What is algebraic about algebraic effects and handlers?

Łukasz Magnuszewski

2023

Outline

Outline

Computation effects as algebraic operations

- Computation are free models
- Sequencing and generic operations

Handlers

What is coalgebraic about algebraic effects and handlers

- Tensoring comodels and models

Algebraic theories

Algebraic theories

Example
Group is a structure $(G, u, *, ^{-1})$ satisfying following equations:

$$(x * y) * z = x * (y * z)$$

$$(u * x) = x = x * u$$

$$x * x^{-1} = u = x^{-1} * x$$

We can define group using other definitions (using quantifiers), but we will stick to this type of definitions, known as algebraic or equational theories.

Signatures

Definition
Signature Σ is a collection of *operation symbols* with *arities* $\{(op_i, ar_i)\}_i$

The operation symbols op_i can by anything, but we usually think about them as syntactic entities

The arities ar_i are non-negative-integers

Operation symbols with 0 arity is constant, while operation symbol with 1, 2, 3 arities are unary, binary, ternary, respectively.

Example
Signature for group is

$$\{u, 0\}, \quad \{^{-1}, 1\}, \quad \{*, 2\}$$

Terms

Definition
Context is (possibly empty) list of distinct variable $x_1, \ldots, x_k$

Definition
Σ-terms in context $x_1, \ldots, x_k$ are build inductively from following rules

- variable x_i is as Σ-term in context $x_1, \ldots, x_k$
- if $t_1, \ldots, t_{ar_i}$ are Σ-terms in context $x_1, \ldots, x_k$ then $op_i(t_1, \ldots, t_{ar_i})$ is a Σ-term in context $x_1, \ldots, x_k$

Example
Using signature of group following terms

$$a * (b * c), \quad a^{-1} * a, \quad u, \quad u * c$$

are Σ-terms in context a, b, c

Σ-Equations

Definition
Σ-equation is a pair of Σ-terms l and r in a context $x_1, \ldots, x_k$. We write

$$x_1, \ldots, x_k | l = r$$

Example
Σ-equations for group:

1. $x, y, z | (x * y) * z = x * (y * z)$
2. $x | x * u = x$
3. $x | u * x = x$
4. $x | x * x^{-1} = u$
5. $x | x^{-1} * x = u$

Algebraic theory

Definition
Algebraic theory $T = (\Sigma_T, \mathcal{E}_T)$ is given by a signature Σ_T and a collection $\mathcal{E}_T$ of Σ_T-equations.

Example
Theory of groups:

$\Sigma_T = \{u, 0\}, \quad \{^{-1}, 1\}, \quad \{*, 2\}$

$\mathcal{E}_T$:

1. $x, y, z \quad | \quad (x * y) * z = x * (y * z)$
2. $x \quad | \quad x * u = x$
3. $x \quad | \quad u * x = x$
4. $x \quad | \quad x * x^{-1} = u$
5. $x \quad | \quad x^{-1} * x = u$

Simple examples of algebraic theories

Example
Theory of pointed set has a constant $\circ$ and no equations

Example
Empty theory has no operations symbols and no equations

Example
Theory of singleton has a constant $\circ$ and the equation $x = y$.

Interpretations and Models

Interpretations of signatures

Definition
An Interpretation I of signature Σ is given by

1. a set $|I|$, called the carrier
2. for each operation symbol op_i a map

$$[\![op_i]\!]_I : \underbrace{|I| \times \ldots \times |I|}_{ar_i} \to |I|$$

called an operation

Interpretations of Σ-terms

Definition
An Interpretation of Σ-term in context is interpreted by a map

$$[\![x_1, \ldots, x_k | t]\!]_I : |I|^k -> |I|$$

The variable x_i is interpreted as the i-th projection

$$[\![x_1, \ldots, x_k | x_i]\!]_I = \pi_i : |I|^k \to |I|$$

A compund term

$$x_1, \ldots, x_k \quad | \quad op_i(t_1, \ldots, t_{ar_i})$$

is interpreted as the composition of maps

$$|I|^k \xrightarrow{([\![t_1]\!]_I, \ldots [\![t_{ar_i}]\!]_I)} |I|^{ar_i} \xrightarrow{[\![op_i]\!]_I} |I|$$

Model of algebraic theories

Definition
Model $|M|$ of an algebraic theory T is an interpretation of the signature Σ_T which validates all the equations $\mathcal{E}_T$. That is for every equation

$$x_1, \ldots, x_k | l = r$$

in $\mathcal{E}_T$, the maps

$$[\![x_1, \ldots, x_k | l]\!]_M \quad \textit{and} \quad [\![x_1, \ldots, x_k | r]\!]_M$$

are equal.

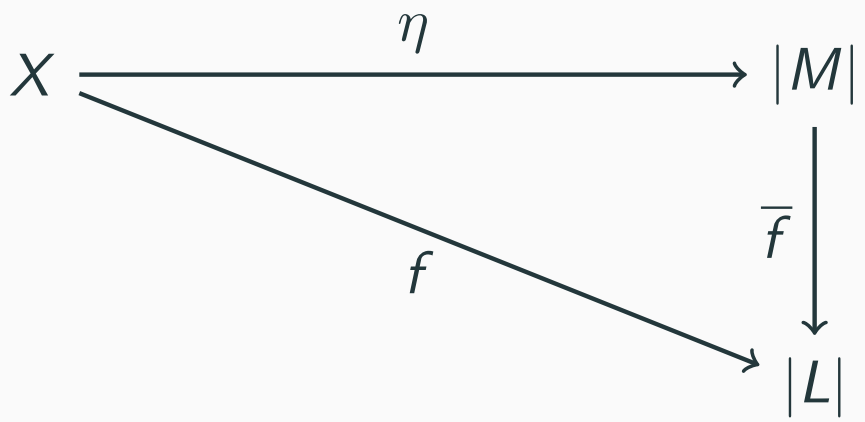
Homomorphisms of models

Definition
Suppose L and M are models of a theory T. A T-homomorphism from L to M is a map $\phi : |L| \to |M|$ which commutes with operations: for every operation symbol op_i of T, we have

$$\phi \circ [\![op_i]\!]_L = \circ [\![op_i]\!]_M \circ \underbrace{\phi, \ldots, \phi}_{ar_i}$$

Free Models

Definition
Given an algebraic theory T and a set X, the free T-model, also called the free T-algebra, generated by X is a model M together with a map $\eta : X \to |M|$ such that, for every T-model L and every map $f : X \to |L|$ there is a unique T-homomorphism $\overline{f} : M \to L$ for which the following diagram commutes:



We can understand this that free T-model generated by x is the "most economical" way of making a T-model out of the set X.

Every algebraic theory have free model

First we take set of all well founded trees with operation symbols from T_Σ and variables from X. Let that be $Tree_\Sigma(X)$ defined as follows

1. for each $x \in X$, there a tree $returnx \in Tree_\Sigma(X)$
2. for each op_i and trees $t_1, \ldots, t_{ar_i}$ theere is a tree denoted by $op_i(t_1, \ldots, t_{ar_i}) \in Tree_\Sigma(X)$, whose root is labeled by op_i and whose subtrees are $t_1, \ldots, t_{ar_i}$

The Σ-terms in context $x_1, \ldots, x_n$ are precisely the trees in $Tree_\Sigma(\{x_1, \ldots, x_n\})$ except that a variable x_i is labeled as return x_i when construed as a tree

Every algebraic theory have free model

Given a theory T, let $\approx_T$ be the least congruence relation such that every equation is satisfied. Define the carrier of the free mode $F_T(X)$ to be the quotient set

$$|F_T(X) = Tree_\Sigma(X)/\approx_T$$

the interpretation of symbols is them map $[\![op_i]\!]_{F_T(X)}$ defined by

$$[\![op_i]\!]_{F_T(X)}(|t_1|, \ldots, |t_{arr_i}|) = |op_i(t_1, \ldots, t_{arr_i})|$$

The map $\eta_X : X \to F_T(X)$ is defined by $\eta_X(x) = |return\, x|$.

It can be verified that such construction is free model.

General arities

One idea to make general arities would be to write

$$op_i(\ldots, t_\alpha, \ldots)_{\alpha \in A}$$

But we know that

$$\underbrace{X \times \ldots \times X}_{|A|} \cong X^A \to X$$

And we can use λ-calculus to write such function. To have A-many elements of a set X is to have a map $\kappa : A -> X$. And to apply the operation symbol op_i to a-many arguments κ we simply write $op_i(\kappa)$.

Operations with parameters

To motivate parameters, consider theory of vector space M.

We need to consider how to define multiplying vectors with scalars from $\mathbb{R}$ There are 2 possibilities:

1. Instead of having a single binary operation taking a scalar and a vector, we could have many unary operations taking a vector, one for each scalar.
2. We could view the scalar as an additional parameter of a unary operation on vec- tors

We shall adopt the second approach. We can generalize terms, equations, interpretations, models, homomorphisms to work with arbitrary arity and parameters.

Theory of vectors over $\mathbb{R}$

Lets define operations of this theory.

Scalar multiplication is unary operation mul parametrized by elements of $\mathbb{R}$. That is fo over $r \in \mathbb{R}$ and term t we may write the term

$$mul(r; t)$$

Other operation doesn't need parameters, buw we can force them to take unit parameter for example addition of vectors

$$add((); t_1, t_2)$$

Computation effects as algebraic operations

Computation effects as algebraic operations

Its start to give some examples from programming. We can define many computations effects by algebraic theories.

We can divide computations into two kinds (we are omitting non-terminating)

1. pure, which terminates and return value v, then we write

$$return\ v$$

2. effectful. Then we model this with operation which take parameter p, for instance the memory location to be read, or the string to be printed, and a continuation κ, which is a suspended computation expecting the result of the operation, for instance the contents of the memory location that has been read.

$$op(p, \kappa)$$

Algebraic theory of state

The algebraic theory of state with locations L and states S has operations

$$\mathit{lookup} : L \rightsquigarrow S \quad \mathit{and} \quad \mathit{update} : L \times S \rightsquigarrow 1$$

Take look at example computation l++

$$\mathit{lookup}(l, \lambda x.\mathit{update}((l, x+1), \lambda\_.\mathit{return} x))$$

Equations for successive lookups and updates

First we have equations which state what happens on successive lookups and updates to the same memory location. For all $l \in L$, $s \in S$ and all continuations κ

$$lookup(l, \lambda s.lookup(l, \lambda t.\kappa st)) = lookup(l, \lambda s.\kappa ss)$$

$$lookup(l, \lambda s.update((l, s), \kappa)) = \kappa()$$

$$update((l, s), \lambda\_.lookup(l, \kappa)) = update((l, s), \lambda\_.\kappa s)$$

$$update((l, s), \lambda\_.update((l, t), \kappa)) = update((l, t), \kappa)$$

Lookups and updates from diffrent locations

There is a second set of equations stating that lookups and updates from different locations $l \neq l'$ distribute over each other:

$$\mathit{lookup}(l, \lambda s.\mathit{lookup}(l', \lambda s'.k\, s\, s')) = \mathit{lookup}(l', \lambda s'.\mathit{lookup}(l, \lambda s.k\, s\, s'))$$

$$\mathit{update}((l, s), \lambda\_.\mathit{lookup}(l', \kappa)) = \mathit{lookup}(l', \lambda t.\mathit{update}((l, s), \lambda\_.\kappa\, t))$$

$$\mathit{update}((l, s), \lambda\_.\mathit{update}((l', s'), \kappa)) = \mathit{update}((l', s'), \lambda\_.\mathit{update}((l, s), \kappa))$$

Combining algebraic theories

Algebraic theories may be combined. For example, we want to combine state and I/O. Sometimes we want to combine theories so that the operations between them interact. To demonstrate this, let us consider the theory of a single stateful memory location holding elements of a set S. The operations are

$$get : 1 \rightsquigarrow S \quad put : S \rightsquigarrow 1$$

Single State theory equations

The equations are:

$$get((), \lambda s.get((), \lambda t.\kappa\, s\, t)) = get((), \lambda s.\kappa\, s\, s)$$

$$get((), \lambda s.put(s, \kappa)) = \kappa()$$

$$put(s, \lambda\_.get((), \kappa)) = put(s, \lambda\_.\kappa s)$$

$$put(s, \lambda\_.put(t, \kappa)) = update(t, \kappa)$$

This is just the first group of equations from previous example, except that we need not specify which memory location to read from.

Combining multiply states

That is, to model I-many states, we combine I-many copies of the theory of a single state, so that for every $l \in \mathsf{I}$ we have operations

$$get_l : 1 \rightsquigarrow S_l \quad \textit{and} \quad get_l : S_l \rightsquigarrow 1$$

We also need to define how diffrent operations combine,

$$get_l((), \lambda s.get_{l'}((), \lambda s'.k\, s\, s')) = get_{l'}((), \lambda s'.get_{;}((), \lambda s.k\, s\, s'))$$

$$put_l(s, \lambda\_.get_{l'}((), \kappa)) = lookup_{l'}((), \lambda t.put_l(s, \lambda\_.\kappa\, t))$$

$$put_l(s, \lambda\_.put_{l'}((), s'), \kappa) = put_{l'}(s', \lambda t.put_l((), \lambda\_., \kappa))$$

This theory is almost the same as theory of multiply locations with two differences, we can have different states in each location. And instead giving parameter to operation, we index our operations.

Computation are free models

Consider the theory State from previous slides. Let us see that free model $F_{State}(V)$ adequately describes stateful computations returning values from V. Every tree from $Tree_{\Sigma_{State}}(V)$ is congruent to one of the form

$$get((), \lambda s.put(f(s), \lambda\_.returng(s)))$$

for some maps $f : S \to S$ and $g : S \to V$. Indeed, by applying the equations, we may contract any two consecutive gets to a single one, and similarly for consecutive puts, we may disregard a get after a put, and cancel a get followed by a put. And give constant continuation.

Therefore the free model $F_{State}(V)$ is isomorphic to the set of functions $S \to S \times V$.

Generic operations

Consider some syntactic sugar for our notation, the same which was presented three weeks ago. Consider an algebraic theory T. For an operation $op : P \rightsquigarrow A$ in Σ_T, define the corresponding generic operation

$$\overline{op}(p) := op(p, \lambda x.return\, x)$$

In words, the generic version performs the operation and returns its result. When the parameter is the unit we write $op()$ instead of the silly looking $op(())$.

Sequencing operations

Suppose that $t \in F_T(X)$ and $h : X \to F_T(X)$. We define syntactic sugar for sequencing. first define helper function.

$$\phi^\dagger([\mathit{return}\, x]) = \phi(x),$$

$$\phi^\dagger([\mathit{op}(p, \kappa)]) = [\mathit{op}(p, \phi^\dagger \circ \kappa)]$$

The we can define do notation

$$\mathit{do}\, x \leftarrow\, t\, \mathit{in}\, h(x) \;:=\; h^\dagger(t)$$

Handlers

Handlers

So far the main take-away is that computations returning values from V and performing operations of a theory T are the elements of the free model $F_T(V)$. What about transformations between computations, what are they?

We can define them as T-homomorphism between models

$$H : F_T(V) \to |F_{T'}(V')|$$

Handlers

So we need to give 3 things

1. a map $f : V \to |F_{T'}(V')|$
2. for every $op_i : P_i \rightsquigarrow A_i \in \Sigma_T$, a map

$$h_i : P_i \times |F_{T'}(V')|^{A_i} \to |F_{T'}(V')|$$

   such that
3. the maps h_i form a T-model on $|F_{T'}(V')|$ meaning the validate the equation of T.

The map $H : F_T(V) \to |F_{T'}(V')|$ induced by that is the unique one satisfying

$$H([return x]) = f(x) \quad \textit{and} \quad H([op(p, \kappa)]) = h_i(p, H \circ \kappa)$$

Handlers notation

We need better notation for handlers. Let us write

$$\mathit{handler}\{\mathit{return}\, x \mapsto f(x),\ (\mathit{op}_i(y;\kappa))_{\mathit{op}_i \in \Sigma_i}\}$$

For handler induced by f and h_i, and

$$(\mathit{with}\, H\, \mathit{handle}\, C)$$

for the application of H to computation C. The defining equations for handlers written in the new notation are:

$$(\mathit{with}\, H\, \mathit{handle}\, \mathit{return}\, v\,) = f(v)$$

$$(\mathit{with}\, H\, \mathit{handle}\, \mathit{do}\, x \leftarrow \overline{\mathit{op}_i}(p)\, \mathit{in}\, \kappa(x)) = h_i(p, \lambda x.\, \mathit{with}\, H\, \mathit{handle}\, \kappa(x))$$

What is coalgebraic about algebraic effects and handlers

Models of algebraic theories in a category

An interpretation I of theory T in category **C**(cartesian closed) is given by

1. an object $|I|$ in **C**, called the carrier,
2. for each operation symbol op_i, a morphism in **C**

$$[\![op_i]\!]_I : |I|^{ar_i} \to |I|$$

An interpretation is extended to Σ-terms in contexts as follows: x_i is interpreted as i-th projection

$$[\![x_1, \ldots x_k | x_i]\!]_I = \pi_i$$

Compound term $x_1, \ldots x_k | op_i(t_1, \ldots, t_{arr_i})$ is interpreted as the composition of morphisms:

$$|I|^k \xrightarrow{([\![t_1]\!]_I),\ldots,([\![t_{arr_i}]\!]_I)} |I|^{arr_i} \xrightarrow{[\![op_i]\!]_I} |I|$$

Comodels of theory

We will model top-level computation effects with comodels. A comodel of theory T in a category $\mathbf{C}$ is a model in the opposite category $\mathbf{C}^{op}$.

Recall that the interpretation of an operation $op_i : P \rightsquigarrow A$ in a model M is a map $[\![op]\!]_M : P \times |M|^A \to |M|$ which in the curried form is $|M|^A \to |M|^P$. In the opposite category the maps turns its direction, and the exponentials become products:

$$A \times |M| \leftarrow P \times |M|$$

Thus, in a comodel W an operation $op : P \rightsquigarrow A$ is interpreted as map

$$[\![op]\!]^W : P \times |W| \to A \times |W|$$

which we call a cooperation

Examples of cooperations

Example
Non-deterministic choice $\mathit{choose} : 1 \rightsquigarrow \mathit{bool}$ is interpreted as a cooperation:

$$1 \times |W| \to \mathit{bool} \times |W|$$

If we think of $|W|$ as the set of all possible worlds, the cooperation choose is the action by which the world produces a boolean value and the next state of the world.

Example
Exception $\mathit{abort} : 1 \rightsquigarrow$ is interpreted as a cooperation:

$$1 \times |W| \to |W| \times$$

Unless $|W|$ is the empty set, there is no such map. An exception cannot propagate to the outer world. The universe is safe from segmentation fault

Examples of cooperations

Example
Printing *print* : $S \rightsquigarrow 1$ is interpreted as a cooperation:

$$S \times |W| \to |W|$$

It is the action by which the world is modified according to the printed message

Example
Reading *read* : $1 \rightsquigarrow S$ is interpreted as a cooperation:

$$1 \times |W| \to |W| \times S$$

This is quite similar to non-deterministic choice, except that the world provides an element of S rather than a boolean value.

Cointerpretations

A Cointerpretation of a signature Σ_T is given by a carrier set $|I|$, and for each operation symbol $op : P \rightsquigarrow A$ a map

$$[\![op]\!]^I : P \times |I| \to a \times |I|$$

The cointerpretation I may be extended to well-founded trees as follows

1. the tree return x is insterpreted as the x-th injection

$$[\![X \mid x]\!]^I : \omega \to (x, \omega) : |X| \times |I|$$

2. the tree $op_i(p, \kappa)$ is interpreted as

$$[\![X \mid op_i(p, \kappa)]\!]^I : |I| \to X \times |I|$$

$$[\![X \mid op_i(p, \kappa)]\!]^I : \omega \to [\![X|\kappa(a)]\!]^I(\overline{\omega}) \quad \textit{where}\, (a, \overline{\omega}) = [\![op_i]\!]^I(p, \omega)$$

Comodel

A comodel W of a theory T is a Σ_T-cointerpretation which validates all the equations $\mathcal{E}_T$. As before, an equation is valid when the interpretations of its left- and right-hand sides yield equal maps

Comodel of state

The operations of State are following $get : 1 \rightsquigarrow S$ and $put : S \rightsquigarrow 1$. so respective cooperations are $g : |W| \to S \times |W|$ and $p : S \times |W| \to |W|$. And cooperation must satisfy equations of State. Right and left side of equation $get((), \lambda s.put(s, \kappa)) = \kappa()$ are insterpreted as maps $|W| \to |W|$, namely

$$[\![\kappa()]\!]^W : w \mapsto w$$

$$[\![get((, \lambda s).put(s, \kappa))]\!]^W : w \mapsto p(g(w))$$

There are equal, precisely when, for all $w \in |W|, \quad p(g(w)) = w$.

Equations of comodel of state

We can treat similarly other equations which gives us

$$p(g(w)) = w$$

$$g(\pi_i(g(w))) = g(w)$$

$$g(p(s, w)) = (s, p(s, w))$$

$$p(t, p(s, w)) = p(t, w)$$

Tensoring comodels and models

If we consider the elments of T-model $|M|$ as effectful computations and the element of T-comodel as external environments it is natural to ask if the combined together can be interpreted as running programs.

Let $\sim_T$ be the least equivalence relation on $|M| \times |W|$ such that, for every symbol $op : P \rightsquigarrow A$ in Σ_T, and for all $p \in P, a \in A, \kappa : A \to |M|.$ and $w, w' \in |W|$ such that

$$[\![op]\!]^W(p, w) = (a, w')$$

then

$$([\![op]\!]_M(p, \kappa), w) \sim_T (\kappa(a), w')$$

Define the tensor $M \otimes W$ to be the quotient set $(|M| \times |W|)/_{\sim_T}$

Tensor of state

Let us compute the tensor of $M = F_{State}(X)$, the free model of the theory of state generated by X, and the comodel W defined by

$$|W| := S, \quad [\![get]\!]^W := \lambda s.(s, s), \quad [\![put]\!]^W := \lambda s.(s, t),$$

We may read the equivalences as rewrite rules, of executing programs, until we are left with $([\![return\, x\,]\!]_M, s)$

$$([\![get((), \kappa)]\!]_M, s) \sim_{State} ([\![\kappa]\!]_M(s), s)$$

$$([\![put(t, \kappa)]\!]_M, s) \sim_{State} ([\![\kappa]\!]_M(), t)$$

Because $([\![return\, x\,]\!]_M, s) \sim_{State} ([\![return\, y\,]\!]_M, t)$ only if $x = y$ *and* $s = t$ (proof ommited). It follows that $M \otimes W \cong X \times S$. In other words, the execution of a program in an initial state always leads to a return value paired with the final state

Homework

Homework

1. Describe some computation effect with algebraic theory
2. Describe free model of the theory T of an associative operation with a unit. It has constant ϵ and binary operation $*$ satisfying equations:

$$(x * y) * z = x * (y * z)$$

$$\epsilon * x = x$$

$$x * \epsilon = x$$

Give a useful description of the free model of T generated by a set X.