Category-Graded Effect System and Algebraic Theory

Takahiro Sanada

RIMS, Kyoto University, `https://www.kurims.kyoto-u.ac.jp/~tsanada/`, tsanada@kurims.kyoto-u.ac.jp

Introduction

An important aspect of the theory of programming languages is the treatment of computational effects. Computational effects are the behavior of a program that breaks its feature as a mathematical function, such as input/output, nondeterministic choice, and state reading/writing. Monads and algebraic theories have been studied since 1990s as a notion to treat such computational effects in a unified manner. Recently, category-graded monads are introduced to unify graded monads and parameterised monads. We define category-graded algebraic theory that corresponds to the category-graded monad, and propose an effect system, *CatEff*, which use it and can safely treat states and input/outputs of varying types.

Examples of Computatinal Effect and Motivation

Consider the following program with a single state. We assume that the initial state has a value of type `Int`.

$$\begin{array}{l} \text{fun } p(a : \texttt{Int}) : \texttt{Int} \\ \quad \text{let}(x : \texttt{Int}) \text{ be } \underline{read}() \\ \quad \text{let}(y : \texttt{Unit}) \text{ be } \underline{write}(a) \\ \quad \text{return } x \end{array}$$

The program p reads an integer from the state and bind it to the variable x of type `Int`, then updates the state with the argument a, and returns the integer x. Next, we consider the following programs:

$$\begin{array}{l} \text{fun } q(a : \texttt{Int}) : \texttt{Int} \\ \quad \text{let}(x : \texttt{Int}) \text{ be } \underline{read}() \\ \quad \text{if } isEven(a) \\ \quad\quad \text{then let}(y : \texttt{Unit}) \text{ be } \underline{write}(true) \\ \quad\quad \text{else let}(y : \texttt{Unit}) \text{ be } \underline{write}(false) \\ \quad \text{return } x \end{array}$$

The program q is intended to write a boolean value to the state. Unfortunately, this program may cause problematic behavior. Consider calling the program q twice: $q(q(42))$. The first call completes execution successfully, In the second call, x is expected to be an integer, but actually a boolean value. There are several solutions to this problem.

1. Keep the programmer aware of the state of the program and make sure that type inconsistencies do not occur. This solution allows the type of the state to change.
2. Fix the type of state, and detect such a type inconsistency automatically by type checking. This solution frees programmers from using their brains. Programs such as q are no longer accepted.

We propose another solution to this problem by introducing novel notions, called category-graded algebraic theory and category-graded effect. This solution allows the type of the state to change and detects type inconsistencies automatically.

References

1. Bauer, A., Pretner, M.: An effect system for algebraic effects and handlers. *Log. Methods Comput. Sci.* **10**(4), 2014.
2. Orchard, D., Wadler, P., Eades III, H.: Unifying graded and parameterised monads. MSFP 2020. EPTCS, vol. 317. pp 18–38, 2020.
3. Street, R.: Two constructions on lax functors. *Cahiers de Topologie et Géométrie Différentielle Catégoriques.* **13**(3), 217–264, 1972.

Category

A category $\mathbb{C}$ is a directed graph satisfying the following conditions:

- For each vertex A of $\mathbb{C}$, there exists an edge $\mathrm{id}_A \colon A \to A$, called identity on A.

$$\mathrm{id}_A \circlearrowright A$$

- For each pair of edges $f \colon A \to B$, and $g \colon B \to C$, there exists an edge $g \circ f \colon A \to C$, called composition of f and g.

$$\begin{tikzcd} A \arrow[r, "f"] \arrow[rr, bend right, "g \circ f"'] & B \arrow[r, "g"] & C \end{tikzcd}$$

- The following equations hold for any vertices A, B and edge $f \colon A \to B$.

$$f \circ \mathrm{id}_A = f, \quad \mathrm{id}_B \circ f = f$$

$$\begin{tikzcd} A \arrow[r, "\mathrm{id}_A"] \arrow[dr, "f"'] & A \arrow[d, "f"] \\ & B \end{tikzcd} \qquad \begin{tikzcd} A \arrow[r, "f"] \arrow[dr, "f"'] & B \arrow[d, "\mathrm{id}_B"] \\ & B \end{tikzcd}$$

We write $\mathrm{ob}(\mathbb{C})$ for the set of vertex of $\mathbb{C}$, and $\mathbb{C}(A, B)$ for the set of edges from $A \in \mathrm{ob}(\mathbb{C})$ to $B \in \mathrm{ob}(\mathbb{C})$ of $\mathbb{C}$. We call a vertex of a category an object, and an edge a morphism.

An Example of Category-Graded Effect

Let $\mathbb{S}$ be the category "freely generated" by the graph: $\begin{tikzcd} \texttt{Int} \arrow[r, bend left, "ib"] & \texttt{Bool} \arrow[l, bend left, "bi"] \end{tikzcd}$. We consider the following read and write operations graded by the morphisms of $\mathbb{S}$.

$$\underline{read}_{\mathrm{id}_{\texttt{Int}}}, \quad \underline{read}_{\mathrm{id}_{\texttt{Bool}}}, \quad \underline{write}_{\mathrm{id}_{\texttt{Int}}}, \quad \underline{write}_{\mathrm{id}_{\texttt{Bool}}}, \quad \underline{write}_{ib}, \quad \underline{write}_{bi}.$$

Intuition of the graded morphism $f \colon A \to B$ of operations $\underline{op}_f$ is that A and B represent precondition and postcondition of the effect $\underline{op}_f$, respectively. For instance, $\underline{write}_{ib}$ can be used under the circumstance that the type of the state is `Int`, and write boolean value to the state. The programs corresponding to p and q in the left column can be written as follows:

$$\begin{array}{l} \text{fun } p'(a : \texttt{Int}) : \texttt{Int};\ \mathrm{id}_{\texttt{Int}} \circ \mathrm{id}_{\texttt{Int}} \\ \quad \text{let}(x : \texttt{Int}) \text{ be } \underline{read}_{\mathrm{id}_{\texttt{Int}}}() \\ \quad \text{let}(y : \texttt{Unit}) \text{ be } \underline{write}_{\mathrm{id}_{\texttt{Int}}}(a) \\ \quad \text{return } x \end{array}$$

$$\begin{array}{l} \text{fun } q'(a : \texttt{Int}) : \texttt{Int};\ ib \circ \mathrm{id}_{\texttt{Int}} \\ \quad \text{let}(x : \texttt{Int}) \text{ be } \underline{read}_{\mathrm{id}_{\texttt{Int}}}() \\ \quad \text{if } isEven(a) \\ \quad\quad \text{then let}(y : \texttt{Unit}) \text{ be } \underline{write}_{ib}(true) \\ \quad\quad \text{else let}(y : \texttt{Unit}) \text{ be } \underline{write}_{ib}(false) \\ \quad \text{return } x \end{array}$$

Program	Graded by
p'	$\mathrm{id}_{\texttt{Int}}$
q'	ib
$p' \circ p'$	$\mathrm{id}_{\texttt{Int}}$
$q' \circ p'$	ib
$p' \circ q'$	⃠
$q' \circ q'$	⃠

The left figure shows some programs and these grading morphisms. ⃠ means that the program is not allowed because the programs cause problematic behavior. Such a type inconsistency is detected by type checking, so we can prevent runtime errors. In summary, by grading effect and program, we can prevent runtime errors by type checking while maintaining flexibility of programs.

CatEff[Sanada, 2021]: A Category-Graded Effect System

We give the formal language for category-grade effect and its type system:

$$\textbf{Values: } V, W ::= x \mid true \mid false \mid n \mid \ldots$$

where x and n ranges over a set of variables and natural numbers, respectively.

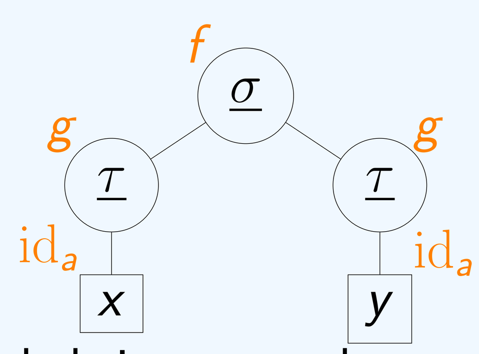
$$\textbf{Computations: } M, N ::= \mathrm{return}_a\, V \mid \text{let } x \text{ be } M \text{ in } N \mid \underline{op}(V) \mid \ldots$$

where a ranges over $\mathrm{ob}(\mathbb{S})$. We define the rule to derive a typing judgement $\Gamma \vdash_f M : \beta$ where $\Gamma = (x_1 : \alpha_1, \ldots, x_n : \alpha_n)$, and α_i and β are type.

$$\frac{\Gamma \vdash V : \alpha}{\Gamma \vdash_{\mathrm{id}_a} \mathrm{return}_a\, V : \alpha} \qquad \frac{\Gamma \vdash_f M : \alpha \quad \Gamma, x : \alpha \vdash_g N : \beta}{\Gamma \vdash_{g \circ f} \text{let } x \text{ be } M \text{ in } N : \beta} \qquad \frac{\Gamma \vdash V : \alpha}{\Gamma \vdash_{f_{op}} \underline{op}(V) : \beta} \qquad \ldots$$

Category-Graded Algebraic Theory

The theoretical background of category-graded effect is category-graded algebraic theory. The right figure is a tree representation of the term $\underline{\sigma}_f(\underline{\tau}_g(x), \underline{\tau}_g(y))$ of a category-graded algebraic theory. The formal definition of terms of category graded algebraic theory is as follows.



Definition[Sanada, 2021] Let X be a set, $\mathbb{S}$ be a category, and Σ be $\mathbb{S}$-graded signature, that is a set of operation symbols. For each operation symbol $\sigma \in \Sigma$, a morphism f_σ of $\mathbb{S}$ and a natural number n_σ are assigned. The set $Term_\Sigma(f, X)$ of f-graded Σ-term is defined inductively as follows:

$$\frac{a \in \mathrm{ob}(\mathbb{S}) \quad x \in X}{\mathrm{e}(a, x) \in Term_\Sigma(\mathrm{id}_a, X)} \qquad \frac{\sigma \in \Sigma \quad \{t_i \in Term_\Sigma(g, X)\}_{i=1}^{n_\sigma}}{\sigma(t_1, \ldots, t_{n_\sigma}) \in Term_\Sigma(g \circ f_\sigma, X)}$$