Category-Graded Algebraic Theories and Effect Handlers

Takahiro Sanada[1,2]

Research Institute for Mathematical Sciences
Kyoto University
Kyoto, Japan

Abstract

We provide an effect system *CatEff* based on a category-graded extension of algebraic theories that correspond to category-graded monads. *CatEff* has category-graded operations and handlers. Effects in *CatEff* are graded by morphisms of the grading category. Grading morphisms represent fine structures of effects such as dependencies or sorts of states. Handlers in *CatEff* are regarded as an implementation of category-graded effects. We define the notion of category-graded algebraic theory to give semantics of *CatEff* and prove soundness and adequacy. We also give an example using category-graded effects to express protocols for sending receiving typed data.

Keywords: Algebraic theory, algebraic effect, effect handler, category-graded monad

1 Introduction

1.1 Background

Moggi [14] used *monads* to capture computational effects. Monads have a close relationship with *algebraic theories* [7]. Algebraic effects [17] are effects based on algebraic theories. Handlers of algebraic effects [18] provide clear ways to implement effects. Algebraic effects and handlers are useful notions to make programs with effects.

There are several extensions of monads [1,8,15,16]. These variations of monads enable us to reason about computational effects in more detail.

Parameterised monads [1] are monads with parameters which represent initial and terminal states of computational effects such as change of type of state. In an effect system based on parameterised monads, each computational term is graded by an object of a parameter category $\mathcal{S}$. For example, we can capture the feature of mutable state of mutable type. To see this, let $\mathcal{S}$ be a discrete category whose objects are $\mathtt{int}$ and $\mathbf{1}$. The parameter indicates the type of state. We can construct computational terms, M with parameter $\mathtt{int}$ and N with parameter $\mathbf{1}$. We can know that the computations M and N have states of type $\mathtt{int}$ and $\mathbf{1}$ respectively. Let us consider lookup and update operations for this mutable state of mutable type. There are two lookup operations and four update operations $\mathrm{lookup}_{\mathtt{int}}(), \mathrm{lookup}_{\mathbf{1}}()$, $\mathrm{update}_{\mathtt{int}\to\mathbf{1}}(V)$, $\mathrm{update}_{\mathbf{1}\to\mathtt{int}}(V)$, $\mathrm{update}_{\mathtt{int}\to\mathtt{int}}(V)$ and $\mathrm{update}_{\mathbf{1}\to\mathbf{1}}(V)$. $\mathrm{lookup}_{\mathtt{int}}()$ is an operation that

[1] I would like to thank the people of computer science group at RIMS, especially Masahito Hasegawa and Soichiro Fujii for discussions and comments, and the anonymous reviewers for comments. This work was supported by JST, the establishment of university fellowships towards the creation of science technology innovation, Grant Number JPMJFS2123 and JST ERATO Grant Number JPMJER1603.

[2] Email: tsanada@kurims.kyoto-u.ac.jp

reads the state of type $\texttt{int}$ and returns the value in it. We can use $\mathrm{lookup}_{\texttt{int}}()$ only when the type of the state is $\texttt{int}$. $\mathrm{lookup}_{\mathbf{1}}()$ is similar. $\mathrm{update}_{\alpha\to\beta}(V)$ is an operation that writes the value V in the state changing the type of state from α to β. The parameter category $\mathcal{S}$ does not need to be a discrete category. If $\mathcal{S}$ has nontrivial morphisms, the intuition is that morphisms mean subtyping relations between the types of the state.

Graded monads [8,12] are monads graded by a partially ordered monoid. Elements of the partially ordered monoid express quantity of effects such as memory locations that effects affect. Its formal theory was given in [5], and its algebraic theories were given in [20,4,9]. For example, let $L = \{l_1, l_2, \dots, l_n\}$ be a set of memory locations. Then we can get a partially ordered monoid 2^L where the product $\cdot$ of 2^L is the union of sets $\cup$ and the order $\le$ of 2^L is the inclusion $\subseteq$. If a computation term M is graded by $A \in 2^L$, we can know that M may access the memory locations contained in A. The role of the order is weakening of a set of locations which may be accessed. If a computation term M is graded by $A \in 2^L$ and $A \le B$, we can deduce that M is also graded by B. The intuition is that computation that may access the locations in A may access the locations in B which is larger than A. Let us consider memory lookup and update operations. Let $\mathrm{lookup}_i()$ be an operation that reads location l_i and returns its value and $\mathrm{update}_j(V)$ be an operation that writes V on the location l_j and returns the unit value. $\mathrm{lookup}_i()$ and $\mathrm{update}_j(V)$ are graded by $\{l_i\}$ and $\{l_j\}$ respectively. If M is graded by $A \in 2^L$, $\mathbf{let}\, x \leftarrow \mathrm{lookup}_i()\, \mathbf{in}\, M$ and $\mathbf{let}\, \_ \leftarrow \mathrm{update}_j(V)\, \mathbf{in}\, M$ are graded by $\{l_i\} \cdot A$ and $\{l_j\} \cdot A$, respectively.

Category-graded monads [15] are introduced to unify parameterised and graded monads. Graded monads are 2-category-graded monads with a single object. Parameterised monads are category-graded monads with *generalised units*. Category-graded monads and the constructions of these Eilenberg-Moore and Kleisli categories are a special case of lax functors and these two constructions are studied by Street [21].

1.2 Overview

In this paper, we provide

- category-graded extensions of algebraic theories,
- a category-graded effect system *CatEff* with effect handlers based on category-graded algebraic theories, and
- operational and denotational semantics of the effect system.

In category-graded algebraic theories, terms are graded by morphisms of a grading category $\mathcal{S}$. In the effect system that corresponds to category-graded algebraic theories, we will define a judgement $\Gamma \vdash_f M : A$ for computational term M and a morphism f in $\mathcal{S}$. This judgement means that the computation M will return a value of type A under the environment Γ and invoke effects graded by f. The term M will be denoted by a map $[\![\Gamma]\!] \to T_f[\![A]\!]$, where T is a category-graded monad. Grading morphisms can express a finer structure of computational effects than elements of monoids in ordinary graded monads or parameters in parameterised monads can, especially structures of dependency and sorts of state.

For instance, let us consider the following morphisms $f = (\mathbf{1} \xrightarrow{\tau^{\mathbf{1}}_{\texttt{int}}} \texttt{int} \xrightarrow{send_{\texttt{int}}} \texttt{int} \xrightarrow{recv^{\texttt{int}}_{\texttt{int}}} \texttt{int})$ and $g = (\mathbf{1} \xrightarrow{recv^{\mathbf{1}}_{\texttt{int}}} \texttt{int} \xrightarrow{send_{\texttt{int}}} \texttt{int})$ in a category $\mathcal{S}$. A computational term M graded by f is a computation that behaves as follows.

(i) The type of the initial state is unit type $\mathbf{1}$.

(ii) Some effects are invoked in M and a value of type $\texttt{int}$ is stored in the state. Thus, the type of state is changed from $\mathbf{1}$ to $\texttt{int}$. These effects are graded by $\tau^{\mathbf{1}}_{\texttt{int}}$, which means internal computation with a change of types of the state.

(iii) An effect sending the value of the state to another process is invoked. It is graded by $send_{\texttt{int}}$. The type of the state is not changed by the sending effect, so the domain and codomain of $send_{\texttt{int}}$ are the same.

(iv) An effect receiving graded by $recv^{\texttt{int}}_{\texttt{int}}$ is invoked. It receives a value of type $\texttt{int}$ and stores it in the state. In this case, the types of state before receiving and after receiving are the same, but in general, they may be changed.

A computational term N graded by g is a computation that behaves as follows.

(i) The type of the initial type is unit type $\mathbf{1}$.

(ii) An effect graded by $recv^{\mathbf{1}}_{\mathtt{int}}$ is invoked. It receives a value of type `int` and stores it in the state. The type of state before receiving and the type of receiving value are different, so the type of state is changed from $\mathbf{1}$ to `int`.

(iii) An effect graded by $send_{\mathtt{int}}$ is invoked. It sends a value of the state.

Thanks to the grading morphisms, we can know the transition of the type of the state, and deduce that M and N can interact with each other and yield values. We think of morphisms in $\mathcal{S}$ as protocols of communication.

We can construct handlers of category-graded effects. As ordinary handlers are the morphisms induced by the universality of free models of algebraic theories, category-graded handlers are the morphisms induced by the universality of free models of category-graded algebraic theories. We can regard category-graded handlers as an implementation of category-graded effects. (Monoid-)graded monads without order and parameterised monads whose parameterising category is discrete are special category-graded monads, so we can get handlers for effects corresponding to these monads automatically.

Contents. In Section 2, we introduce notations and review some categorical notions. In Section 3, we define category-graded algebraic theory and describe the free construction for the theory. In Section 4, we explain our effect system based on category-graded algebraic theory. We call the effect system *CatEff*. The effect system has handlers of category-graded effects. In Section 5 and Section 6, we describe operational and denotational semantics of our effect system, respectively. In Section 7, we show the soundness and adequacy of the semantics.

2 Category-Graded Monads

We assume that readers are familiar with basic notions of category theory such as monads [11]. Throughout this paper, we use the following notations.

- Let $\mathbf{CAT}$ be the 2-category of all categories, functors and natural transformations.
- Let $\mathbf{Set}$ be the category of all sets and maps.
- For a category $\mathcal{C}$, $\mathrm{Id}_{\mathcal{C}}$ is the identity functor on $\mathcal{C}$.
- For a category $\mathcal{C}$, $\mathrm{Endo}(\mathcal{C})$ is the full 2-subcategory of $\mathbf{CAT}$ whose 0-cell is only $\mathcal{C}$.
- For a category $\mathcal{C}$ with finite and countable products and coproducts, an object C of $\mathcal{C}$, and a finite or countable set X, C^X is the X-fold product of C, that is $\prod_{x \in X} C$ and $X \times C$ is the X-fold sum of C, that is $\sum_{x \in X} C$.

2.1 Lax Functors and Category-Graded Monads

Category-graded monads are introduced by Orchard et al. [15]. In this section, we fix a category $\mathcal{C}$ and a small category $\mathcal{S}$.

Definition 2.1 [Lax functor] Let $\mathbf{C}$ be a 2-category. A *lax functor* $F \colon \mathcal{S} \xrightarrow{\text{lax}} \mathbf{C}$ is a tuple $F = (F, \eta^F, \mu^F)$ where

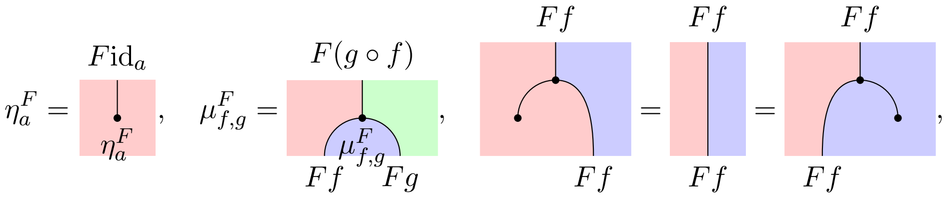
- For each object a of $\mathcal{S}$, Fa is a 0-cell of $\mathbf{C}$.
- For each morphism $f \colon a \to b$ of $\mathcal{S}$, $Ff \colon Fa \to Fb$ is a 1-cell of $\mathbf{C}$.
- For each object a of $\mathcal{S}$, $\eta^F_a \colon \mathrm{id}_{Fa} \Rightarrow F\mathrm{id}_a$ is a 2-cell of $\mathbf{C}$.
- For each morphism $f \colon a \to b$ and $g \colon b \to c$ of $\mathcal{S}$, $\mu^F_{g,f} \colon Fg \circ Ff \Rightarrow F(g \circ f)$ is a 2-cell of $\mathbf{C}$

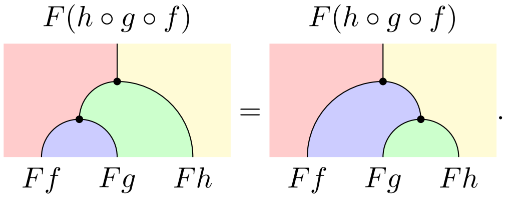
satisfying the following commutative diagrams:

$$\begin{tikzcd}
Ff \arrow[r, Rightarrow, "Ff\eta^F_a"] \arrow[d, Rightarrow, "\eta^F_b Ff"'] \arrow[dr, equal] & Ff \circ F\mathrm{id}_a \arrow[d, Rightarrow, "\mu^F_{f,\mathrm{id}_a}"] \\
F\mathrm{id}_b \circ Ff \arrow[r, Rightarrow, "\mu^F_{\mathrm{id}_b,f}"'] & Ff
\end{tikzcd}
\qquad
\begin{tikzcd}
Fh \circ Fg \circ Ff \arrow[r, Rightarrow, "Fh\mu^F_{g,f}"] \arrow[d, Rightarrow, "\mu^F_{h,g}Ff"'] & Fh \circ F(g \circ f) \arrow[d, Rightarrow, "\mu^F_{h,g\circ f}"] \\
F(h \circ g)Ff \arrow[r, Rightarrow, "\mu^F_{h\circ g,f}"'] & F(h \circ g \circ f)
\end{tikzcd}$$

We call η^F and μ^F *unit* and *multiplication* of F respectively.

We use string diagrams [19] for diagrammatic reasoning. In string diagram, a region of a diagram represents a 0-cell of a 2-category, a string between two regions represents a 1-cell from the 0-cell of the left regions to the 0-cell of the right region, and a node on strings represents a 2-cells from the 1-cell of bottom strings to the 1-cell of top strings. We can depict unit and multiplication, and the axioms of lax functor by string diagrams as follows:





Definition 2.2 [Category-graded monads] A *category-graded monad* (or an *$\mathcal{S}$-graded monad*) on $\mathcal{C}$ is a lax functor $\mathcal{S}^{\mathrm{op}} \xrightarrow{\mathrm{lax}} \mathrm{Endo}(\mathcal{C})$. That is, an $\mathcal{S}$-graded monad consists of mapping of objects and morphisms $T\colon \mathcal{S}^{\mathrm{op}} \to \mathrm{Endo}(\mathcal{C})$ and families of natural transformations $\eta_a\colon \mathrm{Id}_{\mathcal{C}} \Rightarrow T_{\mathrm{id}_a}$ for $a \in \mathcal{S}$ and $\mu_{f,g}\colon T_fT_g \Rightarrow T_{f;g}$ for $f\colon a \to b$ and $g\colon b \to c$ in $\mathcal{S}$ that make the following diagrams commute.

$$\begin{tikzcd}
T_f \arrow[r, Rightarrow, "\eta_a T_f"] \arrow[d, Rightarrow, "T_f\eta_b"'] \arrow[dr, equal] & T_{\mathrm{id}_a}T_f \arrow[d, Rightarrow, "\mu_{\mathrm{id}_a,f}"] \\
T_fT_{\mathrm{id}_b} \arrow[r, Rightarrow, "\mu_{f,\mathrm{id}_b}"'] & T_f
\end{tikzcd}
\qquad
\begin{tikzcd}
T_fT_gT_h \arrow[r, Rightarrow, "T_f\mu_{g,h}"] \arrow[d, Rightarrow, "\mu_{f,g}T_h"'] & T_fT_{g;h} \arrow[d, Rightarrow, "\mu_{f,g;h}"] \\
T_{f;g}T_h \arrow[r, Rightarrow, "\mu_{f;g,h}"'] & T_{f;g;h}
\end{tikzcd}$$

If $\mathcal{S}$ is the trivial category, that is the category with a single object and the identity morphism on it, $\mathcal{S}$-graded monad is a usual monad [3]. If $\mathcal{S}$ is a category with single object and endomorphisms on it, the endomorphisms form a monoid and $\mathcal{S}$-graded monad is a (monoid-)graded monad without order. To consider parameterised monads as category-graded monads introduced in [15], we have to introduce *generalised units*, see [15].

2.2 Eilenberg-Moore Construction on Lax Functors

According to [21], there are two functors obtained from a lax functor. The two constructions correspond to Eilenberg-Moore and Kleisli construction on a monad, respectively. In this subsection, we review the Eilenberg-Moore construction on lax functors.

Let $\mathcal{S}$ be a small category and $F\colon \mathcal{S} \to \mathbf{CAT}$ be a lax functor. The Eilenberg-Moore construction gives a functor $\hat{F}\colon \mathcal{S} \to \mathbf{CAT}$.

Definition 2.3 For a lax functor $F = (F, \eta^F, \mu^F)\colon \mathcal{S} \to \mathbf{CAT}$, the functor $\hat{F}\colon \mathcal{S} \to \mathbf{CAT}$ is defined as follows:

- For an object $a \in \operatorname{Ob}\mathcal{S}$, the category $\hat{F}a$ is defined as follows.
  - · Objects are pairs (A, ξ) where A is a map which assigns to each morphism $f\colon a \to b$ of $\mathcal{S}$ an object $A_f \in \operatorname{Ob} Fb$ and ξ is a family of morphisms $\{\xi_{f,g}\colon F_f A_g \to A_{f \circ g}\}_{f,g}$ such that the following diagrams commute:

$$\begin{tikzcd}
A_f \arrow[r, "\eta^F_b A_f"] \arrow[dr, equal] & F_{\mathrm{id}_b} A_f \arrow[d, "\xi_{\mathrm{id}_b, f}"] & F_h F_g A_f \arrow[r, "F_h \xi_{g,f}"] \arrow[d, "\mu^F_{h,g} A_f"'] & F_h A_{g \circ f} \arrow[d, "\xi_{h, g \circ f}"] \\
 & A_f & F_{h \circ g} A_f \arrow[r, "\xi_{h \circ g, f}"'] & A_{h \circ g \circ f}
\end{tikzcd}$$

  for each $f\colon a \to b$, $g\colon b \to c$, $h\colon c \to d$ in $\mathcal{S}$.
  - · Morphisms are $\alpha\colon (A, h) \to (A', \xi')$ where α is a family of morphisms $\left\{\alpha_f\colon A_f \to A'_f\right\}_f$ such that the following diagram commutes:

$$\begin{tikzcd}
F_g A_f \arrow[r, "F_g \alpha_f"] \arrow[d, "\xi_{g,f}"'] & F_g A'_f \arrow[d, "\xi'_{g,f}"] \\
A_{g \circ f} \arrow[r, "\alpha_{g \circ f}"'] & A'_{g \circ f}
\end{tikzcd}$$

  for each $f\colon a \to b$, $g\colon b \to c$.
- For a morphism $f\colon a \to b$, a functor $\hat{F}f\colon \hat{F}a \to \hat{F}b$ is defined as follows:
  - · $\hat{F}f$ assigns an object $(B, \zeta) := (\hat{F}f)(A, \xi)$ of $\hat{F}b$ to an object (A, ξ) of $\hat{F}a$ where $B_g = A_{g \circ f}$ and $\zeta_{h,g} = \xi_{h, g \circ f}$ for $g\colon b \to c$, $h\colon c \to d$.
  - · For a morphism $\alpha\colon (A, \xi) \to (A', \xi')$ of $\hat{F}a$, a morphism

$$(\hat{F}f)\alpha\colon (\hat{F}f)(A, \xi) \to (\hat{F}f)(A', \xi')$$

  is defined as $((\hat{F}f)\alpha)_g = \alpha_{g \circ f}$ for $g\colon b \to c$ in $\mathcal{S}$.

Since $\mathcal{S}$-graded monad $T\colon \mathcal{S}^{\mathrm{op}} \to \mathrm{Endo}(\mathcal{C})$ is a special case of lax functor $\mathcal{S}^{\mathrm{op}} \to \mathbf{CAT}$, the category of Eilenberg-Moore algebras of $\mathcal{S}$-graded monad T is obtained by the above construction.

Next, we describe an adjunction between Fa and $\hat{F}a$ for $a \in \operatorname{Ob}\mathcal{S}$. A functor $\hat{J}_a\colon Fa \to \hat{F}a$ is defined as follows.

- For an object $X \in \operatorname{Ob} Fa$, $\hat{J}_a X := (A, \xi) \in \operatorname{Ob}\hat{F}a$ where $A_f = F_f X$ and $\xi_{g,f} = \mu^F_{g,f,X}\colon F_g A_f \to A_{g \circ f}$ for $f\colon a \to b$, $g\colon b \to c$ in $\mathcal{S}$.
- For a morphism $x\colon X \to Y$ in Fa, $\hat{J}_a x\colon \hat{J}_a X \to \hat{J}_a Y$ is a family of $\hat{J}_a x_f := F_f x$.

A functor $\hat{E}_a\colon \hat{F}a \to Fa$ is defined as follows.

- For an object $(A, \xi) \in \operatorname{Ob}\hat{F}a$, $\hat{E}_a(A, \xi) := A_{\mathrm{id}_a} \in Fa$.
- For a morphism $\alpha\colon (A, \xi) \to (A', \xi')$ in $\hat{F}a$, $\hat{E}_a \alpha = \alpha_{\mathrm{id}_a}$.

We will show that $\hat{J}$ is a left adjoint to $\hat{E}$. To do so, we construct a unit and a counit of the adjunction. The unit $\hat{\eta}_a\colon \mathrm{Id}_{Fa} \Rightarrow \hat{E}_a \hat{J}_a$ is a natural transformation whose components are $\hat{\eta}_{a,X} := \eta^F_X\colon X \to F_{\mathrm{id}_a} X$. The counit $\hat{\varepsilon}_a\colon \hat{J}_a \hat{E}_a \Rightarrow \mathrm{Id}_{\hat{F}a}$ is a natural transformation whose components are

$$\hat{\varepsilon}_{a,(A,\xi)} := \xi_{(-),\mathrm{id}_a}\colon \left(F_{(-)} A_{\mathrm{id}_a}, (\mu^F_{g,f,A_{\mathrm{id}_a}})_{g,f}\right) \to (A, \xi).$$

Proposition 2.4 *The tuple $(\hat{J}_a, \hat{E}_a, \hat{\eta}_a, \hat{\varepsilon}_a)$ forms an adjunction.*

Proof. Follows by definition. For detail, see [21]. □

For a morphism $f\colon a \to b$, let us calculate the functor $\hat{E}_b\hat{F}f\hat{J}_a\colon Fa \to Fb$. For an object $X \in Fa$, we have $\hat{E}_b\hat{F}f\hat{J}_a(X) = \hat{E}_b\hat{F}f\left(F_{(-)}X, (\mu^F_{g,k,X})_{g,k}\right) = \hat{E}_b\left(F_{(-)\circ f}X, (\mu^F_{g,k\circ f,X})_{g,k}\right) = F_{\mathrm{id}_b\circ f}X = F_fX$. For a morphism $x\colon X \to Y$, we have $\hat{E}_b\hat{F}f\hat{J}_a(x) = \hat{E}_b\hat{F}f(F_{\_}x) = \hat{E}_b(F_{\_\circ f}x) = F_{\mathrm{id}_b\circ f}x = F_fx$. Therefore, we have $F_f = \hat{E}_b\hat{F}f\hat{J}_a$.

2.3 *A Lax Functor Induced by Adjunctions and a Functor*

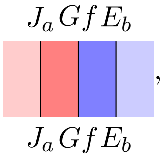
In Section 2.2, a lax functor $F\colon \mathcal{S} \to \mathbf{CAT}$ gives adjunctions and a functor. Conversely, we will show that adjunctions and a functor determine a lax functor. The construction of lax functor is a generalization of the construction [5,13] of lax M-action from an adjunction and a strict M-action where M is a monoid.

Theorem 2.5 *Given a functor $G\colon \mathcal{S} \to \mathbf{CAT}$, a map $F\colon \mathrm{Ob}\,\mathcal{S} \to \mathrm{Ob}\,\mathbf{CAT}$ and adjunctions $(J_a\colon Fa \to Ga, E_a\colon Ga \to Fa, \eta_a, \varepsilon_a)$ for each $a \in \mathrm{Ob}\,\mathcal{S}$, F is extended to a lax functor $F\colon \mathcal{S} \to \mathbf{CAT}$.*

Proof. For each morphism $f\colon a \to b$ of $\mathcal{S}$, we define $Ff := E_b \circ Gf \circ J_a\colon Fa \to Fb$. The unit η^F of F is induced by the units of the adjunctions. For each $a \in \mathrm{Ob}\,\mathcal{S}$, we define $\eta^F_a := \eta_a\colon \mathrm{Id}_{Fa} \Rightarrow F_{\mathrm{id}_a}$.

$$\begin{tikzcd}
Fa \arrow[r, "J_a"] \arrow[d, dashed, "Ff"'] & Ga \arrow[d, "Gf"] \\
Fb & Gb \arrow[l, "E_b"]
\end{tikzcd}$$

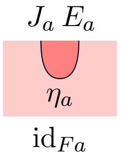
=



,

$$\begin{tikzcd}
Fa \arrow[r, "J_a"] \arrow[d, "\mathrm{Id}_{Fa}"'] \arrow[dr, phantom, "\overset{\eta_a}{\Longrightarrow}"] & Ga \arrow[d, "G\mathrm{id}_a = \mathrm{Id}_{Ga}"] \\
Fa & Ga \arrow[l, "E_a"]
\end{tikzcd}$$

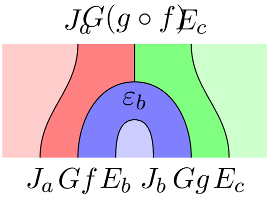
=



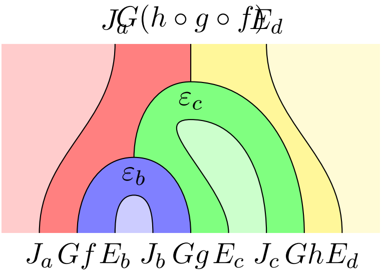
The multiplication μ^F of F is induced by the counits of adjunctions. We define $\mu^F_{g,f} = J_a\varepsilon_bE_c\colon Fg \circ Ff \Rightarrow F(g \circ f)$ for each $f\colon a \to b$ and $g\colon b \to c$ of $\mathcal{S}$.

$$\begin{tikzcd}
Ga \arrow[r, "Gf"] \arrow[rrr, bend left, "G(g\circ f)"] & Gb \arrow[r, "\mathrm{Id}_{Gb}"] \arrow[d, "E_b"'] & Gb \arrow[r, "Gg"] & Gc \arrow[d, "E_c"] \\
Fa \arrow[u, "J_a"] \arrow[r, "Ff"'] \arrow[ur, phantom, "\|"] & Fb \arrow[r, "\mathrm{Id}_{Fb}"'] \arrow[ur, phantom, "\varepsilon_b\Uparrow"] & Fb \arrow[u, "J_b"'] \arrow[r, "Fg"'] \arrow[ur, phantom, "\|"] & Fc
\end{tikzcd}$$

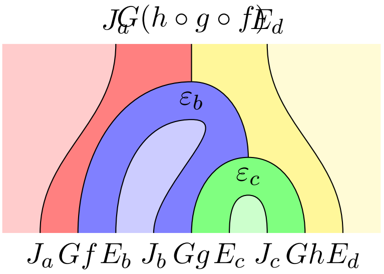
=

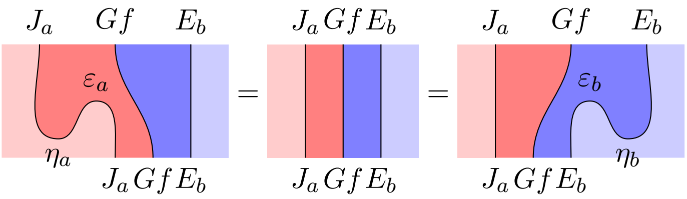


We claim that (F, η^F, μ^F) is a lax functor. To show this claim, we have to show that the axioms of lax functors hold. The following equations of string diagrams imply the axioms hold.


$J_aG(h\circ g\circ f)E_d$
ε_c
ε_b
$J_a\,Gf\,E_b\;J_b\,Gg\,E_c\;J_c\,Gh\,E_d$

=


$J_aG(h\circ g\circ f)E_d$
ε_b
ε_c
$J_a\,Gf\,E_b\;J_b\,Gg\,E_c\;J_c\,Gh\,E_d$



□

Corollary 2.6 *Given a functor $G\colon \mathcal{S}^{\mathrm{op}} \to \mathbf{CAT}$, a category $\mathcal{C}$ and adjunctions $(J_a\colon \mathcal{C} \to Ga, E_a\colon Ga \to \mathcal{C}, \eta_a, \varepsilon_a)$ for each $a \in \operatorname{Ob}\mathcal{S}$, there exists an $\mathcal{S}$-graded monad $T\colon \mathcal{S}^{\mathrm{op}} \to \operatorname{Endo}(\mathcal{C})$ such that $T_f = E_b Gf J_a$ for each $f\colon b \to a$ in $\mathcal{S}$.*

3 Category-Graded Algebraic Theories

We explain category-graded extensions of algebraic theories. In this section, we fix a small category $\mathcal{S}$ and a category $\mathcal{C}$ with countable products.

3.1 Category-Graded Terms

In a category-graded algebraic theory, each term is graded by a morphism in a grading category $\mathcal{S}$. This is analogous to parameterised and (monoid-)graded algebraic theories [1,9].

Definition 3.1 [Signature] An *($\mathcal{S}$-graded) signature* Σ is a set of symbols. For each $\sigma \in \Sigma$, countable or finite sets P and A, and a morphism f in $\mathcal{S}$ are assigned. $\sigma \in \Sigma$ is called an *operation*. P, A and f are called a *parameter*, *arity* and *grade* of σ, respectively. We write $\sigma\colon P \rightsquigarrow A; f$ for an operation σ whose parameter, arity and grade are P, A, f, respectively.

Definition 3.2 [Σ-term] Let X be a set. The set of Σ-*terms* $\mathrm{Term}_\Sigma(f, X)$ for each $f\colon b \to a$ in $\mathcal{S}$ is defined recursively as follows.

$$\frac{a \in \operatorname{Ob}\mathcal{S} \quad x \in X}{\mathbf{e}(a, x) \in \mathrm{Term}_\Sigma(\mathrm{id}_a, X)}$$

$$\frac{p \in P \quad \sigma\colon P \rightsquigarrow A; f\colon c \to b \quad \{t_i\}_{i \in A} \subset \mathrm{Term}_\Sigma(g\colon b \to a, X)}{\mathbf{do}(\sigma, p, \{t_i\}_{i \in A}) \in \mathrm{Term}_\Sigma(f; g, X)}$$

We sometimes write $\mathbf{do}(\sigma, p, \lambda i.t_i)$ instead of $\mathbf{do}(\sigma, p, \{t_i\}_{i \in A})$. Intuitively, $\mathbf{do}(\sigma, p, \lambda i.t_i)$ is the term that performs the operation σ with parameter p, binds the result to i, and invokes the continuation t_i. Note that, when A is the arity of σ, the term $\mathbf{do}(\sigma, p, \lambda i.t_i)$ is a term that takes A-many terms $\{t_i\}_{i \in A}$ as arguments.

Definition 3.3 [Σ-model] Let Σ be a signature. A Σ-*model* $I = (I, |{-}|^I)$ at $a \in \mathcal{S}$ is a pair of a map $I\colon \prod_{b \in \mathcal{S}} \mathcal{S}(b, a) \to \mathcal{C}$ and an interpretation $|\sigma|^I_k\colon P \times I(k)^A \to I(f; k)$ for each operation $\sigma\colon P \rightsquigarrow A; f\colon c \to b \in \Sigma$ and morphism $k\colon b \to a$ in $\mathcal{S}$.

A *homomorphism* $\alpha\colon I \to J$ between two Σ-models I and J at a is a family of morphisms $\{\alpha_k\colon I(k) \to J(k)\}_{k\colon b \to a}$ such that for every operation $\sigma\colon P \rightsquigarrow A; f\colon b \to c$ and morphism $k\colon c \to a$, the following diagram commutes:

$$\begin{tikzcd}
P \times I(k)^A \arrow[r, "P \times \alpha_k^A"] \arrow[d, "|\sigma|^I_k"'] & P \times J(k)^A \arrow[d, "|\sigma|^J_k"] \\
I(f; k) \arrow[r, "\alpha_{f;k}"'] & J(f; k)
\end{tikzcd}$$

The map $I : \prod_b \mathcal{S}(b, a) \to \mathcal{C}$ assigns a "carrier set" $I(k)$ to each $k : b \to a$. Given a Σ-model I, we can interpret Σ-terms by extending the interpretation of I.

Definition 3.4 [Interpretation of Σ-terms] Let Σ be a signature and I be a Σ-model at $a \in \mathrm{Ob}\,\mathcal{S}$. For each Σ-term $t \in \mathrm{Term}_\Sigma(f\colon b \to a, X)$, the element $|t|^I$, called its *interpretation*, of the set $\prod_{a' \in \mathcal{S}, k \in \mathcal{S}(a,a')} \mathcal{C}(I(k)^X, I(f;k))$ is defined recursively as

$$
\begin{aligned}
|\mathbf{e}(a,x)|^I &:= \{\pi_x\colon I(k)^X \to I(k)\}_{k\colon a \to a'},\\
|\mathbf{do}(\sigma, p, \lambda i.t_i)|^I &:= \{(p \times \langle |t_i|^I_k \rangle_{i \in A}); |\sigma|^I_{g;k}\colon I(k)^X \to I(f;g;k)\}_{k\colon a \to a'}
\end{aligned}
$$

where $(\sigma\colon P \rightsquigarrow A; f : c \to b) \in \Sigma$ and $t_i \in \mathrm{Term}_\Sigma(g\colon b \to a, X)$.

Intuitively, grading morphisms are sequences of sorts of effects that will be invoked by executing terms.

Example 3.5 Category graded algebraic theories are useful to deal with "order-sensitive" operations. To illustrate this, we provide an example that contains operations for mutable state and sending and receiving data. In this example, grading morphisms represent orders of sending and receiving effect and types of data, analogously to session types. Let $\mathcal{S}$ be a category whose objects are base types and morphisms are generated by $\alpha \xrightarrow{recv^\alpha_\beta} \beta$, $\alpha \xrightarrow{send_\alpha} \alpha$, $\alpha \xrightarrow{\tau^\alpha_\beta} \beta$, $\tau^\beta_\gamma \circ \tau^\alpha_\beta = \tau^\alpha_\gamma$, and $\tau^\alpha_\alpha = \mathrm{id}_\alpha$. Let $\mathrm{recvint}_\alpha\colon \mathbf{1} \rightsquigarrow \mathbf{1}; \alpha \xrightarrow{recv} \mathtt{int}$, $\mathrm{sendint}\colon \mathbf{1} \rightsquigarrow \mathbf{1}; \mathtt{int} \xrightarrow{send} \mathtt{int}$, $\mathrm{lookupint}\colon \mathbf{1} \rightsquigarrow \mathtt{int}; \mathtt{int} \xrightarrow{\mathrm{id}_{\mathtt{int}}} \mathtt{int}$, $\mathrm{updateint}_\alpha\colon \mathtt{int} \rightsquigarrow \mathbf{1}; \alpha \xrightarrow{\tau} \mathtt{int}$, and Σ be $\{\mathrm{recvint}_\alpha, \mathrm{updateint}_\alpha\}_\alpha \cup \{\mathrm{sendint}, \mathrm{lookupint}\}$. We have the following Σ-terms:

$$
\begin{aligned}
t := {} & \mathbf{do}(\mathrm{updateint}, 2, \lambda\_.\, \mathbf{do}(\mathrm{sendint}, \star, \lambda\_.\, \mathbf{do}(\mathrm{recvint}_{\mathtt{int}}, \star, \lambda\_.\\
& \mathbf{do}(\mathrm{lookupint}, \star, \lambda n.\, \mathbf{e}(\mathtt{int}, n))))) \in \mathrm{Term}_\Sigma(\tau; send_{\mathtt{int}}; recv^{\mathtt{int}}_{\mathtt{int}}, \mathtt{int}),
\end{aligned}
$$

$$
\begin{aligned}
s := {} & \mathbf{do}(\mathrm{recvint}_{\mathbf{1}}, \star, \lambda\_.\, \mathbf{do}(\mathrm{lookupint}, \star, \lambda n.\, \mathbf{do}(\mathrm{updateint}, n+1, \lambda\_.\\
& \mathbf{do}(\mathrm{sendint}_{\mathtt{int}}, \star, \lambda\_.\, \mathbf{e}(\mathtt{int}, \star))))) \in \mathrm{Term}_\Sigma(recv^{\mathbf{1}}_{\mathtt{int}}; send_{\mathtt{int}}, \mathbf{1}).
\end{aligned}
$$

The term t is graded by the morphism $\mathbf{1} \xrightarrow{\tau} \mathtt{int} \xrightarrow{send_{\mathtt{int}}} \mathtt{int} \xrightarrow{recv^{\mathtt{int}}_{\mathtt{int}}} \mathtt{int}$ in $\mathcal{S}$. This means that the term t executes internal effect τ, sends data of type $\mathtt{int}$, and then receives data of type $\mathtt{int}$. The term s is graded by the morphism $\mathbf{1} \xrightarrow{recv^{\mathbf{1}}_{\mathtt{int}}} \mathtt{int} \xrightarrow{send_{\mathtt{int}}} \mathtt{int}$. This means that the term s receives data of type $\mathtt{int}$ and then sends data of type $\mathtt{int}$. We can know from grading morphisms of t and s that they can interact with each other.

3.2 Equations

Next, we introduce equations to represent the equational theory on terms. The equation is defined as a pair of terms as in the non-graded case. However, the pairs of terms must have the same grading morphism.

Definition 3.6 [Equations and category-graded algebraic theory] A graded family of *equations* for Σ is a family of sets $E = (E_f)_f$ where E_f is a set of pairs of terms in $\mathrm{Term}_\Sigma(f, X)$. We write $t = s$ for a pair $(t, s) \in E_f$. An *$\mathcal{S}$-graded algebraic theory* is a pair $\mathcal{T} = (\Sigma, E)$ of $\mathcal{S}$-graded signature Σ and equations E for Σ.

Definition 3.7 [Model for category-graded algebraic theory] Let $\mathcal{T} = (\Sigma, E)$ be an $\mathcal{S}$-graded algebraic theory and a be an object of $\mathcal{S}$. A model for $\mathcal{T}$ at a is a Σ-model I at a that satisfies $|t|^I = |s|^I$ for each morphism $f\colon c \to b$ in $\mathcal{S}$ and equation $t = s \in E_f$. We denote the category of models for $\mathcal{T}$ at a by $\mathrm{Mod}^{\mathcal{T}}_a(\mathcal{C})$.

3.3 Free Models and Adjunctions

We explain free models of a category-graded algebraic theory and its universal property.

Definition 3.8 Let $\mathcal{T} = (\Sigma, E)$ be an $\mathcal{S}$-graded algebraic theory. We define a functor $F^{\mathcal{T}}_a\colon \mathbf{Set} \to \mathrm{Mod}^{\mathcal{T}}_a(\mathbf{Set})$ by $(F^{\mathcal{T}}_a X)(k) = \mathrm{Term}_\Sigma(k, X)/\sim$ for $k\colon b \to a$ in $\mathcal{S}$ and $|\sigma|^{F^{\mathcal{T}}_a X}_k(p, \{[t_i]\}_{i \in A}) = [\mathbf{do}(\sigma, p, \lambda i.t_i)]$ for each $X \in \mathbf{Set}$, $k\colon b \to a$ in $\mathcal{S}$ and $\sigma\colon P \rightsquigarrow A; f$ where $\sim$ is the equivalence relation induced by the equations E and $[t]$ is the equivalence class of t. We also define a map $\eta_X \colon X \to (F_a^{\mathcal{T}} X)(\mathrm{id}_a)$ by $\eta_X(x) = [\mathbf{e}(a, x)] \in \mathrm{Term}_\Sigma(\mathrm{id}_a, X)/\sim$.

We can show that the model $F_a^{\mathcal{T}} X$ with η_X has the universal property of a free model.

Proposition 3.9 *Let $\mathcal{T} = (\Sigma, E)$ be an $\mathcal{S}$-graded algebraic theory. Given a model A in* **Set** *for $\mathcal{T}$ at a and a map $\phi \colon X \to A\mathrm{id}_a$, there exists a unique homomorphism $\overline{\phi} \colon F_a^{\mathcal{T}} X \to A$ such that $\overline{\phi}_{\mathrm{id}_a} \circ \eta_X = \phi$.*

$$\begin{tikzcd} X \arrow[r, "\eta_X"] \arrow[dr, "\phi"'] & (F_a^{\mathcal{T}} X)(\mathrm{id}_a) \arrow[d, "\overline{\phi}_{\mathrm{id}_a}"] & F_a^{\mathcal{T}} X \arrow[d, dashed, "\overline{\phi}"] \\ & A\mathrm{id}_a & A \end{tikzcd}$$

Proof. For each $f \colon b \to a$, we define a map $\hat{\phi}_f \colon \mathrm{Term}_\Sigma(f, X) \to Af$ from the set of Σ-terms to Af recursively by: $\hat{\phi}_{\mathrm{id}_a}(\mathbf{e}(a, x)) = \phi(x)$ and $\hat{\phi}_f(\mathbf{do}(\sigma, p, \lambda i.t_i)) = |\sigma|^A(p, \lambda i.\hat{\phi}(t_i))$. Since A is a model for $\mathcal{T}$, all equations in E holds in A. Therefore, the map $\overline{\phi}([t]) := [\hat{\phi}(t)]$ is well-defined. We can show $\overline{\phi}_{\mathrm{id}_a} \circ \eta_X = \phi$ by definition. □

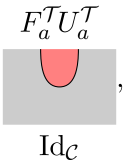
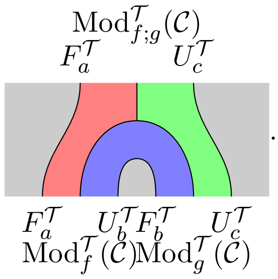
The forgetful functor for models $U_a^{\mathcal{T}} \colon \mathrm{Mod}_a^{\mathcal{T}}(\mathbf{Set}) \to \mathbf{Set}$ is the evaluation at id_a, that is $U_a^{\mathcal{T}}(A) = A_{\mathrm{id}_a}$. By the universality of the free construction, we have isomorphisms $\mathrm{Mod}_a^{\mathcal{T}}(\mathbf{Set})(F_a^{\mathcal{T}} X, A) \cong \mathcal{C}(X, U_a^{\mathcal{T}} A)$ for each $a \in \mathrm{Ob}\,\mathcal{S}$. Thus, we have adjunctions $F_a^{\mathcal{T}} \dashv U_a^{\mathcal{T}}$.

$$\begin{tikzcd} \mathcal{C} \arrow[r, bend left, "F_a^{\mathcal{T}}"] \arrow[r, phantom, "\perp"] & \mathrm{Mod}_a^{\mathcal{T}}(\mathbf{Set}) \arrow[l, bend left, "U_a^{\mathcal{T}}"] \end{tikzcd} .$$

We can define $\mathrm{Mod}_k^{\mathcal{T}}(\mathbf{Set}) \colon \mathrm{Mod}_a^{\mathcal{T}}(\mathbf{Set}) \to \mathrm{Mod}_b^{\mathcal{T}}(\mathbf{Set})$ for a morphism $k \colon b \to a$ in $\mathcal{S}$ to make $\mathrm{Mod}_{-}^{\mathcal{T}}(\mathbf{Set})$ a functor $\mathcal{S}^{\mathrm{op}} \to \mathbf{CAT}$. For a model A in $\mathrm{Mod}_a^{\mathcal{T}}(\mathbf{Set})$, a map $\mathrm{Mod}_k^{\mathcal{T}}(\mathbf{Set})A \colon \prod_c \mathcal{S}(c, b) \to \mathbf{Set}$ is defined as $(\mathrm{Mod}_k^{\mathcal{T}}(\mathbf{Set})A)(f) = A(f; k)$. Interpretation of operations $|-|^{\mathrm{Mod}_k^{\mathcal{T}}(\mathbf{Set})A}$ is defined as $|\sigma|_f^{\mathrm{Mod}_k^{\mathcal{T}}(\mathbf{Set})A} = |\sigma|_{f;k}^A$. It is easy to check that $\mathrm{Mod}_k^{\mathcal{T}}(\mathbf{Set})$ is a functor from $\mathrm{Mod}_a^{\mathcal{T}}(\mathbf{Set})$ to $\mathrm{Mod}_b^{\mathcal{T}}(\mathbf{Set})$.

More generally, we can define a category of models $\mathrm{Mod}_a^{\mathcal{T}}(\mathcal{C})$ for a category $\mathcal{C}$ in the same way as $\mathcal{C} = \mathbf{Set}$, and we can apply the same argument as above if the left adjoint of forgetful functor exists.

Applying Corollary 2.6, we obtain an $\mathcal{S}$-graded monad $\mathcal{S}^{\mathrm{op}} \to \mathrm{Endo}(\mathcal{C})$. We denote the category-graded monad by $T^{\mathcal{T}}$. The unit and multiplication of $T^{\mathcal{T}}$ are depicted as follows.

$\eta_a^{T^{\mathcal{T}}} =$ 
$F_a^{\mathcal{T}} U_a^{\mathcal{T}}$
$\mathrm{Id}_{\mathcal{C}}$
, $\mu_{g,f}^{T^{\mathcal{T}}} =$ 
$\mathrm{Mod}_{f;g}^{\mathcal{T}}(\mathcal{C})$
$F_a^{\mathcal{T}}$
$U_c^{\mathcal{T}}$
$F_a^{\mathcal{T}}$
$U_b^{\mathcal{T}} F_b^{\mathcal{T}}$
$U_c^{\mathcal{T}}$
$\mathrm{Mod}_f^{\mathcal{T}}(\mathcal{C})\mathrm{Mod}_g^{\mathcal{T}}(\mathcal{C})$
.

4 A Category-Graded Effect System

In this section, we introduce an effect system with category-graded operations based on category-graded algebraic theories. We call the effect system *CatEff*. We also construct handlers of category-graded algebraic operations. We fix small categories $\mathcal{S}$ and $\mathcal{S}'$, and $\mathcal{S}$, $\mathcal{S}'$-signatures Σ, Σ'.

4.1 Language

Syntax. Our effect system *CatEff* is based on fine-grained call-by-value calculus [10]. The syntax is divided into two parts, values and computations.

$$\textbf{Values}\quad V, W ::= x \mid \star \mid \mathbf{inl}\, V \mid \mathbf{inr}\, V \mid \langle V, W\rangle \mid \lambda^f x : A.M$$

$$\begin{aligned}\textbf{Computations}\quad M, N ::= {}& \mathbf{val}_a\, V \mid \mathbf{let}\, x \leftarrow M\, \mathbf{in}\, N \mid VW \mid \mathrm{op}(V)\\ &\mid \mathbf{proj}\, V\, \mathbf{as}\, \langle x, y\rangle.M \mid \mathbf{match}\, V\{x.M_1; y.M_2\}\end{aligned}$$

where a is an object in $\mathcal{S}$, and f is a morphism in $\mathcal{S}$. Values are usual except for lambda abstraction. The lambda abstraction $\lambda^f x : A.M$ means that this function has effects represented by the morphism f. We sometimes write V_Σ and M_Σ to specify its signature.

Types. Types are defined by the following BNF:

$$P, Q ::= \mathbf{1} \mid P \times Q \mid P + Q, \quad A, B ::= \mathbf{1} \mid A \times B \mid A + B \mid A \to B; a \xrightarrow{f} b.$$

where $f\colon a \to b$ is a morphism in $\mathcal{S}$. The key idea is that a function type $A \to B; a \xrightarrow{f} b$ indicates that a function of this type consumes an argument of type A and returns a result of type B with effects represented by f. We call P a *primitive type*. We assume that parameters and arities of operations in the signatures Σ and Σ' are primitive types.

Typing rules. There are value judgements of the form $\Gamma \vdash V : A$ and computation judgements of the form $\Gamma \vdash^\Sigma_f M : A$ where Γ is a list of distinct variables with types and f is a morphism in $\mathcal{S}$. The judgement $\Gamma \vdash^\Sigma_f M : A$ means that the computation M returns a result of type A under type environment Γ and causes effects represented by f. We omit Σ in the judgement $\Gamma \vdash^\Sigma_f M : A$ if it is clear from the context. The typing rules for terms in *CatEff* are presented in Figure 1.

Values

$$\frac{}{\Gamma \vdash \star : \mathbf{1}}\ \textsc{Tv-Unit} \qquad \frac{x : A \in \Gamma}{\Gamma \vdash x : A}\ \textsc{Tv-Var} \qquad \frac{\Gamma, x : A \vdash_f M : B}{\Gamma \vdash \lambda^f x.M : A \to B; f}\ \textsc{Tv-Abs}$$

$$\frac{\Gamma \vdash V : A \quad \Gamma \vdash W : B}{\Gamma \vdash \langle V, W\rangle : A \times B}\ \textsc{Tv-Pair}$$

$$\frac{\Gamma \vdash V : A}{\Gamma \vdash \mathbf{inl}\, V : A + B}\ \textsc{Tv-InjL} \qquad \frac{\Gamma \vdash V : B}{\Gamma \vdash \mathbf{inr}\, V : A + B}\ \textsc{Tv-InjR}$$

Computations

$$\frac{\Gamma \vdash V : A}{\Gamma \vdash_{\mathrm{id}_a} \mathbf{val}_a\, V : A}\ \textsc{Tc-Val} \qquad \frac{\Gamma \vdash V : P \quad \mathrm{op}\colon P \rightsquigarrow Q; a \xrightarrow{f} b \in \Sigma}{\Gamma \vdash_f \mathrm{op}(V) : Q}\ \textsc{Tc-Op}$$

$$\frac{\Gamma \vdash_{f : a \to b} M : A \quad \Gamma, x : A \vdash_{g : b \to c} N : B}{\Gamma \vdash_{f;g} \mathbf{let}\, x \leftarrow M\, \mathbf{in}\, N : B}\ \textsc{Tc-Let} \qquad \frac{\Gamma \vdash V : A \to B; f \quad \Gamma \vdash W : A}{\Gamma \vdash_f VW : B}\ \textsc{Tc-App}$$

$$\frac{\Gamma \vdash V : A_1 \times A_2 \quad \Gamma, x : A_1, y : A_2 \vdash_f M : B}{\Gamma \vdash_f \mathbf{proj}\, V\, \mathbf{as}\, \langle x, y\rangle.M : B}\ \textsc{Tc-Proj}$$

$$\frac{\Gamma \vdash V : A_1 + A_2 \quad \Gamma, x : A_1 \vdash_f M_1 : B \quad \Gamma, y : A_2 \vdash_f M_2 : B}{\Gamma \vdash_f \mathbf{match}\, V\{x.M_1; y.M_2\} : B}\ \textsc{Tc-Match}$$

Fig. 1. Typing rules.

4.2 Handlers

Handlers for ordinary algebraic theories [18] are homomorphisms from a free model for a theory to another one. We can also construct handlers for category-graded algebraic theories in a similar way to the ordinary handlers. Let $G\colon \mathcal{S} \to \mathcal{S}'$ be a functor and $\mathcal{T} = (\Sigma, E)$, $\mathcal{T}' = (\Sigma', E')$ be $\mathcal{S}$-, and $\mathcal{S}'$-graded algebraic theories, respectively. For an object a of $\mathcal{S}$ and sets X and Y, a handler from $F^{\mathcal{T}}_a X$ to $(F^{\mathcal{T}'}_{Ga} Y)(G-)$ is obtained by the universality of the free model $F^{\mathcal{T}}_a X$. To obtain the handler, $(F^{\mathcal{T}'}_{Ga} Y)(G-)$ must be a model for $\mathcal{T}$ at a. So we need the following data:

- a map $\phi\colon X \to (F^{\mathcal{T}'}_{Ga} Y)(G\mathrm{id}_a)$, and
- interpretations $|\sigma|^{(F^{\mathcal{T}'}_{Ga} Y)G}_k \colon P \times (F^{\mathcal{T}'}_{Ga} Y)(Gk)^Q \to (F^{\mathcal{T}'}_{Ga} Y)(G(f;k))$ of operations in Σ for every $\sigma\colon P \rightsquigarrow Q; c \xrightarrow{f} b \in \Sigma$ and $k\colon b \to a$.

satisfying all equations in E, that is the interpretations of terms in $\mathrm{Term}_\Sigma(f, X)$ induced by $|\sigma|^{(F^{\mathcal{T}'}_{Ga} Y)G}$ satisfies $|t|^{(F^{\mathcal{T}'}_{Ga} Y)G} = |s|^{(F^{\mathcal{T}'}_{Ga} Y)G}$ for every $t = s \in E$.

Together with the above data, $(F^{\mathcal{T}'}_{Ga} Y)(G-)$ becomes a model for $\mathcal{T}$ at a, and there is a homomorphism $\overline{\phi}\colon F^{\mathcal{T}}_a X \to (F^{\mathcal{T}'}_{Ga} Y)G$ such that $\overline{\phi}_{\mathrm{id}_a} \circ \eta_X = \phi$ by the universal property of free model. We can calculate the homomorphism $\overline{\phi} = \{\overline{\phi}_l\}_{l\colon b \to a}$ as $\overline{\phi}_{\mathrm{id}_a}([\mathbf{e}(a, v)]) = \phi(v)$ and $\overline{\phi}_{l\colon b \to a}([\mathbf{do}(\sigma, p, \lambda i.t_i)]) = |\sigma|^{(F^{\mathcal{T}'}_{Ga} Y)G}_k (p, \lambda i.\overline{\phi}_k([t_i]))$ where $t_i \in (F^{\mathcal{T}'}_{Ga} Y)(Gk)$.

We add syntax and typing rules for handlers as follows. Note that the following constructions of handlers don't care about equations of algebraic theories. Programmers must ensure on their responsibilities that handlers they are constructing respect proper equations of effects.

Additional syntax. To construct handlers for *CatEff*, we need data that corresponds to $\phi\colon X \to (F^{\mathcal{T}'}_{Ga} Y)(G\mathrm{id}_a)$ and $|\sigma|^{(F^{\mathcal{T}'}_{Ga} Y)G}_k \colon P \times (F^{\mathcal{T}'}_{Ga} Y)(Gk)^Q \to (F^{\mathcal{T}'}_{Ga} Y)(G(f;k))$ as argued above. Thus, we extend the syntax of *CatEff* as follows:

$$\begin{array}{lrl}
\textbf{Computations} & M_{\Sigma'} & ::= \cdots \mid \mathbf{handle}\, M_\Sigma \,\mathbf{with}\, H_{\Sigma \Rightarrow \Sigma'} \\
\textbf{Handlers} & H_{\Sigma \Rightarrow \Sigma'} & ::= \{\mathbf{val}_b\, x \mapsto M_{\Sigma'}\} \\
& & \quad \cup \{\mathrm{op}(p), r \mapsto_k (M_{\Sigma'})^k_{\mathrm{op}}\}^{k\colon c \to b}_{\mathrm{op}\colon P \rightsquigarrow Q; d \xrightarrow{g} c \in \Sigma}
\end{array}$$

Additional typing rules. We add a new judgement $\Gamma \vdash^G_b H : R \Rightarrow R'$ where R and R' are primitive types. This means that the handler H handles operations in computation of type R and then produces a computation of type R'. Let Δ be an environment $x_1 : P_1, \dots, x_n : P_n$.

$$\frac{\Delta \vdash^\Sigma_{f\colon a \to b} M : R \quad \Gamma \vdash^G_b H : R \Rightarrow R' \quad \Delta \subseteq \Gamma}{\Gamma \vdash^{\Sigma'}_{G(f)} \mathbf{handle}\, M \,\mathbf{with}\, H : R'}\ \textsc{Tc-Handle}$$

$$\frac{\begin{array}{c}\Gamma, x : R \vdash^{\Sigma'}_{\mathrm{id}_{Gb}} M : R' \\ \left\{\Gamma, p : P, r : Q \to R'; Gk \vdash^{\Sigma'}_{G(g;k)} M^k_{\mathrm{op}} : R'\right\}^{k\colon c \to b}_{\mathrm{op}\colon P \rightsquigarrow Q; d \xrightarrow{g} c \in \Sigma}\end{array}}{\Gamma \vdash^G_b \{\mathbf{val}_b\, x \mapsto M\} \cup \{\mathrm{op}(p), r \mapsto_k M^k_{\mathrm{op}}\}^k_{\mathrm{op} \in \Sigma} : R \Rightarrow R'}\ \textsc{Th-Handler}$$

Note that the environment Δ contains only variables typed by primitive types. If it contained a type $A \to B; f$, the morphism f is in $\mathcal{S}$ while the morphism must be in another grading category $\mathcal{S}'$ after a computation graded by f is handled. This is impossible in general.

We omit the morphism k in $\mathrm{op}(p), r \mapsto_k M^k_{\mathrm{op}}$ and write simply $\mathrm{op}(p), r \mapsto M_{\mathrm{op}}$ when $M_{\mathrm{op}} = M^k_{\mathrm{op}}$ for all k. This convention is used in Example 5.9.

$$\begin{array}{lrl}
\textsc{S-App} & (\lambda^f x.M)V & \to M[V/x] \\
\textsc{S-Let} & \mathbf{let}\, x \leftarrow \mathbf{val}_a\, V\, \mathbf{in}\, M & \to M[V/x] \\
\textsc{S-Proj} & \mathbf{proj}\, \langle V_1, V_2\rangle\, \mathbf{as}\, \langle x, y\rangle.M & \to M[V_1/x, V_2/y] \\
\textsc{S-MatchLeft} & \mathbf{match}\, (\mathbf{inl}\, V)\{x.M_1; y.M_2\} & \to M_1[V/x] \\
\textsc{S-MatchRight} & \mathbf{match}\, (\mathbf{inr}\, V)\{x.M_1; y.M_2\} & \to M_2[V/y] \\
\textsc{S-HandleRet} & \mathbf{handle}\, (\mathbf{val}_a\, V)\, \mathbf{with}\, H & \to M[V/x] \\
\textsc{S-HandleOp} & \mathbf{handle}\, \mathcal{E}[\mathrm{op}(V)]\, \mathbf{with}\, H & \to N \\
\textsc{S-Lift} & \mathcal{F}[M] & \to \mathcal{F}[M'] \quad \text{if } M \to M'
\end{array}$$

where $H = \{\mathbf{val}_a\, x \mapsto M\} \cup \{\mathrm{op}(p), r \mapsto_k M^k_{\mathrm{op}}\}^k_{\mathrm{op}}$ in S-HandleRet and S-HandleOp, and
$N = M^{k\colon c \to b}_{\mathrm{op}}[V/p, \lambda^{Gk} y.\, \mathbf{handle}\, \mathcal{E}[\mathbf{val}_b\, y]\, \mathbf{with}\, H/r]$ in S-HandleOp.

Fig. 2. Small-step operational semantics

5 Operational Semantics

To define the operational semantics of *CatEff*, we need some auxiliary notions.

Definition 5.1 [Evaluation context] Evaluation contexts $\mathcal{E}$ and $\mathcal{F}$ are defined by the following BNF:

$$\mathcal{E}_\Sigma ::= [\,] \mid \mathbf{let}\, x \leftarrow \mathcal{E}_\Sigma\, \mathbf{in}\, M_\Sigma$$

$$\mathcal{F}_{\Sigma'} ::= [\,] \mid \mathbf{let}\, x \leftarrow \mathcal{F}_{\Sigma'}\, \mathbf{in}\, M_{\Sigma'} \mid \mathbf{handle}\, \mathcal{F}_\Sigma\, \mathbf{with}\, H_{\Sigma \Rightarrow \Sigma'}$$

We write $\Gamma \vdash \mathcal{F} : A \rightsquigarrow B; G(b') \xrightarrow{f} a$ if $\Gamma, x : A \vdash_f \mathcal{F}[\mathbf{val}_{b'}\, x] : B$ is derived where G is a functor that corresponds to handlers in $\mathcal{F}$ and x does not appear in $\mathcal{F}$ as a free variable.

Figure 2 gives the small-step operational semantics for *CatEff*. It is based on [6]. The rules of operational semantics are usual except for S-HandleOp. In S-HandleOp, the grading morphism plays an important role. Let $\mathcal{E}$ be $\mathbf{let}\, x_n \leftarrow (\ldots(\mathbf{let}\, x_1 \leftarrow [\,]\, \mathbf{in}\, M_1)\ldots)\, \mathbf{in}\, M_n$ and f_i be the grading morphism of M_i for each $i = 1, \ldots, n$. Consider a term $\mathbf{handle}\, \mathcal{E}[\mathrm{op}(V)]\, \mathbf{with}\, H$. The handler H handles $\mathrm{op}(V)$ with the term $M^{f_1;\ldots;f_n}_{\mathrm{op}}$ in H. For example, see Example 5.8.

The goal of the rest of this section is to show the progress lemma and the preservation lemma for *CatEff*. We start by proving the following lemmas by induction on derivations.

Lemma 5.2 (Substitution) *If* $\Gamma, x_1 : A_1, \ldots, x_n : A_n \vdash_f M : B$ *and* $\Gamma \vdash V_i : A_i$ *for each* $i = 1, \ldots, n$, *then* $\Gamma \vdash_f M[V_1/x_1, \ldots, V_n/x_n] : B$.

Lemma 5.3 *If* $\Gamma \vdash \mathcal{F} : A \rightsquigarrow B; b \xrightarrow{f} a$, $\Gamma \vdash_{g'\colon c' \to b'} M : A$, *and* $G(b') = b$, *then* $\Gamma \vdash_{G(g');f} \mathcal{F}[M] : B$

Lemma 5.4 *If* $\Gamma \vdash_h \mathcal{F}[M] : B$, *then we have* $\Gamma \vdash_{g'} M : A$ *and* $\Gamma \vdash \mathcal{F} : A \rightsquigarrow B; f$ *satisfying* $h = G(g'); f$.

Lemma 5.5 (Progress) *If* $\vdash_{f\colon b \to a} M : A$ *then one of the following holds.*

(i) $f = \mathrm{id}_a$ *and* $M = \mathbf{val}_a\, V$ *for some value term* V,

(ii) $M = \mathcal{E}[op(V)]$ *for some* $\mathcal{E}$, op *and* V, *or*

(iii) *there exists a computation term* N *such that* $M \to N$.

Lemma 5.6 (Preservation) *If* $\vdash_{f\colon b \to a} M : A$ *and* $M \to N$ *then* $\vdash_f N : B$ *is derivable.*

Together with the progress lemma(Lemma 5.5) and preservation lemma(Lemma 5.6), we have a safety theorem. The safety theorem says that if a computation term is well-typed then the term comes from value ($\mathbf{val}_a\, V$) or is about to perform an operation.

Theorem 5.7 (Safety) *If $\vdash_{f\colon b\to a} M : A$ is a terminating term then there exists a value term V such that $M = \mathbf{val}_a V$, or M calls an operation, that is $M = \mathcal{E}[\mathrm{op}(V)]$ for some $\mathcal{E}$, $\mathrm{op} \in \Sigma$ and V.*

Example 5.8 [Handler] We present an example of small-step evaluation with handlers. Let $\mathcal{S}$ be a category such that $\mathrm{Ob}\,\mathcal{S} = \{c, d, e\}$ and morphisms are identities and $g\colon c \to d$ and $h\colon d \to e$, and $\mathcal{S}'$ be a category such that $\mathrm{Ob}\,\mathcal{S}' = \{\bullet\}$ and the identity is the only morphism. There is a unique functor $G\colon \mathcal{S} \to \mathcal{S}'$, which sends all objects in $\mathcal{S}$ to $\bullet$ in $\mathcal{S}'$. Let $\mathcal{S}$-signature Σ be $\{\mathrm{op}_1\colon P \rightsquigarrow A; g, \mathrm{op}_2\colon Q \rightsquigarrow B; h\}$ and $\mathcal{S}'$-signature Σ' be $\emptyset$. Given terms

$$\begin{gathered}
N = \mathbf{let}\, x \leftarrow \mathrm{op}_1(V)\,\mathbf{in}\,\mathbf{let}\, y \leftarrow \mathrm{op}_2(W)\,\mathbf{in}\,\mathbf{val}_e\langle x, y\rangle,\\
M^h_{\mathrm{op}_1} = rV', \quad M^{\mathrm{id}_e}_{\mathrm{op}_2} = rW',\\
H = \{\mathbf{val}_e\, z \mapsto \mathbf{val}_\bullet\, z;\ \mathrm{op}_1(p), r \mapsto_h M^h_{\mathrm{op}_1};\ \mathrm{op}_2(p), r \mapsto_{\mathrm{id}_e} M^{\mathrm{id}_e}_{\mathrm{op}_2}\}
\end{gathered}$$

such that $\vdash V : P$, $\vdash W : Q$, $\vdash V' : A$, $\vdash W' : B$, $\vdash^{\Sigma}_{g;h} N : A \times B$ and $\vdash^{G}_{e} H : A \times B \Rightarrow A \times B$. By the handler H, op_1 and op_2 are implemented as constant operations that always return values V' and W' for any arguments, respectively. By Tc-Handle, we have a judgement $\vdash^{\Sigma'}_{\mathrm{id}_\bullet} \mathbf{handle}\, N\, \mathbf{with}\, H : A \times B$. Therefore, the term $\mathbf{handle}\, N\, \mathbf{with}\, H$ is evaluated to the form $\mathbf{val}_\bullet U$ by the safety theorem. Indeed, $\mathbf{handle}\, N\, \mathbf{with}\, H$ is evaluated as follows:

$$\begin{aligned}
&\mathbf{handle}\, N\, \mathbf{with}\, H\\
&\to (\lambda^{Gh} v.\, \mathbf{handle}\,(\mathbf{let}\, x \leftarrow \mathbf{val}_d\, v\,\mathbf{in}\,\mathbf{let}\, y \leftarrow \mathrm{op}_2(W)\,\mathbf{in}\,\mathbf{val}_e\langle x, y\rangle)\,\mathbf{with}\, H)V'\\
&\to^* \mathbf{handle}\,(\mathbf{let}\, y \leftarrow \mathrm{op}_2(W)\,\mathbf{in}\,\mathbf{val}_e\langle V', y\rangle)\,\mathbf{with}\, H\\
&\to M^h_{\mathrm{op}_2}[W/p, \lambda^{G\mathrm{id}_e} w.\, \mathbf{handle}\,\mathbf{let}\, y \leftarrow \mathbf{val}_e\, w\,\mathbf{in}\,\mathbf{val}_e\langle V', y\rangle\,\mathbf{with}\, H/r]\\
&= (\lambda^{G\mathrm{id}_e} w.\, \mathbf{handle}\,\mathbf{let}\, y \leftarrow \mathbf{val}_e\, w\,\mathbf{in}\,\mathbf{val}_e\langle a, y\rangle\,\mathbf{with}\, H)W' \to^* \mathbf{val}_\bullet\langle V', W'\rangle.
\end{aligned}$$

We have $\vdash^{\Sigma'}_{\mathrm{id}_\bullet} \mathbf{val}_\bullet\langle V', W'\rangle : A \times B$.

Example 5.9 [Mutable store of mutable type with a plan] We can make a program with a mutable store of mutable type with a mutation plan. Let A and B be primitive types, V and W are value terms such that $\vdash V : A$ and $\vdash W : B$. Let $\mathcal{S}$ be a category such that $\mathrm{Ob}\,\mathcal{S} = \{\mathbf{1}, A, B\}$ and morphisms are generated by $f^\alpha_\beta\colon \alpha \to \beta$ for $\alpha, \beta \in \mathrm{Ob}\,\mathcal{S}$ and $\mathcal{S}'$ be a category such that $\mathrm{Ob}\,\mathcal{S}' = \{\mathbf{1} + (A + B)\}$ and morphism is only identity. There is a unique functor $G\colon \mathcal{S} \to \mathcal{S}'$, which sends all objects in $\mathcal{S}$ to $\mathbf{1} + (A + B)$ in $\mathcal{S}'$. Intuitively, objects are possible types of mutable store and morphisms are plans of mutations of types of the store. Let Σ be an $\mathcal{S}$-signature $\{\mathrm{update}^\alpha_\beta\}_{\alpha,\beta\in\mathrm{Ob}\,\mathcal{S}} \cup \{\mathrm{lookup}_\alpha\}_{\alpha\in\mathrm{Ob}\,\mathcal{S}}$ where the type of $\mathrm{update}^\alpha_\beta \in \Sigma$ is $\beta \rightsquigarrow \mathbf{1}; f^\alpha_\beta\colon \alpha \to \beta$ and the type of lookup_α is $\mathbf{1} \rightsquigarrow \alpha; \mathrm{id}_\alpha$ for each $\alpha, \beta \in \mathrm{Ob}\,\mathcal{S}$. Let Σ' be an $\mathcal{S}'$-signature $\{\mathrm{update}, \mathrm{lookup}\}$ where the type of update is $\mathbf{1} + (A + B) \rightsquigarrow \mathbf{1}; \mathrm{id}$ and the type of lookup is $\mathbf{1} \rightsquigarrow \mathbf{1} + (A + B); \mathrm{id}$.

We can implement the behavior of mutable types by the following handlers: $H_{\mathbf{1}} = \{\mathbf{val}_{\mathbf{1}}\, x \mapsto \mathbf{val}_{\mathbf{1}+(A+B)}\,\mathbf{inl}\, x\} \cup H_{\mathrm{op}}$, $H_A = \{\mathbf{val}_A\, x \mapsto \mathbf{val}_{\mathbf{1}+(A+B)}\,\mathbf{inr}(\mathbf{inl}\, x)\} \cup H_{\mathrm{op}}$ and $H_B = \{\mathbf{val}_B\, x \mapsto \mathbf{val}_{\mathbf{1}+(A+B)}\,\mathbf{inr}(\mathbf{inr}\, x)\} \cup H_{\mathrm{op}}$ where

$$\begin{aligned}
H_{\mathrm{op}} = \{&\mathrm{update}^{\mathbf{1}}_{\mathbf{1}}(\_), r \mapsto \mathbf{let}\, \_ \leftarrow \mathrm{update}(\mathbf{inl}\,\star)\,\mathbf{in}\, r\star\\
&\vdots\\
&\mathrm{update}^B_B(b), r \mapsto \mathbf{let}\, \_ \leftarrow \mathrm{update}(\mathbf{inr}(\mathbf{inr}\, b))\,\mathbf{in}\, r\star\\
&\mathrm{lookup}_{\mathbf{1}}(\_), r \mapsto \mathbf{let}\, x \leftarrow \mathrm{lookup}(\star)\,\mathbf{in}\\
&\qquad\qquad\mathbf{match}\, x\{\mathbf{inl}\, y.r\star; \mathbf{inr}\, y.\,\mathbf{match}\, y\{\mathbf{inl}\, a.r\star; \mathbf{inr}\, b.r\star\}\}\\
&\mathrm{lookup}_A(\_), r \mapsto \mathbf{let}\, x \leftarrow \mathrm{lookup}(\star)\,\mathbf{in}\\
&\qquad\qquad\mathbf{match}\, x\{\mathbf{inl}\, y.rV; \mathbf{inr}\, y.\,\mathbf{match}\, y\{\mathbf{inl}\, a.ra; \mathbf{inr}\, b.rV\}\}\\
&\mathrm{lookup}_B(\_), r \mapsto \mathbf{let}\, x \leftarrow \mathrm{lookup}(\star)\,\mathbf{in}\\
&\qquad\qquad\mathbf{match}\, x\{\mathbf{inl}\, y.rW; \mathbf{inr}\, y.\,\mathbf{match}\, y\{\mathbf{inl}\, a.rW; \mathbf{inr}\, b.rb\}\}
\end{aligned}$$

We have judgements $\vdash^G_\alpha H_\alpha : \alpha \Rightarrow \mathbf{1} + (A + B)$ for all $\alpha \in \mathrm{Ob}\,(\mathcal{S})$. Let us consider the term

$$N = \mathbf{let}\,\_ \leftarrow \mathrm{update}^1_A(V)\,\mathbf{in}\,\mathbf{let}\,\_ \leftarrow \mathrm{update}^A_B W\,\mathbf{in}\,\mathbf{let}\,x \leftarrow \mathrm{lookup}_B(\star)\,\mathbf{in}\,\mathbf{val}_B\,x$$

We have a judgement $\vdash^\Sigma_{f^{\mathbf{1}}_A; f^A_B} N : B$. The grading morphism $f^{\mathbf{1}}_A; f^A_B : \mathbf{1} \to A \to B$ implies that the program N stores a value of type A and then stores a value of type B. Handling operations in Σ by H_B, a mutable store of mutable type with a plan is transformed to a mutable store of fixed type $\mathbf{1} + (A + B)$. We have $\vdash^{\Sigma'}_{\mathrm{id}_{\mathbf{1}+(A+B)}} \mathbf{handle}\,N\,\mathbf{with}\,H_B : \mathbf{1} + (A + B)$.

6 Denotational Semantics

In this section, we give denotational semantics for *CatEff*. The denotational semantics is based on [2]. Let Σ, Σ' be $\mathcal{S}$- and $\mathcal{S}'$-signatures, respectively. For the sake of simplicity, we work in **Set**. To interpret computation terms that return a value in X, we use the sets $\mathrm{Term}_\Sigma(f \colon b \to a, X) = U_b \mathrm{Mod}^\Sigma_f(\mathbf{Set}) F_a X$ defined in section 3. We don't care about equations and so consider free models without any equations. Each computational term is interpreted by a term of a category-graded algebraic theory. We can think of elements of $\mathrm{Term}_\Sigma(f \colon b \to a, X)$ as trees whose leaves are labelled by elements of X, and internal nodes are labelled by operations and their arguments. For example, $\mathbf{e}(a, v) \in \mathrm{Term}_\Sigma(\mathrm{id}_a \colon a \to a, X)$ is a tree with only one node labelled by v. Note that, for any tree $t \in \mathrm{Term}_\Sigma(f, X)$, we obtain f by composing morphisms of label op of nodes in a path from the root to a leaf.

Definition 6.1 We define the interpretation of types $[\![A]\!] \in \mathbf{Set}$ as follows:

$$[\![\mathbf{1}]\!] = \{\star\}, \quad [\![A \to B; b \xrightarrow{f} a]\!] = \mathrm{Term}_\Sigma(f, [\![B]\!])^{[\![A]\!]},$$
$$[\![A + B]\!] = [\![A]\!] \sqcup [\![B]\!], \quad [\![A \times B]\!] = [\![A]\!] \times [\![B]\!].$$

For a context $\Gamma = x_1 : A_1, \ldots, x_n : A_n$, we define $[\![\Gamma]\!] = [\![A_1]\!] \times \cdots \times [\![A_n]\!]$.

We interpret value $\Gamma \vdash V : A$, computation $\Gamma \vdash_f M : A$ and handler $\Gamma \vdash^G_a H : R \Rightarrow R'$ as maps $[\![\Gamma \vdash V : A]\!] \colon [\![\Gamma]\!] \to [\![A]\!]$, $[\![\Gamma \vdash_f M : A]\!] \colon [\![\Gamma]\!] \to \mathrm{Term}_\Sigma(f, [\![A]\!])$ and $[\![\Gamma \vdash^G_a H : R \Rightarrow R']\!] \colon [\![\Gamma]\!] \to \mathrm{Term}_{\Sigma'}(Gf, [\![R']\!])^{\mathrm{Term}_\Sigma(f, [\![R]\!])}$, respectively. We write the lift of a morphism $\phi \colon X \to \mathrm{Term}_\Sigma(g, Y)$ as $\phi^{\dagger_f} := \mu_{f,g}(\mathrm{Term}_\Sigma(f, \phi))$ and often simply write $\phi^\dagger$ when f is clear from the context.

Definition 6.2 Let Γ be a typing context. Given $s \in [\![\Gamma]\!]$, we define the interpretation of terms as follows:

$$[\![\Gamma \vdash \star : \mathbf{1}]\!]s = \star, \quad [\![\Gamma \vdash \langle V_1, V_2 \rangle : A \times B]\!]s = \langle [\![V_1]\!]s, [\![V_2]\!]s \rangle,$$
$$[\![\Gamma \vdash \mathbf{inl}\,V : A + B]\!]s = \mathrm{in}_1\,[\![V]\!]s, \quad [\![\Gamma \vdash \mathbf{inr}\,V : A + B]\!]s = \mathrm{in}_2\,[\![V]\!]s,$$
$$[\![\Gamma, x : A \vdash x : A]\!]s = \pi_x(s), \quad [\![\Gamma \vdash \lambda^f x.M : A \to B; f]\!]s = [\![M]\!](s, -),$$

$$[\![\Gamma \vdash_{\mathrm{id}_a} \mathbf{val}_a\,V : A]\!]s = \mathbf{e}(a, [\![V]\!]s), \quad [\![\Gamma \vdash_f VW : B]\!]s = ([\![V]\!]s)([\![W]\!]s),$$
$$[\![\Gamma \vdash_f \mathrm{op}(V) : A]\!]s = \mathbf{do}(\mathrm{op}, [\![V]\!]s, \{\mathbf{e}(a, x)\}_{x \in [\![A]\!]}),$$
$$[\![\Gamma \vdash_{f;g} \mathbf{let}\,x \leftarrow M\,\mathbf{in}\,N : B]\!]s = ([\![N]\!](s, -))^\dagger([\![M]\!]s),$$
$$[\![\Gamma \vdash_f \mathbf{proj}\,V\,\mathbf{as}\,\langle x, y \rangle.M : B]\!]s = [\![M]\!](s, \pi_1([\![V]\!]s), \pi_2([\![V]\!]s)),$$
$$[\![\Gamma \vdash_f \mathbf{match}\,V\{x.y; M_1.M_2\} : B]\!]s = [[\![M_1]\!](s, -), [\![M_2]\!](s, -)]([\![V]\!]s),$$
$$[\![\Gamma \vdash^{\Sigma'}_{Gf} \mathbf{handle}\,M\,\mathbf{with}\,H : R']\!]s = ([\![H]\!]s)([\![M]\!]s),$$

$$[\![\Gamma \vdash^G_a H : R \Rightarrow R']\!]s = \begin{cases} \mathbf{e}(a, v) & \mapsto [\![M]\!](s, v) \\ \mathbf{do}(\mathrm{op}, p, \{t_i\}_i) & \mapsto [\![M^k_{\mathrm{op}}]\!](s, p, \{[\![H]\!](s, t_i)\}_i) \end{cases}$$

where $H = \{\mathbf{val}_a\,x \mapsto M\} \cup \{\mathrm{op}(p), r \mapsto M^k_{\mathrm{op}}\}^k_{\mathrm{op} \in \Sigma}$.

7 Soundness and Adequacy

We show soundness and adequacy. These theorems assert that operational semantics and denotational semantics are compatible with each other.

Theorem 7.1 (Soundness) *If* $\vdash_f M : A$ *and* $M \to M'$ *then* $[\![\vdash_f M : A]\!] = [\![\vdash_f M' : A]\!]$.

To show adequacy(Theorem 7.5), we define relations $\lhd_A$ for values and $\lhd_A^f$ for computations as done in [2].

Definition 7.2 For $v \in [\![A]\!]$ and a closed value term $\vdash V : A$, we define $v \lhd_A V$ as follows:

- $v \lhd_{\mathbf{1}} V$ if $v = \mathbf{1}$ and $V = \star$.
- $v \lhd_{A\times B} V$ if $v = \langle v_1, v_2\rangle$, $V = \langle V_1, V_2\rangle$, $v_1 \lhd_A V_1$ and $v_2 \lhd_B V_2$.
- $v \lhd_{A+B} V$ if either $v = \mathrm{in}_1\, v_1$, $V = \mathbf{inl}\, V_1$ and $v_1 \lhd_A V_1$, or $v = \mathrm{in}_2\, v_2$, $V = \mathbf{inr}\, V_2$ and $v_2 \lhd_B V_2$.
- $v \lhd_{A\to B;f} V$ if $v(w) \lhd_B^f VW$ for each $w \in [\![A]\!]$ and closed value $\vdash W : A$ satisfying $w \lhd_A W$.

Simultaneously, for $c \in \mathrm{Term}_\Sigma(f, [\![A]\!])$ and a closed computation term $\vdash_f M : A$, $c \lhd_A^f M$ holds if

(i) $f = \mathrm{id}_a$, $c = \mathrm{e}(a, v)$, $M \to^* \mathbf{val}_a\, V$ and $v \lhd_A V$, or

(ii) $c = \mathbf{do}(\sigma, v, \{t_x\}_{x\in[\![C]\!]})$, $M \to^* \mathcal{E}[\mathrm{op}(V)]$, $v \lhd_P V$, and if $w \lhd_C W$ then $t_w \lhd_A^k \mathcal{E}[\mathbf{val}_b\, W]$.

Lemma 7.3 *If* $c \lhd_A^f M'$ *and* $M \to M'$ *then* $c \lhd_A^f M$.

Proof. By assumption $c \lhd_A^f M'$, we have two cases:

(i) $f = \mathrm{id}_a$, $c = \mathrm{e}(a, v)$, $M' \to^* \mathbf{val}_a\, V$ and $v \lhd_A V$. In this case, we have $M \to^* \mathbf{val}_a\, V$.

(ii) $c = \mathbf{do}(\sigma, v, \{t_x\}_{x\in[\![C]\!]})$, $M' \to^* \mathcal{E}[\mathrm{op}(V)]$, $v \lhd_P V$, and if $w \lhd_C W$ then $t_w \lhd_A^k \mathcal{E}[\mathbf{val}_b\, W]$. In this case, we have $M \to^* \mathcal{E}[\mathrm{op}(V)]$.

In both cases, we can conclude $c \lhd_A^f M$ as required. □

Lemma 7.4 *Let* Γ *be a typing context* $x_1 : A_1, \ldots, x_n : A_n$, *and* $\vdash W_i : A_i$ *be a closed value term and* w_i *be an element of* $[\![A_i]\!]$ *with* $w_i \lhd_{A_i} W_i$ *for each* $1 \le i \le n$. *Then followings hold.*

(i) *If* $\Gamma \vdash V : A$ *then* $[\![V]\!](w_1, \ldots, w_n) \lhd_A V[W_1/x_1, \ldots, W_n/x_n]$.

(ii) *If* $\Gamma \vdash_f M : A$ *then* $[\![M]\!](w_1, \ldots, w_n) \lhd_A^f M[W_1/x_1, \ldots, W_n/x_n]$.

Theorem 7.5 (Adequacy) *If* $\vdash_{\mathrm{id}_a} M : \mathbf{1}$ *and* $[\![\vdash_{\mathrm{id}_a} M : \mathbf{1}]\!] = \mathbf{e}(a, \star)$ *then* $M \to^* \mathbf{val}_a\, \star$.

Proof. By Lemma 7.4, we have $[\![M]\!] \lhd_{\mathbf{1}}^{\mathrm{id}_a} M$. Thus, we have $\mathbf{e}(a, \star) \lhd_{\mathbf{1}}^{\mathrm{id}_a} M$ by assumption. By the definition of $\lhd_{\mathbf{1}}^{\mathrm{id}_a}$, we obtain $M \to^* \mathbf{val}_a\, V$ and $\star \lhd_{\mathbf{1}} V$. Therefore, by definition of $\lhd_{\mathbf{1}}$, we conclude $V = \star$ and $M \to^* \mathbf{val}_a\, \star$. □

8 Future Work

User-defined grading category and graded operations. The grading categories of *CatEff* are considered to be built-in in this paper. To provide user-defined grading operations, the syntax to write grading categories and graded operations is needed. This is future work.

Category-graded Lawvere theories. Graded Lawvere theories which correspond to graded algebraic theories were developed in [9]. Category-graded extensions of Lawvere theories are future work.

2-category-graded monads. Graded monads are graded by partially ordered monoids in general. We must consider *2-category-graded monads*, whose grading category is 2-category, to generalise partially ordered monoid-graded monads. This situation is beyond [21]. Thus, the Eilenberg-Moore and Kleisli constructions on these monads are not trivial.

References

[1] Atkey, R., *Algebras for parameterised monads*, in: A. Kurz, M. Lenisa and A. Tarlecki, editors, *CALCO 2009*, Lecture Notes in Computer Science **5728** (2009), pp. 3–17.

[2] Bauer, A. and M. Pretnar, *An effect system for algebraic effects and handlers*, Log. Methods Comput. Sci. **10** (2014).

[3] Bénabou, J., *Introduction to bicategories*, in: *Reports of the Midwest Category Seminar*, Lecture Notes in Mathematics **47** (1967), pp. 1–77.

[4] Dorsch, U., S. Milius and L. Schröder, *Graded monads and graded logics for the linear time - branching time spectrum*, in: W. J. Fokkink and R. van Glabbeek, editors, *CONCUR 2019*, LIPIcs **140** (2019), pp. 36:1–36:16.

[5] Fujii, S., S. Katsumata and P. Melliès, *Towards a formal theory of graded monads*, in: B. Jacobs and C. Löding, editors, *FoSSaCS 2016*, Lecture Notes in Computer Science **9634** (2016), pp. 513–530.

[6] Hillerström, D. and S. Lindley, *Shallow effect handlers*, in: S. Ryu, editor, *APLAS 2018*, Lecture Notes in Computer Science **11275** (2018), pp. 415–435.

[7] Hyland, M. and J. Power, *The category theoretic understanding of universal algebra: Lawvere theories and monads*, Electron. Notes Theor. Comput. Sci. **172** (2007), pp. 437–458.

[8] Katsumata, S., *Parametric effect monads and semantics of effect systems*, in: S. Jagannathan and P. Sewell, editors, *POPL '14* (2014), pp. 633–646.

[9] Kura, S., *Graded algebraic theories*, in: J. Goubault-Larrecq and B. König, editors, *FoSSaCS 2020*, Lecture Notes in Computer Science **12077** (2020), pp. 401–421.

[10] Levy, P. B., J. Power and H. Thielecke, *Modelling environments in call-by-value programming languages*, Inf. Comput. **185** (2003), p. 182–210.

[11] MacLane, S., "Categories for the Working Mathematician," Springer-Verlag, New York, 1971, ix+262 pp., graduate Texts in Mathematics, Vol. 5.

[12] Melliès, P., *Parametric monads and enriched adjunction*, LOLA 2012. Manuscript available on the author's web page.

[13] Melliès, P., *The parametric continuation monad*, Math. Struct. Comput. Sci. **27** (2017), pp. 651–680.

[14] Moggi, E., *Notions of computation and monads*, Inf. Comput. **93** (1991), pp. 55–92.

[15] Orchard, D., P. Wadler and H. Eades, III, *Unifying graded and parameterised monads*, in: M. S. New and S. Lindley, editors, *MSFP 2020*, EPTCS **317**, 2020, pp. 18–38.

[16] Orchard, D. A., T. Petricek and A. Mycroft, *The semantic marriage of monads and effects*, CoRR **abs/1401.5391** (2014). URL http://arxiv.org/abs/1401.5391

[17] Plotkin, G. D. and J. Power, *Adequacy for algebraic effects*, in: F. Honsell and M. Miculan, editors, *FoSSaCS 2001*, Lecture Notes in Computer Science **2030** (2001), pp. 1–24.

[18] Plotkin, G. D. and M. Pretnar, *Handlers of algebraic effects*, in: G. Castagna, editor, *ESOP 2009*, Lecture Notes in Computer Science **5502** (2009), pp. 80–94.

[19] Selinger, P., *A survey of graphical languages for monoidal categories*, in: B. Coecke, editor, *New Structures for Physics*, Lecture Notes in Physics **813**, Springer, 2010 pp. 289–355.

[20] Smirnov, A. L., *Graded monads and rings of polynomials*, J. Math. Sci. **151** (2008), pp. 3032–3051.

[21] Street, R., *Two constructions on lax functors*, Cahiers de topologie et géométrie différentielle catégoriques **13** (1972), pp. 217–264.

A Generalised Units and Generalised Counits

Generalised units were introduced to unify category-graded monads and parameterised monads in [15]. In this section, we introduce *generalised counits of adjunctions* that corresponds to generalised units of monads and investigate the role of generalised units in *CatEff*.

A.1 Generalised Units of Category-Graded Monads

First, we review the definition of categroy-graded monads with generalised units.

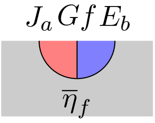
Definition A.1 [Category-graded monads with generalised unit, [15]] A *category-graded monads with generalised unit* is a category-graded monad $(T\colon \mathcal{S}^{\mathrm{op}} \xrightarrow{\text{lax}} \mathrm{Endo}(\mathcal{C}), \eta, \mu)$ together with the following data:

- A *wide subcategory* $\mathcal{R}$ of $\mathcal{S}$, that is a subcategory $\mathcal{R} \subseteq \mathcal{S}$ satisfying $\mathrm{Ob}\,\mathcal{R} = \mathrm{Ob}\,\mathcal{S}$. We denote the inclusion functor by $\iota : \mathcal{R} \hookrightarrow \mathcal{S}$ and usually omit ι when no confusion arises.
- For each morphism f in $\mathcal{R}$ and object C in $\mathcal{C}$, a morphism $(\overline{\eta}_f)_C\colon C \to T_f C$, satisfying the following commutative diagrams.

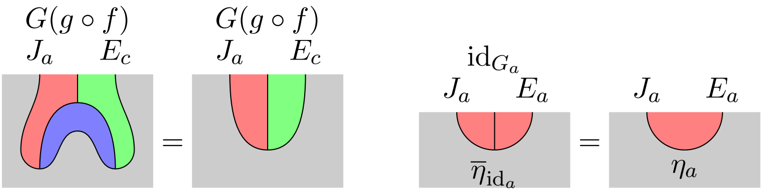
$$\begin{tikzcd}
C \arrow[r, "(\overline{\eta}_f)_C"] \arrow[d, "(\overline{\eta}_{g\circ f})_C"'] & T_f C \arrow[d, "T_f(\overline{\eta}_g)_C"] \\
T_{g\circ f} C & T_f T_g C \arrow[l, "\mu_{f,g}"]
\end{tikzcd}
\qquad
\begin{tikzcd}
C \arrow[r, "(\eta_a)_C"] \arrow[dr, "(\overline{\eta}_{\mathrm{id}_a})_C"'] & T_{\mathrm{id}_a} C \arrow[d, equal] \\
 & T_{\mathrm{id}_a} C
\end{tikzcd}$$

Let $G\colon \mathcal{S}^{\mathrm{op}} \to \mathbf{CAT}$ be a functor and $(J_a\colon \mathcal{C} \to Ga, Ea\colon Ga \to \mathcal{C}, \eta_a, \varepsilon_a)$ be adjunctions for each $a \in \mathrm{Ob}\,\mathcal{S}$ such that $T_f = E_b Gf Ja$ for every $f\colon a \to b$ in $\mathcal{S}$. We depict the generalised units by string diagrams as follows.

$$\begin{tikzcd}
\mathcal{C} \arrow[r, "J_a"] \arrow[dd, "\mathrm{Id}_{\mathcal{C}}"'] & Ga \arrow[dd, "Gf"] \\
{} \arrow[r, Rightarrow, "\overline{\eta}_f"] & {} \\
\mathcal{C} & Gb \arrow[l, "E_b"]
\end{tikzcd}
=$$



Then, the above two rules are depicted as follows.



A category-graded monad with generalised unit whose wide subcategory is a discrete category is an ordinary parameterised monad. Every parameterised monad can be seen a category-graded monad with generalised units using *pair completion*, see [15] and Appendix C.

A.2 Generalised Counits of Adjunctions

We introduce *generalised counits* of adjunctions and show that generalised units of monads correspond to generalised counits of adjunctions.

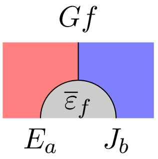
Definition A.2 [Generalised counits of adjunctions] Let $G\colon \mathcal{S}^{\mathrm{op}} \to \mathbf{CAT}$ be a functor and $(J_a\colon \mathcal{S} \to Ga, E_a\colon Ga \to \mathcal{S}, \eta_a, \varepsilon_a)$ be adjunctions. A *generalised counit* of those adjunctions consists of the following data.

- A wide subcategory $\mathcal{R}$ of $\mathcal{S}$.
- For each morphism $f\colon b \to a$ in $\mathcal{R}$, a natural transformation $\overline{\varepsilon}_f\colon J_b E_a \Rightarrow Gf$, satisfying the following commutative diagrams.

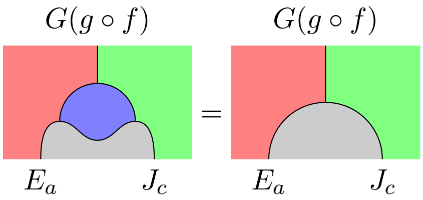
$$\begin{tikzcd}
J_c E_a A \arrow[r, "J_c(\eta_b)_{E_a A}"] \arrow[d, "(\overline{\varepsilon}_{f\circ g})_A"] & J_c E_b J_b E_a A \arrow[r, "J_c E_b(\overline{\varepsilon}_g)_A"] & J_c E_b Gg A \arrow[d, "(\overline{\varepsilon}_f)_{GgA}"] \\
G(f\circ g)A \arrow[rr, equal] & & Gf Gg A
\end{tikzcd}
\qquad
\begin{tikzcd}
J_a E_a A \arrow[r, "(\varepsilon_a)_A"] \arrow[dr, "(\overline{\varepsilon}_{\mathrm{id}_a})_A"'] & A \arrow[d, equal] \\
 & A
\end{tikzcd}$$

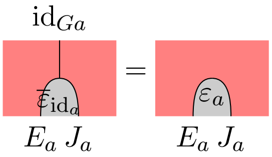
We depict generalised counits by string diagrams as follows:

$$\begin{tikzcd}
\mathcal{C} \arrow[d, "\mathrm{Id}_{\mathcal{C}}"'] & Ga \arrow[l, "E_a"'] \arrow[d, "Gf"] \arrow[dl, phantom, "\xRightarrow{\overline{\varepsilon}_f}"] \\
\mathcal{C} \arrow[r, "J_b"] & Gb
\end{tikzcd} =$$

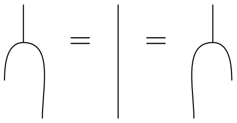


The following string diagrams are equivalent to the above diagrams:

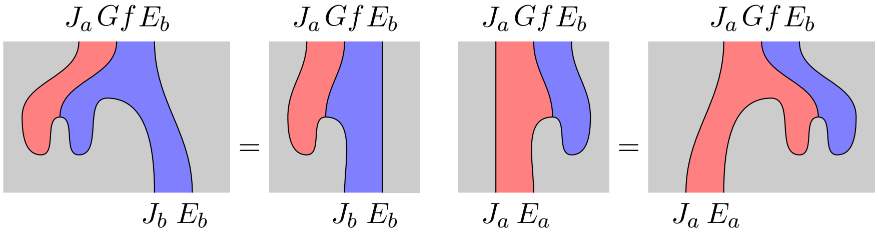




Next proposition is analogous to the usual unit law of monads:



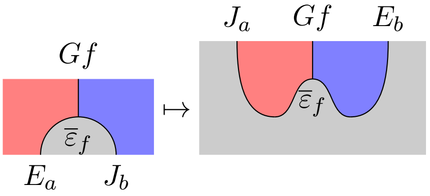
Proposition A.3 *The following equations hold.*



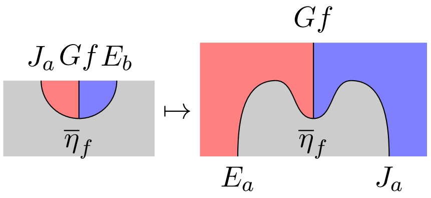
Proof. We can easily check by deformation of strings. □

Theorem A.4 *Let $G\colon \mathcal{S}^{\mathrm{op}} \to$ **CAT** be a functor and $(J_a\colon \mathcal{S} \to Ga, E_a\colon Ga \to \mathcal{S}, \eta_a, \varepsilon_a)$ be adjunctions. There is a one to one correspondence between generalised units of the category-graded monad induced by G and $(J_a, E_a, \eta_a, \varepsilon_a)$ and generalised counits of the adjunctions.*

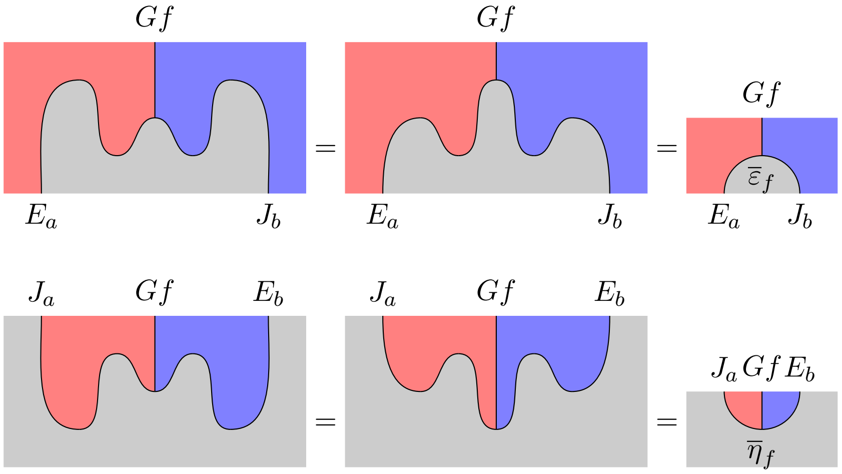
Proof. We can define a map from generalised counits to generalised units as follows.



Indeed we can check that the right-hand side satisfies the axioms of generalised units using the axioms generalised counits and deformation of strings. The inverse of this map is as follows.



The following equations show that the above two maps are inverse of each other.



□

A.3 *Generalised Units in CatEff*

We can introduce generalised units to *CatEff*. We fix a grading category $\mathcal{S}$ and wide subcategory $\mathcal{R}$ of it. The typing rule corresponding to generalised units is as follows.

$$\frac{g\colon a' \to a \text{ in } \mathcal{R} \quad \Gamma \vdash_{f\colon a \to b} M : A \quad h\colon b \to b' \text{ in } \mathcal{R}}{\Gamma \vdash_{h \circ f \circ g} M : A}\ \textsc{Tc-GUnit}$$

If we think of objects in grading category as conditions of states, the rule Tc-GUnit represents weakening of the condition along the grading morphism. For example, if objects in the grading category are types of states, generalised units represent the subtyping relation of the types.

The denotation of a judgement derived by Tc-GUnit is defined as follows.

$$[\![\Gamma \vdash_{h \circ f \circ g} M : A]\!]s = \mu_{f \circ g, h}(\mathrm{Term}_\Sigma(f \circ g, \overline{\eta}_h)(\mu_{g,f}(\overline{\eta}_g([\![\Gamma \vdash_f M : A]\!]s))))$$

B Recursion

In this section, we add recursion to *CatEff*. We have to take care of grading morphisms when we construct recursive functions.

We add value terms to construct recursive functions.

$$\textbf{Values} \quad V ::= \ldots \mid \mathbf{rec}^{k\colon a \to b}\, f(x : A).M$$

Typing rule for recursive functions is given as follows:

$$\frac{\Gamma, f : A \to B; k, x : A \vdash_k M : B}{\Gamma \vdash \mathbf{rec}^k\, f(x : A).M : A \to B; k}\ \textsc{Tv-Rec}$$

The key point is that the grading morphisms of f and M are the same in the premise of Tv-Rec.

Reduction of application of recursive functions is defined as follows:

$$\text{S-RecApp} \quad (\mathbf{rec}^k\, f(x : A).M)W \to M[W/x, \mathbf{rec}^k\, f(x : A).M/f]$$

We can show the safety theorem for this extension.

To give denotational semantics, we need domain theoretical construction to interpret recursively defined function. We believe that we can construct such domains as done in [2] and show soundness and adequacy for *CatEff* with recursion.

C Correspondence of Parameterised Monads to Category-Graded Monads with Generalised Units

In this section, we describe correspondence of parameterised monads to category-graded monads with generalised units. This correspondence was shown by [15], but it needs some modification.

Definition C.1 [[1]] A *parameterised monad* consists of the following data:

- a functor $P \colon \mathcal{S}^{\mathrm{op}} \times \mathcal{S} \to [\mathcal{C}, \mathcal{C}]$,
- for each $a \in \mathrm{Ob}\, \mathcal{S}$, a natural transformation $\eta^P_a \colon \mathrm{Id}_{\mathcal{C}} \Rightarrow P(a, a)$, and
- for each $a, b, c \in \mathrm{Ob}\, \mathcal{S}$, a natural transformation $\mu^P_{a,b,c} \colon P(a, b)P(b, c) \Rightarrow P(a, c)$

satisfying the following commutative diagrams:

$$\begin{tikzcd}
P(a,b) \arrow[r, Rightarrow, "\eta^P_a P(a,b)"] \arrow[d, Rightarrow, "P(a,b)\eta^P_b"'] \arrow[dr, equal] & P(a,a)P(a,b) \arrow[d, Rightarrow, "\mu^P_{a,a,b}"] \\
P(a,b)P(b,b) \arrow[r, Rightarrow, "\mu^P_{a,b,b}"'] & P(a,a)
\end{tikzcd}
\qquad
\begin{tikzcd}
P(a,b)P(b,c)P(c,d) \arrow[r, "\mu^P_{a,b,c}P(c,d)"] \arrow[d, "P(a,b)\mu^P_{b,c,d}"'] & P(a,c)P(c,d) \arrow[d, "\mu^P_{a,c,d}"] \\
P(a,b)P(b,d) \arrow[r, "\mu^P_{a,b,d}"'] & P(a,d)
\end{tikzcd}$$

and η^P_a is dinatural in a and $\mu^P_{a,b,c}$ is dinatural in b and natural in a, c, that is satisfying the following commutative diagrams:

$$\begin{tikzcd}
\mathrm{Id}_{\mathcal{C}} \arrow[r, Rightarrow, "\eta^P_a"] \arrow[d, Rightarrow, "\eta^P_{a'}"'] & P(a,a) \arrow[d, Rightarrow, "P(a,f)"] \\
P(a',a') \arrow[r, Rightarrow, "P(f,a')"'] & P(a,a')
\end{tikzcd}
\qquad
\begin{tikzcd}
P(a,b)P(b',c) \arrow[r, Rightarrow, "P(a,g)P(b',c)"] \arrow[d, Rightarrow, "P(a,b)P(g,c)"'] & P(a,b')P(b',c) \arrow[d, Rightarrow, "\mu^P_{a,b',c}"] \\
P(a,b)P(b,c) \arrow[r, Rightarrow, "\mu^P_{a,b,c}"'] & P(a,c)
\end{tikzcd}$$

$$\begin{tikzcd}
P(a',b)P(b,c) \arrow[r, Rightarrow, "\mu^P_{a',b,c}"] \arrow[d, Rightarrow, "P(f,b)P(b,c)"'] & P(a',c) \arrow[d, Rightarrow, "P(f,c)"] \\
P(a,b)P(b,c) \arrow[r, Rightarrow, "\mu^P_{a,b,c}"'] & P(a,c)
\end{tikzcd}
\qquad
\begin{tikzcd}
P(a,b)P(b,c) \arrow[r, Rightarrow, "\mu^P_{a,b,c}"] \arrow[d, Rightarrow, "P(a,b)P(b,h)"'] & P(a,c) \arrow[d, Rightarrow, "P(a,h)"] \\
P(a,b)P(b,c') \arrow[r, Rightarrow, "\mu^P_{a,b,c'}"'] & P(a,c')
\end{tikzcd}$$

for all $a, a', b, b', c, c' \in \mathrm{Ob}\, \mathcal{S}$ and $f \colon a \to a'$, $g \colon b \to b'$ and $h \colon c \to c'$.

We introduce pair completion to unify parameterised monads and category-graded monads with generalised units.

Definition C.2 [[15]] Let $\mathcal{S}$ be a small category. The *pair completion* $\mathcal{S}^{\nabla}$ of $\mathcal{S}$ is a category whose objects are the objects of $\mathcal{S}$ and homsets are $\mathcal{S}^{\nabla}(a, b) = \mathcal{S}(a, b) \sqcup \{(a, b)\}$. Composition of morphisms are defined as follows:

$$\begin{aligned}
\mathrm{in}_1\, g \circ \mathrm{in}_1\, f &= \mathrm{in}_1(g \circ f)\\
\mathrm{in}_1\, g \circ \mathrm{in}_2\, g &= \mathrm{in}_2(a,c)\\
\mathrm{in}_2(b,c) \circ \mathrm{in}_1\, f &= \mathrm{in}_2(a,c)\\
\mathrm{in}_2(b,c) \circ \mathrm{in}_2(a,b) &= \mathrm{in}_2(a,c)
\end{aligned}$$

where $a, b, c \in \mathrm{Ob}\,\mathcal{S}$, $f\colon a \to b$ and $g\colon b \to c$.

The following two propositions show the correspondence of parameterised monads to category-graded monads with generalised units.

Proposition C.3 (Category-graded monads with generalised units from parameterised monads) *Let $P\colon \mathcal{S}^{\mathrm{op}} \times \mathcal{S} \to [\mathcal{C}, \mathcal{C}]$ be a parameterised monad. We define $(T_P, \eta^{T_P}, \mu^{T_P}, \overline{\eta}^{T_P})$ as follows:*

- *$T_P(a) := \mathcal{C}$, $T_P(f) := P(a,b)$ for $f\colon a \to b$ in $\mathcal{S}^{\nabla}$,*
- *$\eta^{T_P}_a := \eta^P_a\colon \mathrm{Id}_{\mathcal{C}} \Rightarrow T_P(\mathrm{id}_a)$ for $a \in \mathrm{Ob}\,\mathcal{S}$,*
- *$\mu^{T_P}_{f,g} := \mu^P_{a,b,c}\colon T_P(f) \circ T_P(g) \Rightarrow T_P(g \circ f)$ for $a, b, c \in \mathrm{Ob}\,\mathcal{S}$, $f\colon a \to b$ and $g\colon b \to c$ in $\mathcal{S}^{\nabla}$ and*
- *$\overline{\eta}^{T_P}_{\mathrm{in}_1 f} := P(a,f) \circ \eta^P_a\colon \mathrm{Id}_{\mathcal{C}} \Rightarrow T_P(f)$ for $a, b \in \mathcal{S}$ and $f\colon a \to b$ in $\mathcal{S}$.*

Then $(T_P, \eta^{T_P}, \mu^{T_P}, \overline{\eta}^{T_P})$ is a $\mathcal{S}^{\nabla}$-graded monad with a generalised unit.

Proposition C.4 (Parameterised monads from category-graded monads with generalised units) *Let $T\colon (\mathcal{S}^{\nabla})^{\mathrm{op}} \to \mathrm{Endo}(\mathcal{C})$ with $\left(\overline{\eta}^T_f\colon \mathrm{Id}_{\mathcal{C}} \Rightarrow T_f\right)_{f \in \mathcal{S}}$ be a category-graded monad with generalised unit satisfying $T_{\mathrm{in}_1\, \mathrm{id}_a} = T_{\mathrm{in}_2(a,a)}$.* [3] *We define a functor $P_T\colon \mathcal{S}^{\mathrm{op}} \times \mathcal{S} \to [\mathcal{C}, \mathcal{C}]$ and natural transformations η^{P_T} and μ^{P_T} as follows:*

- *$P_T(a,b) := T_{\mathrm{in}_2(a,b)}$,*
- *$P_T(a,a) := T_{\mathrm{in}_1\, \mathrm{id}_a} = T_{\mathrm{in}_2(a,a)}$,*
- *$P_T(f,g)$ is*

$$\begin{tikzcd}
P_T(a,b) = T_{\mathrm{in}_2(a,b)} \arrow[r, Rightarrow, "\overline{\eta}^T_f T_{\mathrm{in}_2(a,b)}"] & T_{\mathrm{in}_1 f} T_{\mathrm{in}_2(a,b)} \arrow[r, Rightarrow, "T_{\mathrm{in}_1 f} T_{\mathrm{in}_2(a,b)} \overline{\eta}^T_g"] & T_{\mathrm{in}_1 f} T_{\mathrm{in}_2(a,b)} T_{\mathrm{in}_1 g} \arrow[d, Rightarrow, "\mu^T_{\mathrm{in}_1 f, \mathrm{in}_2(a,b)} T_{\mathrm{in}_1 g}"] \\
 & & T_{\mathrm{in}_2(a',b)} T_{\mathrm{in}_1 g} \arrow[d, Rightarrow, "\mu^T_{\mathrm{in}_2(a',b), \mathrm{in}_1 g}"] \\
 & & T_{\mathrm{in}_2(a',b')} = P_T(a',b')
\end{tikzcd}$$

for $a, a', b, b' \in \mathrm{Ob}\,\mathcal{S}$ and $f\colon a' \to a$ and $g\colon b \to b'$,

- *$\eta^{P_T}_a := \eta^T_a\colon \mathrm{Id}_{\mathcal{C}} \Rightarrow P(a,a)$,*
- *$\mu^{P_T}_{a,b,c} := \mu^T_{\mathrm{in}_2(a,b), \mathrm{in}_2(b,c)}\colon P(a,b)P(b,c) \Rightarrow P(a,c)$*

Then $(P_T, \eta^{P_T}, \mu^{P_T})$ is a parameterised monad.

Theorem C.5 *Parameterised monads correspond to category-graded monads with generalised units. More precisely, there is a one to one correspondence between the set of $\mathcal{S}$-parameterised monads and the set of $\mathcal{S}^{\nabla}$-graded monads with generalised units $(T, \overline{\eta})$ satisfying $T_{\mathrm{in}_1 f} = T_{\mathrm{in}_2(a,b)}$ for all $a, b \in \mathrm{Ob}\,\mathcal{S}$ and $f\colon a \to b$ in $\mathcal{S}$.*

[3] The condition $T_{\mathrm{in}_1\, \mathrm{id}_a} = T_{\mathrm{in}_2(a,a)}$ is necessary for the construction of $P_T(f,g)$ to be well-defined, but it was not imposed in [15].

C.1 Correspondence of Eilenberg-Moore Constructions

Let $P\colon \mathcal{S}^{\mathrm{op}} \times \mathcal{S} \to [\mathcal{C}, \mathcal{C}]$ be a parameterised monad. We obtain the functor $\hat{T^P}\colon (\mathcal{S}^{\nabla})^{\mathrm{op}} \to \mathbf{CAT}$ by Eilenberg-Moore construction on $T_P\colon (\mathcal{S}^{\nabla})^{\mathrm{op}} \xrightarrow{\text{lax}} \mathrm{Endo}(\mathcal{C})$. In this section, we describe the relation between P-algebras and T_P-algebras. For $a \in \mathrm{Ob}\,\mathcal{S}$, Recall that the category of T^P-algebra at a, $\hat{T^P}(a)$, consists of the following data.

- Objects are (A, h) where for $f\colon a \to b$, $g\colon b \to c$ in $\mathcal{S}^{\nabla}$ $A\colon \sum_b (\mathcal{S}^{\nabla})(b, a) \to \mathcal{C}$ and $\xi_{f,g}\colon T_f A_g \to A_{g\circ f}$ are compatible with η^{T_P} and μ^{T_P}.
- A morphism from (A, ξ) to (A', ξ') is a natural transformation $\alpha\colon A \Rightarrow A'$ which compatible with ξ and ξ'.

On the other hand, the category of P-algebra $\mathcal{C}^P$ consists of the following data [1].

- Objects are (B, ζ) where $B\colon \mathcal{S}^{\mathrm{op}} \to \mathcal{C}$ and $\zeta_{a,b}\colon P(a, b)B_b \to B_a$ are compatible with η^P and μ^P.
- A morphism from (B, ζ) to (B', ζ') is a natural transformation $\beta\colon B \Rightarrow B'$ which compatible with ζ and ζ'.

Construction of T_P-algebras from P-algebras. Given a P-algebra (B, ζ), we can construct a T_P-algebra (A, ξ) at a for any $a \in \mathrm{Ob}\,\mathcal{S}$ as follows.

$$A_g := B_b, \quad \xi_{f,g} := \zeta_{c,b}\colon T_{Pf} A_g = P(c, b)B_b \to B_c = A_{g\circ f}$$

where $f\colon c \to b$ and $g\colon b \to a$ are morphisms in $\mathcal{S}^{\nabla}$. This T_P-algebra (A, ξ) satisfies $A_{\mathrm{in}_1 f} = A_{\mathrm{in}_2(b,a)}$ for all $b \in \mathrm{Ob}\,\mathcal{S}$ and $f\colon b \to a$ in $\mathcal{S}$

Next, we consider a construction of morphisms of T_P-algebras from morphisms of P-algebras. Given a morphism $\beta\colon (B, \zeta) \to (B', \zeta')$ between P-algebras. Recall that β is a natural transformation $\beta\colon B \Rightarrow B'$ satisfying the following commutative diagram.

$$\begin{tikzcd}
P(b,c)B_c \arrow[r, "P(b,c)\beta_c"] \arrow[d, "\zeta_{b,c}"'] & P(b,c)B'_c \arrow[d, "\zeta'_{b,c}"] \\
B_b \arrow[r, "\beta_b"'] & B'_b
\end{tikzcd}$$

Let (A, ξ) and (A', ξ') be the T_P-algebras constructed from (B, ζ) and (B', ζ'), respectively. We define a natural transformation $\alpha\colon A \Rightarrow A'$ to be $\alpha_{f\colon b\to a} := \beta_b\colon A_f = B_b \to B'_b = A'_f$. This α becomes a T_P-homomorphism at a. Indeed, for all $f\colon b \to c$ and $g\colon c \to a$ in $\mathcal{S}^{\nabla}$ the following commutative diagram hold.

$$\begin{tikzcd}
T_{Pf}A_g \arrow[r, "T_{Pf}\alpha_f"] \arrow[d, equal] \arrow[ddd, bend right=60, "h_{f,g}"'] & T_{Pf}A'_g \arrow[d, equal] \arrow[ddd, bend left=60, "h'_{f,g}"] \\
P(b,c)B_c \arrow[r, "P(b,c)\beta_c"] \arrow[d, "k_{b,c}"] & P(b,c)B'_c \arrow[d, "k'_{b,c}"] \\
B_b \arrow[r, "\beta_b"] \arrow[d, equal] & B'_b \arrow[d, equal] \\
A_{gf} \arrow[r, "\alpha_{gf}"] & A'_{gf}
\end{tikzcd}$$

Summarizing the above discussion, we obtain a functor $F_a\colon \mathcal{C}^P \to \hat{T_P}(a)$ where $a \in \mathrm{Ob}\,\mathcal{S}$. In particular, we get $F_a\colon \mathcal{C}^P \to \mathcal{C}_a^{T_P}$ where $\mathcal{C}_a^{T_P}$ be the full subcategory of $\hat{T_P}(a)$ whose objects are (A, ξ) satisfying $A_{\mathrm{in}_1 f} = A_{\mathrm{in}_2(b,a)}$ for all $b \in \mathrm{Ob}\,\mathcal{S}$ and $f\colon b \to a$ in $\mathcal{S}$

Construction P-algebras from T_P-algebras. Conversely, given a T_P-algebra (A, ξ) at a satisfying $A_{\mathrm{in}_1 f} = A_{\mathrm{in}_2(b,a)}$ for all $b \in \mathrm{Ob}\,\mathcal{S}$ and $f\colon b \to a$ in $\mathcal{S}$, we can construct a P-algebra (B, ζ) as follows.

$$B_b := A_{\mathrm{in}_2(b,a)}$$

$$B_f := \left(B_c = A_{\mathrm{in}_2(c,a)} \xrightarrow{\overline{\eta}_f^{T_P}} T_{Pf} A_{\mathrm{in}_2(c,a)} \xrightarrow{\xi_{f,\mathrm{in}_2(c,a)}} A_{\mathrm{in}_2(c,a)\circ f} = B_b \right)$$

$$\zeta_{b,c} := \xi_{\mathrm{in}_2(b,c),\mathrm{in}_2(c,a)} \colon P(b,c)B_c = T_{P\mathrm{in}_2(b,c)} A_{\mathrm{in}_2(c,a)} \to A_{\mathrm{in}_2(b,a)} = B_b$$

We prove that B is a functor. Firstly, we have

$$B_{\mathrm{id}_b} = \xi_{\mathrm{id}_b,\mathrm{in}_2(b,a)} \circ \overline{\eta}_{\mathrm{id}_b}^{T_P} A_{\mathrm{in}_2(b,a)} = \xi_{\mathrm{id}_b,\mathrm{in}_2(b,a)} \circ \eta_b^{T_P} A_{\mathrm{in}_2(b,a)} = \mathrm{id}_{A_{\mathrm{in}_2(b,a)}},$$

$$\begin{tikzcd}
A_{ba} \arrow[r, "\eta_b^{T_P} A_{ba}"] \arrow[rr, bend left=50, "B_{\mathrm{id}_b}"] \arrow[rr, bend right=50, equal] & T_{\mathrm{id}_b} A_{ba} \arrow[r, "\xi_{\mathrm{id}_b,ba}"] & A_{ba}
\end{tikzcd} .$$

Secondly, we have

$$B_f \circ B_g = \left(\begin{tikzcd}
A_{\mathrm{in}_2(d,a)} \arrow[r, "\overline{\eta}_g^{T_P} A_{\mathrm{in}_2(d,a)}"] & T_{Pg} A_{\mathrm{in}_2(d,a)} \arrow[r, "h_{g,\mathrm{in}_2(d,a)}"] & A_{\mathrm{in}_2(d,a)\circ g} \arrow[d, "\overline{\eta}_f^{T_P} A_{\mathrm{in}_2(d,a)\circ g}"] \\
 & & T_{Pf} A_{\mathrm{in}_2(d,a)\circ g} \arrow[d, "\xi_{f,\mathrm{in}_2(d,a)\circ g}"] \\
 & & A_{\mathrm{in}_2(d,a)\circ g\circ f}
\end{tikzcd} \right)$$

and

$$B_{g\circ f} = \left(A_{\mathrm{in}_2(d,a)} \xrightarrow{\overline{\eta}_{gf}^{T_P} A_{\mathrm{in}_2(d,a)}} T_{Pg\circ f} A_{\mathrm{in}_2(d,a)} \xrightarrow{\xi_{g\circ f,\mathrm{in}_2(d,a)}} A_{\mathrm{in}_2(d,a)\circ g\circ f} \right)$$

for $f\colon b \to c$ and $g\colon c \to d$ in $\mathcal{S}$. These two morphisms are equal because the following diagram commutes.

$$\begin{tikzcd}
A_{\mathrm{in}_2(d,a)} \arrow[d, "\overline{\eta}_g^{T_P} A_{\mathrm{in}_2(d,a)}"'] \arrow[rrd, bend left, "\overline{\eta}_{g\circ f}^{T_P} A_{\mathrm{in}_2(d,a)}"] & & \\
T_{Pg} A_{\mathrm{in}_2(d,a)} \arrow[r, "\overline{\eta}_f^{T_P} T_g A_{\mathrm{in}_2(d,a)}"] \arrow[d, "\xi_{g,\mathrm{in}_2(d,a)}"'] & T_{Pf} T_{Pg} A_{\mathrm{in}_2(d,a)} \arrow[r, "\mu_{f,g}^{T_P} A_{\mathrm{in}_2(d,a)}"] \arrow[d, "T_f \xi_{g,\mathrm{in}_2(d,a)}"] & T_{Pg\circ f} A_{\mathrm{in}_2(d,a)} \arrow[d, "\xi_{g\circ f,\mathrm{in}_2(d,a)}"] \\
A_{\mathrm{in}_2(d,a)\circ g} \arrow[r, "\overline{\eta}_f^{T_P} A_{\mathrm{in}_2(d,a)\circ g}"'] & T_{Pf} A_{\mathrm{in}_2(d,a)\circ g} \arrow[r, "\xi_{f,\mathrm{in}_2(d,a)\circ g}"'] & A_{\mathrm{in}_2(d,a)\circ g\circ f}
\end{tikzcd}$$

In the above diagram, the top triangle is commutative by the definition of generalised unit, the left bottom square is commutative by the naturality of $\overline{\eta}^{T_P}$ and the right bottom square is commutative the definition of T_P-algebra (A, ξ).

Summarizing the above discussion, we obtain a functor $G_a\colon \mathcal{C}_a^{T_P} \to \mathcal{C}^P$.

We can easily check the following theorem by the definition of F_a and G_a.

Theorem C.6 *The category $\mathcal{C}^P$ of P-algebras is isomorphic to the category $\mathcal{C}_a^{T_P}$ of T_P-algebras at a.*

$$F_a : \mathcal{C}^P \cong \mathcal{C}_a^{T_P} : G_a$$

D Proofs of propositions

D.1 Progress Lemma

Proof. By induction on the derivation of $\vdash_f M : A$.

Case Tc-Val. In this case, $f = \mathrm{id}_a$, $M = \mathbf{val}_a\, V$ for some V. Thus, the claim holds.

Case Tc-Op. We have $M = \mathrm{op}(V)$ for some V. Thus, the claim holds.

Case Tc-Let. We have $M = \mathbf{let}\, x \leftarrow M' \,\mathbf{in}\, N$ for some x, M' and N, and $f = g; h$ for some morphisms $g \colon b \to b'$ and $h \colon b' \to a$. The root of the derivation is as follows:

$$\dfrac{\vdash_g M' : B \quad x : B \vdash_h N : A}{\vdash_{g;h} \mathbf{let}\, x \leftarrow M' \,\mathbf{in}\, N : A}\ \textsc{Tc-Let}$$

By applying the induction hypothesis to $\vdash_g M' : B$, we obtain three cases:

(i) $g = \mathrm{id}_b$ and $M' = \mathbf{val}_b\, V$. We have $f = h$ and $M = \mathbf{let}\, x \leftarrow \mathbf{val}_b\, V \,\mathbf{in}\, N \to N[V/x]$ by S-Let.

(ii) $M' = \mathcal{E}'[\mathrm{op}(V)]$. We have $M = \mathcal{E}[\mathrm{op}(V)]$ where $\mathcal{E}[-] = \mathbf{let}\, x \leftarrow \mathcal{E}'[-] \,\mathbf{in}\, N$.

(iii) There exists M'' such that $M' \to M''$. Using S-Lift, we have $M = \mathcal{E}[M'] \to \mathcal{E}[M'']$ where $\mathcal{E}[-] = \mathbf{let}\, x \leftarrow [-] \,\mathbf{in}\, N$.

Case Tc-App. We can show that the root of the derivation is as follows:

$$\dfrac{\dfrac{x : B \vdash_f M' : A}{\vdash \lambda^f x : B.M' : B \to A; f}\ \textsc{Tv-Abs} \qquad \vdash W : B}{\vdash_f (\lambda^f x : B.M')W : A}\ \textsc{Tc-App}$$

Thus we have $M = (\lambda^f x : B.M')W \to M'[W/x]$ by S-App.

Case Tc-Proj and Tc-Match. Straightforward.

Case Tc-Handle. We have $f = G(f')$ for some $f' \colon b' \to a'$, and $M = \mathbf{handle}\, M' \,\mathbf{with}\, H$ for some handler H. The root of the derivation is as follows:

$$\dfrac{\vdash^{\Sigma'}_{f' \colon b' \to a'} M' : B \quad \vdash^{G}_{a'} H : B \Rightarrow A}{\vdash^{\Sigma}_{G(f')} \mathbf{handle}\, M' \,\mathbf{with}\, H : A}\ \textsc{Tc-Handle}$$

By applying the induction hypothesis to $\vdash^{\Sigma'}_{f'} M' : B$, we obtain three cases:

(i) $f' = \mathrm{id}_{a'}$ and $M' = \mathbf{val}_{a'}\, V$. We have $M = \mathbf{handle}\,(\mathbf{val}_{a'}\, V)\,\mathbf{with}\, H$. By Th-Handler, H contains $\mathbf{val}_{a'}\, x \mapsto N$. So we get

$$M = \mathbf{handle}\,(\mathbf{val}_{a'}\, V)\,\mathbf{with}\, H \to N[V/x]$$

by S-HandleRet.

(ii) $M' = \mathcal{E}[\mathrm{op}(V)]$ for some $\mathcal{E}$, $\mathrm{op} \in \Sigma'$ and V. Let $k \colon c' \to b'$ be the grading morphism that corresponds to $\mathcal{E}$. By Th-Handler, H contains $(\mathrm{op}(p), r \mapsto_k M^k_{\mathrm{op}})$. So we get $M = \mathbf{handle}\, \mathcal{E}[\mathrm{op}(V)] \,\mathbf{with}\, H$ and

$$M \to M^k_{\mathrm{op}}[V/p, \lambda^{Gk} y.\, \mathbf{handle}\, \mathcal{E}[\mathbf{val}_{c'}\, y] \,\mathbf{with}\, H/r]$$

by S-HandleOp.

(iii) There exists M'' such that $M' \to M''$. By S-Lift, $M = \mathbf{handle}\, M' \,\mathbf{with}\, H \to \mathbf{handle}\, M'' \,\mathbf{with}\, H$.

□

D.2 Preservation Lemma

Proof. By induction on the derivation of $\vdash_f M : A$.

Case Tc-Val. We have $M = \mathbf{val}_a\, V$ for some a and V, but there is no term N such that $M \to N$. So this cannot happen.

Case Tc-Op. We have $M = \mathrm{op}(V)$ for some op and V, but there is no term N such that $M \to N$. So this cannot happen.

Case Tc-Let. We have $M = \mathbf{let}\, x \leftarrow M' \,\mathbf{in}\, L$ for some x, M', N. The root of the derivation is as follows:

$$\cfrac{\vdash_{g:\, b \to c} M' : B \quad x : B \vdash_{h:\, c \to a} L : A}{\vdash_{g;h} \mathbf{let}\, x \leftarrow M' \,\mathbf{in}\, L : A}\ \textsc{Tc-Let}$$

By the assumption that there is a term N such that $M = \mathbf{let}\, x \leftarrow M' \,\mathbf{in}\, L \to N$, we obtain two cases:

(i) $M \to N$ is derived by S-Let. In this case, $M' = \mathbf{val}_b\, V$ for some V, and $N = L[V/x]$. The root of the derivation is as follows:

$$\cfrac{\cfrac{\vdash V : B}{\vdash_{\mathrm{id}_b} \mathbf{val}_b\, V : B}\ \textsc{Tc-Val} \quad x : B \vdash_f L : A}{\vdash_f \mathbf{let}\, x \leftarrow \mathbf{val}_b\, V \,\mathbf{in}\, L : A}\ \textsc{Tc-Let}$$

We have $\vdash V : B$ and $x : B \vdash_f L : A$, so we have $\vdash_f L[V/x] : A$ by substitution lemma.

(ii) $M \to N$ is derived by S-Lift. In this case, $M' \to N'$ and $M = \mathcal{F}[M'] \to \mathcal{F}[N'] = N$ for some N' where $\mathcal{F} = \mathbf{let}\, x \leftarrow [-] \,\mathbf{in}\, L$. Applying the induction hypothesis to $\vdash_g M' : B$ and $M' \to N'$, we have $\vdash_g N' : B$. We obtain the following derivation:

$$\cfrac{\vdash_g N' : B \quad x : B \vdash_h L : A}{\vdash_{g;h} \mathbf{let}\, x \leftarrow N' \,\mathbf{in}\, L : A}\ \textsc{Tc-Let}$$

Thus we conclude $\vdash_{g;h} N : A$.

Case Tc-App. We have $M = VW$ for some V and W. By the assumption that $VW \to N$ and $\vdash_f VW : A$, we have $V = \lambda^f x : B.M'$ for some B and M' and $N = M'[W/x]$. The root of the derivation is as follows:

$$\cfrac{\cfrac{x : B \vdash_f M' : A}{\vdash \lambda^f x : B.M' : B \to A; f}\ \textsc{Tv-Abs} \quad \vdash W : B}{\vdash_f (\lambda^f x : B.M')W : A}\ \textsc{Tc-App}$$

Thus we have $\vdash_f M'[W/x] : A$ by substitution lemma, as required.

Case Tc-Proj. We have $M = \mathbf{proj}\, V \,\mathbf{as}\, \langle x, y\rangle.M'$ for some V, x, y and M'. The root of the derivation is as follows:

$$\cfrac{\vdash V : B_1 \times B_2 \quad x : B_1, y : B_2 \vdash_f M : A}{\vdash_f \mathbf{proj}\, V \,\mathbf{as}\, \langle x, y\rangle.M' : A}\ \textsc{Tc-Proj}$$

By the assumption that there is N such that $M = \mathbf{proj}\, V \,\mathbf{as}\, \langle x, y\rangle.M' : A \to N$, we have $V = \langle V_1, V_2\rangle$, $\vdash V_1 : B_1$, $\vdash V_2 : B_2$ and $N = M'[V_1/x, V_2/y]$. Thus, we have $\vdash_f M'[V_1/x, V_2/y] : A$ by substitution lemma.

Case Tc-Match. Straightforward.

Case Tc-Handle. We have $M = \mathbf{handle}\, M' \,\mathbf{with}\, H$ for some M' and H. The root of the derivation is as follows:

$$\cfrac{\vdash^{\Sigma'}_{f':\, b' \to a'} M' : B \quad \vdash^{G}_{a'} H : B \Rightarrow A}{\vdash^{\Sigma}_{G(f')} \mathbf{handle}\, M' \,\mathbf{with}\, H : A}\ \textsc{Tc-Handle}$$

By the assumption that there is a term N such that $M = \mathbf{handle}\, M' \,\mathbf{with}\, H \to N$, we have three cases:

(i) $M \to N$ is derived by S-HandleRet. In this case, we have $M' = \mathbf{val}_{a'}\, V$ for some V and $N = L[V/x]$ where $(\mathbf{val}_{a'}\, x \mapsto L) \in H$. The root of the derivation is as follows:

$$\dfrac{\dfrac{\vdash V : B}{\vdash^{\Sigma'}_{\mathrm{id}_{a'}} \mathbf{val}_{a'}\, V : B} \qquad \dfrac{\begin{array}{c} x : B \vdash^{\Sigma'}_{\mathrm{id}_a} L : A \\ \left\{\begin{array}{l} p : P, \\ r : Q \to A; Gk \end{array} \vdash_{G(k;g)} L^k_{\mathrm{op}} : A \right\} \end{array}}{\vdash^{G}_{a'} H : B \Rightarrow A}\ \textsc{Th-Handler}}{\vdash^{\Sigma}_{\mathrm{id}_a} \mathbf{handle}\,(\mathbf{val}_{a'}\, V)\,\mathbf{with}\, H : A}\ \textsc{Tc-Handle}$$

Thus, we have $\vdash_{\mathrm{id}_a} L[V/x] : A$ by substitution lemma as required.

(ii) $M \to N$ is derived by S-HandleOp. In this case, we have $M' = \mathcal{E}[\mathrm{op}(V)]$ for some $\mathcal{E}$ and V. Let $k\colon c' \to a'$ be the grading morphism that corresponds to $\mathcal{E}$. By the assumption that $M = \mathbf{handle}\,\mathcal{E}[\mathrm{op}(V)]\,\mathbf{with}\, H \to N$, we have $(\mathrm{op}(p), r \mapsto_k L^k_{\mathrm{op}}) \in H$ and

$$N = L^k_{\mathrm{op}}[V/p, \lambda^{Gk} y.\, \mathbf{handle}\,\mathcal{E}[\mathbf{val}_{c'}\, y]\,\mathbf{with}\, H/r].$$

Applying Lemma 5.4 to $\vdash_{f'} \mathcal{E}[\mathrm{op}(V)] : B$, we have $\vdash_{g'} \mathrm{op}(V) : Q$ and $\vdash \mathcal{E} : Q \rightsquigarrow B; k$ satisfying $f' = g'; k$. The derivation of $\Gamma \vdash H\colon B \Rightarrow A$ is as follows:

$$\dfrac{\begin{array}{c} x : B \vdash^{\Sigma'}_{\mathrm{id}_a} L : A \\ \{p : P, r : Q \to A; Gk \vdash_{G(g';k)} L^k_{\mathrm{op}} : A\} \end{array}}{\vdash^{G}_{a'} H : B \Rightarrow A}\ \textsc{Th-Handler}$$

By applying Lemma 5.3 to $\vdash \mathcal{E} : Q \rightsquigarrow B; c' \xrightarrow{k} a'$ and $y : Q \vdash \mathbf{val}_{c'}\, y : Q$, we obtain $y : Q \vdash_k \mathcal{E}[\mathbf{val}_{c'}\, y] : B$. We have the following derivation:

$$\dfrac{\dfrac{y : Q \vdash_k \mathcal{E}[\mathbf{val}_{c'}\, y] : B \qquad \vdash H : B \Rightarrow A}{y : Q \vdash_{Gk} \mathbf{handle}\,\mathcal{E}[\mathbf{val}_{c'}\, y]\,\mathbf{with}\, H : A}\ \textsc{Tc-Handle}}{\vdash \lambda^{Gk} y.\, \mathbf{handle}\,\mathcal{E}[\mathbf{val}_{c'}\, y]\,\mathbf{with}\, H : Q \to A; Gk}\ \textsc{Tv-Abs}$$

Then we obtain

$$\vdash_{G(f')} L^k_{\mathrm{op}}[V/p, \lambda^{Gk} y.\, \mathbf{handle}\,\mathcal{E}[\mathbf{val}_{c'}\, y]\,\mathbf{with}\, H] : A$$

by substitution lemma as required.

(iii) $M \to N$ is derived by S-Lift. In this case, we have $M' \to N'$ and $M = \mathcal{F}[M'] \to \mathcal{F}[N'] = N$ for some N' where $\mathcal{F} = \mathbf{handle}\,[-]\,\mathbf{with}\, H$. Applying the induction hypothesis to $\vdash_{f'} M' : B$ and $M' \to N'$, we have $\vdash_{f'} N' : B$. Thus, we obtain the following derivation:

$$\dfrac{\vdash^{\Sigma'}_{f'} N' : B \qquad \vdash^{G}_{a'} H : B \Rightarrow A}{\vdash^{\Sigma}_{G(f')} \mathbf{handle}\, N'\,\mathbf{with}\, H' : A}\ \textsc{Tc-Handle}$$

Therefore we have $\vdash_{G(f')} N : A$ as required.

□

D.3 Soundness

Proof. We prove by case analysis of the rule used to derive $M \to M'$.

Case S-App.

$$[\![(\lambda^f x.M)V]\!] = [\![\lambda^f x.M]\!]([\![V]\!]) = [\![M]\!]([\![V]\!]) = [\![M[V/x]]\!].$$

Case S-Let.

$$\begin{aligned}[\![\mathbf{let}\, x \leftarrow \mathbf{val}_a\, V\, \mathbf{in}\, M]\!] &= [\![M]\!]^\dagger([\![\mathbf{val}_a\, V]\!]) \\ &= [\![M]\!]^\dagger(\mathrm{e}(a, [\![V]\!])) \\ &= [\![M]\!]([\![V]\!]) \\ &= [\![M[V/x]]\!].\end{aligned}$$

Case S-Proj.

$$\begin{aligned}[\![\mathbf{proj}\, \langle V_1, V_2\rangle\, \mathbf{as}\, \langle x, y\rangle.M]\!] &= [\![M]\!](\pi_1[\![\langle V_1, V_2\rangle]\!], \pi_2[\![\langle V_1, V_2\rangle]\!]) \\ &= [\![M]\!](\pi_1\langle[\![V_1]\!], [\![V_2]\!]\rangle, \pi_2\langle[\![V_1]\!], [\![V_2]\!]\rangle) \\ &= [\![M]\!]([\![V_1]\!], [\![V_2]\!]) \\ &= [\![M[V_1/x, V_2/y]]\!].\end{aligned}$$

Case S-MatchLeft.

$$\begin{aligned}[\![\mathbf{match}\, (\mathbf{inl}\, V)\{x.M_1; y.M_2\}]\!] &= [[\![M_1]\!], [\![M_2]\!]](\mathrm{in}_1[\![V]\!]) \\ &= [\![M_1]\!]([\![V]\!]) \\ &= [\![M_1[V/x]]\!].\end{aligned}$$

Case S-MatchRight. Similar to S-MatchLeft.
Case S-HandleRet. Let $H = \{\mathbf{val}_a\, x \mapsto M\} \cup \{\mathrm{op}(p), r \mapsto M^k_{\mathrm{op}}\}^k_{\mathrm{op}}$.

$$\begin{aligned}[\![\mathbf{handle}\, \mathbf{val}_a\, V\, \mathbf{with}\, H]\!] &= [\![H]\!]([\![\mathbf{val}_a\, V]\!]) \\ &= [\![H]\!](\mathrm{e}(a, [\![V]\!])) \\ &= [\![M]\!]([\![V]\!]) \\ &= [\![M[V/x]]\!].\end{aligned}$$

Case S-HandleOp. Let H be $\{\mathbf{val}_a\, x \mapsto M\} \cup \{\mathrm{op}(p), r \mapsto M^k_{\mathrm{op}}\}^k_{\mathrm{op}}$, $\mathcal{E}$ be $\mathbf{let}\, x_n \leftarrow (\dots(\mathbf{let}\, x_1 \leftarrow [\,]\, \mathbf{in}\, N_1)\dots)\, \mathbf{in}\, N_n$, and k be the grading morphism that corresponds to $\mathcal{E}$.

$$\begin{aligned}&[\![\mathbf{handle}\, \mathcal{E}[\mathrm{op}(V)]\, \mathbf{with}\, H]\!] \\ &= [\![H]\!]([\![\mathcal{E}[\mathrm{op}(V)]]\!]) \\ &= [\![H]\!]([\![N_n]\!]^\dagger(\dots([\![N_1]\!]^\dagger(\mathbf{do}(\mathrm{op}, [\![V]\!], \{\mathrm{e}(a, x)\}_{x \in [\![A]\!]})))\dots)) \\ &= [\![H]\!](\mathbf{do}(\mathrm{op}, [\![V]\!], \{[\![N_n]\!]^\dagger(\dots([\![N_1]\!]^\dagger\, \mathrm{e}(a, x))\dots)\}_{x \in [\![A]\!]})) \\ &= [\![M^k_{\mathrm{op}}]\!]([\![V]\!], \{[\![H]\!]([\![N_n]\!]^\dagger(\dots([\![N_1]\!]^\dagger\, \mathrm{e}(a, x))\dots))\}_{x \in [\![A]\!]}) \\ &= [\![M^k_{\mathrm{op}}]\!]([\![V]\!], [\![\lambda^{Gk} x.\, \mathbf{handle}\, \mathcal{E}[\mathbf{val}_a\, x]\, \mathbf{with}\, H]\!]) \\ &= [\![M^k_{\mathrm{op}}[V/p, \lambda^{Gk} x.\, \mathbf{handle}\, \mathcal{E}[\mathbf{val}_a\, x]\, \mathbf{with}\, H/r]]\!].\end{aligned}$$

Case S-Lift. Obvious. □

D.4 Lemma 7.4

Proof. Let s be a tuple $(w_1, \dots, w_n)$ and σ be a substitution $W_1/x_1, \dots, W_n/x_n$. We prove by induction on the derivation of the judgements.

Case Tv-Unit, Tv-Id, Tv-Pair, Tv-InjL and Tv-InjR. Straightforward.

Case Tv-Abs. By the induction hypothesis, for all $w \in [\![A]\!]$ and value terms $\vdash W : A$ with $w \lhd_A W$, we have

$$\begin{aligned}([\![\Gamma \vdash \lambda^f x : A.M : B]\!](s))(w) &= [\![\Gamma, x : A \vdash_f M : B]\!](s, w)\\ &\lhd_B^f M[\sigma, W/x].\end{aligned}$$

Here, we have

$$\begin{aligned}((\lambda^f x : A.M)[\sigma])W &= (\lambda^f x : A.M[\sigma])W\\ &\to M[\sigma, W/x]\end{aligned}$$

Then we can apply Lemma 7.3 and obtain

$$[\![\lambda^f x : A.M]\!](s) \lhd_{A \to B;f} (\lambda^f x : A.M : B)[s, W/x].$$

Case Tc-Val. We have

$$[\![\Gamma \vdash_{\mathrm{id}_a} \mathbf{val}_a\, V : A]\!](s) = \mathrm{e}(a, [\![V]\!](s)).$$

By the induction hypothesis, we have $[\![V]\!](w_1, \dots, w_n) \lhd_A V[\sigma]$. Thus, we have $[\![\Gamma \vdash_{\mathrm{id}_a} \mathbf{val}_a\, V : A]\!](s) \lhd_A^{\mathrm{id}_a}$ $(\mathbf{val}_a\, V)[\sigma]$ by definition of $\lhd_A^{\mathrm{id}_a}$.

Case Tc-Op. We have

$$[\![\Gamma \vdash_f \mathrm{op}(V) : Q]\!](s) = \mathbf{do}(\mathrm{op}, [\![V]\!](s), \{\mathrm{e}(a, x)\}_{x \in [\![Q]\!]}).$$

By the induction hypothesis, $[\![V]\!](s) \lhd_Q V[\sigma]$. Furthermore, for any $w \in [\![Q]\!]$ and closed value term $\vdash W : Q$, we have $e(a, w) \lhd_Q^{\mathrm{id}_a} \mathbf{val}_a\, W$. Thus, we have

$$[\![\Gamma \vdash_f \mathrm{op}(V)]\!](s) \lhd_Q^f \mathrm{op}(V)[\sigma].$$

Case Tc-Let. We have

$$[\![\Gamma \vdash_{f;g} \mathbf{let}\, x \leftarrow M\, \mathbf{in}\, N : B]\!](s) = ([\![N]\!](s, -))^\dagger([\![M]\!](s)).$$

By the induction hypothesis, we have $[\![M]\!](s) \lhd_A^f M[\sigma]$ and $[\![N]\!](s, v) \lhd_B^g N[\sigma, V/x]$ for any $v \in [\![A]\!]$ and $\vdash V : A$ with $v \lhd_A V$. By the definition of $\lhd_A^f$, there are two cases:

(i) $[\![M]\!](s) = \mathrm{e}(a, v)$, $f = \mathrm{id}_a$, $M[\sigma] \to^* \mathbf{val}_a\, V$ and $v \lhd_P V$. We have

$$\begin{aligned}([\![N]\!](s, -))^\dagger([\![M]\!](s)) &= ([\![N]\!](s, -))^\dagger(\mathrm{e}(a, v))\\ &= [\![N]\!](s, v)\end{aligned}$$

and

$$\begin{aligned}(\mathbf{let}\, x \leftarrow M\, \mathbf{in}\, N)[\sigma] &= \mathbf{let}\, x \leftarrow (M[\sigma])\, \mathbf{in}\, (N[\sigma])\\ &\to^* \mathbf{let}\, x \leftarrow \mathbf{val}_a\, V\, \mathbf{in}\, (N[\sigma])\\ &\to N[\sigma, V/x].\end{aligned}$$

Therefore, by Lemma 7.3 we obtain

$$[\![\mathbf{let}\, x \leftarrow M\, \mathbf{in}\, N]\!](s) \lhd_B^{f;g} (\mathbf{let}\, x \leftarrow M\, \mathbf{in}\, N)[\sigma].$$

(ii) $[\![M]\!](s) = \mathbf{do}(\mathrm{op}, v, \{t_x\}_{x \in [\![Q]\!]})$, $M[\sigma] \to^* \mathcal{E}[\mathrm{op}(V)]$, $v \lhd_P V$, and if $w \lhd_Q W$ then $t_w \lhd_A^k \mathcal{E}[\mathbf{val}_b\, W]$. Then, we have

$$\begin{aligned}([\![N]\!](s,-))^\dagger([\![M]\!](s)) &= ([\![N]\!](s,-))^\dagger(\mathbf{do}(\mathrm{op}, v, \{t_x\}_x)\\ &= \mathbf{do}(\mathrm{op}, v, \{([\![N]\!](s,-))^\dagger(t_x)\}_x)\end{aligned}$$

and

$$\begin{aligned}(\mathbf{let}\, x \leftarrow M \,\mathbf{in}\, N)[\sigma] &= \mathbf{let}\, x \leftarrow (M[\sigma]) \,\mathbf{in}\, (N[\sigma])\\ &\to^* \mathbf{let}\, x \leftarrow \mathcal{E}[\mathrm{op}(V)] \,\mathbf{in}\, (N[\sigma]).\end{aligned}$$

Given $w \in [\![Q]\!]$ and $\vdash W : Q$ with $w \lhd_Q W$, we show

$$[\![N]\!](s,-)^\dagger(t_w) \lhd_B^{k;g} \mathbf{let}\, x \leftarrow \mathcal{E}[\mathbf{val}_b\, W] \,\mathbf{in}\, (N[\sigma])$$

by induction on the height of the tree t_w.

- If $t_w = \mathrm{e}(c, u)$ then we have $\mathrm{e}(c, u) = t_w \lhd_A^k \mathcal{E}[\mathbf{val}_b\, W]$ and then $c = b$, $\mathcal{E} = [\,]$ and $u \lhd_Q W$ by definition of $\lhd_A^k$. We have

$$\begin{aligned}[\![N]\!](s,-)^\dagger(t_w) &= [\![N]\!](s,-)^\dagger(\mathrm{e}(b, u))\\ &= [\![N]\!](s, u)\\ &\lhd_B^{k;g} N[\sigma, W/x]\end{aligned}$$

  and

$$\mathbf{let}\, x \leftarrow \mathcal{E}[\mathbf{val}_b\, W] \,\mathbf{in}\, (N[\sigma]) \to^* N[\sigma, W/x].$$

  Therefore, by Lemma 7.3, we have

$$[\![N]\!](s,-)^\dagger(t_w) \lhd_B^{k;g} \mathbf{let}\, x \leftarrow \mathcal{E}[\mathbf{val}_b\, W] \,\mathbf{in}\, (N[\sigma]).$$

- If $t_w = \mathbf{do}(\mathrm{op}', u, \{t'_y\}_{y \in [\![Q']\!]})$ then we have $\mathbf{do}(\mathrm{op}', u, \{t'_y\}_y) = t_w \lhd_A^k \mathcal{E}[\mathbf{val}_b\, W]$ and then $\mathcal{E}[\mathbf{val}_b\, W] \to^* \mathcal{E}'[\mathrm{op}'(U)]$, $u \lhd_{P'} U$ and if $w' \lhd_{Q'} W'$ then $t'_{w'} \lhd_{A'}^{k'} \mathcal{E}'[\mathbf{val}_{b'}\, W']$. We have

$$\mathbf{let}\, x \leftarrow \mathcal{E}[\mathbf{val}_b\, W] \,\mathbf{in}\, (N[\sigma]) \to^* \mathbf{let}\, x \leftarrow \mathcal{E}'[\mathbf{val}_{b'}\, W'] \,\mathbf{in}\, (N[\sigma])$$

  and by induction hypothesis,

$$[\![N]\!](s,-)^\dagger(t'_{w'}) \lhd_B^{k';g} \mathbf{let}\, x \leftarrow \mathcal{E}'[\mathbf{val}_{b'}\, W'] \,\mathbf{in}\, (N[\sigma]).$$

  Thus, by Lemma 7.3, we have

$$[\![N]\!](s,-)^\dagger(t'_{w'}) \lhd_B^{k';g} \mathbf{let}\, x \leftarrow \mathcal{E}[\mathbf{val}_b\, W] \,\mathbf{in}\, (N[\sigma]).$$

  Therefore, we have

$$[\![N]\!](s,-)^\dagger(t_w) \lhd_B^{k;g} \mathbf{let}\, x \leftarrow \mathcal{E}[\mathbf{val}_b\, W] \,\mathbf{in}\, (N[\sigma])$$

  by definition of $\lhd_B^{k;g}$.

Case Tc-App. By the induction hypothesis, we have

$$[\![V]\!](s) \lhd_{A \to B;f} V[\sigma], \quad [\![W]\!](s) \lhd_A W[\sigma].$$

Thus, by definition of $\lhd_{A\to B;f}$, we have

$$\begin{aligned}[\![\Gamma \vdash_f VW : B]\!](s) &= ([\![V]\!](s))([\![W]\!](s))\\ &\lhd^f_B (V[\sigma])(W[\sigma])\\ &= (VW)[\sigma]\end{aligned}$$

as required.

Case Tc-Proj, Tc-Match. Straightforward.

Case Tc-Handle. Let H be a handler $\{\mathbf{val}_a\, x \mapsto N\} \cup \{\mathrm{op}(p), r \mapsto M^k_{\mathrm{op}}\}^k_{\mathrm{op}}$. We have $[\![\mathbf{handle}\, M\, \mathbf{with}\, H]\!](s) = ([\![H]\!](s))([\![M]\!](s))$ and $\Gamma \vdash^{\Sigma'}_{Gf} \mathbf{handle}\, M\, \mathbf{with}\, H : B$. The root of derivation is as follows:

$$\dfrac{\Delta \vdash^{\Sigma}_{f} M : A \quad \Gamma \vdash^{G}_{a} H : A \Rightarrow B \quad \Delta \subseteq \Gamma}{\Gamma \vdash^{\Sigma'}_{Gf} \mathbf{handle}\, M\, \mathbf{with}\, H : B}\ \text{Tc-Handle}$$

where $\Delta = x_1 : P_1, \ldots, x_n : P_n$. We write the restriction of s and σ to Δ by $s\restriction_\Delta$ and $\sigma\restriction_\Delta$, respectively. By the induction hypothesis, we have

$$\begin{aligned}[\![M]\!](s\restriction_\Delta) &\lhd^f_A M[\sigma\restriction_\Delta],\\ [\![N]\!](s, v) &\lhd^{\mathrm{id}_{Ga}}_B N[\sigma, V/x],\\ [\![M^k_{\mathrm{op}}]\!](s, v, w) &\lhd^{Gk}_B M^k_{\mathrm{op}}[\sigma, V/p, W/r]\end{aligned}$$

where $v \in [\![A]\!]$ and $\vdash V : A$ satisfying $v \lhd_A V$ in the second relation, and $v \in [\![P]\!]$, $\vdash V : P$, $w \in [\![Q \to B; Gk]\!]$ and $\vdash W : Q \to B; Gk$ satisfying $v \lhd_P V$ and $w \lhd_{Q\to B;Gk} W$ in the third relation. We want to show

$$[\![\mathbf{handle}\, M\, \mathbf{with}\, H]\!](s) \lhd^{Gf}_B (\mathbf{handle}\, M\, \mathbf{with}\, H)[\sigma].$$

By the definition of $\lhd^f_A$, there are two cases:

(i) $[\![M]\!](s\restriction_\Delta) = \mathrm{e}(a, v)$, $f = \mathrm{id}_a$, $M[\sigma\restriction_\Delta] \to^* \mathbf{val}_a\, V$ and $v \lhd_P V$. We have

$$\begin{aligned}([\![H]\!](s))([\![M]\!](s\restriction_\Delta) &= [\![H]\!](s)(\mathrm{e}(a, v))\\ &= [\![N]\!](s, v)\\ &\lhd^{\mathrm{id}_{Ga}}_B N[\sigma, V/x]\end{aligned}$$

and

$$\begin{aligned}(\mathbf{handle}\, M\, \mathbf{with}\, H)[\sigma] &= \mathbf{handle}\, M[\sigma\restriction_\Delta]\, \mathbf{with}\, H[\sigma]\\ &\to^* \mathbf{handle}\, \mathbf{val}_a\, V\, \mathbf{with}\, H[\sigma]\\ &\to N[\sigma, V/x].\end{aligned}$$

Therefore, by Lemma 7.3 we obtain

$$[\![\mathbf{handle}\, M\, \mathbf{with}\, H]\!](s) \lhd^{Gf}_B (\mathbf{handle}\, M\, \mathbf{with}\, H)[\sigma].$$

(ii) $[\![M]\!](s\restriction_\Delta) = \mathbf{do}(\mathrm{op}, v, \{t_x\}_{x\in[\![Q]\!]})$, $M[\sigma\restriction_\Delta] \to^* \mathcal{E}[\mathrm{op}(V)]$, $v \lhd_P V$ and if $w \lhd_Q W$ then $t_w \lhd^k_A \mathcal{E}[\mathbf{val}_b\, W]$. We can show $\lambda x.[\![H]\!](s)(t_x) \lhd^{Gk}_B \lambda^{Gk} y : Q.\, \mathbf{handle}\, \mathcal{E}[\mathbf{val}_b\, y]\, \mathbf{with}\, H$ by induction on the height of the tree t_x similarly to the case Tc-Let. We have

$$\begin{aligned}([\![H]\!](s))([\![M]\!](s\restriction_\Delta)) &= ([\![H]\!](s))(\mathbf{do}(\mathrm{op}, v, \{t_x\}_{x\in[\![Q]\!]})\\ &= [\![M^k_{\mathrm{op}}]\!](s, v, \{[\![H]\!](s)(t_x)\}_{x\in[\![Q]\!]})\\ &\lhd^{Gk}_B M^k_{\mathrm{op}}[\sigma, V/p, \lambda^{Gk} y : Q.\, \mathbf{handle}\, \mathcal{E}[\mathbf{val}_b\, y]\, \mathbf{with}\, H/r]\end{aligned}$$

and

$$\begin{aligned}(\mathbf{handle}\, M\, \mathbf{with}\, H)[\sigma] &= \mathbf{handle}\, M[\sigma{\restriction}_{\Delta}]\, \mathbf{with}\, H[\sigma] \\ &\to^{*} \mathbf{handle}\, \mathcal{E}[\mathrm{op}(V)]\, \mathbf{with}\, H[\sigma] \\ &\to M^{k}_{\mathrm{op}}[\sigma, V/p, \lambda^{Gk} y : Q.\, \mathbf{handle}\, \mathcal{E}[\mathbf{val}_{b}\, y]\, \mathbf{with}\, H/r].\end{aligned}$$

Therefore, we obtain

$$[\![\mathbf{handle}\, M\, \mathbf{with}\, H]\!](s) \lhd^{Gf}_{B} (\mathbf{handle}\, M\, \mathbf{with}\, H)[\sigma]$$

by Lemma 7.3 as required.

□