How to construct graded monads

Dylan McDermott
Computational effects, with grades

Each computation has a grade \( e \in G \), where \( (G, \leq, 1, \cdot) \) is an ordered monoid.

\[
\frac{\Gamma \vdash V : A}{\Gamma \vdash \text{return} V : A \& 1} \quad \frac{\Gamma \vdash M_1 : A \& e_1 \quad \Gamma, x : A \vdash M_2 : A \& e_2}{\Gamma \vdash (\text{do } x \leftarrow M_1 ; M_2) : A \& (e_1 \cdot e_2)} \quad \frac{\Gamma \vdash M : A \& e \quad e \leq e'}{\Gamma \vdash M : A \& e'}
\]

Interpret computations using a graded monad \( \mathcal{R} \):

\[
\llbracket \Gamma \vdash M : A \& e \rrbracket : \llbracket \Gamma \rrbracket \to \mathcal{R} \llbracket A \rrbracket
\]

instead of a monad \( T \):

\[
\llbracket \Gamma \vdash M : A \& e \rrbracket : \llbracket \Gamma \rrbracket \to T \llbracket A \rrbracket
\]

Example: may analysis for global state uses the ordered monoid

\( (\mathcal{P}\{\text{get, put}\}, \subseteq, \emptyset, \cup) \)

so that \( e \subseteq \{\text{get, put}\} \)
Example: backtracking with cut

\[
\text{or(or(or(or(return11, return12), fail), \\
or(return13, cut)), return14) : int & } \top
\]

Effectful operations:

\[
\begin{align*}
\Gamma \vdash M_1 : A &\& e_1 & \quad \Gamma \vdash M_2 : A &\& e_2 \\
\quad \quad \quad \quad \quad \top &\quad \quad \quad \quad \quad \top \\
\Gamma \vdash \text{or}(M_1, M_2) : A &\& (e_1 \sqcap e_2) & \quad \Gamma \vdash \text{fail} : A &\& \top \\
\Gamma \vdash \text{cut} : A &\& \bot
\end{align*}
\]

Ordered monoid \((\{\bot, 1, \top\}, \leq, 1, \cdot)\):

\[
\begin{align*}
\top & \quad \text{don’t know anything} \\
\forall & \quad \top \cdot e = \top \\
1 & \quad \text{definitely cuts or returns something} \\
\forall & \quad 1 \cdot e = e \\
\bot & \quad \text{definitely cuts} \\
\forall & \quad \bot \cdot e = \bot
\end{align*}
\]
Monads

A *monad* $T$ consists of

- a set $TX$ for each set $X$;
- a function $\text{return} : X \to TX$ for each $X$;
- a function $(\gg) : TX \to (X \to TY) \to TY$ for each $X, Y$;

satisfying the monad laws:

\[
\begin{align*}
\text{return} x \gg f &= f x & \text{(left unit)} \\
 t &= t \gg \text{return} & \text{(right unit)} \\
 (t \gg f) \gg g &= t \gg \lambda y.(f y \gg g) & \text{(associativity)}
\end{align*}
\]
Graded monads

A graded monad $R$ consists of

- an ordered monoid $(G, \leq, 1, \cdot)$ (grades);
- a set $RX$ for each grade $e \in G$, set $X$ (computations of grade $e$);
- a function $\text{return} : X \rightarrow R1X$ for each $X$;
- a function $(\prod_{e_1, e_2} : R(e_1 \cdot e_2)Y \rightarrow R1X)$ (bind) for each $e_1, e_2 \in G$, $X, Y$;
- a function $R(e \leq e')X : RX \rightarrow Re'X$ (subgrading) for each $e \leq e'$, $X$

satisfying the monad laws

\[
\text{return}^{1,e}x \prod_{e} f = fx \quad \text{(left unit)}
\]
\[
r = r \prod_{e, 1} \text{return} \quad \text{(right unit)}
\]
\[
(r \prod_{e_1, e_2} e_1 \cdot e_2, e_3) \prod_{e} g = r \prod_{e_1, e_2, e_3} \lambda y.(fy \prod_{e_2, e_3} g) \quad \text{(associativity)}
\]

and some laws about subgrading
Example: may analysis for global state

For a non-empty set $S$ of states:

- ordered monoid $(\mathcal{P}\{\text{get, put}\}, \subseteq, \emptyset, \cup)$
- sets of computations

\[
\begin{align*}
R\emptyset X &= X & R\{\text{get}\} X &= S \Rightarrow X \\
R\{\text{put}\} X &= (1 + S) \times X & R\{\text{get, put}\} X &= S \Rightarrow S \times X
\end{align*}
\]

- return $= \text{id} : X \rightarrow R\emptyset X$
- 16 cases of $\gg$
- 9 cases of $R(e \leq e')$
- 64 associativity laws
- some other laws
Example: backtracking with cut

\[ R_e X = \{(xs, c) \in \text{List}_X \times \{\text{cut, nocut}\} \mid (e = \bot \Rightarrow c = \text{cut}) \land (e = 1 \Rightarrow c = \text{cut} \lor xs \neq [])\} \]

- \top: don’t know anything
- \bot: definitely cuts
- 1: definitely cuts or returns something
Gradings of monads

A grading $R$ of a (non-graded) monad $T$ consists of

- an ordered monoid $(G, \leq, 1, \cdot)$
- a subset $R e X \subseteq T X$ for each $e \in G$, set $X$

such that

- $R e X \subseteq R e' X$ for $e \leq e'$
- the return and bind functions

\[
\text{return} : X \rightarrow TX \quad (\gg) : TX \rightarrow (X \rightarrow TY) \rightarrow TY
\]

restrict to functions

\[
\text{return} : X \rightarrow R 1 X \quad (e_1, e_2) : Re_1 X \rightarrow (X \rightarrow Re_2 Y) \rightarrow R(e_1 \cdot e_2) Y
\]

The restricted functions are the return and bind of a graded monad $R$, with subgrading functions $R(e \leq e') X : Re X \subseteq Re' X$
Gradings of monads

▶ Backtracking:

\[ ReX = \{(xs, c) \in TX \mid (e = \bot \Rightarrow c = \text{cut}) \land (e = 1 \Rightarrow c = \text{cut} \lor xs \neq [])\} \]

where

\[ TX = \text{List} X \times \{\text{cut}, \text{nocut}\} \]

▶ Global state:

\[ ReX \cong \{t \in TX \mid (\text{put} \notin e \Rightarrow (\text{fst} \circ t) \text{ is identity}) \land (\text{get} \notin e \Rightarrow (\text{fst} \circ t) \text{ is constant or identity} \land (\text{snd} \circ t) \text{ is constant})\} \]

where

\[ TX = S \Rightarrow S \times X \]
Gradings are good for program reasoning

If $T$ forms an adequate model

$$
\begin{align*}
\llbracket \Gamma \vdash M : A \rrbracket_T : \llbracket \Gamma \rrbracket_T \rightarrow T \llbracket A \rrbracket_T \\
\llbracket M \rrbracket_T = \llbracket N \rrbracket_T \quad \Rightarrow \quad M \simeq_{\text{ctx}} N
\end{align*}
$$

then any grading $R$ of $T$ also forms an adequate model

$$
\llbracket M \rrbracket_R = \llbracket N \rrbracket_R \quad \Rightarrow \quad M \simeq_{\text{ctx}} N \quad \text{where} \quad \llbracket \Gamma \vdash M : A \& e \rrbracket_R : \llbracket \Gamma \rrbracket_R \rightarrow R e \llbracket A \rrbracket_R
$$

but $\llbracket M \rrbracket_R = \llbracket N \rrbracket_R$ is usually weaker (and easier to prove) than $\llbracket M \rrbracket_T = \llbracket N \rrbracket_T$
How to construct graded monads

Supply some data:

1. a (non-graded) monad $T$;
2. an ordered set of grades $(\mathcal{G}, \leq)$, and unit grade $1$;
3. a subset $R_eX \subseteq TX$ for each $e \in \mathcal{G}$;
4. a multiplication $(\cdot) : \mathcal{G} \times \mathcal{G} \to \mathcal{G}$

such that $(\mathcal{G}, \leq, 1, \cdot)$ and $R$ form a grading of $T$
The canonical grading of a monad

For each monad $T$, there is\(^1\) an ordered monoid ($\text{Sub}(T), \subseteq, I, \otimes$), where

- $\text{Sub}(T)$ is the set of *subfunctors* $P$ of $T$, i.e. set-indexed families of subsets
  \[ PX \subseteq TX \]
  closed under $Tf = (\lambda t. t \gg (f \circ \text{return})) : TX \to TY$ for each $f : X \to Y$
- $P \subseteq P'$ iff $\forall X. PX \subseteq P'X$
- $IX = \{ \text{return} x \mid x \in X \}$
- $(P_1 \otimes P_2)X = \{ t \gg f \mid Y \in \text{Set}, t \in P_1 Y, f : Y \to P_2 X \}$

\(^1\)ignoring some size issues
Constructing (·)

A grading of $T$ is equivalently

- an ordered monoid $(\mathcal{G}, \leq, 1, \cdot)$
- together with a lax homomorphism of ordered monoids $R : \mathcal{G} \to \text{Sub}(T)$

\[
eq e' \Rightarrow Re \sqsubseteq Re' \quad I \sqsubseteq R1 \quad Re_1 \otimes Re_2 \sqsubseteq R(e_1 \cdot e_2)
\]

So if the following is associative and unital, we get a grading:

\[
e_1 \cdot e_2 = LRe_1 \otimes Re_2
\]

assuming $R$ has a left adjoint $L : \text{Sub}(T) \to \mathcal{G}$:

\[
\forall e \in \mathcal{G}. \ LP \leq e \iff P \sqsubseteq Re
\]
Constructing (·)

- For

\[ ReX = \{(xs, c) \in TX \mid (e = \bot \Rightarrow c = \text{cut}) \land (e = 1 \Rightarrow c = \text{cut} \lor xs \neq [])\} \]

we get

\[ LP = \begin{cases} \bot & \text{if } \exists X. ([], \text{nocut}) \in PX \\ 1 & \text{if } \exists X, xs. (xs, \text{nocut}) \in PX \\ \top & \text{otherwise} \end{cases} \]

\[ \bot \cdot e = \bot \quad \top \cdot e = \top \quad 1 \cdot e = e \]

- For

\[ ReX \cong \{ t \in TX \mid (\text{put} \notin e \Rightarrow (\text{fst} \circ t) \text{ is identity}) \land (\text{get} \notin e \Rightarrow (\text{fst} \circ t) \text{ is constant or identity} \land (\text{snd} \circ t) \text{ is constant})\} \]

we get

\[ LP = \{ \text{get, put} \} \setminus \{ \text{op} \mid \forall X. R\{\text{op}\}X \subseteq PX \} \]

\[ e_1 \cdot e_2 = L(Re_1 \otimes Re_2) = e_1 \cup e_2 \]
How to construct graded monads

Supply some data:

1. a (non-graded) monad $T$;
2. an ordered set of grades $(\mathcal{G}, \leq)$, and unit grade $1$;
3. a subset $ReX \subseteq TX$ for each $e \in \mathcal{G}$;

such that $R : \mathcal{G} \to \text{Sub}(T)$, and such that $(\mathcal{G}, \leq, 1, \cdot)$ and $R$ form a grading of $T$.
Constructing the subsets $ReX \subseteq TX$

Given a collection of operations $(op : n)$, each with

- an algebraic operation
  \[
  \llbracket op \rrbracket : (TX)^n \to TX
  \]

- a choice of grading function
  \[
  \theta_{op} : \mathcal{G}^n \to \mathcal{G}
  \]

we can define $R$ as the smallest family of subsets such that

- return restricts to a function $return : X \to R1X$
- $\llbracket op \rrbracket$ restricts to a function $\llbracket op \rrbracket : Rd_1X \times \cdots \times Rd_nX \to R(\theta_{op}(d_1, \ldots, d_n))X$
How to construct graded monads

Supply some data:

1. a (non-graded) monad T;
2. an ordered set of grades \((G, \leq)\), and unit grade 1;
3. a subset \(R_e X \subseteq TX\) for each \(e \in G\)
   (in many cases, generated by considering algebraic operations);

such that \(R : G \rightarrow \text{Sub}(T)\), and such that \((G, \leq, 1, \cdot)\) and \(R\) form a grading of \(T\)

An alternative: present a graded monad by graded operations and equations