Flexible Presentations of Graded Monads

SHIN-YA KATSUMATA, National Institute of Informatics, Japan
DYLAN MCDERMOTT, Reykjavik University, Iceland
TARMO UUSTALU, Reykjavik University, Iceland and Tallinn University of Technology, Estonia
NICOLAS WU, Imperial College London, United Kingdom

A large class of monads used to model computational effects have natural presentations by operations and equations, for example, the list monad can be presented by a constant and a binary operation subject to unitality and associativity. Graded monads are a generalization of monads that enable us to track quantitative information about the effects being modelled. Correspondingly, a large class of graded monads can be presented using an existing notion of graded presentation. However, the existing notion has some deficiencies, in particular many effects do not have natural graded presentations.

We introduce a notion of flexibly graded presentation that does not suffer from these issues, and develop the associated theory. We show that every flexibly graded presentation induces a graded monad equipped with interpretations of the operations of the presentation, and that all graded monads satisfying a particular condition on colimits have a flexibly graded presentation. As part of this, we show that the usual algebra-preserving correspondence between presentations and a class of monads transfers to an algebra-preserving correspondence between flexibly graded presentations and a class of flexibly graded monads.

CCS Concepts: • **Theory of computation** → *Categorical semantics*; *Denotational semantics*; *Equational logic and rewriting*.

Additional Key Words and Phrases: monad, graded monad, flexible grading, algebraic theory, presentation, computational effect

ACM Reference Format:
Shin-ya Katsumata, Dylan McDermott, Tarmo Uustalu, and Nicolas Wu. 2022. Flexible Presentations of Graded Monads. *Proc. ACM Program. Lang.* 6, ICFP, Article 123 (August 2022), 29 pages. https://doi.org/10.1145/3547654

1 INTRODUCTION

Consider a language in which we can write backtracking computations using an operation `or` for nondeterministic choice, and an operation `cut` for pruning any remaining choices. Let t be the computation $\texttt{or}(\texttt{return}\ 17, \texttt{cut})$, which offers only 17 as a possible result, and prunes the rest of the search space. The computation $\texttt{or}(t, \texttt{return}\ 42)$ is equivalent to t, and more generally, the equation $\texttt{or}(x, y) \approx x$ is valid whenever we know that x definitely `cut`s.

We may seek to analyse a computation statically to determine whether it definitely `cut`s, and whether we can therefore apply the equation $\texttt{or}(x, y) \approx x$ to simplify a program. One way of performing such an analysis is through *grading*, an approach that goes back to effect systems [Lucassen and Gifford 1988]. We assign a grade $\perp$ to each computation we know will cut, assign some other grade 1 to computations that might not cut, and propagate this information throughout the

Authors' addresses: Shin-ya Katsumata, s-katsumata@nii.ac.jp, National Institute of Informatics, Tokyo, Japan; Dylan McDermott, dylanm@ru.is, Reykjavik University, Reykjavík, Iceland; Tarmo Uustalu, tarmo@ru.is, Reykjavik University, Reykjavík, Iceland and Tallinn University of Technology, Tallinn, Estonia; Nicolas Wu, n.wu@imperial.ac.uk, Imperial College London, London, United Kingdom.

2475-1421/2022/8-ART123
https://doi.org/10.1145/3547654

program.[1] For example, we can assign the grade $\perp$ to t because one of the arguments to `or` has grade $\perp$.

This approach has a well-established semantics using *graded monads*, which were introduced by Smirnov [2008] in connection with graded rings, and independently by Katsumata [2014] to model effects for which we track quantitative information, like whether a computation `cuts`. There is a graded monad Cut for modelling our backtracking example above (we define it in Section 2.4 below). It is a graded version of a monad described by Piróg and Staton [2017], and is somewhat similar to the familiar list monad.

Piróg and Staton [2017] show that their monad has a *presentation* in terms of operations for nondeterministic choice and cut, with several equations. Presentations of ordinary monads have number of important applications, for example in proving program equivalences [Kammar and Plotkin 2012] and in combining effects [Hyland et al. 2006]. We may expect there to be a similar presentation of Cut, using the notions of presentation for graded monads defined by Smirnov [2008], and subsequently by Milius et al. [2015], Dorsch et al. [2019] and Kura [2020]. However, all of these (roughly equivalent) notions of presentation have a deficiency, which makes them unsuitable for the backtracking example: they require all arguments to each operation to have the same grade. This is not immediately a problem for t above, since while `return` 17 has grade 1 and `cut` has grade $\perp$, we can also assign the grade 1 to `cut` as a safe overapproximation. But then we do not know that either argument of `or` definitely `cuts`, and have to assign the grade 1 to t. We would therefore fail to notice that t definitely `cuts`, and would not be able to apply the equation $\mathtt{or}(x, y) \approx x$. In fact, in the equational logics associated with these presentations, it is not possible to express equations with restrictions on the grades of variables. We cannot even *state* the equation $\mathtt{or}(x, y) \approx x$, where x stands for a computation of grade $\perp$.

We introduce a notion of *flexibly graded* presentation that does not suffer from these issues. Flexibly graded presentations are more general than the existing graded presentations (which we call 'rigidly graded' below for clarity). The leading idea is to relax the condition that all arguments to each operation have the same grade. Hence we can allow, for example, `or` to be applied to arguments of possibly different grades, and assign the grade $\perp$ when at least one of the arguments has grade $\perp$. We would then assign the grade $\perp$ to the computation t above. Flexibly graded presentations also allow equations over variables of different grades, like the example above.

Every flexibly graded presentation induces a graded monad, but relationship between the two is more subtle than for rigidly graded presentations. For the latter, there is a *correspondence* with a class of rigidly graded monads. The fact that enables such a correspondence is that, for each rigidly graded presentation, there is a graded monad with the same *algebras*, where an algebra for a presentation is a space equipped with interpretations of the operations, satisfying the equations. By generalizing to flexibly graded presentations we lose this property. We instead get a correspondence with a class of *flexibly graded monads* of McDermott and Uustalu [2022], which are different from graded monads (below we call the latter 'rigidly graded monads', again for clarity). Despite this, as we show, every flexibly graded monad does induce a canonical rigidly graded monad, and moreover the induced rigidly graded monad comes with interpretations of the operations of the presentation. In this sense, flexibly graded presentations do present rigidly graded monads. In fact, they present exactly the same class of rigidly graded monads as rigidly graded presentations do. We introduce flexibly graded presentations not in order to present more rigidly graded monads, but to enable more *natural* presentations of rigidly graded monads.

The purpose of this paper is to introduce flexibly graded presentations and to develop their theory. As well as the backtracking example, we give several other instances. Graded monads

[1] We in fact also need a third grade $\top$ to get everything to work correctly, but this does not come up in the introduction.

have several applications, for example in semantics of type-and-effect systems [Katsumata 2014; Mycroft et al. 2016], in process semantics [Dorsch et al. 2019; Milius et al. 2015], and in probability theory [Fritz and Perrone 2019]. We intend for our results to be applicable to all of these.

Contributions.

- We introduce the examples of rigidly graded monads we use (Section 2). Some of these are new (e.g. our gradings of global state, of the stack monad, and of backtracking with cut).
- We outline the theory of rigidly graded presentations (Section 4), filling in some gaps from the existing literature. In particular, we show that operations of rigidly graded presentations induce *rigidly graded algebraic operations.*
- Our main contribution is the introduction of flexibly graded presentations (Section 6). We develop much of their theory, defining a flexibly graded equational logic and notions of algebra and of flexibly graded algebraic operation. We show that our examples have natural flexibly graded presentations.
- As a step towards proving a correspondence for flexibly graded monads, we introduce a notion of *flexibly graded clone* (Section 7). We also introduce *rigidly graded clones* in the same section.
- We prove a correspondence between flexibly graded presentations and those flexibly graded monads that a colimit condition (Section 8).

2 GRADED MONADS AND EXAMPLES

The *grades* are elements of a *partially ordered monoid* (pomonoid) $\mathbb{E} = (|\mathbb{E}|, \leq, 1, \cdot)$; that is, $(|\mathbb{E}|, \leq)$ is a partially ordered set (poset), and $(|\mathbb{E}|, 1, \cdot)$ is a monoid for which $\cdot$ is monotone. (More generally, we could consider an arbitrary small monoidal category $\mathbb{E}$ of grades, but we restrict to pomonoids for simplicity.) The order enables overapproximation of grades. The grade of a trivial computation is 1, and the grade of a sequence of two computations is provided by the $\cdot$ operator.

Definition 2.1. An $\mathbb{E}$*-graded set* X consists of a set Xe for each grade $e \in |\mathbb{E}|$ (the *elements of grade* e), and a *coercion* function $(e \leq e')^* : Xe \longrightarrow Xe'$ for each $e \leq e' \in |\mathbb{E}|$, functorial in the sense that $(e \leq e)^* = \mathrm{id}_{Xe}$ and $(e \leq e'')^* = (e' \leq e'')^* \circ (e \leq e')^*$. A *grade-preserving function* $f : X \Rightarrow Y$ is a family of functions $f_e : Xe \longrightarrow Ye$, natural in the sense that $f_{e'} \circ (e \leq e')^* = (e \leq e')^* \circ f_e$.

We often omit the prefix $\mathbb{E}$- from $\mathbb{E}$-graded.

Remark. In other words, graded sets are functors from $\mathbb{E}$ to **Set** (presheaves over $\mathbb{E}$). Grade-preserving functions are natural transformations between such functors.

The following is the notion of graded monad introduced by Smirnov [2008], Melliès [2012], and Katsumata [2014]. They are similar to ordinary monads, except that instead of having a set TX of computations for every set X of values, they have a *graded* set RX of computations for every set X. We give the definition only for the category of sets and functions (which is all we need), and in terms of a *Kleisli extension* operation $(-)^\dagger$. We say 'rigidly graded monad' instead of just 'graded monad', to more clearly distinguish between these and the *flexibly* graded monads defined below.

Definition 2.2. A *rigidly* $\mathbb{E}$*-graded monad* R consists of an $\mathbb{E}$-graded set RX and *unit* function $\eta_X : X \longrightarrow RX1$ for each set X, and a *Kleisli extension* operator that maps every function $f : X \longrightarrow RYe$ to a grade-preserving function $f^\dagger : RX \Rightarrow RY(- \cdot e)$ (i.e., a family of functions $f^\dagger_d : RXd \longrightarrow RY(d \cdot e)$ natural in d); Kleisli extension is required to be natural in e, and to satisfy the following unitality and associativity laws.

$$f^\dagger_1 \circ \eta_X = f \qquad \mathrm{id}_{RXd} = (\eta_X)^\dagger_d \qquad (g^\dagger_e \circ f)^\dagger_d = g^\dagger_{d\cdot e} \circ f^\dagger_d \quad (\text{for } f : X \longrightarrow RYe, g : Y \longrightarrow RZe')$$

The unit is also known as *return* and the Kleisli extension as *bind*, written $t \gg\!= f$ instead of $f^\dagger t$.

Instead of having a graded set RX for each ordinary (ungraded) set X, *flexibly graded monads* [McDermott and Uustalu 2022] have a graded set TX for each graded set X. The intuition is that the elements of X may themselves be computations, and may have grades.

Definition 2.3. A *flexibly* $\mathbb{E}$*-graded monad* T consists of an $\mathbb{E}$-graded set TX and *unit* grade-preserving function $\eta_X : X \Rightarrow TX$ for each $\mathbb{E}$-graded set X, and a *Kleisli extension* operator that sends every grade-preserving function $f : X \Rightarrow TY(- \cdot e)$ to a grade-preserving function $f^\dagger : TX \Rightarrow TY(- \cdot e)$, natural in e, and satisfying the following unitality and associativity laws:

$$f^\dagger \circ \eta_X = f \qquad \mathrm{id}_{TX} = (\eta_X)^\dagger \qquad \left(g^\dagger_{-\cdot e} \circ f\right)^\dagger = g^\dagger_{-\cdot e} \circ f^\dagger \quad (f : X \Rightarrow TY(- \cdot e), g : Y \Rightarrow TZ(- \cdot e'))$$

2.1 Example: Writer

Our first example involves a graded writer monad Wr^{M}; these are analogous to the usual non-graded writer monads. We use this example in our discussion of rigidly graded presentations in Section 4. As such, the rigidly graded presentation of Wr^{M} is perfectly natural, and there is no obvious benefit to giving a flexibly graded presentation of Wr^{M}. This is not the case for our other examples.

For each monoid $\mathsf{M} = (M, \varepsilon, \otimes)$, there is an non-graded writer monad given by $T^{\mathsf{M}}X = M \times X$, for which we can define an operation $\mathit{tell}_{p,X} : TX \to TX$ for producing an output $p \in M$, by $\mathit{tell}_{p,X}(p', x) = (p \otimes p', x)$. These monads can be presented by the operations tell_p (with suitable equations).

Rigidly graded writer monads Wr^{M} are analogous. Fix a $\mathbb{E}$*-graded monoid* M, that is, an $\mathbb{E}$-graded set M, together with a *unit* element $\varepsilon \in M1$ and family of *multiplication* functions $\otimes_{e_1,e_2} : Me_1 \times Me_2 \to M(e_1 \cdot e_2)$ natural in $e_1, e_2 \in \mathbb{E}$, satisfying $\varepsilon \otimes p = p = p \otimes \varepsilon$ and $(p \otimes q) \otimes r = p \otimes (q \otimes r)$ for each $p \in Me_1, q \in Me_2, r \in Me_3$. (We write multiplication infix and omit the subscripts.) For example, we could let $\mathbb{E}$ be the pomonoid of natural numbers with addition and their usual ordering, and let Me be the set of strings of length at most e (over some set of characters), with concatenation as the multiplication. We could also, with the same $\mathbb{E}$, let Me be the powerset of some fixed set, restricted to subsets of cardinality at most e, with union as multiplication. We define Wr^{M} in exactly the same way as the ungraded writer monad, except that at grade e we use Me:

$$\mathrm{Wr}^{\mathsf{M}}Xe = Me \times X \quad (e \le e')^*(p, x) = ((e \le e')^* p, x)$$
$$\eta_X x = (\varepsilon, x) \quad f^\dagger_d(p, x) = \text{let } (p', y) = f\,x \text{ in } (p \otimes p', y)$$

We again define an operation for outputting an element $p \in Me'$, but with the appropriate grading:

$$\mathit{tell}_{p,X,d} : \mathrm{Wr}^{\mathsf{M}}Xd \to \mathrm{Wr}^{\mathsf{M}}X(e' \cdot d) \qquad \mathit{tell}_{p,X,d}(p', x) = (p \otimes p', x)$$

In Section 4, we give a rigidly graded presentation of Wr^{M} involving the operations tell_p (which are *rigidly graded algebraic operations* in the sense of Section 4.4).

2.2 Example: Global State

For our second example, we consider computations that read from and write to a global V-valued state, where V is a finite set. These computations can be interpreted using the ordinary state monad $V \Rightarrow V \times (-)$. We give rigidly and flexibly graded versions of this monad.

The grades e in this example are binary relations on V, which we represent as functions $e : V \to \mathcal{P}V$. The idea is that, if a computation of grade e is run with initial state v, and terminates with final state v', then $v' \in ev$. Grades are allowed to overapproximate these relations, so the ordering is given by set inclusion. The unit grade 1 is the diagonal relation, and the multiplication $e \cdot e'$ of grades is the composition of relations. We write Rel_V for the pomonoid of grades.

$$e \le e' \text{ iff } \forall v \in V.\, ev \subseteq e'v \qquad 1 = \lambda v.\,\{v\} \qquad e \cdot e' = \lambda v.\{v'' \in V \mid \exists v'.\, v' \in ev \wedge v'' \in e'v'\}$$

The rigidly Rel_V-graded monad State is defined analogously to the ordinary state monad, so a computation is a function that sends each state v to a pair of a state v' and a result x, except that the initial state v and final state v' are required to be related by the grade e:

$$\mathsf{State}\,X\,e = \textstyle\prod_{v:V} \coprod_{v':ev} X \qquad (e \le e')^* t = \lambda v.\, tv$$
$$\eta_X x = \lambda v.\,(v, x) \qquad f_d^\dagger t = \lambda v.\,\mathrm{let}\ (v', x) = tv\ \mathrm{in}\ fxv'$$

(Grades e in which ev is not a singleton are in a sense unnecessary: each computation maps each initial state to a single final state, so we could also work with functions $e : V \longrightarrow V$. The reason for allowing ev to contain multiple values is to enable overapproximation of grades.)

There is also a flexibly Rel_V-graded monad $\mathsf{State}_{\mathrm{flex}}$. The definition is again similar to that of the ordinary state monad, except that X is a graded set. In the definition, we use grades $(v, v')\blacktriangleright e$.

$$(v, v')\blacktriangleright e = \lambda w'.\,\mathrm{if}\ w' = v'\ \mathrm{then}\ ev\ \mathrm{else}\ V$$
$$\mathsf{State}_{\mathrm{flex}}\,X\,e = \textstyle\prod_{v:V} \coprod_{v':V} X((v, v')\blacktriangleright e) \qquad (e \le e')^* t = \lambda v.\, tv$$
$$\eta_{X,d} x = \lambda v.\,(v, (d \le (v, v)\blacktriangleright d)^* x) \qquad f_d^\dagger t = \lambda v.\,\mathrm{let}\ (v', x) = tv\ \mathrm{in}\ f_{(v,v')\blacktriangleright d} xv'$$

A computation $t \in \mathsf{State}Xe$ therefore takes an initial state v and produces another state v', and some $x \in X((v, v')\blacktriangleright e)$. Here x should be thought of as a further computation that can interact with the state; v' is unrestricted, but the grade of x ensures that, if we run x with initial state v' and get a final state v'', then we have $v'' \in ev$.

For an example, we define for any set X a graded set $\hat{X}$, by $\hat{X}e = X$ if $\forall v.\, v \in ev$, and $\hat{X}e = \emptyset$ otherwise. This graded set should be thought of as containing computations that return elements of X without interacting with the state at all, so $\hat{X}e$ only contains a computation when e allows the state to be left unchanged (in other words, contains the diagonal relation), and then computations are just elements of X. The grade $(v, v')\blacktriangleright e$ contains the diagonal exactly when $v' \in ev$. A computation $t \in \mathsf{State}_{\mathrm{flex}}\hat{X}e$ sends each initial state v to a state v' and element $x \in \hat{X}((v, v')\blacktriangleright e)$, but the latter condition amounts to $x \in X$ and $v' \in ev$. Here the final state is v', which is required to be related to v by e, so t is equivalently an element of $\mathsf{State}Xe$. (This is the basis of the construction in Section 5 below.) For a more interesting example, we can nest computations over $\mathsf{State}_{\mathrm{flex}}$:

$$\mathsf{State}_{\mathrm{flex}}(\mathsf{State}_{\mathrm{flex}}\hat{X})e \cong \textstyle\prod_{v:V} \coprod_{v':V} \prod_{w':V} \coprod_{w'':((v,v')\blacktriangleright e)\,w'} X$$

Here the condition on w'' says if we pass the final state v' of the outer computation as the initial state w' of the inner computation, then we get $w'' \in ev$.

The ordinary state monad has a presentation by operations for reading from and writing to the state [Plotkin and Power 2002]. The rigidly graded monad State similarly has a flexibly graded presentation, by an operation get_e for each $e : V \longrightarrow \mathsf{Rel}_V$, and an operation put_w for each $w \in V$:

$$get_{e,X,d} : \textstyle\prod_v \mathsf{State}X(ev \cdot d) \longrightarrow \mathsf{State}X(\lambda w.\,(ew \cdot d)w) \qquad get_{e,X,d} f = \lambda v.\, fvv$$
$$put_{w,X,d} : \mathsf{State}Xd \longrightarrow \mathsf{State}X(\lambda\_.\, dw) \qquad put_{w,X,d} t = \lambda\_.\, tw$$

The computation $get_{e,X,d}(\lambda w.tw)$ gets the initial value v of the state, and then continues as tv. The computation $put_{w,X,d}t$ sets the state to w and then continues as t. The initial state is irrelevant, and this is reflected in the grade $(\lambda\_.\, dw)$. For example, if $V = \{\mathrm{true}, \mathrm{false}\}$, then we have a computation $t_{\mathrm{neg}} = get(\lambda v.\, put_{\neg v}(\eta_V v)) \in \mathsf{State}V(\lambda w.\,\{\neg w\})$ that negates the state.

In get, the computations given as arguments are allowed to have different grades. This is essential to avoid overapproximating the grade in examples like t_{neg}. If we had instead said that all of these computations had to have the same grade, the best grade we would be able to give to t_{neg} is the total relation $\lambda\_.\,V$. In fact, if we had required the grades to be equal in this way, *get* and *put* would not suffice for a presentation of State. This is part of the motivation for flexibly graded presentations, in which we allow arguments to have different grades. While there is a rigidly graded presentation of State, there is no rigidly graded presentation in terms of *get* and *put*. Flexibly graded operations are essential to give a natural presentation of State.

2.3 Example: A Stack

Our next example involves computations interacting with a stack of values drawn from a finite set V. Each computation either: pushes a value $v \in V$ onto the stack, and then continues with another computation; or attempts to pop a value from the stack, continuing as one computation if the stack was empty, and continuing as one of $|V|$ other computations if there was a value to pop. This example is a grading of Goncharov's stack monad [2013].

Grades are pairs (ℓ, u) of an lower bound $\ell \in \{-\infty\} \cup \mathbb{Z}$ and an upper bound $u \in \mathbb{Z} \cup \{\infty\}$ on the net change in the height of the stack. The ordering is given by $(\ell, u) \le (\ell', u')$ if $\ell' \le \ell$ and $u \le u'$. The unit grade is $(0, 0)$, and multiplication of grades is given by $(\ell_1, u_1) \cdot (\ell_2, u_2) = (\ell_1 + \ell_2, u_1 + u_2)$. We call this pomonoid $\mathsf{Interval}$.

We give a rigidly $\mathsf{Interval}$-graded monad Stk that has interpretations of the push and pop operations above, as functions

$$\begin{aligned} push_{v,X,(\ell,u)} &: \mathrm{Stk}X(\ell, u) \to \mathrm{Stk}X(\ell + 1, u + 1) && \text{(for each } v \in V\text{)} \\ pop_{X,(\ell,u)} &: \mathrm{Stk}X(\ell, u) \times (V \Rightarrow \mathrm{Stk}X(\ell + 1, u + 1)) \to \mathrm{Stk}X(\ell, u) \end{aligned}$$

We first define some notation. For a set V, let $\mathrm{List}V$ be the set of finite, possibly empty lists over the set V. We write $|\vec{v}|$ for the length of a list $\vec{v}$, and $\mathrm{List}_{\ell..u}V$ and $\mathrm{List}_{\rho}V$ for the subsets of $\mathrm{List}V$ containing the lists $\vec{v}$ such that $\ell \le |\vec{v}| \le u$ and such that $|\vec{v}| = \rho$ respectively. We also write the cons operation as $v :: \vec{v}$, write concatenation of lists as $\vec{v} +\!\!+ \vec{v}'$, and write $\mathrm{head}\,\vec{v}$ and $\mathrm{tail}\,\vec{v}$ for the first and remaining elements of a non-empty list $\vec{v}$.

A computation of grade (ℓ, u) that returns elements of X is in particular a function $t : \prod_{\vec{v}:\mathrm{List}V}(\mathrm{List}_{|\vec{v}|+\ell..|\vec{v}|+u}V \times X)$, which sends each initial stack $\vec{v}$ to a pair of a final stack of the appropriate length and a result from X. The graded monad Stk restricts to only a subset of these computations; this restriction has to do with the fact that computations are finite: for every t, there is some natural number ρ, such that t only uses at most the first ρ values on the initial stack. The graded set $\mathrm{Stk}X$ of computations that return elements of the set X is given by:

$$\begin{aligned} \mathrm{Stk}X(\ell, u) = \{t : {}& \textstyle\prod_{\vec{v}:\mathrm{List}V}(\mathrm{List}_{|\vec{v}|+\ell..|\vec{v}|+u}V \times X) \\ & \mid \exists \rho \in \mathbb{N}.\, \forall \vec{v} \in \mathrm{List}_{\rho}V, \vec{w} \in \mathrm{List}V.\, \mathrm{fst}(t(\vec{v} +\!\!+ \vec{w})) = \mathrm{fst}(t\,\vec{v}) +\!\!+ \vec{w} \\ & \qquad \wedge\, \mathrm{snd}(t(\vec{v} +\!\!+ \vec{w})) = \mathrm{snd}(t\,\vec{v})\} \end{aligned}$$

Coercions $((\ell, u) \le (\ell', u'))^*$ just use the inclusions $\mathrm{List}_{|\vec{v}|+\ell..|\vec{v}|+u}V \subseteq \mathrm{List}_{|\vec{v}|+\ell'..|\vec{v}|+u'}V$. The unit and Kleisli extension are similar to those of the state monad:

$$\eta_X x = \lambda\vec{v}.(x, \vec{v}) \qquad f^{\dagger}_{(\ell,u)} t = \lambda\vec{v}.\ \mathrm{let}\ (\vec{v}', x) = t\,\vec{v}\ \mathrm{in}\ f\,x\,\vec{v}'$$

The interpretations of the push and pop operations are the following:

$$push_{v,X,(\ell,u)}\, t = \lambda\vec{v}'.\, t\,(v :: \vec{v}') \qquad pop_{X,(\ell,u)}\,(t_0, t) = \lambda\vec{v}.\ \mathrm{if}\ |\vec{v}| = 0\ \mathrm{then}\ t_0\,\vec{v}\ \mathrm{else}\ t\,(\mathrm{head}\,\vec{v})\,(\mathrm{tail}\,\vec{v})$$

We also define a flexibly graded monad $\mathrm{Stk}_{\mathrm{flex}}$ for this example. It is similar to the rigidly graded monad Stk, except that there is no restriction on the stack produced by the computation (which should be thought of as an intermediate stack). Instead, the grade of the element of X compensates for having a stack of the wrong length.

$$\begin{aligned}\mathrm{Stk}_{\mathrm{flex}}X(\ell,u) = \{t : \textstyle\prod_{\vec{v}:\mathrm{List}V}\coprod_{\vec{v}':\mathrm{List}V} X(\ell-(|\vec{v}'|-|\vec{v}|),\ u-(|\vec{v}'|-|\vec{v}|)) \\ \mid \exists\rho\in\mathbb{N}.\,\forall\vec{v}\in\mathrm{List}_\rho V, \vec{w}\in\mathrm{List}V.\,\mathrm{fst}(t(\vec{v} +\!\!+ \vec{w})) = \mathrm{fst}(t\,\vec{v}) +\!\!+ \vec{w} \\ \wedge\, \mathrm{snd}(t(\vec{v} +\!\!+ \vec{w})) = \mathrm{snd}(t\,\vec{v})\}\end{aligned}$$

There are also push and pop operations for the flexibly graded monad $\mathrm{Stk}_{\mathrm{flex}}$. These are defined in exactly the same way as the operations for the rigidly graded monad above, and have almost identical types (the key difference being that X now ranges over graded sets):

$$\begin{aligned}push_{v,X,(\ell,u)} &: \mathrm{Stk}_{\mathrm{flex}}X(\ell,u) \longrightarrow \mathrm{Stk}_{\mathrm{flex}}X(\ell+1,u+1) && \text{(for each } v\in V\text{)}\\ pop_{X,(\ell,u)} &: \mathrm{Stk}_{\mathrm{flex}}X(\ell,u)\times(V\Rightarrow \mathrm{Stk}_{\mathrm{flex}}X(\ell+1,u+1)) \longrightarrow \mathrm{Stk}_{\mathrm{flex}}X(\ell,u)\end{aligned}$$

$\mathrm{Stk}_{\mathrm{flex}}$ is actually *presented* by these operations. The reason we can give such a presentation is that every $t\in\mathrm{Stk}_{\mathrm{flex}}X(\ell,u)$ can be written using only η and finitely many pushes and pops. Indeed, if ρ is a witness from the definition of $\mathrm{Stk}_{\mathrm{flex}}X(\ell,u)$, then:

$$\begin{aligned}t &= push_{\mathrm{fst}(t[\,])}(\eta(\mathrm{snd}(t[\,]))) && \text{(if } \rho = 0\text{)}\\ t &= pop\big(push_{\mathrm{fst}(t[\,])}(\eta(\mathrm{snd}(t[\,]))),\ \lambda v.\,\lambda\vec{w}.\,t(v :: \vec{w})\big) && \text{(if } \rho > 0\text{)}\end{aligned}$$

Here we have omitted some of the subscripts, and written $push_{[w_1,\dots,w_n]}$ for $push_{w_n}\circ\cdots\circ push_{w_1}$. This provides a recursive procedure for writing t using $push_v$, pop and η. The recursion is well-founded because the witness ρ for $\lambda\vec{w}.\,t(v :: \vec{w})$ is strictly smaller than the witness for t.

2.4 Example: Backtracking Nondeterminism, with Cut

We give a rigidly graded monad for finite nondeterminism with a cut operator, similar to the ungraded monad described by Piróg and Staton [2017].

For this example, the pomonoid of grades is $\{\bot \le 1 \le \top\}$, with multiplication given by $\bot\cdot e = \bot$, $1\cdot e = e$ and $\top\cdot e = \top$. The grade $\bot$ means 'definitely cuts'; the unit grade 1 means 'definitely cuts or produces at least one value'; and $\top$ imposes no restrictions. The rigidly graded monad Cut is defined as follows.

$$\mathrm{Cut}Xe = \{(\vec{x},c)\in\mathrm{List}X\times\{\bot,\top\} \mid (e=\bot \Rightarrow c=\bot)\wedge(e=1\Rightarrow c=\bot\vee\vec{x}\neq[\,])\}$$

$$(e\le e')^*(\vec{x},c) = (\vec{x},c) \qquad \eta_X x = ([x],\top) \qquad f_d^\dagger([\,],c) = ([\,],c)$$

$$f_d^\dagger(x :: \vec{x}',c) = \mathrm{let}\ (\vec{y},c') = fx\ \mathrm{in\ if}\ c'=\bot\ \mathrm{then}\ (\vec{y},c')\ \mathrm{else\ let}\ (\vec{y}',c'') = f_d^\dagger(\vec{x}',c)\ \mathrm{in}\ (\vec{y} +\!\!+ \vec{y}',c'')$$

A computation over X is a list of values drawn from X, plus a tag that indicates whether the computation cuts ($\bot$ for 'cuts', $\top$ for 'does not cut'). Kleisli extension applies f to each of the elements of the list $\vec{x}$ and concatenates the results, except that everything after the first cut is discarded.

3 LOCALLY GRADED CATEGORIES AND ALGEBRAS

In classical universal algebra, the monad T corresponding to a presentation (Σ,E) is determined by the fact that it has the same algebras as the presentation. Precisely, (Σ,E)-algebras and T-algebras both form concrete categories (categories equipped with a functor into **Set**), and there is an isomorphism between these concrete categories.

Similar considerations apply to both rigidly graded presentations and flexibly graded presentations. In this case however, ordinary category theory does not suffice. Instead of just having sets of morphisms between algebras, we instead have graded sets of morphisms between algebras. Algebras form *locally graded categories* in the sense of the following definition.

Definition 3.1 (Wood [1976]). A *locally* $\mathbb{E}$*-graded category* C consists of

- a collection $|C|$ of *objects*;
- for each $X, Y \in |C|$, an $\mathbb{E}$-graded set $C(X, Y)$ of *morphisms* from X to Y; we write $f : X \mathrel{-e\!\!\to} Y$ to indicate that $f \in C(X, Y)e$;
- for each $X \in |C|$, a morphism $\mathrm{id}_X : X \mathrel{-1\!\!\to} X$;
- for each $f : X \mathrel{-e\!\!\to} Y$ and $g : Y \mathrel{-e'\!\!\to} Z$, a morphism $g \circ f : X \mathrel{-e \cdot e'\!\!\to} Z$;

such that composition is unital ($\mathrm{id}_Y \circ f = f = f \circ \mathrm{id}_X$), associative ($(h \circ g) \circ f = h \circ (g \circ f)$), and natural ($(e_1 \cdot e_2 \le e_1' \cdot e_2')^*(g \circ f) = (e_1' \le e_2')^* g \circ (e_1 \le e_2)^* f$).

Locally graded categories were introduced by Wood [1976] and used by Gaboardi et al. [2021] for the semantics of graded Hoare logic. The terminology we use is from Levy [2019]. Instead of locally graded categories, it is possible to work instead with *actegories*, like [Fujii et al. 2016], but locally graded categories turn out to be more convenient for us.

Remark. Locally graded categories are an instance of enriched categories. In detail, the category $[\mathbb{E}, \mathbf{Set}]$ forms a monoidal category with Day convolution as tensor:

$$I = \mathbb{E}(1, -) \qquad X \otimes Y = \int^{e_1, e_2 \in |\mathbb{E}|} \mathbb{E}(e_1 \cdot e_2, -) \times X e_1 \times Y e_2$$

Locally $\mathbb{E}$-graded categories are $[\mathbb{E}, \mathbf{Set}]$-enriched categories. This correspondence extends to locally graded and enriched functors and natural transformations (we define locally graded functors below), so the 2-categories of $\mathbb{E}$-graded categories and $[\mathbb{E}, \mathbf{Set}]$-enriched categories are equivalent.

We use the following locally graded category of graded sets throughout. It has the same role here as **Set** does in the classical presentation–monad correspondence.

Definition 3.2. The locally graded category $\mathbf{GSet}_{\mathbb{E}}$ has as objects $\mathbb{E}$-graded sets, and as morphisms $X \mathrel{-e\!\!\to} Y$ grade-preserving functions $X \Rightarrow Y(- \cdot e)$. The coercion $(e \le e')^* f : X \mathrel{-e'\!\!\to} Y$ of $f : X \mathrel{-e\!\!\to} Y$ uses coercions in Y on the left below. The identity on X is the identity grade-preserving function $\mathrm{id}_X : X \Rightarrow X$, the composition $g \circ f : X \mathrel{-e \cdot e'\!\!\to} Z$ of $f : X \mathrel{-e\!\!\to} Y$ and $g : Y \mathrel{-e'\!\!\to} Z$ is defined on the right below.

$$((e \le e')^* f)_d : Xd \xrightarrow{f_d} Y(d \cdot e) \xrightarrow{(d \cdot e \le d \cdot e')^*} Y(d \cdot e') \qquad (g \circ f)_d : Xd \xrightarrow{f_d} Y(d \cdot e) \xrightarrow{g_{d \cdot e}} Z(d \cdot e \cdot e')$$

Using the notion of morphism of graded sets given in this definition, the Kleisli extensions of each rigidly graded monad R and each flexibly graded monad T have the following types:

$$\frac{f : X \longrightarrow RYe}{f^\dagger : RX \mathrel{-e\!\!\to} RY} \qquad \frac{f : X \mathrel{-e\!\!\to} TY}{f^\dagger : TX \mathrel{-e\!\!\to} TY}$$

The appropriate notion of functor between locally graded categories is as follows:

Definition 3.3 (Wood [1976]). A *functor* $F : C \longrightarrow \mathcal{D}$ between locally graded categories consists of an object mapping $F : |C| \longrightarrow |\mathcal{D}|$ and a mapping of morphisms as on the left below; these are required to preserve identities and composition, and to be natural, as on the right below.

$$\frac{f : X \mathrel{-e\!\!\to} Y}{Ff : FX \mathrel{-e\!\!\to} FY} \qquad \begin{gathered} F\mathrm{id}_X = \mathrm{id}_{FX} \\ F(g \circ f) = Fg \circ Ff \\ F((e \le e')^* f) = (e \le e')^*(Ff) \end{gathered}$$

For an example, let T be a flexibly $\mathbb{E}$-graded monad. The assignment $X \mapsto TX$ of graded sets to graded sets extends to a functor $T : \mathbf{GSet}_{\mathbb{E}} \longrightarrow \mathbf{GSet}_{\mathbb{E}}$ by defining $Tf = (\eta_Y \circ f)^\dagger : TX \mathrel{-e\!\!\to} TY$ for each morphism $f : X \mathrel{-e\!\!\to} Y$ of graded sets. We use this construction in Section 8 below. (There is also a similar construction for rigidly graded monads, see Section 8.2.)

We define the locally graded categories of algebras for rigidly graded monads and for flexibly graded monads. In both cases an algebra is essentially a graded set in which we can interpret computations. These are defined in terms of an extension operator analogous to Kleisli extension.

Definition 3.4. Let R be a rigidly $\mathbb{E}$-graded monad. An R-*algebra* A consists of an $\mathbb{E}$-graded set A (the *carrier*) and an *extension* operator $(-)^{\ddagger}$ that maps each function $f : X \to Ae$ to a grade-preserving function $f^{\ddagger} : RX \Rightarrow A(- \cdot e)$; this is required to satisfy the following equations:

$$f^{\ddagger}_1 \circ \eta_X = f \qquad g^{\ddagger}_{d\cdot e} \circ f^{\dagger}_d = (g^{\ddagger}_e \circ f)^{\ddagger}_d$$

A *morphism* $f : \mathsf{A} \mathrel{-e\!\!\to} \mathsf{A}'$ of R-algebras is a morphism $f : A \mathrel{-e\!\!\to} A'$ of graded sets satisfying $f_{d\cdot e} \circ g^{\ddagger}_d = (f_e \circ g)^{\ddagger}_d$. These form a locally $\mathbb{E}$-graded category $\mathbf{EM}(\mathsf{R})$, with coercions, identities and composition as in $\mathbf{GSet}_{\mathbb{E}}$. There is a *forgetful functor* $U_{\mathsf{R}} : \mathbf{EM}(\mathsf{R}) \to \mathbf{GSet}_{\mathbb{E}}$ that sends each R-algebra to its carrier, and each morphism of R-algebras to itself.

The *free* R-*algebra* on a set X is the R-algebra consisting of the graded set RX equipped with the Kleisli extension operator $(-)^{\dagger}$.

Instead of defining algebras in terms of the extension operator $(-)^{\ddagger}$, there is also a definition in terms of a family of functions $a_{e_1,e_2} : R(Ae_2)e_1 \to A(e_1 \cdot e_2)$ [Fujii et al. 2016] (analogous to the usual definition of monad algebra in terms of a function $TA \to A$), but we do not need this here.

Example 3.5. Recall the rigidly $\mathbb{E}$-graded writer monad Wr^{M} (Section 2.1), and let A be a Wr^{M}-algebra. By specializing the extension operator $(-)^{\ddagger} : \mathbf{Set}(X, Ae) \to \mathbf{GSet}_{\mathbb{E}}(M(-) \times X, A)e$ to $X = 1$ and defining $[\![\mathsf{tell}_p]\!]_e a = (\lambda\_.a)^{\ddagger}_d(p, \star)$, we obtain a grade-preserving function $[\![\mathsf{tell}_p]\!] : A \Rightarrow A(d \cdot -)$ for each $p \in Md$; this satisfies the equations

$$(d\cdot e \le d'\cdot e)^*([\![\mathsf{tell}_p]\!]_e a) = [\![\mathsf{tell}_{(d\le d')^* p}]\!]_e a \qquad a = [\![\mathsf{tell}_\varepsilon]\!]_e a \qquad [\![\mathsf{tell}_p]\!]_{d'\cdot e}([\![\mathsf{tell}_{p'}]\!]_e a) = [\![\mathsf{tell}_{p\otimes p'}]\!]_e a$$

This construction is one half of a bijection. For the other half, given a graded set A and grade-preserving functions $[\![\mathsf{tell}_p]\!] : A \Rightarrow A(d \cdot -)$ satisfying these equations, A is the carrier of a Wr^{M}-algebra A with extension operator $f^{\ddagger}_d(p, x) = [\![\mathsf{tell}_p]\!]_e(fx)$. In the case of the free Wr^{M}-algebra $(\mathrm{Wr}^{\mathsf{M}}X, (-)^{\dagger})$, we have $[\![\mathsf{tell}_p]\!]_e = tell_{p,X,e}$ where $tell_p$ is as defined in Section 2.1.

Under this bijection, a morphism $f : \mathsf{A} \mathrel{-e\!\!\to} \mathsf{A}'$ of Wr^{M}-algebras is equivalently a morphism $f : A \mathrel{-e\!\!\to} A'$ of graded sets that preserves each $[\![\mathsf{tell}_p]\!]$ in the sense that $f_{d\cdot d'}([\![\mathsf{tell}_p]\!]_{d'} a) = [\![\mathsf{tell}_p]\!]_{d'\cdot e}(f_{d'} a)$. The bijection between the two forms of algebras therefore extends to an isomorphism of locally graded categories (between $\mathbf{EM}(\mathsf{Wr}^{\mathsf{M}})$ and the locally graded category of graded sets equipped with $[\![\mathsf{tell}_p]\!]$). This isomorphism moreover preserves carriers and sends morphisms to themselves.

Below we show that Wr^{M} has a rigidly graded presentation by a family of unary operations tell_p; this isomorphism is precisely the reason why such a presentation induces Wr^{M}. As the notation suggests, the functions $[\![\mathsf{tell}_p]\!]$ are the interpretations of the operations tell_p.

In the above example, we have an isomorphism of locally graded categories that preserves carriers and morphisms. Functors like this appear throughout this paper, we say they are *over* $\mathbf{GSet}_{\mathbb{E}}$. They are analogous to the concrete functors from the classical correspondence. (The functors U, U' in the following definition are typically forgetful functors, like U_{R} above.)

Definition 3.6. Let C, C' be locally graded categories equipped with functors $U : C \to \mathbf{GSet}_{\mathbb{E}}$ and $U' : C' \to \mathbf{GSet}_{\mathbb{E}}$. A functor $F : C \to C'$ is *over* $\mathbf{GSet}_{\mathbb{E}}$ when $U' \cdot F = U$.

Definition 3.7. Let T be a flexibly $\mathbb{E}$-graded monad. A T-*algebra* A consists of an $\mathbb{E}$-graded set A (the *carrier*) and an *extension* operator $(-)^{\ddagger}$ that maps each grade-preserving function $f : X \Rightarrow A(- \cdot e)$

to a grade-preserving function $f^{\ddagger} : TX \Rightarrow A(- \cdot e)$; this is required to satisfy the following equations:

$$f^{\ddagger} \circ \eta_X = f \qquad g^{\ddagger}_{-\cdot e} \circ f^{\dagger} = (g^{\ddagger}_{-\cdot e} \circ f)^{\ddagger} \quad (f : X \Rightarrow TY(- \cdot e), g : Y \Rightarrow A(- \cdot e'))$$

A *morphism* $f : \mathsf{A} \xrightarrow{e} \mathsf{A}'$ of T-algebras is a morphism $f : A \xrightarrow{e} A'$ of graded sets satisfying $f \circ g^{\ddagger} = (f \circ g)^{\ddagger}$. These form a locally $\mathbb{E}$-graded category $\mathbf{EM}(\mathsf{T})$, with coercions, identities and composition as in $\mathbf{GSet}_{\mathbb{E}}$. There is a *forgetful functor* $U_{\mathsf{T}} : \mathbf{EM}(\mathsf{T}) \longrightarrow \mathbf{GSet}_{\mathbb{E}}$ that sends each T-algebra to its carrier, and each morphism of T-algebras to itself.

The *free* T-*algebra* on a graded set X is the T-algebra consisting of the graded set TX equipped with the Kleisli extension operator $(-)^{\dagger}$.

Example 3.8. In Example 3.5, we characterize algebras for the rigidly graded monad Wr^{M} in terms of operations tell_p. We give a similar characterization for the flexibly graded monad $\mathsf{Stk}_{\mathsf{flex}}$ for V-valued stacks (Section 2.3). In this case, the operations are push_v and pop.

Let A be a $\mathsf{Stk}_{\mathsf{flex}}$-algebra. Define[2] a graded set P by $Pe = \{\star \mid (0,0) \le e\} \cup \{v \in V \mid (1,1) \le e\}$, where $\star$ is not in V (so the union is disjoint). There are bijections $\vartheta : A(\ell,u) \times (V \Rightarrow A(\ell{+}1, u{+}1)) \cong \mathbf{GSet}_{\mathsf{Interval}}(P, A)(\ell, u)$, defined by

$$(\vartheta(a,f))_e \star = ((\ell,u) {\le} (e{\cdot}(\ell,u)))^* a \qquad (\vartheta(a,f))_e v = ((\ell{+}1,u{+}1) {\le} (e{\cdot}(\ell,u)))^* (f v)$$

By applying the extension operator $(-)^{\ddagger} : \mathbf{GSet}_{\mathsf{Interval}}(P, A)(\ell,u) \longrightarrow \mathbf{GSet}_{\mathsf{Interval}}(\mathsf{Stk}_{\mathsf{flex}} P, A)(\ell,u)$ we can therefore define for each (ℓ,u) a function

$$[\![\mathsf{pop}]\!]_{(\ell,u)} : A(\ell,u) \times (V \Rightarrow A(\ell+1, u+1)) \longrightarrow A(\ell,u)$$
$$[\![\mathsf{pop}]\!]_{(\ell,u)}(a,f) = (\vartheta(a,f))^{\ddagger}_{(0,0)} (\lambda \vec{v}.\, \text{if } \vec{v} = [\,] \text{ then } ([\,], \star) \text{ else } (\mathrm{tail}\, \vec{v}, \mathrm{head}\, \vec{v}))$$

We can similarly define a graded set P' by $P'e = \{\star \mid (0,0) \le e\}$, so that there are bijections $\vartheta : A(\ell,u) \cong \mathbf{GSet}_{\mathsf{Interval}}(P', A)(\ell,u)$, and use these to define

$$[\![\mathsf{push}_w]\!]_{(\ell,u)} : A(\ell,u) \longrightarrow A(\ell{+}1, u{+}1) \qquad [\![\mathsf{push}_w]\!]_{(\ell,u)} a = (\vartheta a)^{\ddagger}_{(1,1)} (\lambda \vec{v}.\, (w :: \vec{v}, \star))$$

These functions are natural in (ℓ,u), and satisfy the following equations:

$$[\![\mathsf{push}_v]\!]_{(\ell,u)} ([\![\mathsf{pop}]\!]_{(\ell,u)}(a,f)) = f\, v \qquad (\text{for each } v \in V)$$
$$[\![\mathsf{pop}]\!]_{(\ell,u)}(a, (\lambda v.\, [\![\mathsf{push}_v]\!]_{(\ell,u)} a)) = a \qquad [\![\mathsf{pop}]\!]_{(\ell,u)}([\![\mathsf{pop}]\!]_{(\ell,u)}(a,f), g) = [\![\mathsf{pop}]\!]_{(\ell,u)}(a,g)$$

Every morphism $\mathsf{A} \xrightarrow{e} \mathsf{A}'$ of $\mathsf{Stk}_{\mathsf{flex}}$-algebras preserves the functions $[\![\mathsf{pop}]\!]$ and $[\![\mathsf{push}_v]\!]$.

This is half of an isomorphism of locally graded categories over $\mathbf{GSet}_{\mathsf{Interval}}$, essentially because every $\mathsf{Stk}_{\mathsf{flex}}$-computation can be written using finitely many pops and pushes, as in Section 2.3.

When A is the free $\mathsf{Stk}_{\mathsf{flex}}$-algebra on a graded set X (with $\mathsf{Stk}_{\mathsf{flex}} X$ as the carrier), then we recover the operations defined for $\mathsf{Stk}_{\mathsf{flex}}$ in Section 2.3 as $pop_X = [\![\mathsf{pop}]\!]$ and $push_{v,X} = [\![\mathsf{push}_v]\!]$.

4 RIGIDLY GRADED PRESENTATIONS

Before introducing flexibly graded presentations, we consider the less general rigidly graded presentations. The theory of these is quite well-developed. They are discussed (at varying levels of generality) by Dorsch et al. [2019]; Kura [2020]; Smirnov [2008], all of whom show a correspondence between rigidly graded presentations and a class of rigidly graded monads. Our discussion mostly follows Kura [2020]. We do however reformulate several definitions (primarily to ease the comparison with flexibly graded presentations), for example we work with locally graded categories

[2]The graded sets P, P' and bijections ϑ defined here are special cases of the graded sets $K_{\mathbb{E}}\vec{e}$ and bijections ϑ of Definition 8.5. Specifically, we have $P \cong K_{\mathsf{Interval}}((0,0), \underbrace{(1,1), \ldots, (1,1)}_{|V|})$ and $P' \cong K_{\mathsf{Interval}}((0,0))$.

(Kura uses actegories). We also fill in a few gaps. In particular, we show that the operations of the presentation induce *algebraic operations* (Section 4.4).

A presentation firstly has a collection of *operations*, forming a *signature*.

Definition 4.1. A *rigidly $\mathbb{E}$-graded signature* consists of a set $\Sigma(n;e')$ for each natural number n and grade e'. We call the elements of $\Sigma(n;e')$ the *$(n;e')$-ary operations.*

(In this section we often use e' even when there is no e, for easier comparison with later sections.) Here n is the number of arguments the operation has, and e' is the grade associated with applying the operation. All of the arguments are required to have the same grade d when applying the operation (hence rigid). The result of applying the operation has grade $e' \cdot d$ when the arguments have grade d.

For each rigidly graded signature Σ we have a notion of *term* (derived operation) over Σ. A *context* $\Gamma = x_1, \ldots, x_n$ is a list of variable names (without repetitions). (We often call the ith variable x_i, but permit ourselves to use other variable names and identify terms up to α-equivalence, so that a context is equivalently just a natural number specifying the number of variables.) We write $\Gamma \vdash t : e$ to indicate that t is a term of grade e in context Γ; these are generated inductively by the following rules for variables, application of operations, and coercion:

$$\frac{x \in \Gamma}{\Gamma \vdash x : 1} \qquad \frac{\mathrm{op} \in \Sigma(n;e') \quad \Gamma \vdash t_1 : d \ \cdots \ \Gamma \vdash t_n : d}{\Gamma \vdash \mathrm{op}(d; t_1, \ldots, t_n) : e' \cdot d} \qquad \frac{e \leq e' \quad \Gamma \vdash t : e}{\Gamma \vdash (e \leq e')^* t : e'}$$

In particular, variables have the unit grade 1, and operations can only be applied when all of their arguments have the same grade d. There is no restriction at all on which grade d is; this is crucial when defining substitution of terms, because even though variables have grade 1, we can substitute terms of arbitrary grades d. The grade d in fact has exactly the same role as the d in the definition of rigidly graded monad (Definition 2.2). Given a term $x_1, \ldots, x_n \vdash t : e'$ and a list of terms $\Gamma \vdash u_i : d$ (all of the same grade), by substituting we obtain a term $\Gamma \vdash t\{d; x_1 \mapsto u_1, \ldots, x_n \mapsto u_n\} : e' \cdot d$.

Now that we have a notion of term, we can say what an *equation* is (just a pair of terms that we wish to be equal), and then define *rigidly graded presentations.*

Definition 4.2. An *$(n;e')$-ary term* over a rigidly graded signature Σ is a term $x_1, \ldots, x_n \vdash t : e'$. An $(n;e')$-ary *equation* over Σ is a pair (t, u) of $(n;e')$-ary terms over Σ.

We will often write an $(n;e')$-ary equation (t, u) as $x_1, \ldots, x_n \vdash t \approx u : e'$, and, for convenience, permit ourselves to give different names to the variables.

Definition 4.3. A *rigidly $\mathbb{E}$-graded presentation* (Σ, E) consists of

- a rigidly $\mathbb{E}$-graded signature Σ;
- for each natural number n and grade e', a set $E(n;e')$ of $(n;e')$-ary equations over Σ.

Example 4.4. Recall the rigidly $\mathbb{E}$-graded writer monad Wr^{M} from Section 2.1. We give a rigidly graded presentation (Σ, E) of Wr^{M}. The signature Σ consists of a $(1;e')$-ary operation tell_p for each $p \in Me'$, so that terms are generated by variables, coercions, and

$$\frac{\Gamma \vdash t : d}{\Gamma \vdash \mathsf{tell}_p(d; t) : e' \cdot d} \quad (p \in Me')$$

There is a $(1;e')$-ary equation for each $p \in Me$ and $e' \geq e$, a single $(1;1)$-ary equation, and a $(1;e \cdot e')$-ary equation for each $p \in Me$ and $p' \in Me'$:

$$x \vdash (e \leq e')^*(\mathsf{tell}_p(1;x)) \approx \mathsf{tell}_{(e \leq e')^* p}(1;x) : e'$$

$$x \vdash \mathsf{tell}_\varepsilon(1;x) \approx x : 1 \qquad x \vdash \mathsf{tell}_p(e'; \mathsf{tell}_{p'}(1;x)) \approx \mathsf{tell}_{p \otimes p'}(1;x) : e \cdot e'$$

4.1 Rigidly Graded Equational Logic

A rigidly graded presentation (Σ, E) specifies a collection E of equations as axioms, but further equations $\Gamma \vdash t \approx u : e$ can be derived from these axioms in the rigidly graded *equational logic* over (Σ, E). This is given by the following rules, plus reflexivity, transitivity and symmetry. The first two rules are congruences, the next two express that coercions are compatible with reflexivity and transitivity of $\leq$, the fifth rule says that operator application is natural in the grade d, and the final rule is application of the axioms of the rigidly graded presentation.

$$\frac{\mathrm{op} \in \Sigma(n; e') \qquad \Gamma \vdash t_1 \approx u_1 : d \quad \cdots \quad \Gamma \vdash t_n \approx u_n : d}{\Gamma \vdash \mathrm{op}(d; t_1, \ldots, t_n) \approx \mathrm{op}(d; u_1, \ldots, u_n) : e' \cdot d} \qquad \frac{e \leq e' \quad \Gamma \vdash t \approx u : e}{\Gamma \vdash (e{\leq}e')^* t \approx (e{\leq}e')^* u : e'}$$

$$\frac{\Gamma \vdash t : e}{\Gamma \vdash t \approx (e{\leq}e)^* t : e} \qquad \frac{e \leq e' \leq e'' \quad \Gamma \vdash t : e}{\Gamma \vdash (e'{\leq}e'')^*((e{\leq}e')^* t) \approx (e{\leq}e'')^* t : e''}$$

$$\frac{\mathrm{op} \in \Sigma(n; e') \qquad d \leq d' \qquad \Gamma \vdash t_1 : d \quad \cdots \quad \Gamma \vdash t_n : d}{\Gamma \vdash (e'{\cdot}d \leq e'{\cdot}d')^*(\mathrm{op}(d; t_1, \ldots, t_n)) \approx \mathrm{op}(d'; (d \leq d')^* t_1, \ldots, (d \leq d')^* t_n) : e' \cdot d'}$$

$$\frac{(t, u) \in E(n; e') \qquad \Gamma \vdash s_1 : d \quad \cdots \quad \Gamma \vdash s_n : d}{\Gamma \vdash t\{d; x_1 \mapsto s_1, \ldots, x_n \mapsto s_n\} \approx u\{d; x_1 \mapsto s_1, \ldots, x_n \mapsto s_n\} : e' \cdot d}$$

For every natural number n and grade e' we write $\mathrm{Term}^{(\Sigma,E)} n e'$ for the set of $(n; e')$-ary terms over Σ, quotiented by $\approx$. These form graded sets $\mathrm{Term}^{(\Sigma,E)} n$.

4.2 Algebras

An *algebra* for a rigidly graded presentation (Σ, E) is a graded set equipped with interpretations of the operations in Σ, satisfying the equations in E. We define this notion in two steps, by first defining an notion of algebra for a signature Σ. This matches the definition given by Kura [2020], but, differently from Kura, we work with locally graded categories of algebras.

Definition 4.5. Let Σ be a rigidly $\mathbb{E}$-graded signature. A Σ-*algebra* A consists of a graded set A (the *carrier*), together with a grade-preserving function $[\![\mathrm{op}]\!] : A^n \to A(e' \cdot -)$ for every operation $\mathrm{op} \in \Sigma(n; e')$. The Σ-algebras form a locally $\mathbb{E}$-graded category $\mathbf{Alg}(\Sigma)$, in which a *morphism* $f : \mathsf{A} \xrightarrow{-e} \mathsf{A}'$ is a morphism $f : A \xrightarrow{-e} A'$ between the carriers, i.e. a grade-preserving function $f : A \Rightarrow A'(- \cdot e)$, preserving interpretations of operations as follows:

$$f_{e' \cdot d}([\![\mathrm{op}]\!]_d(a_1, \ldots, a_n)) = [\![\mathrm{op}]\!]_{d \cdot e}(f_d a_1, \ldots, f_d a_n)$$

Every Σ-algebra A admits interpretations of terms over Σ. For each $(n; e')$-ary term t and grade d, the interpretation of t is the grade-preserving function $[\![t]\!] : A^n \Rightarrow A(e' \cdot -)$ defined as follows:

$$\begin{aligned}
[\![x_i]\!]_d(a_1, \ldots, a_n) &= a_i \\
[\![\mathrm{op}(d'; t_1, \ldots, t_m)]\!]_d(a_1, \ldots, a_n) &= [\![\mathrm{op}]\!]_{d' \cdot d}([\![t_1]\!]_d(a_1, \ldots, a_n), \ldots, [\![t_m]\!]_d(a_1, \ldots, a_n))) \\
[\![(e{\leq}e')^* t]\!]_d(a_1, \ldots, a_n) &= (e{\cdot}d \leq e'{\cdot}d)^*([\![t]\!]_d(a_1, \ldots, a_n))
\end{aligned}$$

Definition 4.6. Let (Σ, E) be a rigidly graded presentation. A (Σ, E)-*algebra* is a Σ-algebra A that *satisfies* all of the equations in E, in the sense that for every equation $(t, u) \in E(n; e')$ and grade d, we have $[\![t]\!]_d = [\![u]\!]_d$. We write $\mathbf{Alg}(\Sigma, E)$ for the locally graded category of (Σ, E)-algebras, which has the same morphisms as $\mathbf{Alg}(\Sigma)$. There is a forgetful functor $U_{(\Sigma,E)} : \mathbf{Alg}(\Sigma, E) \to \mathbf{GSet}_{\mathbb{E}}$ which sends algebras to carriers and morphisms to themselves.

Example 4.7. Recall the presentation (Σ, E) of Wr^{M}, from Example 4.4. The locally graded category $\mathbf{Alg}(\Sigma, E)$ is exactly the one described in Example 3.5: objects are graded sets equipped with grade-preserving functions $[\![\mathsf{tell}_p]\!]$ satisfying the equations given there. Morphisms are $[\![\mathsf{tell}_p]\!]$-preserving morphisms of graded sets. In particular, there is an isomorphism $\mathbf{Alg}(\Sigma, E) \cong \mathbf{EM}(\mathsf{Wr}^{\mathsf{M}})$ over $\mathbf{GSet}_{\mathbb{E}}$.

For every natural number n, the terms in context $x_1, \ldots, x_n$, quotiented by $\approx$, form a (Σ, E)-algebra. The carrier is the graded set $\mathrm{Term}^{(\Sigma,E)} n$ defined above, and operations are interpreted by $[\![\mathrm{op}]\!]_d(u_1, \ldots, u_n) = \mathrm{op}(d; u_1, \ldots, u_n)$. It follows from this definition that arbitrary terms are interpreted by substitution as $[\![t]\!]_d(u_1, \ldots, u_m) = t\{d; x_1 \mapsto u_1, \ldots, x_m \mapsto u_m\}$. In particular, the rule for application of axioms in the equational logic amounts to the fact that, in these Σ-algebras, $[\![t]\!]_d = [\![u]\!]_d$ for every $(t, u) \in E(n; e'')$ and d, so $\mathrm{Term}^{(\Sigma,E)} n$ does indeed form a (Σ, E)-algebra.

4.3 Rigidly Graded Monads from Rigid Presentations

Our motivation for rigidly graded presentations is to present rigidly graded monads, and indeed every rigidly graded presentation (Σ, E) induces a rigidly graded monad $\mathsf{R}^{(\Sigma,E)}$. Of course we do not want just any rigidly graded monad: in particular the operations of (Σ, E) have interpretations in $\mathsf{R}^{(\Sigma,E)}$. It turns out that $\mathsf{R}^{(\Sigma,E)}$ is canonical in the sense that to equip a graded set A with the structure of an $\mathsf{R}^{(\Sigma,E)}$-algebra is equivalently to equip A with the structure of a (Σ, E)-algebra. In other words, to interpret computations over $\mathsf{R}^{(\Sigma,E)}$ is equivalently to interpret the operations of Σ in a way that satisfies the equations of E. Formally, we have the following:

Theorem 4.8. *For every rigidly $\mathbb{E}$-graded presentation (Σ, E), there is a rigidly graded monad $\mathsf{R}^{(\Sigma,E)}$ and isomorphism $\mathbf{Alg}(\Sigma, E) \cong \mathbf{EM}(\mathsf{R}^{(\Sigma,E)})$ over $\mathbf{GSet}_{\mathbb{E}}$.*

(We do not prove this yet, instead we show a stronger theorem (Theorem 8.8) that there is a correspondence between rigidly graded presentations and a class of rigidly graded monads. Similar correspondences can be found e.g. in [Kura 2020].)

Example 4.9. For the presentation (Σ, E) introduced for Wr^{M} (Example 4.4), there is an isomorphism $\mathbf{Alg}(\Sigma, E) \cong \mathbf{EM}(\mathsf{Wr}^{\mathsf{M}})$ over $\mathbf{GSet}_{\mathbb{E}}$ (Example 4.7). Since rigidly graded monads are determined by their algebras, the rigidly graded monad $\mathsf{R}^{(\Sigma,E)}$ induced by this presentation is Wr^{M} (up to isomorphism). Hence (Σ, E) is indeed a presentation of Wr^{M}.

4.4 Algebraic Operations

Every rigidly graded presentation (Σ, E) induces a rigidly graded monad $\mathsf{R}^{(\Sigma,E)}$ that is canonical in the sense that (Σ, E)-algebras are equivalently $\mathsf{R}^{(\Sigma,E)}$-algebras. It follows that $\mathsf{R}^{(\Sigma,E)}$ admits an interpretation of each of the operations in Σ. To make this precise, we define a notion of rigidly graded *algebraic operation.* These are a special case of Katsumata's [2014] algebraic operations, and are analogous to Plotkin and Power's [2003] algebraic operations for non-graded monads.

Definition 4.10. If R is a rigidly $\mathbb{E}$-graded monad, then a $(n; e')$*-ary algebraic operation* consists of a grade-preserving function $\alpha_X : (RX)^n \Rightarrow RX(e' \cdot -)$ for each set X, and such that

$$f^{\dagger}_{e' \cdot d}(\alpha_{X,d}(r_1, \ldots, r_n)) = \alpha_{Y,d \cdot e}(f^{\dagger}_d r_1, \ldots, f^{\dagger}_d r_n) \qquad (f : X \to RYe,\ r_i \in RXd)$$

Consider the rigidly graded monad $\mathsf{R}^{(\Sigma,E)}$ presented by (Σ, E). For every set X, the graded set $R^{(\Sigma,E)}X$ is the carrier of the free $\mathsf{R}^{(\Sigma,E)}$-algebra on X, and hence also forms a (Σ, E)-algebra. Every operation $\mathrm{op} \in \Sigma(n; e')$ therefore has an interpretation in $R^{(\Sigma,E)}X$:

$$[\![\mathrm{op}]\!]^X : (R^{(\Sigma,E)}X)^n \Rightarrow R^{(\Sigma,E)}X(e' \cdot -)$$

These collectively (for all X) form an $(n; e')$-ary algebraic operation for the rigidly graded monad $\mathsf{R}^{(\Sigma,E)}$.

Example 4.11. Recall the presentation (Σ, E) of Wr^{M} (Example 4.4). Since in this case we have $\mathsf{R}^{(\Sigma,E)} = \mathsf{Wr}^{\mathsf{M}}$, for every $p \in Me'$ we obtain a $(1; e')$-ary algebraic operation tell_p for Wr^{M}:

$$\mathit{tell}_{p,X} = [\![\mathsf{tell}_p]\!]^X : \mathrm{Wr}^{\mathsf{M}}X \Rightarrow \mathrm{Wr}^{\mathsf{M}}X(e' \cdot -)$$

These are exactly the grade-preserving functions $\mathit{tell}_{p,X}$ defined in Section 2.1. We have recovered them directly from the presentation of Wr^{M}.

5 FROM FLEXIBLE TO RIGID

Our goal is to develop a more flexible notion of presentation. The extra flexibility comes at a cost: there cannot be a precise correspondence between flexibly graded presentations and a class of rigidly graded monads. In particular, for a given flexibly graded presentation, there is in general no rigidly graded monad that has the same algebras. Despite this, each flexibly graded presentation induces a rigidly graded monad that carries interpretations of the operations of the presentation. To show this, we use flexibly graded monads. We show that there *is* a correspondence between flexibly graded presentations and flexibly graded monads, so every flexibly graded presentation induces a flexibly graded monad (with the same algebras). To get a rigidly graded monad, it then suffices to show that each flexibly graded monad induces an appropriate rigidly graded monad. The goal of the present section is explain how to do this. (This construction is discussed in more detail in [McDermott and Uustalu 2022].)

This is possible because every set X induces a graded set $\hat{X}$ as follows, intuitively by considering the elements of X to have grade 1. (This is the same $\hat{X}$ we define in Section 2.2, but here we generalize to arbitrary $\mathbb{E}$.)

Definition 5.1. Every set X induces an $\mathbb{E}$-graded set $\hat{X}$, defined by

$$\hat{X}e = X \quad \text{if } 1 \le e \qquad \hat{X}e = \emptyset \quad \text{if } 1 \not\le e \qquad (e{\le}e')^* x = x$$

If Y is a graded set and e is a grade, then there is a bijection θ as follows:

$$\theta : \mathbf{Set}(X, Ye) \cong \mathbf{GSet}_{\mathbb{E}}(\hat{X}, Y)e : \theta^{-1} \qquad (\theta f)_d x = fx \qquad \theta^{-1}gx = g_1 x$$

Now suppose that T is a flexibly graded monad; we define a rigidly graded monad $\lfloor \mathsf{T} \rfloor$, essentially by restricting T to graded sets of the form $\hat{X}$. For every set X, we have a graded set $\hat{X}$, and hence a graded set $\lfloor T \rfloor X = T\hat{X}$. The latter intuitively contains computations that return elements of X, where elements of X have grade 1. The unit of T induces a family of functions $\eta_X : \hat{X} \to T\hat{X} = \lfloor T \rfloor X$, which is equivalently a family of functions $\theta^{-1}\eta_{\hat{X}} : X \to \lfloor T \rfloor X1$; these form the unit of $\lfloor T \rfloor$. Finally, the Kleisli extension of $\lfloor T \rfloor$ sends $f : X \to \lfloor T \rfloor Ye$ to $(\theta f)^\dagger : \lfloor T \rfloor X \Rightarrow \lfloor T \rfloor Y(- \cdot e)$.

$$\frac{\eta_{\hat{X}} : \hat{X} \Rightarrow T\hat{X}}{\theta^{-1}\eta_{\hat{X}} : X \to T\hat{X}1} \qquad \frac{\dfrac{f : X \to T\hat{Y}e}{\theta f : \hat{X} \Rightarrow T\hat{Y}(- \cdot e)}}{(\theta f)^\dagger : T\hat{X} \Rightarrow T\hat{Y}(- \cdot e)}$$

Example 5.2. As we explain in Section 2.2, if we apply this construction to the flexibly graded monad $\mathsf{State}_{\mathrm{flex}}$, the rigidly graded monad $\lfloor \mathsf{State}_{\mathrm{flex}} \rfloor$ we get is State. There is an equivalent statement for all of the flexibly graded monads T we define in Section 2: in each case, $\lfloor \mathsf{T} \rfloor$ is the corresponding rigidly graded monad.

We show that this construction is in a sense canonical. First, every T-algebra A induces an $\lfloor \mathsf{T} \rfloor$-algebra $Q_{\mathsf{T}}\mathsf{A}$, again essentially by restricting the structure to graded sets of the form $\hat{X}$. The carrier of $Q_{\mathsf{T}}\mathsf{A}$ is the carrier A of A, while the extension operator maps each function $f : X \to Ae$

to the natural transformation $(\theta f)^{\ddagger} : \lfloor T \rfloor X \Rightarrow A(- \cdot e)$.

$$\frac{\dfrac{f : X \to Ae}{\theta f : \hat{X} \Rightarrow A(- \cdot e)}}{(\theta f)^{\ddagger} : T\hat{X} \Rightarrow A(- \cdot e)}$$

This construction forms a functor $Q_{\mathsf{T}} : \mathbf{EM}(\mathsf{T}) \to \mathbf{EM}(\lfloor \mathsf{T} \rfloor)$, by sending each morphism $f : \mathsf{A} \xrightarrow{e} \mathsf{A}'$ to itself. In general Q_{T} is not an isomorphism, since in general, there is no rigidly graded monad with the same algebras as T. However, we do have the following universal property for $\lfloor \mathsf{T} \rfloor$, which informally says that $\lfloor \mathsf{T} \rfloor$ is as close as we can get to T.

Lemma 5.3. *Let* T *be a flexibly* $\mathbb{E}$*-graded monad. For every rigidly* $\mathbb{E}$*-graded monad* R *and functor* $Q' : \mathbf{EM}(\mathsf{T}) \to \mathbf{EM}(\mathsf{R})$ *over* $\mathbf{GSet}_{\mathbb{E}}$*, there is a unique functor* $F : \mathbf{EM}(\lfloor \mathsf{T} \rfloor) \to \mathbf{EM}(\mathsf{R})$ *over* $\mathbf{GSet}_{\mathbb{E}}$ *such that such that* $Q' = F \cdot Q_{\mathsf{T}}$.

$$\begin{tikzcd}
\mathbf{EM}(\mathsf{T}) \arrow[r, "Q_{\mathsf{T}}"] \arrow[dr, "Q'"'] & \mathbf{EM}(\lfloor \mathsf{T} \rfloor) \arrow[d, dashed, "F"] \\
& \mathbf{EM}(\mathsf{R})
\end{tikzcd}$$

It is in this sense that $\lfloor \mathsf{T} \rfloor$ is canonical (and since rigidly graded monads are completely determined by their algebras, this lemma actually characterizes $\lfloor \mathsf{T} \rfloor$ up to isomorphism of rigidly graded monads). We show in Section 6.4 that algebraic operations for T induce algebraic operations for $\lfloor \mathsf{T} \rfloor$, which will ensure that operations of a flexibly graded presentation can be interpreted in the induced rigidly graded monad.

6 FLEXIBLY GRADED PRESENTATIONS

We turn now to the main contribution of this paper: the notion of *flexibly graded presentation*. As we describe in the introduction, these generalize rigidly graded presentations so that their arguments do not all need to have the same grade. Instead of having natural numbers n in the arities of operations, we instead have lists $\vec{e} = (e_1, \dots, e_n)$ of grades.

Definition 6.1. A *flexibly* $\mathbb{E}$*-graded signature* consists of a set $\Sigma(\vec{e}; e')$ for each list $\vec{e}$ of grades and grade e'. We call the elements of $\Sigma(\vec{e}; e')$ the $(\vec{e}; e')$*-ary operations*.

For each flexibly graded signature Σ, we have a notion of *term* (derived operation) over Σ. A *context* $\Gamma = x_1 : e_1, \dots, x_n : e_n$ is a list of (variable name, grade)-pairs. (We often call the ith variable x_i, but permit ourselves to use other variable names and identify terms up to α-equivalence.) We write $\Gamma \vdash t : e$ to indicate that t is a term of grade e in context Γ; these are generated inductively by the following rules for variables, application of operations, and coercions:

$$\frac{(x : e) \in \Gamma}{\Gamma \vdash x : e} \qquad \frac{\mathrm{op} \in \Sigma(e_1, \dots, e_n; e') \quad \Gamma \vdash t_1 : e_1 \cdot d \ \cdots \ \Gamma \vdash t_n : e_n \cdot d}{\Gamma \vdash \mathrm{op}(d; t_1, \dots, t_n) : e' \cdot d} \qquad \frac{e \leq e' \quad \Gamma \vdash t : e}{\Gamma \vdash (e{\leq}e')^{*} t : e'}$$

The grade d has a crucial role in the definition of substitution below.

Example 6.2. Let $V = \{\mathsf{v}_1, \dots, \mathsf{v}_{|V|}\}$ be a finite set. We have a flexibly Interval-graded signature Σ that we use below as part of a flexibly graded presentation for our V-valued stack example (Section 2.3). This consists of a $((0,0); (1,1))$-ary operation push_v for each $v \in V$, and a $((0,0), \underbrace{(1,1), \dots, (1,1)}_{|V|}; (0,0))$-ary operation pop. The operations push_v are actually rigidly graded in the sense that all arguments have the same grade (because there is only one argument). However, pop is genuinely flexibly graded, since one argument has a different grade from the rest. Terms over Σ are generated by rules for variables, coercions, and the following rules for the operations:

$$\frac{\Gamma \vdash t : (\ell, u)}{\Gamma \vdash \mathsf{push}_v((\ell, u); t) : (\ell + 1, u + 1)} \qquad \frac{\Gamma \vdash t : (\ell, u) \quad \Gamma \vdash t'_1 : (\ell{+}1, u{+}1) \ \cdots \ \Gamma \vdash t'_{|V|} : (\ell{+}1, u{+}1)}{\Gamma \vdash \mathsf{pop}((\ell, u); t, t'_1, \ldots, t'_{|V|}) : (\ell, u)}$$

The term $\mathsf{push}_v((\ell, u); t)$ pushes v onto the stack and continues as t; the term $\mathsf{pop}((\ell, u); t, t'_1, \ldots, t'_{|V|})$ attempts to pop a value, continues as t if the stack was empty, and continues as t'_i if the stack had the value v_i at the top.

Definition 6.3. An $(e_1, \ldots, e_n; e')$*-ary term* over Σ is a term $x_1 : e_1, \ldots, x_n : e_n \vdash t : e'$. An $(\vec{e}; e')$-ary *equation* over a signature Σ is a pair (t, u) of $(\vec{e}; e')$-ary terms over Σ.

We write $(\vec{e}; e')$-ary equations (t, u) as $x_1 : e_1, \ldots, x_n : e_n \vdash t \approx u : e'$, and allow ourselves to use different names for the variables. We now come to our main definition.

Definition 6.4. A *flexibly* $\mathbb{E}$*-graded presentation* (Σ, E) consists of

- a flexibly $\mathbb{E}$-graded signature Σ;
- for each $\vec{e}$ and e', a set $E(\vec{e}; e')$ of $(\vec{e}; e')$-ary equations over Σ.

Example 6.5. Recall the signature Σ for V-valued stacks, from Example 6.2. We make this into a flexibly Interval-graded presentation (Σ, E), with E consisting of the following equations:

$$x : (0,0), y_1 : (1,1), \ldots, y_{|V|} : (1,1) \vdash \mathsf{push}_{\mathsf{v}_i}((0,0); \mathsf{pop}((0,0); x, y_1, \ldots, y_{|V|})) \approx y_i : (0,0) \quad \text{for each } i \leq |V|$$

$$x : (0,0) \vdash \mathsf{pop}((0,0); x, \mathsf{push}_{\mathsf{v}_1}((0,0); x), \ldots, \mathsf{push}_{\mathsf{v}_{|V|}}((0,0); x)) \approx x : (0,0)$$

$$\begin{aligned} &x : (0,0), y_1 : (1,1), \ldots, y_{|V|} : (1,1), z_1 : (1,1), \ldots, z_{|V|} : (1,1) \vdash \\ &\quad \mathsf{pop}((0,0); \mathsf{pop}((0,0); x, y_1, \ldots, y_{|V|}), z_1, \ldots, z_{|V|}) \approx \mathsf{pop}((0,0); x, z_1, \ldots, z_{|V|}) : (0,0) \end{aligned}$$

These equations are analogous to the equations that $[\![\mathsf{push}_v]\!]$ and $[\![\mathsf{pop}]\!]$ satisfy in Example 3.8. This is of course not a coincidence, as we explain in Example 6.8 below.

The appropriate notion of substitution for terms over a flexibly graded signature Σ is the following. Given a term $x_1 : e_1, \ldots, x_n : e_n \vdash t : e'$, grade d, and a list of terms $\Gamma \vdash u_i : e_i \cdot d$, we have a term $\Gamma \vdash t\{d; x_1 \mapsto u_1, \ldots, x_n \mapsto u_n\} : e' \cdot d$, defined by

$$\begin{aligned} x_i\{d; x_1 \mapsto u_1, \ldots, x_n \mapsto u_n\} &= u_i \\ (\mathsf{op}(d'; t_1, \ldots, t_m))\{d; x_1 \mapsto u_1, \ldots, x_n \mapsto u_n\} &= \mathsf{op}(d' {\cdot} d; t_1\{d; x_1{\mapsto}u_1, \ldots\}, \ldots, t_m\{d; x_1{\mapsto}u_1, \ldots\}) \\ ((e' {\leq} e'')^* t)\{d; x_1 \mapsto u_1, \ldots, x_n \mapsto u_n\} &= (e' {\cdot} d {\leq} e'' {\cdot} d)^* (t\{d; x_1 \mapsto u_1, \ldots, x_n \mapsto u_n\}) \end{aligned}$$

6.1 Flexibly Graded Equational Logic

We define an *equational logic* for proving equalities between terms over Σ, where (Σ, E) is a flexibly graded presentation. We write $\Gamma \vdash t \approx u : e$ to indicate that the terms $\Gamma \vdash t : e$ and $\Gamma \vdash u : e$ are equal in this equational logic. This is defined inductively by the following rules: two congruence rules, two rules for identity and composition of coercions, one rule for naturality of operations, and one rule for application of the axioms of E; plus reflexivity, symmetry and transitivity.

$$\frac{\mathrm{op} \in \Sigma(e_1, \ldots, e_n; e') \quad \Gamma \vdash t_1 \approx u_1 : e_1 \cdot d \ \cdots \ \Gamma \vdash t_n \approx u_n : e_n \cdot d}{\Gamma \vdash \mathrm{op}(d; t_1, \ldots, t_n) \approx \mathrm{op}(d; u_1, \ldots, u_n) : e' \cdot d} \qquad \frac{e \le e' \quad \Gamma \vdash t \approx u : e}{\Gamma \vdash (e{\le}e')^* t \approx (e{\le}e')^* u : e'}$$

$$\frac{\Gamma \vdash t : e}{\Gamma \vdash t \approx (e{\le}e)^* t : e} \qquad \frac{e \le e' \le e'' \quad \Gamma \vdash t : e}{\Gamma \vdash (e'{\le}e'')^*((e{\le}e')^* t) \approx (e{\le}e'')^* t : e''}$$

$$\frac{\mathrm{op} \in \Sigma(e_1, \ldots, e_n; e') \quad d \le d' \quad \Gamma \vdash t_1 : e_1 \cdot d \ \cdots \ \Gamma \vdash t_n : e_n \cdot d}{\Gamma \vdash (e'{\cdot}d \le e'{\cdot}d')^*(\mathrm{op}(d; t_1, \ldots, t_n)) \approx \mathrm{op}(d'; (e_1{\cdot}d \le e_1{\cdot}d')^* t_1, \ldots, (e_n{\cdot}d \le e_n{\cdot}d')^* t_n) : e' \cdot d'}$$

$$\frac{(t, u) \in E(e_1, \ldots, e_n; e') \quad \Gamma \vdash s_1 : e_1 \cdot d \ \cdots \ \Gamma \vdash s_n : e_n \cdot d}{\Gamma \vdash t\{d; x_1 \mapsto s_1, \ldots, x_n \mapsto s_n\} \approx u\{d; x_1 \mapsto s_1, \ldots, x_n \mapsto s_n\} : e' \cdot d}$$

(We do not need a general rule for closure of $\approx$ under substitution, because this is admissible.) We show that this equational logic is sound and complete in Theorem 6.9 below.

6.2 Algebras

Definition 6.6. Let Σ be a flexibly $\mathbb{E}$-graded signature. A Σ-*algebra* A consists of a graded set A (the *carrier*), together with an assignment to every operation $\mathrm{op} \in \Sigma(\vec{e}; e')$ of a grade-preserving function $[\![\mathrm{op}]\!] : \prod_i A(e_i \cdot -) \Rightarrow A(e' \cdot -)$.

A *morphism* $f : \mathrm{A} {-}e{\nrightarrow} \mathrm{A}'$ is a morphism $f : A {-}e{\nrightarrow} A'$ between the carriers, i.e., a grade-preserving function $f : A \Rightarrow A'(- \cdot e)$, preserving interpretations of operations as follows:

$$f_{e' \cdot d}([\![\mathrm{op}]\!]_d(a_1, \ldots, a_n)) = [\![\mathrm{op}]\!]_{d \cdot e}(f_{e_1 \cdot d} a_1, \ldots, f_{e_n \cdot d} a_n)$$

Every Σ-algebra A admits interpretations of terms over Σ. The interpretation of a $(\vec{e}; e')$-ary term t in A at grade d is the function $[\![t]\!]_d : \prod_i A(e_i \cdot d) \to A(e' \cdot d)$ defined as follows:

$$\begin{aligned} [\![x_i]\!]_d(a_1, \ldots, a_n) &= a_i \\ [\![\mathrm{op}(d'; t_1, \ldots, t_m)]\!]_d(a_1, \ldots a_n) &= [\![\mathrm{op}]\!]_{d' \cdot d}([\![t_1]\!]_d(a_1, \ldots, a_n), \ldots, [\![t_m]\!]_d(a_1, \ldots, a_n))) \\ [\![(e{\le}e')^* t]\!]_d(a_1, \ldots, a_n) &= (e{\cdot}d \le e'{\cdot}d)^*([\![t]\!]_d(a_1, \ldots, a_n)) \end{aligned}$$

Definition 6.7. Let (Σ, E) be a flexibly graded presentation. A (Σ, E)-*algebra* is a Σ-algebra A that *satisfies* all of the equations in E, in the sense that for every equation $(t, u) \in E(\vec{e}; e')$ and grade d, we have $[\![t]\!]_d = [\![u]\!]_d$.

Example 6.8. For the flexibly graded presentation (Σ, E) of V-valued stacks (Example 6.5), the (Σ, E)-algebras are exactly as described in Example 3.8: they are graded sets A equipped with grade-preserving functions $[\![\mathsf{pop}]\!]$, $[\![\mathsf{push}_v]\!]$, satisfying equations. The isomorphism described there is an isomorphism $\mathbf{Alg}(\Sigma, E) \cong \mathbf{EM}(\mathsf{Stk}_{\mathsf{flex}})$ over $\mathbf{GSet}_{\mathsf{Interval}}$.

Using the equational logic, the terms over the signature Σ form (Σ, E)-algebras as follows. For each list of grades $\vec{e}$, let $\mathrm{Term}^{(\Sigma,E)}\vec{e}$ be the graded set of terms over the signature Σ, quotiented by $\approx$. Coercions $(e \le e')^* t$ are just those provided in the syntax of terms. This graded set forms a Σ-algebra by interpreting operations as $[\![\mathrm{op}]\!]_d(u_1, \ldots, u_n) = \mathrm{op}(d; u_1, \ldots, u_n)$. It follows that arbitrary terms are interpreted in this Σ-algebra by substitution:

$$[\![t]\!]_d(u_1, \ldots, u_m) = t\{d; x_1 \mapsto u_1, \ldots, x_m \mapsto u_m\} \tag{1}$$

In particular, the rule for application of axioms in the equational logic amounts to the fact that $[\![t]\!]_d = [\![u]\!]_d$ for every $(t, u) \in E(\vec{e}'; e'')$, so $\mathrm{Term}^{(\Sigma,E)}\vec{e}$ in fact forms a (Σ, E)-algebra. It follows that flexibly graded equational logic is sound and complete in the following sense.

Theorem 6.9. *For every flexibly graded presentation (Σ, E), the judgment $x_1 : e_1, \ldots, x_n : e_n \vdash t \approx u : e'$ is derivable if and only if, for every (Σ, E)-algebra and grade d, we have $[\![t]\!]_d = [\![u]\!]_d$.*

Proof. Only if (soundness): this is proved by induction on the derivation of $\approx$. In the case of an axiom from E, we use the following substitution lemma, which is proved by induction on t':

$$[\![t'\{d; x_1 \mapsto s_1, \ldots, x_m \mapsto s_m\}]\!]_{d'}(a_1, \ldots, a_k) = [\![t']\!]_{d \cdot d'}([\![s_1]\!]_{d'}(a_1, \ldots, a_k), \ldots, [\![s_m]\!]_{d'}(a_1, \ldots, a_k))$$

If (completeness): the graded set $\mathrm{Term}^{(\Sigma,E)}\vec{e}$ of terms quotiented by $\approx$ forms a (Σ, E)-algebra as above. If we have $[\![t]\!]_d = [\![u]\!]_d$ in every (Σ, E)-algebra, then in particular $[\![t]\!]_1(x_1, \ldots, x_n) = [\![u]\!]_1(x_1, \ldots, x_n)$ in this (Σ, E)-algebra so, using Eq. (1), we have $t = t\{1; x_1 \mapsto x_1, \ldots, x_n \mapsto x_n\} \approx u\{1; x_1 \mapsto x_1, \ldots, x_n \mapsto x_n\} = u$. □

6.3 Rigidly Graded Monads from Flexibly Graded Presentations

We now turn to the main theorem of this paper: that each flexibly graded presentation induces a canonical flexibly graded monad, and hence a canonical rigidly graded monad.

Theorem 6.10. *For every flexibly $\mathbb{E}$-graded presentation (Σ, E), we have:*

(1) *a flexibly $\mathbb{E}$-graded monad $\mathsf{T}^{(\Sigma,E)}$, and an isomorphism $\mathbf{Alg}(\Sigma, E) \cong \mathbf{EM}(\mathsf{T}^{(\Sigma,E)})$ over $\mathbf{GSet}_{\mathbb{E}}$;*
(2) *a rigidly $\mathbb{E}$-graded monad $\lfloor \mathsf{T}^{(\Sigma,E)} \rfloor$, and a functor $Q_{(\Sigma,E)} : \mathbf{Alg}(\Sigma, E) \to \mathbf{EM}(\lfloor \mathsf{T}^{(\Sigma,E)} \rfloor)$ over $\mathbf{GSet}_{\mathbb{E}}$. The latter are universal in the sense that, for every rigidly $\mathbb{E}$-graded monad R and functor $Q' : \mathbf{Alg}(\Sigma, E) \to \mathbf{EM}(\mathsf{R})$ over $\mathbf{GSet}_{\mathbb{E}}$, there is a unique functor $F : \mathbf{EM}(\lfloor \mathsf{T}^{(\Sigma,E)} \rfloor) \to \mathbf{EM}(\mathsf{R})$ over $\mathbf{GSet}_{\mathbb{E}}$ such that $Q' = F \cdot Q_{(\Sigma,E)}$.*

We postpone the proof of the first part to Section 8, while the second part follows from the first by Lemma 5.3. The characterizations of algebras uniquely determine both $\mathsf{T}^{(\Sigma,E)}$ and $\lfloor \mathsf{T}^{(\Sigma,E)} \rfloor$ (up to structure-preserving isomorphism), so we can call these *the* flexibly graded monad and *the* rigidly graded monad presented by (Σ, E).

Example 6.11. Let (Σ, E) be the flexibly graded presentation for V-valued stacks (Example 6.5). Due to the isomorphism described in Example 6.8, the induced flexibly graded monad $\mathsf{T}^{(\Sigma,E)}$ is $\mathsf{Stk}_{\mathrm{flex}}$, and the induced flexibly graded monad $\lfloor \mathsf{T}^{(\Sigma,E)} \rfloor$ is Stk. Hence (Σ, E) presents $\mathsf{Stk}_{\mathrm{flex}}$ and Stk.

6.4 Algebraic Operations

We have shown that every flexibly graded presentation (Σ, E) induces a rigidly graded monad $\lfloor \mathsf{T}^{(\Sigma,E)} \rfloor$ that is in some sense canonical. We now show that $\lfloor \mathsf{T}^{(\Sigma,E)} \rfloor$ carries an interpretation of each of the operations in Σ. We introduce a notion of flexibly graded *algebraic operation* for a rigidly graded monad. These are analogous to Plotkin and Power's [2003] algebraic operations for non-graded monads, and are related to effect-function graded algebraic operations for rigidly graded monads [Katsumata 2014]. We show that every operation in Σ induces a flexibly graded algebraic operation for $\lfloor \mathsf{T}^{(\Sigma,E)} \rfloor$. The construction takes two steps: first we show that operations in Σ induce algebraic operations for the *flexibly* graded monad $\mathsf{T}^{(\Sigma,E)}$, and then that these induce flexibly graded algebraic operations for the restriction $\lfloor \mathsf{T}^{(\Sigma,E)} \rfloor$.

Definition 6.12. If T is a flexibly $\mathbb{E}$-graded monad, then a $(\vec{e}; e')$*-ary algebraic operation* consists of a grade-preserving function $\alpha_X : \prod_i TX(e_i \cdot -) \Rightarrow TX(e' \cdot -)$ for each graded set X, satisfying

$$f^{\dagger}_{e' \cdot d}(\alpha_{X,d}(t_1, \ldots, t_n)) = \alpha_{Y,d \cdot e}(f^{\dagger}_{e_1 \cdot d} t_1, \ldots, f^{\dagger}_{e_n \cdot d} t_n) \qquad (f : X \Rightarrow TY(- \cdot e),\, t_i \in TX(e_i \cdot d))$$

If R is a rigidly $\mathbb{E}$-graded monad, then a $(\vec{e}; e')$*-ary algebraic operation* consists of a grade-preserving function $\alpha_X : \prod_i RX(e_i \cdot -) \Rightarrow RX(e' \cdot -)$ for each set X, such that

$$f^{\dagger}_{e' \cdot d}(\alpha_{X,d}(r_1, \ldots, r_n)) = \alpha_{Y,d \cdot e}(f^{\dagger}_{e_1 \cdot d} r_1, \ldots, f^{\dagger}_{e_n \cdot d} r_n) \qquad (f : X \to RYe,\, r_i \in RX(e_i \cdot d))$$

Every $(\vec{e}; e')$-ary algebraic operation α for a flexibly graded monad T induces a $(\vec{e}; e')$-ary algebraic operation for the rigidly graded restriction $\lfloor \mathsf{T} \rfloor$—by restricting α to the graded sets $\hat{X}$.

Now suppose that (Σ, E) is a presentation, and consider the induced flexibly graded monad $\mathsf{T}^{(\Sigma,E)}$. Since there is an isomorphism $\mathbf{Alg}(\Sigma, E) \cong \mathbf{EM}(\mathsf{T}^{(\Sigma,E)})$ over $\mathbf{GSet}_{\mathbb{E}}$, the carrier of every $\mathsf{T}^{(\Sigma,E)}$-algebra forms a (Σ, E)-algebra. In particular, for every graded set X, the graded set $T^{(\Sigma,E)}X$ is the carrier of the free $\mathsf{T}^{(\Sigma,E)}$-algebra on X, and hence forms a (Σ, E)-algebra. For every $(\vec{e}; e')$-ary operation $\mathsf{op} \in \Sigma(\vec{e}; e')$, we therefore have grade-preserving functions

$$[\![\mathsf{op}]\!]^X : \textstyle\prod_i T^{(\Sigma,E)}X(e_i \cdot -) \Rightarrow T^{(\Sigma,E)}X(e' \cdot -)$$

The assignment of a grade-preserving function $[\![\mathsf{op}]\!]^X$ to each graded set X constitutes a $(\vec{e}; e')$-ary flexibly graded algebraic operation α_{op} for the flexibly graded monad $\mathsf{T}^{(\Sigma,E)}$. The assignment of $[\![\mathsf{op}]\!]^{\hat{X}}$ to each set X makes a $(\vec{e}; e')$-ary flexibly graded operation for the rigidly graded monad $\lfloor \mathsf{T}^{(\Sigma,E)} \rfloor$. That is, the rigidly graded monad induced by a flexibly graded presentation carries interpretations of all of the operations of the presentation.

Example 6.13. For the presentation (Σ, E) of V-valued stacks (Example 6.5), the induced algebraic operations for $\mathsf{Stk}_{\mathsf{flex}}$ and for Stk are the *push* and *pop* operations defined in Section 2.3.

Remark. Katsumata [2014] discussed the inconvenience of rigid grading. He introduced a notion of algebraic operation for rigidly graded monads based on effect functions.

In Katsumata's proposal, an algebraic operation is graded by an effect function. An n-ary *effect function* is a monotone function $\phi : |\mathbb{E}|^n \longrightarrow |\mathbb{E}|$ such that $\phi(e_1 \cdot d, \ldots, e_n \cdot d) = \phi(e_1, \ldots, e_n) \cdot d$ for all $e_1, \ldots, e_n, d$. A ϕ-ary algebraic operation of a rigidly graded monad R is a family of functions $\alpha_{X,e_1,\ldots,e_n} : RXe_1 \times \ldots \times RXe_n \longrightarrow RX(\phi(e_1, \ldots, e_n))$ for every $X, e_1, \ldots, e_n$ appropriately agreeing with coercion and the Kleisli extension.

An algebraic operation like this induces a flexibly graded algebraic operation in our sense for every individual point $(e_1, \ldots, e_n; \phi(e_1, \ldots, e_n))$ in the graph of ϕ. To capture one effect-function graded algebraic operation, we need a flexibly graded algebraic operation for each tuple $(e_1, \ldots, e_n)$ of mutually prime grades.

While the idea of effect functions is appealing, they are too restrictive to capture rigidly graded algebraic operations: an effect function ϕ has to be total, one cannot choose to define it only for $(e_1, \ldots, e_n)$ of the form $(d, \ldots, d)$.

This shortcoming can be solved by switching to effect relations. We could define an n-ary effect relation as a relation $\rho \subseteq |\mathbb{E}|^n \times |\mathbb{E}|$ such that $e_1 \leq e'_1, \ldots, e_n \leq e'_n$ and $\rho(e_1, \ldots, e_n; e)$ imply existence of e' such that $e \leq e'$ and $\rho(e'_1, \ldots, e'_n; e')$ for all $e_1, \ldots, e_n, e, e'_1, \ldots, e'_n$, and also $\rho(e_1, \ldots, e_n; e)$ implies $\rho(e_1 \cdot d, \ldots, e_n \cdot d; e \cdot d)$ for all $e_1, \ldots, e_n, e, d$. A rigidly graded algebraic operation of grade e' could then be graded by the relation $\{d, \ldots, d; e' \cdot d \mid d \in |\mathbb{E}|\}$.

6.5 Examples

We give some further examples of flexibly graded presentations, and the flexibly and rigidly graded monads they induce.

6.5.1 Global state. We give a flexibly Rel_V-graded presentation (Σ, E) for global V-valued state, where V is a finite set $\{\mathsf{v}_1, \ldots, \mathsf{v}_{|V|}\}$. This is analogous to the ungraded presentation described by Plotkin and Power [2002], and to the definition of a *mnemoid* [Melliès 2010]. The flexibly graded monad presented by (Σ, E) is $\mathsf{State}_{\mathsf{flex}}$, and the rigidly graded monad is State, both defined in Section 2.2. The operations are the following:

- a $(e_1, \ldots, e_{|V|};\ \gamma(e_1, \ldots, e_{|V|}))$-ary operation $\mathsf{get}_{\vec{e}}$ for each $e_1, \ldots, e_{|V|}$ where $\gamma : \mathsf{Rel}_V^{|V|} \longrightarrow \mathsf{Rel}_V$ is given by $\gamma(e_1, \ldots, e_{|V|}) = \lambda \mathsf{v}_i.\, e_i \mathsf{v}_i$;

- a $(1;\ (\lambda\_.\,\{w\}))$-ary operation put_w for each $w \in V$.

The equations are:

$$x_1 : e_1 \cdot d, \dots, x_{|V|} : e_{|V|} \cdot d \vdash \mathsf{get}_{\vec{e}}(d; x_1, \dots, x_{|V|}) \approx \mathsf{get}_{\vec{e}\cdot d}(1; x_1, \dots, x_{|V|}) : \gamma(\vec{e}) \cdot d \quad \text{(for each } \vec{e}, d)$$

$$x_1 : e_1, \dots, x_{|V|} : e_{|V|} \vdash (\gamma(\vec{e}) \le \gamma(\vec{e}'))^*(\mathsf{get}_{\vec{e}}(1; x_1, \dots)) \approx \mathsf{get}_{\vec{e}'}(1; (e_1 {\le} e_1')^* x_1, \dots) : \gamma(\vec{e}') \quad \text{(for each } \vec{e} \le \vec{e}')$$

$$x : 1 \vdash \mathsf{get}_{\lambda\_.\{\mathsf{v}_1\},\dots,\lambda\_.\{\mathsf{v}_{|V|}\}}(1; \mathsf{put}_{\mathsf{v}_1}(1; x), \dots, \mathsf{put}_{\mathsf{v}_{|V|}}(1; x)) \approx x : 1$$

$$x : 1 \vdash \mathsf{put}_{w'}((\lambda\_.\,\{w\});\ \mathsf{put}_w(1; x)) \approx \mathsf{put}_w(1; x) : (\lambda\_.\,\{w\}) \qquad \text{(for each } w, w' \in V)$$

$$x_1 : e_1, \dots, x_{|V|} : e_{|V|} \vdash \mathsf{put}_{\mathsf{v}_i}(\gamma(\vec{e});\ \mathsf{get}_{\vec{e}}(1; x_1, \dots)) \approx \mathsf{put}_{\mathsf{v}_i}(e_i; x_i) : (\lambda\_.e_i\mathsf{v}_i) \quad \text{(for each } i \le |V|, \vec{e})$$

The operations of the presentation (Σ, E) induce flexibly graded algebraic operations for the flexibly graded monad $\mathrm{State}_{\mathrm{flex}}$ and for the rigidly graded monad State. For State, these are exactly $get_{\vec{e}}$ and put_w, as defined in Section 2.2.

6.5.2 Backtracking nondeterminism, with cut. We give a flexibly graded presentation (Σ, E), similar to Piróg and Staton's ungraded presentation [2017], for the rigidly graded monad Cut defined in Section 2.4. (There is some redundancy; we do not claim this is the most efficient presentation.) The operations are

- a $(\ ;\bot)$-ary operation cut;
- a $(\ ;\top)$-ary operation fail;
- for each $e_1, e_2 \in \{\bot, 1, \top\}$, a $(e_1, e_2;\ e_1 \sqcap e_2)$-ary operation or_{e_1,e_2} where $\sqcap$ denotes meet.

The equations are:

$$x : e_1 \cdot d, y : e_2 \cdot d \vdash \mathsf{or}_{e_1,e_2}(d; x, y) \approx \mathsf{or}_{e_1\cdot d, e_2\cdot d}(1; x, y) : (e_1 \sqcap e_2) \cdot d \qquad \text{(for each } e_1, e_2, d)$$

$$x : e_1, y : e_2 \vdash (e_1{\sqcap}e_2{\le}e_1'{\sqcap}e_2')^*(\mathsf{or}_{e_1,e_2}(1; x, y)) \approx \mathsf{or}_{e_1',e_2'}(1; (e_1{\le}e_1')^* x, (e_2{\le}e_2')^* y) : e_1' \sqcap e_2' \quad \text{(for each } e_1 \le e_1', e_2 \le e_2')$$

$$x : e \vdash \mathsf{or}_{\top,e}(1; \mathsf{fail}(1;\ ), x) \approx x : e \qquad x : e \vdash x \approx \mathsf{or}_{e,\top}(1; x, \mathsf{fail}(1;\ )) : e \qquad \text{(for each } e)$$

$$x : e_1, y : e_2, z : e_3 \vdash \mathsf{or}_{e_1 \sqcap e_2, e_3}(1; \mathsf{or}_{e_1,e_2}(1; x, y), z) \approx \mathsf{or}_{e_1, e_2 \sqcap e_3}(1; x, \mathsf{or}_{e_2,e_3}(1; y, z)) : e \quad \text{(for each } e_1, e_2, e_3)$$

$$x : \bot, y : e \vdash \mathsf{or}_{\bot,e}(1; x, y) \approx x : \bot \qquad \text{(for each } e)$$

A notable aspect of this presentation is the equation at the bottom. We should of course have $\mathsf{or}_{\bot,e}(1; \mathsf{cut}, y) \approx \mathsf{cut}$, since this is precisely the behaviour of a cut. If we had this as an axiom instead, we would get strictly fewer equalities in the equational theory. We impose the equation $\mathsf{or}_{\bot,e}(1; x, y) \approx x$ for *every* x that definitely cuts—where the fact that x cuts is indicated only by having grade $\bot$. It is not possible to express equations like this, that hold only for variables of certain grades, in a rigidly graded presentation.

The flexibly graded presentation (Σ, E) does indeed induce the rigidly graded monad Cut, and hence also flexibly graded algebraic operations for Cut. These are:

$$[\![\mathsf{cut}]\!]_d : 1 \to \mathrm{Cut}X\bot \qquad [\![\mathsf{cut}]\!]_d\star = ([\,], \bot) \qquad [\![\mathsf{fail}]\!]_d : 1 \to \mathrm{Cut}X\top \qquad [\![\mathsf{fail}]\!]_d\star = ([\,], \top)$$

$$[\![\mathsf{or}_{e_1,e_2}]\!]_d : \mathrm{Cut}X(e_1 \cdot d) \times \mathrm{Cut}X(e_2 \cdot d) \to \mathrm{Cut}X((e_1 \sqcap e_2) \cdot d)$$

$$[\![\mathsf{or}_{e_1,e_2}]\!]_d((\vec{v}, \bot), (\vec{v}', c')) = (\vec{v}, \bot) \qquad [\![\mathsf{or}_{e_1,e_2}]\!]_d((\vec{v}, \top), (\vec{v}', c')) = (\vec{v} \mathbin{+\!\!+} \vec{v}', c')$$

6.6 Flexibly Graded Presentations from Rigidly Graded Presentations

We show that flexibly graded presentations are more general than rigidly graded presentations, in that every rigidly graded presentation induces a flexibly graded presentation with the same algebras.

Suppose that (Σ^r, E^r) is a rigidly graded presentation. We define a flexibly graded presentation (Σ^f, E^f), by treating every $(n; e')$-ary operation or equation of (Σ^r, E^r) as a $(1, \ldots, 1; e')$-ary operation or equation of (Σ^f, E^f) (the sets $\Sigma^f(\vec{e}, e')$ and $E^f(\vec{e}, e')$ are empty when $\vec{e}$ contains a grade that is not 1).

$$\Sigma^f(\underbrace{1, \ldots, 1}_{n}; e') = \Sigma^r(n; e') \qquad E^f(\underbrace{1, \ldots, 1}_{n}; e') = E^r(n; e')$$

Here we are treating the $(n; e')$-ary terms over Σ^r as $(1, \ldots, 1; e')$-ary terms over Σ^f. It is in fact trivial to show that (Σ^r, E^r)-algebras are the same as (Σ^f, E^f)-algebras – the definitions expand to the same thing. We therefore also obtain a flexibly graded monad $\mathsf{T}^{(\Sigma^f, E^f)}$ with the same algebras, along with the rigidly graded monad presented by (Σ^r, E^r), which is in fact $\lfloor \mathsf{T}^{(\Sigma^f, E^f)} \rfloor$.

Theorem 6.14. *For every rigidly $\mathbb{E}$-graded presentation (Σ^r, E^r), there is a flexibly $\mathbb{E}$-graded presentation (Σ^f, E^f), flexibly $\mathbb{E}$-graded monad $\mathsf{T}^{(\Sigma^f, E^f)}$, and rigidly $\mathbb{E}$-graded monad $\lfloor \mathsf{T}^{(\Sigma^f, E^f)} \rfloor$, together with isomorphisms over* $\mathbf{GSet}_{\mathbb{E}}$*:*

$$\mathbf{Alg}(\Sigma^r, E^r) \cong \mathbf{Alg}(\Sigma^f, E^f) \cong \mathbf{EM}(\mathsf{T}^{(\Sigma^f, E^f)}) \cong \mathbf{EM}(\lfloor \mathsf{T}^{(\Sigma^f, E^f)} \rfloor)$$

The operations of Σ^r induce algebraic operations. We say that an $(n; e')$*-ary algebraic operation* for a flexibly graded monad T is a $(1, \ldots, 1; e)$-ary algebraic operation for T (with n arguments). (A $(n; e')$-ary algebraic operation for a rigidly graded monad R, as defined in Section 4.4, is the same an $(1, \ldots, 1; e')$-ary algebraic operation for R). Since each $(n; e')$-ary operation $\mathrm{op} \in \Sigma^r(n; e')$ is an operation in Σ^f, and these induce algebraic operations as in Section 6.4, op induces a $(n; e')$-ary rigidly graded algebraic operation for the flexibly graded monad $\mathsf{T}^{(\Sigma^f, E^f)}$ (Hence also an algebraic operation for the rigidly $\mathbb{E}$-graded monad $\lfloor \mathsf{T}^{(\Sigma^f, E^f)} \rfloor \cong \mathsf{R}^{(\Sigma^r, E^r)}$; this is the same as the algebraic operation constructed in Section 4.4.)

7 GRADED CLONES

The goal of the remainder of this paper is to prove a correspondence between flexibly graded presentations and a class of flexibly graded monads. This correspondence in particular provides the proof that every flexibly graded presentation presents a flexibly graded monad (Theorem 6.10). We prove the correspondence in two steps: first we prove a correspondence between flexibly graded presentations and *flexibly graded clones* (Theorem 7.7), and then between *flexibly graded clones* and a class of flexibly graded monads. We also do the same for rigidly graded presentations. This section discusses the first step.

The *(abstract) clones* [Cohn 1981] of classical universal algebra axiomatize collections of terms, with variables and substitution. There is a folklore correspondence between classical presentations and clones. In one direction, given a presentation (Σ, E), the terms over Σ, quotiented by the equations, form a clone. In the other direction, the terms of a clone are the operations of the corresponding presentation. Clones provide a presentation-independent notion of algebraic theory.

7.1 Rigidly Graded Clones

We give the rigidly graded version of the correspondence, by first introducing the appropriate notion of *rigidly graded clone.*

Definition 7.1. A *rigidly $\mathbb{E}$-graded clone* R consists of

- for each natural number n, an $\mathbb{E}$-graded set Rn of *terms*;
- for each natural number n and positive integer $i \le n$, a term $\mathrm{var}_i \in Rn1$ (the ith *variable*);
- for each term $t \in Rne''$, grade $d \in |\mathbb{E}|$, and tuple of terms $u_1, \ldots, u_n \in Rmd$, a term $t[d; u_1, \ldots, u_n] \in Rm(e'' \cdot d)$ (*substitution*)

such that substitution is natural in $d \in \mathbb{E}$ and $e'' \in \mathbb{E}$, and satisfies the following equations:

$$\mathrm{var}_i[d; u_1, \ldots, u_n] = u_i \qquad t = t[1; \mathrm{var}_1, \ldots, \mathrm{var}_n]$$
$$(t[d; u_1, \ldots, u_n])[d'; v_1, \ldots, v_m] = t[(d \cdot d'); u_1[d'; v_1, \ldots, v_m], \ldots, u_n[d'; v_1, \ldots, v_m]]$$

Since these are *rigidly* graded, substitution requires the terms $u_1, \ldots, u_n$ to have the same grade. The main examples of rigidly graded clones are those induced by presentations as in the following construction, which is half of the rigidly graded presentation–monad correspondence.

Definition 7.2. Let (Σ, E) be a rigidly graded presentation. The rigidly graded clone $\mathsf{Term}^{(\Sigma,E)}$ of terms over (Σ, E) is defined as follows.

- The graded sets $\mathsf{Term}^{(\Sigma,E)}n$ are those of terms over Σ, quotiented by $\approx$ (as defined in Section 4).
- The ith variable var_i is the term $x_1, \ldots, x_n \vdash x_i : 1$.
- Substitution is defined by $t[d; u_1, \ldots, u_n] = t\{d; x_1 \mapsto u_1, \ldots, x_n \mapsto u_n\}$.

Each rigidly graded clone R induces a notion of R-*algebra*, consisting of a carrier equipped with interpretations of the terms of R. We omit the definition (it is similar to Definition 7.6 below). We state the correspondence between rigidly graded presentations and clones. (The proof is analogous to the proof of Theorem 7.7 below.)

Theorem 7.3. *We have the following correspondence between rigidly graded presentations and rigidly graded clones:*

(1) *For each rigidly $\mathbb{E}$-graded presentation (Σ, E), there is a rigidly $\mathbb{E}$-graded clone* $\mathsf{Term}^{(\Sigma,E)}$ *and isomorphism* $\mathbf{Alg}(\mathsf{Term}^{(\Sigma,E)}) \cong \mathbf{Alg}(\Sigma, E)$ *over* $\mathbf{GSet}_{\mathbb{E}}$.
(2) *For each rigidly $\mathbb{E}$-graded clone* R*, there is a rigidly $\mathbb{E}$-graded presentation* $(\Sigma^{\mathsf{R}}, E^{\mathsf{R}})$ *and isomorphism* $\mathbf{Alg}(\Sigma^{\mathsf{R}}, E^{\mathsf{R}}) \cong \mathbf{Alg}(\mathsf{R})$ *over* $\mathbf{GSet}_{\mathbb{E}}$.

7.2 Flexibly Graded Clones

We now turn to the analogous correspondence for flexibly graded presentations, which is similar to the correspondence for rigidly graded presentations. First, we introduce *flexibly graded clones.*

Definition 7.4. A *flexibly $\mathbb{E}$-graded clone* T consists of

- for each list $\vec{e} = (e_1, \ldots, e_n)$ of grades, an $\mathbb{E}$-graded set $T\vec{e}$ of *terms*;
- for each list of grades $(e_1, \ldots, e_n)$ and positive integer $i \le n$, a term $\mathrm{var}_i \in T(e_1, \ldots, e_n)e_i$ (the ith *variable*);
- for each term $t \in T(e'_1, \ldots, e'_n)e''$, grade d, and tuple of terms $u_1 \in T\vec{e}(e'_1 \cdot d), \ldots, u_n \in T\vec{e}(e'_n \cdot d)$, a term $t[d; u_1, \ldots, u_n] \in T\,\vec{e}\,(e'' \cdot d)$ (*substitution*)

such that substitution is natural in $d \in \mathbb{E}$ and $e'' \in \mathbb{E}$, and satisfies the following equations:

$$\mathrm{var}_i[d; u_1, \ldots, u_n] = u_i \qquad t = t[1; \mathrm{var}_1, \ldots, \mathrm{var}_n]$$
$$(t[d; u_1, \ldots, u_n])[d'; v_1, \ldots, v_m] = t[(d \cdot d'); u_1[d'; v_1, \ldots, v_m], \ldots, u_n[d'; v_1, \ldots, v_m]]$$

Here substitution does not require the terms $u_1, \ldots, u_n$ to all have the same grade.

Definition 7.5. Let (Σ, E) be a flexibly graded presentation. The flexibly graded clone $\mathsf{Term}^{(\Sigma,E)}$ of terms over (Σ, E) is defined as follows.

- The graded sets $\mathsf{Term}^{(\Sigma,E)}\vec{e}$ are those of terms over Σ, quotiented by $\approx$ (Section 6.1).
- The ith variable var_i is the term $x_1 : e_1, \ldots, x_n : e_n \vdash x_i : e_i$.
- Substitution is defined by $t[d; u_1, \ldots, u_n] = t\{d; x_1 \mapsto u_1, \ldots, x_n \mapsto u_n\}$.

Definition 7.6. Let T be a flexibly $\mathbb{E}$-graded clone. A T-*algebra* A is an $\mathbb{E}$-graded set A (the *carrier*), together with, for each term $t \in T(e_1, \dots, e_n)e'$ and grade d, a family of functions $[\![t]\!]_d : \prod_i A(e_i \cdot d) \to A(e' \cdot d)$. This family is required to be natural in $d \in \mathbb{E}$ and $e' \in \mathbb{E}$, and to respect variables and substitution in the sense that $[\![\mathrm{var}_i]\!]_d(a_1, \dots, a_n) = a_i$ and

$$[\![t[d; u_1, \dots, u_m]]\!]_{d'}(a_1, \dots, a_n) = [\![t]\!]_{d \cdot d'}([\![u_1]\!]_{d'}(a_1, \dots, a_n), \dots, [\![u_m]\!]_{d'}(a_1, \dots, a_n))$$

A *morphism* $f : \mathsf{A} -e'' \!\to \mathsf{A}'$ of grade e'' is a morphism $f : A -e\!\to A'$ of graded sets that satisfies

$$f_{e' \cdot d}([\![t]\!]_d(a_1, \dots, a_n)) = [\![t]\!]_{d \cdot e''}(f_{e_1 \cdot d} a_1, \dots, f_{e_n \cdot d} a_n) \qquad (t \in T\vec{e}e',\ a_i \in A(e_i \cdot d))$$

These form a locally $\mathbb{E}$-graded category $\mathbf{Alg}(\mathsf{T})$, and there is a forgetful functor $U_{\mathsf{T}} : \mathbf{Alg}(\mathsf{T}) \to \mathbf{GSet}_{\mathbb{E}}$, which sends each T-algebra to its carrier, and each morphism to itself.

Theorem 7.7. *We have the following correspondence between flexibly graded presentations and flexibly graded clones:*

(1) *For each flexibly $\mathbb{E}$-graded presentation (Σ, E), there is a flexibly $\mathbb{E}$-graded clone* $\mathsf{Term}^{(\Sigma,E)}$ *and isomorphism* $\mathbf{Alg}(\mathsf{Term}^{(\Sigma,E)}) \cong \mathbf{Alg}(\Sigma, E)$ *over* $\mathbf{GSet}_{\mathbb{E}}$.
(2) *For each flexibly $\mathbb{E}$-graded clone* T, *there is a flexibly $\mathbb{E}$-graded presentation* $(\Sigma^{\mathsf{T}}, E^{\mathsf{T}})$ *and isomorphism* $\mathbf{Alg}(\Sigma^{\mathsf{T}}, E^{\mathsf{T}}) \cong \mathbf{Alg}(\mathsf{T})$ *over* $\mathbf{GSet}_{\mathbb{E}}$.

Proof. For (1), the clone $\mathsf{Term}^{(\Sigma,E)}$ is defined in Definition 7.5. It remains to show that (Σ, E) and $\mathsf{Term}^{(\Sigma,E)}$ have the same algebras. Every (Σ, E)-algebra with carrier A admits interpretations of terms that respect $\approx$ by Theorem 6.9; these make A into a $\mathsf{Term}^{(\Sigma,E)}$-algebra. Conversely, given a $\mathsf{Term}^{(\Sigma,E)}$-algebra with carrier A, we can interpret the operations op using the terms $\mathrm{op}(1; x_1, \dots, x_n)$:

$$[\![\mathrm{op}]\!]_d = [\![\mathrm{op}(1; x_1, \dots, x_n)]\!]_d : \prod A(e_i \cdot d) \to A(e' \cdot d) \qquad (\mathrm{op} \in \Sigma(e_1, \dots, e_n; e'))$$

Simple calculations show that these constructions form the required isomorphism over $\mathbf{GSet}_{\mathbb{E}}$.

For (2), let T be a flexibly graded clone. We construct the corresponding flexibly graded presentation $(\Sigma^{\mathsf{T}}, E^{\mathsf{T}})$. The sets of operations are given by $\Sigma^{\mathsf{T}}(\vec{e}; e') = T\,\vec{e}\,e'$, so a $(\vec{e}; e')$-ary operation is a term $t \in T\vec{e}e'$. The collection of equations E^{T} consists of:

- For each $\vec{e}$ and i, a $(\vec{e}; e_i)$-ary equation $x_i \approx \mathrm{var}_i(1; x_1, \dots, x_n)$.
- For each term $t \in T(e'_1, \dots, e'_m)e''$ and tuple of terms $u_i \in T\vec{e}(e'_i \cdot d)$, a $(\vec{e}; e'' \cdot d)$-ary equation

$$t(d; u_1(1; x_1, \dots, x_n), \dots, u_m(1; x_1, \dots, x_n)) \approx (t[d; u_1, \dots, u_m])(1; x_1, \dots, x_n)$$

- For each $e' \leq e'' \in \mathbb{E}$ and term $t \in T(\vec{e}; e')$, a $(\vec{e}; e'')$-ary equation $((e' {\leq} e'')^* t)(1; x_1, \dots, x_n) \approx (e' {\leq} e'')^*(t(1; x_1, \dots, x_n))$.

Both T-algebras and $(\Sigma^{\mathsf{T}}, E^{\mathsf{T}})$-algebras have interpretations $[\![t]\!]_d : \prod_i A(e_i \cdot d) \to A(e' \cdot d)$ of each $t \in T\vec{e}'$. It follows from this that we have $\mathbf{Alg}(\Sigma^{\mathsf{T}}, E^{\mathsf{T}}) \cong \mathbf{Alg}(\mathsf{T})$ over $\mathbf{GSet}_{\mathbb{E}}$. □

We do not give the details here but, just as flexibly graded monads induce rigidly graded monads, each flexibly graded clone T induces a rigidly graded clone $\lfloor \mathsf{T} \rfloor$, and $\lfloor \mathsf{T} \rfloor$ satisfies a universal property that can be expressed in terms of its algebras. We can also go in the other direction, but this is not as simple as constructing a flexibly graded presentation from a rigidly graded presentation.

We end this section by noting that the substitution of a flexibly graded clone T restricts to a *variable renaming* operation. This will be useful in the next section. We define a locally graded category of flexibly graded contexts (viewed as lists of grades), with variable renamings as morphisms.

Definition 7.8. We write $\mathbf{FCtx}_{\mathbb{E}}$ for the locally $\mathbb{E}$-graded category in which objects are lists of grades, and morphisms, identities, and composition are given as follows:

$$\mathbf{FCtx}_{\mathbb{E}}(\vec{e}, \vec{e}')d = \textstyle\prod_i \{j \mid e'_j \leq e_i \cdot d\} \quad \mathrm{id}_{\vec{e}} = (1, \dots, n) \quad (k_1, \dots, k_m) \circ (j_1, \dots, j_n) = (k_{j_1}, \dots, k_{j_n})$$

The assignment $\vec{e} \mapsto T\vec{e}$ extends to a functor $T : \mathbf{FCtx}_{\mathbb{E}} \to \mathbf{GSet}_{\mathbb{E}}$, by

$$(T(j_1, \ldots, j_n))_{e''} t = t[d; (e'_{j_1} \leq e_1 \cdot d)^* \mathrm{var}_{j_1}, \ldots, (e'_{j_n} \leq e_n \cdot d)^* \mathrm{var}_{j_n}] \quad (j \in \mathbf{FCtx}_{\mathbb{E}}(\vec{e}, \vec{e}')d,\ t \in T\vec{e}\, e'')$$

8 GRADED MONAD–PRESENTATION CORRESPONDENCES

It is well-known that there is a correspondence between ordinary (ungraded) presentations and finitary monads on **Set**: given any presentation there is a finitary monad with the same algebras, and vice-versa. There are analogous correspondences for flexibly graded presentations and for rigidly graded presentations, which we give in this section.

8.1 Flexibly Graded Correspondence

We first consider the flexibly graded correspondence (Theorem 8.4 below), which is between flexibly graded presentations and flexibly graded monads satisfying a condition on colimits. To say which condition, we need some more machinery for locally graded categories.

Definition 8.1. Every locally graded category C has an *underlying* ordinary category $\underline{C}$ with the same objects; a morphism $f : X \to Y$ in $\underline{C}$ is a morphism $f : X \mathrel{-1\!\!\to} Y$ in C. Every functor $F : C \to \mathcal{D}$ between locally graded categories restricts to an ordinary functor $\underline{F} : \underline{C} \to \underline{\mathcal{D}}$.

The underlying ordinary category of $\mathbf{GSet}_{\mathbb{E}}$ is the category $[\mathbb{E}, \mathbf{Set}]$ of $\mathbb{E}$-graded sets and grade-preserving functions between them. We now define *conical colimits* in locally graded categories.

Definition 8.2. Let C be a locally graded category, and let $D : \mathbb{I} \to \underline{C}$ be an ordinary functor. A *cocone* of grade e is an object $X \in |C|$ equipped with a morphism $c_i : Di \mathrel{-e\!\!\to} X$ for each $i \in |\mathbb{I}|$, such that $c_{i'} \circ Df = c_i$ for each $f : i \to i'$ in $\mathbb{I}$. A cocone $\mathrm{in}_i : Di \mathrel{-1\!\!\to} \operatorname{colim} D$ of grade 1 is the *conical colimit* of D if, for every grade e and cocone $c_i : Di \mathrel{-e\!\!\to} X$ of grade e, there is a unique morphism $[c] : \operatorname{colim} D \mathrel{-e\!\!\to} X$ such that $c_i = [c] \circ \mathrm{in}_i$ for all i. A functor $F : C \to \mathcal{D}$ *preserves* this conical colimit if the cocone $(F(\operatorname{colim} D), F\mathrm{in}_i)$ is the conical colimit of $\underline{F} \circ D$ in $\mathcal{D}$.

(This is an instance of the notion of conical colimit given by Gordon and Power [1999], which generalizes the standard notion for categories enriched over a symmetric monoidal category [Kelly 1982].)

The locally graded category $\mathbf{GSet}_{\mathbb{E}}$ has conical colimits of small diagrams, given pointwise by ordinary colimits in **Set**. If $F : C \to \mathbf{GSet}_{\mathbb{E}}$ is a functor, then F preserves a conical colimit $\operatorname{colim} D$ whenever the ordinary functor $\underline{F} : \underline{C} \to [\mathbb{E}, \mathbf{Set}]$ preserves the ordinary colimit of D in $\underline{C}$. (However, existence of a ordinary colimit in $\underline{C}$ is not enough to guarantee existence of a conical colimit in C.)

The ordinary presentation–monad correspondence is usually stated in terms of *filtered* colimits, but can equivalently be stated in terms of the more general *sifted* colimits, because an endofunctor on **Set** preserves filtered colimits exactly when it preserves sifted colimits [Lack and Rosický 2011]. Here we have no choice; we need to use sifted colimits.

Definition 8.3. An ordinary small category $\mathbb{I}$ is *sifted* when ordinary colimits of shape $\mathbb{I}$ commute with finite products in **Set**. Explicitly this means, for all finite sets $\mathbb{J}$ and functors $D : \mathbb{I} \times \mathbb{J} \to \mathbf{Set}$, the canonical function

$$\operatorname{colim}_i \left(\textstyle\prod_j D(i,j) \right) \xrightarrow{[\prod_j \mathrm{in}_i]_i} \textstyle\prod_j \operatorname{colim}_i D(i,j)$$

is a bijection. A *conical sifted colimit* is a conical colimit of a diagram with sifted domain.

Theorem 8.4. *We have the following correspondence between flexibly graded presentations and a class of flexibly graded monads.*

(1) *For each flexibly $\mathbb{E}$-graded presentation (Σ, E), there is a flexibly $\mathbb{E}$-graded monad $\mathsf{T}^{(\Sigma,E)}$ such that $T^{(\Sigma,E)}$ preserves conical sifted colimits and $\mathbf{Alg}(\Sigma, E) \cong \mathbf{EM}(\mathsf{T}^{(\Sigma,E)})$ over $\mathbf{GSet}_{\mathbb{E}}$.*

(2) *For each flexibly $\mathbb{E}$-graded monad* T *such that* T *preserves conical sifted colimits, there is a flexibly* $\mathbb{E}$*-graded presentation* $(\Sigma^{\mathsf{T}}, E^{\mathsf{T}})$ *such that* $\mathbf{Alg}(\Sigma^{\mathsf{T}}, E^{\mathsf{T}}) \cong \mathbf{EM}(\mathsf{T})$ *over* $\mathbf{GSet}_{\mathbb{E}}$.

We outline the proof of the correspondence. Since we have already proved a correspondence between flexibly graded presentations and flexibly graded clones (Theorem 7.7), it actually suffices to prove a correspondence between flexibly graded clones and flexibly graded monads.

The first step is to characterize the conical-sifted-colimit-preserving functors $\mathbf{GSet}_{\mathbb{E}} \longrightarrow \mathbf{GSet}_{\mathbb{E}}$: they are equivalently functors $\mathbf{FCtx}_{\mathbb{E}} \longrightarrow \mathbf{GSet}_{\mathbb{E}}$.

Definition 8.5. For every list $\vec{e}$ of grades, we define an $\mathbb{E}$-graded set $K_{\mathbb{E}}\vec{e}$ by $K_{\mathbb{E}}\vec{e}e' = \{i \mid e_i \le e'\}$, with inclusions for coercions. If A is a graded set and d a grade, then there is a bijection ϑ between tuples $a = (a_i)_i \in \prod_i A(e_i \cdot d)$ and morphisms $K_{\mathbb{E}}\vec{e} \,{-d\!\!\to}\, A$ of graded sets as follows:

$$\vartheta : \textstyle\prod_i A(e_i \cdot d) \cong \mathbf{GSet}_{\mathbb{E}}(K_{\mathbb{E}}\vec{e}, A)d : \vartheta^{-1} \qquad (\vartheta a)_{e'}\, i = (e_i{\cdot}d{\le}e'{\cdot}d)^* a_i \qquad \vartheta^{-1} f = (f_{e_i} i)_i$$

We extend $K_{\mathbb{E}}$ to a functor $K_{\mathbb{E}} : \mathbf{FCtx}_{\mathbb{E}} \longrightarrow \mathbf{GSet}_{\mathbb{E}}$ by defining $K_{\mathbb{E}}(j_1, \dots, j_n) = \vartheta(j_1, \dots, j_n)$.

Given any functor $\mathbf{GSet}_{\mathbb{E}} \longrightarrow \mathbf{GSet}_{\mathbb{E}}$, we can compose with $K_{\mathbb{E}}$, to obtain a functor $\mathbf{FCtx}_{\mathbb{E}} \longrightarrow \mathbf{GSet}_{\mathbb{E}}$. To go in the other direction, we take the *left Kan extension* along $K_{\mathbb{E}}$.

Definition 8.6. Let $J : \mathcal{J} \longrightarrow C$ and $F : \mathcal{J} \longrightarrow \mathcal{D}$ be functors between locally graded categories. A functor $\mathrm{Lan}_J F : C \longrightarrow \mathcal{D}$ equipped with a natural family $\lambda_Z : FZ \,{-1\!\!\to}\, \mathrm{Lan}_J F(JZ)$ is the (pointwise) *left Kan extension* of F along J when, for all $X \in |C|$, $Y \in |\mathcal{D}|$, grades e, and natural families of functions $\alpha_{Z,d} : C(JZ, X)d \longrightarrow \mathcal{D}(FZ, Y)(d \cdot e)$, there is a unique morphism $[\alpha] : \mathrm{Lan}_J FX \,{-e\!\!\to}\, Y$ such that $\alpha_{Z,d} f = [\alpha] \circ \mathrm{Lan}_J Ff \circ \lambda_Z$.

The crucial property of $K_{\mathbb{E}}$ is the following lemma.

Lemma 8.7. *The functor* $K_{\mathbb{E}} : \mathbf{FCtx}_{\mathbb{E}} \longrightarrow \mathbf{GSet}_{\mathbb{E}}$ *is the cocompletion of* $\mathbf{FCtx}_{\mathbb{E}}$ *under conical sifted colimits. Explicitly this means, for every locally graded category* $\mathcal{D}$ *with conical sifted colimits and functor* $F : \mathbf{FCtx}_{\mathbb{E}} \longrightarrow \mathcal{D}$*, the left Kan extension of* F *along* $K_{\mathbb{E}}$ *exists, and is, up to isomorphism, the unique conical-sifted-colimit-preserving functor* $F^\sharp : \mathbf{GSet}_{\mathbb{E}} \longrightarrow \mathcal{D}$ *such that* $F \cong F^\sharp \cdot K_{\mathbb{E}}$.

Proof sketch. First consider the ordinary functor $\underline{K_{\mathbb{E}}} : \underline{\mathbf{FCtx}_{\mathbb{E}}} \longrightarrow [\mathbb{E}, \mathbf{Set}]$. The category $\underline{\mathbf{FCtx}_{\mathbb{E}}}$ has finite coproducts. By [Adámek and Rosický 2001, Section 2], the cocompletion of $\underline{\mathbf{FCtx}_{\mathbb{E}}}$ under sifted colimits is therefore given by the Yoneda embedding $y : \underline{\mathbf{FCtx}_{\mathbb{E}}} \longrightarrow \mathbf{Sind}(\underline{\mathbf{FCtx}_{\mathbb{E}}})$, where the codomain is the full subcategory of $[\underline{\mathbf{FCtx}_{\mathbb{E}}}^{\mathrm{op}}, \mathbf{Set}]$ on the finite-product-preserving functors. The singleton functor $\mathbb{E} \hookrightarrow \underline{\mathbf{FCtx}_{\mathbb{E}}}^{\mathrm{op}}$ is the completion of $\mathbb{E}$ under finite products, so we have an equivalence $\mathbf{Sind}(\underline{\mathbf{FCtx}_{\mathbb{E}}}) \simeq [\mathbb{E}, \mathbf{Set}]$. It follows that the ordinary functor $\underline{K_{\mathbb{E}}} : \underline{\mathbf{FCtx}_{\mathbb{E}}} \longrightarrow [\mathbb{E}, \mathbf{Set}]$ is the cocompletion of $\underline{K_{\mathbb{E}}}$ under sifted colimits.

It follows that $K_{\mathbb{E}}$ is itself a cocompletion, by noting that the left Kan extension of a functor $F : K_{\mathbb{E}} \longrightarrow C$ along $K_{\mathbb{E}}$ is actually given by the left Kan extension of $\underline{F}$ along $\underline{K_{\mathbb{E}}}$.[3] □

From this it follows that functors $\mathbf{GSet}_{\mathbb{E}} \longrightarrow \mathbf{GSet}_{\mathbb{E}}$ preserve conical sifted colimits exactly when they are left Kan extensions along $K_{\mathbb{E}}$, and that left Kan extension provides an equivalence between such functors and functors $\mathbf{FCtx}_{\mathbb{E}} \longrightarrow \mathbf{GSet}_{\mathbb{E}}$.

The next step is to show that this equivalence restricts, so that to make a conical-sifted-colimit-preserving functor $T : \mathbf{GSet}_{\mathbb{E}} \longrightarrow \mathbf{GSet}_{\mathbb{E}}$ into the underlying functor of a flexibly graded monad T is equivalently to make $T' = T \cdot K_{\mathbb{E}} : \mathbf{FCtx}_{\mathbb{E}} \longrightarrow \mathbf{GSet}_{\mathbb{E}}$ into the underlying functor of a flexibly graded clone T', in such a way that morphisms of flexibly graded monads become morphisms of flexibly graded clones. Since $K_{\mathbb{E}}$ is a cocompletion (in particular, it is *dense*), families of morphisms $\eta_X : X \,{-1\!\!\to}$

[3]This fact is particular to $K_{\mathbb{E}}$. As far as we know it does not hold for Kan extensions along other functors.

TX natural in X are completely determined by the components $\eta_{K_{\mathbb{E}}\vec{e}} : K_{\mathbb{E}}\vec{e} -1\to T'\vec{e}$, equivalently $\vartheta\eta_{K_{\mathbb{E}}\vec{e}} \in \prod_i T'\vec{e}e_i$. The latter gives the variables of the flexibly graded clone T'. Since T is a left Kan extension along $K_{\mathbb{E}}$, each natural family of morphisms $(-)^\dagger : \mathbf{GSet}_{\mathbb{E}}(X, TY)d \longrightarrow \mathbf{GSet}_{\mathbb{E}}(TX, TY)d$ is determined by its restriction to the components $(-)^\dagger : \mathbf{GSet}_{\mathbb{E}}(K_{\mathbb{E}}\vec{e}', TY)d \longrightarrow \mathbf{GSet}_{\mathbb{E}}(T'\vec{e}', TY)d$ Since $K_{\mathbb{E}}$ is a cocompletion, the functors $\mathbf{GSet}_{\mathbb{E}}(K_{\mathbb{E}}\vec{e}', -)$ preserve left Kan extensions along $K_{\mathbb{E}}$, so $(-)^\dagger$ is determined by its further restriction to $Y = K_{\mathbb{E}}\vec{e}$. Hence $(-)^\dagger$ is determined by the functions

$$\textstyle\prod_i T'\vec{e}(e'_i \cdot d) \overset{\vartheta}{\cong} \mathbf{GSet}_{\mathbb{E}}(K_{\mathbb{E}}\vec{e}', T'\vec{e})d \xrightarrow{(-)^\dagger} \mathbf{GSet}_{\mathbb{E}}(T'\vec{e}', T'\vec{e})d$$

These functions give substitution in T': for a term $t \in T'\vec{e}'e''$ and tuple of terms $u_i \in T'\vec{e}(e'_i \cdot d)$, we have $t[d; u_1, \ldots, u_n] = (\vartheta u)^\dagger_{e''} t \in T'\vec{e}(e'' \cdot d)$.

Finally, we need to show that T and T' have the same algebras, in the sense that there is an isomorphism $\mathbf{EM}(\mathsf{T}) \cong \mathbf{Alg}(\mathsf{T})$ over $\mathbf{GSet}_{\mathbb{E}}$. Given an algebra A for the flexibly graded monad T, we can use a construction similar to the definition of substitution above to make the carrier A into an algebra for the flexibly graded clone T'. Specifically, we specialize the extension operator $(-)^\ddagger$ to the graded sets $K_{\mathbb{E}}\vec{e}'$, and then define $[\![t]\!]_d(a_1, \ldots, a_n) = (\vartheta a)^\ddagger_{e''} t \in A(e'' \cdot d)$ for $t \in T'\vec{e}'e''$. Since T is the left Kan extension of T' along $K_{\mathbb{E}}$, $(-)^\ddagger$ is completely determined by its restriction to the graded sets $K_{\mathbb{E}}\vec{e}'$. It follows that this construction forms an isomorphism $\mathbf{EM}(\mathsf{T}) \cong \mathbf{Alg}(\mathsf{T}')$ over $\mathbf{GSet}_{\mathbb{E}}$.

8.2 Rigidly Graded Correspondence

We also outline the correspondence for rigidly graded presentations. This is essentially the same as the correspondence proved by Kura [2020], except that we rephrase it in terms of sifted colimits and locally graded categories.

Let $\mathbf{RSet}_{\mathbb{E}}$ be the locally $\mathbb{E}$-graded category in which objects are sets, morphisms $f : X -e\to Y$ only exist when $1 \leq e$, in which case they are functions $f : X \longrightarrow Y$, and identities and composition are as in $\mathbf{Set}$. Every rigidly graded monad R has an underlying functor $R : \mathbf{RSet}_{\mathbb{E}} \longrightarrow \mathbf{GSet}_{\mathbb{E}}$, and $\mathbf{RSet}_{\mathbb{E}}$ has conical sifted colimits computed as in $\mathbf{Set}$.

Theorem 8.8. *We have the following correspondence between rigidly graded presentations and a class of rigidly graded monads.*

(1) *For each rigidly $\mathbb{E}$-graded presentation (Σ, E), there is a rigidly $\mathbb{E}$-graded monad $\mathsf{R}^{(\Sigma,E)}$ such that $R^{(\Sigma,E)}$ preserves conical sifted colimits and $\mathbf{Alg}(\Sigma, E) \cong \mathbf{EM}(\mathsf{R}^{(\Sigma,E)})$ over $\mathbf{GSet}_{\mathbb{E}}$.*
(2) *For each rigidly $\mathbb{E}$-graded monad R such that R preserves conical sifted colimits, there is a rigidly $\mathbb{E}$-graded presentation $(\Sigma^{\mathsf{R}}, E^{\mathsf{R}})$ such that $\mathbf{Alg}(\Sigma^{\mathsf{R}}, E^{\mathsf{R}}) \cong \mathbf{EM}(\mathsf{R})$ over $\mathbf{GSet}_{\mathbb{E}}$.*

The proof is similar to the proof of the flexible correspondence. There is a locally graded category $\mathbf{RCtx}_{\mathbb{E}}$ in which objects are natural numbers and morphisms are given by $\mathbf{RCtx}_{\mathbb{E}}(n, m)e = \prod_{i \leq n}\{1, \ldots, m\}$ when $1 \leq e$, with $\mathbf{RCtx}_{\mathbb{E}}(n, m)e$ empty otherwise. This embeds into $\mathbf{RSet}_{\mathbb{E}}$ via a functor $J_{\mathbb{E}} : n \mapsto \{1, \ldots, n\} : \mathbf{RCtx}_{\mathbb{E}} \longrightarrow \mathbf{RSet}_{\mathbb{E}}$, which is the cocompletion of $\mathbf{RCtx}_{\mathbb{E}}$ under conical sifted colimits. It follows that conical-sifted-colimit-preserving functors $\mathbf{RSet}_{\mathbb{E}} \longrightarrow \mathbf{GSet}_{\mathbb{E}}$ are equivalently functors $\mathbf{RCtx}_{\mathbb{E}} \longrightarrow \mathbf{GSet}_{\mathbb{E}}$. This equivalence extends to the required correspondence.

9 RELATED WORK

(Rigidly) graded monads and presentations. Rigidly graded monads are a special case of *lax functors*. The formal properties of the latter were studied by Street [1972], and adapted to graded monads by Fujii et al. [2016]. In mathematics, graded monads were used as a generalization of rings by Durov [2007] to study Arakelov geometry. Later, Smirnov [2008] studied the free construction from generators of monads graded by commutative monoids. Graded algebraic theories with rigid

grading of operations were studied by Dorsch et al. [2019]; Milius et al. [2015], who graded only by natural numbers with addition. Kura [2020] generalized these to grading by strict monoidal categories, and established a correspondence with graded monads and a notion of graded Lawvere theory.

Algebraic operations. Katsumata [2014] discussed the inconvenience of rigid grading of algebraic operations for rigidly graded monads and introduced effect-function graded algebraic operations. As we discussed in Section 6.4, our rigidly graded algebraic operations are a special case of Katsumata's if one replaces functions with relations, but our flexibly graded algebraic operations are not. Plotkin and Power [2003] define a general notion of algebraic operation for monads enriched over a symmetric monoidal category. As remarked in Section 3, flexibly $\mathbb{E}$-graded monads are monads enriched over $[\mathbb{E}, \mathbf{Set}]$ with Day convolution as the tensor product. For general non-symmetric $\mathbb{E}$, the monoidal category $[\mathbb{E}, \mathbf{Set}]$ is not symmetric, and symmetry appears to be essential in Plotkin and Power's definition. The relationship between the latter and our flexibly graded algebraic operations is therefore unclear.

Presentation–monad correspondences. To prove the correspondences between graded presentations and graded monads, we go via graded clones. This is essentially the same as the technique used by Kelly and Power [1993], though they do not mention clones explicitly. Staton [2013] proves a correspondence for a particular notion of presentation, explicitly using a notion of *enriched clone*. Staton also notes that enriched clones generalize *relative monads* [Altenkirch et al. 2015]. Altenkirch et al. [2015] show a correspondence between J-relative monads and a class of monads, when J satisfies certain *well-behavedness* conditions, which Szlachányi [2017] shows are equivalent to J being a cocompletion. The correspondence between classical abstract clones and finitary monads is an instance, because abstract clones are J-relative monads where J is the inclusion of finite sets in **Set**. Similar considerations apply to our flexibly graded correspondence, and our proof of the correspondence is similar to Altenkirch et al.'s proof. For the appropriate locally graded notion of relative monad, flexibly graded clones are $K_{\mathbb{E}}$-relative monads. The functor $K_{\mathbb{E}}$ is a cocompletion, so is well-behaved (in a locally graded sense). The rigidly graded correspondence is between two classes of relative monad, so the situation is slightly more complicated. Rigidly graded clones are $(\hat{(-)} \cdot J_{\mathbb{E}})$-relative monads, while rigidly graded monads are $\hat{(-)}$-relative monads.

10 CONCLUSIONS

This paper contributes the new notion of flexibly graded presentation, to enable more natural presentation of many (rigidly) graded monads, such as the length-graded list monad and many others. Flexibly graded present the same class of rigidly graded monads as rigidly graded presentations, but they also present a wide class of McDermott and Uustalu's [2022] flexibly graded monads.

Central to our development here were two tools, which we suppressed somewhat, in order to keep the exposition elementary: locally graded categories and relative monads (as adapted to locally graded category theory). Relative monads appear in two places: while flexibly graded monads are simply monads on the locally graded category $\mathbf{GSet}_{\mathbb{E}}$, rigidly graded monads are monads relative to the inclusion $\hat{(-)} : \mathbf{RSet}_{\mathbb{E}} \longrightarrow \mathbf{GSet}_{\mathbb{E}}$; flexibly graded clones are monads relative to $K_{\mathbb{E}} : \mathbf{FCtx}_{\mathbb{E}} \longrightarrow \mathbf{GSet}_{\mathbb{E}}$.

ACKNOWLEDGMENTS

We thank Nathanael Arkor and Georgy Lukyanov for helpful discussions. D.M. and T.U. were supported by the Icelandic Research Fund grants no. 196323-053 and 228684-051. S.K. was supported by the JST project no. JPMJER1603 (ERATO HASUO), N.W. by the EPSRC grant no. EP/S028129/1.

REFERENCES

Jiří Adámek and Jiří Rosický. 2001. On Sifted Colimits and Generalized Varieties. *Theory Appl. Categ.* 8, 3 (2001), 33–53. http://www.tac.mta.ca/tac/volumes/8/n3/8-03abs.html Revised 2007.

Thorsten Altenkirch, James Chapman, and Tarmo Uustalu. 2015. Monads Need Not Be Endofunctors. *Log. Methods Comput. Sci.* 11, 1, Article 3 (2015), 40 pages. https://doi.org/10.2168/lmcs-11(1:3)2015

Paul M. Cohn. 1981. *Universal Algebra* (revised ed.). Mathematics and Its Applications, Vol. 6. D. Reidel Publ. Co., Dordrecht, Boston, London. https://doi.org/10.1007/978-94-009-8399-1

Ulrich Dorsch, Stefan Milius, and Lutz Schröder. 2019. Graded Monads and Graded Logics for the Linear Time–Branching Time Spectrum. In *30th Int. Conf. on Concurrency Theory, CONCUR 2019*, Wan Fokkink and Rob van Glabbeek (Eds.). Leibniz Int. Proc. in Informatics, Vol. 140. Dagstuhl Publishing, Saarbrücken/Wadern, 36:1–36:16. https://doi.org/10.4230/lipics.concur.2019.36

Nikolai Durov. 2007. New Approach to Arakelov Geometry. arXiv eprint 0704.2030. arXiv:0704.2030 [math.AG] https://arxiv.org/abs/0704.2030

Tobias Fritz and Paolo Perrone. 2019. A Probability Monad as the Colimit of Spaces of Finite Samples. *Theory Appl. Categ.* 34, 7 (2019), 170–220. http://www.tac.mta.ca/tac/volumes/34/7/34-07abs.html

Soichiro Fujii, Shin-ya Katsumata, and Paul-André Melliès. 2016. Towards a Formal Theory of Graded Monads. In *Foundations of Software Science and Computation Structures: 19th Int. Conf., FoSSaCS 2016, Eindhoven, The Netherlands, April 2–8, 2016, Proceedings*, Bart Jacobs and Christof Löding (Eds.). Lect. Notes in Comput. Sci., Vol. 9634. Springer, Cham, 513–530. https://doi.org/10.1007/978-3-662-49630-5_30

Marco Gaboardi, Shin-ya Katsumata, Dominic Orchard, and Tetsuya Sato. 2021. Graded Hoare Logic and Its Categorical Semantics. In *Programming Languages and Systems: 30th Europ. Symp. on Programming, ESOP 2021, Luxembourg City, Luxembourg, March 27 – April 1, 2021, Proceedings*, Nobuko Yoshida (Ed.). Lect. Notes in Comput. Sci., Vol. 12648. Springer, Cham, 234–263. https://doi.org/10.1007/978-3-030-72019-3_9

Sergey Goncharov. 2013. Trace Semantics via Generic Observations. In *Algebra and Coalgebra in Computer Science: 5th Int. Conf. CALCO 2013, Warsaw, Poland, September 3–6, 2013, Proceedings*, Reiko Heckel and Stefan Milius (Eds.). Lect. Notes in Comput. Sci., Vol. 8089. Springer, Berlin, Heidelberg, 158–174. https://doi.org/10.1007/978-3-642-40206-7_13

Robert Gordon and A. John Power. 1999. Gabriel-Ulmer Duality for Categories Enriched in Bicategories. *J. Pure Appl. Alg.* 137, 1 (1999), 29–48. https://doi.org/10.1016/s0022-4049(97)00167-9

Martin Hyland, Gordon Plotkin, and John Power. 2006. Combining Computational Effects: Commutativity and Sum. *Theor. Comput. Sci.* 357, 1–3 (2006), 70–99. https://doi.org/10.1016/j.tcs.2006.03.013

Ohad Kammar and Gordon D. Plotkin. 2012. Algebraic Foundations for Effect-Dependent Optimisations. In *Proc. of 39th Ann. ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages, POPL '12, Philadelphia, PA, USA, January 22–28, 2012.* ACM Press, New York, 349–360. https://doi.org/10.1145/2103656.2103698

Shin-ya Katsumata. 2014. Parametric Effect Monads and Semantics of Effect Systems. In *Proc. of 41st Ann. ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages, POPL '14, San Diego, CA, USA, January 20-21, 2014.* ACM Press, New York, 633–645. https://doi.org/10.1145/2535838.2535846

G. Max Kelly. 1982. *Basic Concepts of Enriched Category Theory.* London Math. Soc. Lecture Note Series, Vol. 64. Cambridge University Press, Cambridge. Reprinted (2005) as: *Reprints in Theory and Applications of Categories* 10, http://www.tac.mta.ca/tac/reprints/articles/10/tr10abs.html.

G. Max Kelly and A. John Power. 1993. Adjunctions Whose Counits Are Coequalizers, and Presentations of Finitary Enriched Monads. *J. Pure Appl. Alg.* 89, 1–2 (1993), 163–179. https://doi.org/10.1016/0022-4049(93)90092-8

Satoshi Kura. 2020. Graded Algebraic Theories. In *Foundations of Software Science and Computation Structures: 23rd Int. Conf., FoSSaCS 2020, Dublin, Ireland, April 25–30, 2020, Proceedings*, Jean Goubault-Larrecq and Barbara König (Eds.). Lect. Notes in Comput. Sci., Vol. 12077. Springer, Cham, 401–421. https://doi.org/10.1007/978-3-030-45231-5_21

Stephen Lack and Jiří Rosický. 2011. Notions of Lawvere Theory. *Appl. Categ. Struct.* 19, 1 (2011), 363–391. https://doi.org/10.1007/s10485-009-9215-2

Paul Blain Levy. 2019. Locally Graded Categories. Slides. https://www.cs.bham.ac.uk/~pbl/papers/locgrade.pdf

John M. Lucassen and David K. Gifford. 1988. Polymorphic Effect Systems. In *Conf. Record of 15th Ann. ACM Symp. on Principles of Programming Languages, POPL '88, San Diego, CA, USA, January 10–13, 1988* (San Diego, California, USA). ACM Press, New York, 47–57. https://doi.org/10.1145/73560.73564

Dylan McDermott and Tarmo Uustalu. 2022. Flexibly Graded Monads and Graded Algebras. In *Mathematics of Program Construction: 14th Int. Conf., MPC 2022, Tbilisi, Georgia, September 26–28, 2022, Proceedings*, Ekaterina Komendantskaya (Ed.). Lect. Notes in Comput. Sci. Springer, Cham. (to appear).

Paul-André Melliès. 2010. Segal Condition Meets Computational Effects. In *Proc. of 25th Ann. IEEE Symp. on Logic in Computer Science, LICS '10, 11–14 July 2010, Edinburgh, United Kingdom.* IEEE, Los Alamitos, CA, 150–159. https://doi.org/10.1109/lics.2010.46

Paul-André Melliès. 2012. Parametric Monads and Enriched Adjunctions. Manuscript. https://www.irif.fr/~mellies/tensorial-logic/8-parametric-monads-and-enriched-adjunctions.pdf

Stefan Milius, Dirk Pattinson, and Lutz Schröder. 2015. Generic Trace Semantics and Graded Monads. In *6th Conf. on Algebra and Coalgebra in Computer Science, CALCO 2015*, Lawrence S. Moss and Paweł Sobociński (Eds.). Leibniz Int. Proceedings in Informatics, Vol. 35. Dagstuhl Publishing, Saarbrücken/Wadern, 253–269. https://doi.org/10.4230/lipics.calco.2015.253

Alan Mycroft, Dominic Orchard, and Tomas Petricek. 2016. Effect Systems Revisited: Control-Flow Algebra and Semantics. In *Semantics, Logics, and Calculi: Essays Dedicated to Hanne Riis Nielson and Flemming Nielson on the Occasion of Their 60th Birthdays*, Christian W. Probst, Chris Hankin, and René Rydhof Hansen (Eds.). Lect. Notes in Comput. Sci., Vol. 9560. Springer, Cham, 1–32. https://doi.org/10.1007/978-3-319-27810-0_1

Maciej Piróg and Sam Staton. 2017. Backtracking with Cut via a Distributive Law and Left-Zero Monoids. *J. Funct. Program.* 27, Article e17 (2017), 15 pages. https://doi.org/10.1017/s0956796817000077

Gordon Plotkin and John Power. 2002. Notions of Computation Determine Monads. In *Foundations of Software Science and Computation Structures: 5th Int. Conf., FoSSaCS 2002, Grenoble, France, April 8–12, 2002, Proceedings*, Mogens Nielsen and Uffe Engberg (Eds.). Lect. Notes in Comput. Sci., Vol. 2303. Springer, Berlin, Heidelberg, 342–356. https://doi.org/10.1007/3-540-45931-6_24

Gordon Plotkin and John Power. 2003. Algebraic Operations and Generic Effects. *Appl. Categ. Struct.* 11 (2003), 69–94. https://doi.org/10.1023/a:1023064908962

A.L. Smirnov. 2008. Graded Monads and Rings of Polynomials. *J. Math. Sci.* 151, 3 (2008), 3032–3051. https://doi.org/10.1007/s10958-008-9013-7

Sam Staton. 2013. An Algebraic Presentation of Predicate Logic. In *Foundations of Software Science and Computation Structures: 16th Int. Conf., FOSSACS 2013, Rome, Italy, March 16–24, 2013, Proceedings*, Frank Pfenning (Ed.). Lect. Notes in Comput. Sci., Vol. 7794. Springer, Berlin, Heidelberg, 401–417. https://doi.org/10.1007/978-3-642-37075-5_26

Ross Street. 1972. Two Constructions on Lax Functors. *Cah. Topol. Géom. Diff. Catég.* 13, 3 (1972), 217–264. http://www.numdam.org/item/CTGDC_1972__13_3_217_0

Kornél Szlachányi. 2017. On the Tensor Product of Modules over Skew Monoidal Actegories. *J. Pure Appl. Algebra* 221, 1 (2017), 185–221. https://doi.org/10.1016/j.jpaa.2016.06.003

Richard J. Wood. 1976. *Indicial Methods for Relative Categories.* Ph. D. Dissertation. Dalhousie University. http://hdl.handle.net/10222/55465