}, \eta\cdot\mathsf{inr}]^\sharp]^\sharp \cdot \rho^\mathsf{cv} \cdot \Sigma_\mathsf{c}(\langle \iota^\mathsf{v}\cdot\Sigma_\mathsf{v}\,\mathsf{fst},\ \rho^\mathsf{v}\rangle, \mathsf{id})\cdot \\
&\qquad\qquad \Sigma_\mathsf{c}(\Sigma_\mathsf{v}(\eta\times\mathsf{id}), \eta) \cdot \Sigma_\mathsf{c}(\Sigma_\mathsf{v}(\mathsf{id}\times(\mu \mathrel{-\!\!\bullet} \mathsf{id})), \mathsf{id}) \cdot \Sigma_\mathsf{c}(\Sigma_\mathsf{v}\langle \mathsf{id}, \mathsf{subst}^\flat\rangle, \mathsf{id}) \\
&\qquad = T\nabla^\sharp \cdot T[\Sigma^\star\,\mathsf{inl}, [\Sigma^\star\,\mathsf{inl}, \eta\cdot\mathsf{inr}]^\sharp]^\sharp \cdot \rho^\mathsf{cv} \cdot \Sigma_\mathsf{c}(\langle \iota^\mathsf{v}\cdot\Sigma_\mathsf{v}\,\mathsf{fst},\ \rho^\mathsf{v}\rangle, \mathsf{id})\cdot \\
&\qquad\qquad \Sigma_\mathsf{c}(\Sigma_\mathsf{v}(\mathsf{id}\times(\mu \mathrel{-\!\!\bullet} \mathsf{id})), \mathsf{id}) \cdot \Sigma_\mathsf{c}(\Sigma_\mathsf{v}(\eta\times\mathsf{id}), \eta) \cdot \Sigma_\mathsf{c}(\Sigma_\mathsf{v}\langle \mathsf{id}, \mathsf{subst}^\flat\rangle, \mathsf{id}) \\
&\qquad = T\nabla^\sharp \cdot \rho^\mathsf{cv} \cdot \Sigma_\mathsf{c}(\mathsf{id}\times D(\mathsf{id}, \nabla^\sharp), \mathsf{id})\cdot \\
&\qquad\qquad \Sigma_\mathsf{c}(\langle \iota^\mathsf{v}\cdot\Sigma_\mathsf{v}\,\mathsf{fst},\ \rho^\mathsf{v}\rangle, \mathsf{id}) \cdot \Sigma_\mathsf{c}(\Sigma_\mathsf{v}\langle \mathsf{id}, \mathsf{subst}^\flat\rangle, \mathsf{id}) \\
&\qquad = T\nabla^\sharp \cdot \rho^\mathsf{cv} \cdot \Sigma_\mathsf{c}(\langle \iota^\mathsf{v}\cdot\Sigma_\mathsf{v}\,\mathsf{fst},\ D(\mathsf{id}, \nabla^\sharp)\cdot\rho^\mathsf{v}\rangle, \mathsf{id}) \cdot \Sigma_\mathsf{c}(\Sigma_\mathsf{v}\langle \mathsf{id}, \mathsf{subst}^\flat\rangle, \mathsf{id}) \\
&\qquad = T\nabla^\sharp \cdot \rho^\mathsf{cv} \cdot \Sigma_\mathsf{c}(\langle \iota^\mathsf{v},\ D(\mathsf{id}, \nabla^\sharp)\cdot\rho^\mathsf{v}\cdot\Sigma_\mathsf{v}\langle \mathsf{id}, \mathsf{subst}^\flat\rangle\rangle, \mathsf{id}) \\
&\qquad = T\nabla^\sharp \cdot \rho^\mathsf{cv} \cdot \Sigma_\mathsf{c}(\langle \iota^\mathsf{v}, \gamma^\mathsf{v}\rangle, \mathsf{id}),
\end{aligned}$$

as desired. □