Local states in string diagrams

Paul-André Melliès *

Laboratoire Preuves, Programmes, Systèmes
CNRS, Université Paris Diderot

Abstract. We establish that the local state monad introduced by Plotkin and Power is a monad with graded arities in the category $[Inj, Set]$. From this, we deduce that the local state monad is associated to a graded Lawvere theory מ which is presented by generators and relations, depicted in the graphical language of string diagrams.

1 Introduction

In this paper, we elaborate an algebraic and graphical account of the local state monad on the category $[Inj, Set]$ of covariant presheaves on the category Inj

$$L \quad : \quad [Inj, Set] \quad \longrightarrow \quad [Inj, Set]$$

formulated ten years ago by Plotkin and Power [12] themselves inspired by seminal ideas developed by O'Hearn and Tennent [11] on the presheaf semantics of local states. Much work has been dedicated in the past decade in order to understand the algebraic nature of this specific local state monad, in particular by Power [14, 15] and by Staton [17, 18]. One main purpose of the present paper is to recast these two lines of work in the language of *monads with arities*. An immediate benefit of the reformulation is that every monad with arities comes together with a notion of *Lawvere theory with arities* formulated in [9]. By proving that the local state monad is a monad with graded arities, we are thus able to define its graded Lawvere theory מ (a letter pronounced *mem* in hebrew). The whole point of the paper is that the category מ is sufficiently simple to be presented by generators and relations easily adapted from [12, 15, 18]. The shift from finitely presentable arities to graded arities is fundamental to that purpose.

Recall that the category Inj has natural numbers $n, p, q \in \mathbb{N}$ as objects, and injective functions $f : [p] \to [q]$ as morphisms, where $[n]$ denotes the finite set $[n] = \{1, \ldots, n\}$ of cardinal n. As any presheaf category, the category $[Inj, Set]$ is cartesian closed and thus provides a model of the simply-typed λ-calculus, where every simple type is interpreted as a presheaf A. Moreover, the local state monad L defines a computational monad on the category $[Inj, Set]$ in the sense of Moggi [10]. For that reason, the category $[Inj, Set]$ together with the monad L defines an interpretation of an imperative call-by-value λ-calculus where registers may be alternatively written, read, allocated and collected, see [12] for details.

* This work has been partly supported by the ANR RECRE.

The elements M in A_n are called elements of degree n in the presheaf A. The idea underlying the model is that a program M with n registers in the language should be interpreted as an element of degree n in the presheaf A associated to the type of the program.

Now, let us recall how the monad L is defined. By convention, we suppose that every register of our language may be assigned the same finite set V of values. In that situation, the covariant presheaf LA obtained by applying the monad L on a covariant presheaf A is conveniently expressed by the formula

$$LA \quad : \quad n \ \mapsto \ \ S^n \ \Rightarrow \ \left(\ \int^{p \in Inj} \ S^p \times A_p \times Inj(n,p) \ \right) \tag{1}$$

where the contravariant presheaf

$$S \quad : \quad n \quad \mapsto \quad V^n \quad : \quad Inj^{\,op} \quad \longrightarrow \quad Set$$

transports every number n to the set S^n of states possibly taken by n registers:

$$S^n \quad := \quad V^n \quad = \quad \{ \ (val_1, \ldots, val_n) \ \mid \ \forall i \in \{1, ..., n\}, \, val_i \in V \ \}.$$

Although the formula (1) may appear slightly intimidating, the intuition underlying it is easy to grasp. It simply reflects the idea that a program M of type LA with n registers behaves in the following way: first, the program M reads the state $s_{in} \in S^n$ of its n registers, then, depending on the value $s_{in} = (val_1, \ldots, val_n)$ which has been just read, the program M allocates a number $p - n$ of registers (with $p \geq n$) and returns three pieces of information to the context:

1. a state $s_{out} = (wal_1, \ldots, wal_p) \in S^p$ of the p registers,
2. a return value $M(s_{in}) \in A_p$ depending on the p registers,
3. and finally, an injective function $f : [n] \to [p]$ which tracks the n registers originally appearing in the program M among the p registers of the returned program $M(s_{in})$.

A nice aspect of the formula (1) is that it takes care of the fact that the $p - n$ registers may be allocated with different names in the memory. This is indeed the purpose of the colimit (or more precisely the coend) formula

$$\int^{p \in Inj} \ S^p \times A_p \times Inj(n,p) \quad = \quad \Big(\coprod_{p \in \mathbb{N}} \ S^p \times A_p \times Inj(n,p) \Big) \, / \, \sim \tag{2}$$

which is defined as the set of triples $(val_1, \ldots, val_p, M, f)$ in $S^p \times A_p \times Inj(n,p)$ modulo the least equivalence relation $\sim$ identifying all triples

$$(val_{h(1)}, \cdots, val_{h(p)}, M, f : [n] \to [p]) \ \sim \ (val_1, \cdots, val_q, A_h(M), h \circ f : [n] \to [q])$$

for an injection $h : [p] \to [q]$. Here, the element $A_h(M) \in A_q$ denotes the image (or pushforward) of the element $M \in A_p$ of degree p along the injection $h : [p] \to [q]$, which is typically obtained in the case of a program M by h-reindexing the names of its p registers.

2 The global state monad

In their work, Plotkin and Power [12] made the important observation that the local state monad L may be alternatively presented by a series of well-chosen generators and relations. One ambition of this paper is to illustrate the additional principle advocated by the author in [9] that any concise formulation such as (1) of a monad L presented by generators and relations can be derived from the existence of a class of *canonical forms* for the terms of the associated algebraic theory. This general principle was succesfully applied in [9] on the global state monad

$$T \quad : \quad A \quad \mapsto \quad S \Rightarrow (S \times A) \quad : \quad Set \quad \longrightarrow \quad Set \tag{3}$$

induced by a finite set S of states on the category Set. For simplicity, we will suppose from now on that all the registers manipulated by the language are boolean, and thus that the set of values is equal to $V = \{true, false\}$. We will also suppose for the sake of the discussion that $S = V = \{true, false\}$. This leads to the following definition. A *mnemoid* in a cartesian category $\mathscr{C}$ is defined as an object A equipped with a binary operation `lookup` and a unary operation $\texttt{update}_{\langle val \rangle}$ for each value $val \in \{true, false\}$ of the register:

$$\texttt{lookup} : A \times A \longrightarrow A \qquad\qquad \texttt{update}_{\langle val \rangle} : A \longrightarrow A$$

moreover satisfying three families of equations.

1. Creation lookup – update. Reading the value val of the register and then writing the very same value val in the register is like doing nothing at all. This leads to the equation below:

$$\texttt{lookup}(\texttt{update}_{\langle true \rangle}(term), \texttt{update}_{\langle false \rangle}(term)) \; = \; term$$

2. Interaction update – update. Storing a value val_1 and then a value val_2 inside the register is just like storing directly the value val_2. In particular, the value val_1 is lost in the process.

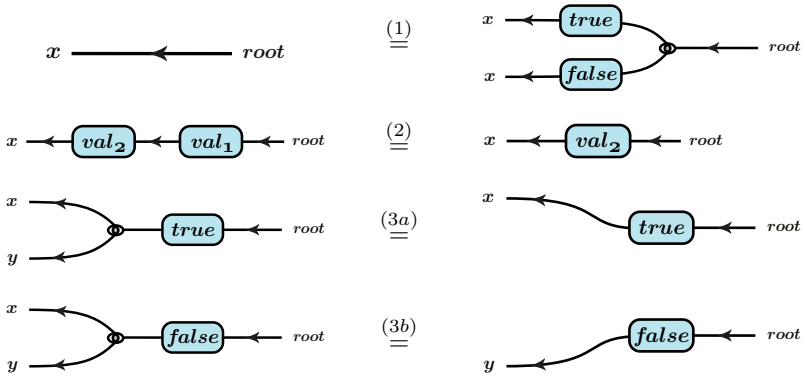
$$\texttt{update}_{\langle val_1 \rangle} \; \circ \; \texttt{update}_{\langle val_2 \rangle} = \texttt{update}_{\langle val_2 \rangle}$$

3. Interaction update – lookup. When one stores a value val in the register and then reads the value of the register, one gets back the value val.

$$\texttt{update}_{\langle val \rangle} \circ \texttt{lookup} \left[\, term(true) \, , \, term(false) \, \right] \; = \; \texttt{update}_{\langle val \rangle}(\, term(val) \,).$$

The two operations `lookup` and $\texttt{update}_{\langle val \rangle}$ of a mnemoid may be conveniently depicted in the language of string diagrams in the following way:

Here, the `lookup` operation is depicted as an "eye" which inspects the value of the register at the root and then branches on the left when the value is $true$ and on the right when the value is $false$. The $\texttt{update}_{\langle val \rangle}$ operation is depicted as a "cartouche" which erases the value of the register at the root and writes its own value $val \in \{true, false\}$ on the branch. The arrows on the wires indicate the direction of execution, which goes from the root to the leaves of the tree of operations. The three equations 1, 2 and $3(a, b)$ required of a mnemoid are depicted as follows in the language of string diagrams:



The main theorem established by Plotkin and Power [12] for the global state monad may be formulated as follows for the cartesian category $\mathscr{C} = Set$:

Theorem 1. *The category of mnemoids is equivalent to the category of algebras of the global state monad for $S = \{true, false\}$.*

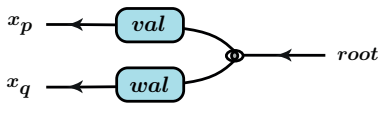
The theorem is obtained in the original paper by Plotkin and Power [12] by applying the Beck theorem in order to establish the monadicity of an adjunction of interest. We advocate in [9] that a more conceptual way to obtain the same result is to deduce it from two separate facts. First of all, the global state monad is finitary. Then, the following *canonical form* theorem for mnemoids:

Theorem 2. *Every term of the theory of mnemoids with n variables $x_1, \ldots, x_p$ is equivalent to a term of the form*

$$\texttt{lookup} \quad \left[\quad \texttt{update}_{\langle val \rangle}\,(x_p)\,,\, \texttt{update}_{\langle wal \rangle}\,(x_q) \quad \right]$$

Moreover, this canonical form is unique for a given term.

Expressed graphically, this means that every such term with n variables of the theory of mnemoids is equivalent to a unique term of the form:



with $val, wal \in \{true, false\}$ and $p, q \in [n]$. In other words, every sequence of operations performed in the theory of mnemoids on the finite set $[n] = \{1, \ldots, n\}$

of variables is equal to a lookup operation followed on each branch $true$ and $false$ by an update operation and a choice of an element in $[n]$. We explain in [9] how to apply the philosophy of Lawvere in order to deduce from this canonical form theorem that the free mnemoid generated by a given set A coincides in fact with the result TA of applying the global state monad T to the set A. More conceptually, one recovers in this way the result by Plotkin and Power (Thm 1) that the category of mnemoids is equivalent to the category of algebras of the global state monad T. The existence of the canonical form was independently observed by Pretnar [16] who uses it in order to establish that the theory is Hilbert-Post complete.

3 The five operations of the local state monad

One main purpose of the present article is to establish a similar result for the local state monad L. In particular, we would like to derive the formula (1) for the monad L from the existence of a canonical form for a particular algebraic presentation of its operations. To this effect, we start from a mild adaptation of the algebraic presentation given by Plotkin and Power in their seminal paper [12]. The resulting algebraic presentation is based on the distinction between four families of operations. First of all, for each natural number $n \in \mathbb{N}$ and each location $loc \in [n]$, the operations of a mnemoid:

$$\texttt{lookup}_{\langle loc \rangle} : A_n \times A_n \longrightarrow A_n \qquad \texttt{update}_{\langle loc,val \rangle} : A_n \longrightarrow A_n \tag{4}$$

where $val \in \{true, false\}$. Note that these two operations do not alter the degree n of the elements. Then, for each natural number $n \in \mathbb{N}$, for each location $loc \in [n+1]$ and for each value $val \in \{true, false\}$, an operation

$$\texttt{fresh}_{\langle loc,val \rangle} \quad : \quad A_{n+1} \quad \longrightarrow \quad A_n \tag{5}$$

whose intuitive purpose is to allocate among n registers a fresh register at location $loc \in [n+1]$ moreover initialized with the value $val \in \{true, false\}$. Then, for each natural number $n \in \mathbb{N}$ and for each location $loc \in [n+1]$, an operation

$$\texttt{collect}_{\langle loc \rangle} \quad : \quad A_n \quad \longrightarrow \quad A_{n+1} \tag{6}$$

whose intuitive purpose is to deallocate or garbage collect the register at location $loc \in [n+1]$. Finally, for each natural number $n \in \mathbb{N}$ and for each pair of locations $loc, loc+1 \in [n]$, an operation

$$\texttt{permute}_{\langle loc,loc+1 \rangle} \quad : \quad A_n \quad \longrightarrow \quad A_n \tag{7}$$

whose intuitive purpose is to permute the two registers at location loc and $loc+1$. One main conceptual difficulty of the local state monad, and also one main reason for studying it so closely, is that it entangles in a sophisticated way the read/write operations (4) of the mnemoid to the structural operations (5–6–7) whose function is to reorganize the shape of the memory by allocating, collecting or permuting registers. There seems to be here a general principle of interaction between effects and resources, which one would like to better understand.

4 A notion of graded arity in $[\boldsymbol{Inj}, \boldsymbol{Set}]$

An apparent obstruction to the idea of canonical form in the case of the local state monad L is the fact that its definition relies on the coend formula (2). According to a naive but reasonable understanding of Lawvere's principles, there should be *exactly* one canonical form for each element of the set $(TA)_n$ and such a one-to-one correspondence seems difficult to achieve for a general covariant presheaf A because of the presence of the non-trivial equivalence relation $\sim$ in the formula (2). In order to tackle the situation, we thus need to *specialize* the formula (1) to a specific class of covariant presheaves, provided in this cas by the finite sums

$$[\, p_0 \,,\, \ldots \,,\, p_m \,] \quad = \quad \underbrace{\langle 0 \rangle + \cdots + \langle 0 \rangle}_{p_0 \text{ times}} \;+\; \cdots \;+\; \underbrace{\langle m \rangle + \cdots + \langle m \rangle}_{p_m \text{ times}}$$

of *representable* covariant presheaves

$$\langle\, k \,\rangle \;:=\; \mathbf{y}_k \quad : \quad n \;\mapsto\; Inj(k,n) \quad : \quad Inj \;\longrightarrow\; Set.$$

Note in particular that it follows from the Yoneda lemma that

$$[Inj, Set]\,(\,[p_0,\,\ldots\,,p_m]\,,\;A\,) \quad = \quad \prod_{k=0}^{m} \underbrace{A_k \times \ldots \times A_k}_{p_k \text{ times}} \quad = \quad \prod_{k=0}^{m} A_k^{p_k}$$

for every covariant presheaf A over the category Inj. This equation justifies thinking of the presheaves $[\,p_0,\ldots,p_m\,]$ as an appropriate notion of generalized arity in the category of presheaves $[Inj, Set]$ which we call *graded arity*. Indeed, in the same way a function from $[n] = \{1,\ldots,n\}$ to a set A defines a word of length n in the alphabet A, a morphism from $[\,p_0,\ldots,p_m\,]$ to a covariant presheaf A defines a word of length $n = \sum p_k$ in the *graded* alphabet A, consisting of m words of length p_k in the alphabet A_k of elements of grade $k \in [m]$ in the covariant presheaf A. Note that the full subcategory of such arities in the category $[Inj, Set]$ is isomorphic to the free category with finite sums $\Sigma\, Inj^{op}$ generated by the category Inj^{op}. Moreover, the resulting full and faithful functor

$$i \quad : \quad \Sigma\, Inj^{op} \quad \longrightarrow \quad [Inj, Set]$$

is dense in the category $[Inj, Set]$. As such, the category ΣInj^{op} together with the functor i defines a notion of arities on the presheaf category $[Inj, Set]$ in the sense of Weber [20] who developed the notion in his study of globular operads, see also [9].

5 From a coend formula to a coproduct formula

It appears that when applied to a graded arity $A = [\,p_0,\ldots,p_m]$, the coend formula (2) suddenly becomes a coproduct formula. Let us briefly explain why.

The phenomenon is in fact slightly more general. Consider the category Nat with natural numbers $n \in \mathbb{N}$ as objects and no morphisms except for the identities. The object-preserving functor $\ell : Nat \longrightarrow Inj$ induces an adjunction

$$[Nat, Set] \underset{\ell^*}{\overset{\exists_\ell}{\underset{\longleftarrow}{\overset{\longrightarrow}{\bot}}}} [Inj, Set] \tag{8}$$

where the forgetful functor ℓ^* is defined by precomposition with ℓ and its left adjoint functor $\exists_\ell$ is defined by left Kan extension along ℓ. The computation of this left Kan extension is easy. Namely, given a presheaf A on the category Nat, one obtains:

$$\exists_\ell A \;:\; n \;\mapsto\; \coprod_{m\in\mathbb{N}} A_m \times Inj(m,n) \;=\; \{\, (M,f) \,|\, M \in A_m,\; f \in Inj(m,n) \,\}$$

Observe that the covariant presheaves of the form $\exists_\ell A$ are precisely the (possibly infinite) sums of representable presheaves over Inj. In particular, every graded arity $[\, p_0, \ldots, p_m \,]$ is of that form. Now, a simple computation shows that the coend formula (2) applied to such a covariant presheaf $\exists_\ell A$ yields a coproduct formula:

$$\int^{p\in Inj} S^p \times \Big[\coprod_{m\in\mathbb{N}} A_m \times Inj(m,p) \Big] \times Inj(n,p) \quad\cong\quad \coprod_{m\in\mathbb{N}} A_m \;\times\; \langle\, m,n \,|\, S \,\rangle$$

where the set $\langle\, m,n \,|\, S \,\rangle$ is defined as

$$\langle\, m,n \,|\, S \,\rangle \quad:=\quad \int^{p\in Inj} Inj(m,p) \;\times Inj(n,p) \times S^p.$$

This slightly enigmatic result convinces us to study more closely the monad

$$L_{Nat} \quad:=\quad \ell^* \circ L \circ \exists_\ell \quad:\quad [Nat, Set] \quad\longrightarrow\quad [Nat, Set]$$

obtained by pre and post-composing the monad L with the two components of the adjunction $\exists_\ell \dashv \ell^*$. The image of a presheaf A on Nat (also called a graded set) is thus defined as

$$L_{Nat}\, A \;:\; n \;\mapsto\; S^n \;\Rightarrow \coprod_{m\in\mathbb{N}} A_m \;\times\; \langle\, m,n \,|\, S \,\rangle. \tag{9}$$

From now on, and in order to differentiate the two local state monads, we write L_{Inj} for the local state monad L on the category $[Inj, Set]$.

6 The category *Res* of resource management

The monad L_{Nat} is so simple that it deserves further analysis. In particular, remember from §3 that we are interested in clarifying the intricate interplay

between read/write effects (lookup,update) and resource management (fresh, collect, permute) in the local state monad. This idea has been already explored quite far by Power [15] in his work on indexed Lawvere theories. Here, it provides us with a precious guide in our analysis. Indeed, it is folklore that the category Inj is presented (in some sense which will be later elaborated) by the `collect` and `permute` operations. This preliminary observation leads us to introduce a category Res already considered by Staton [18] whose intuitive purpose is to reflect all the resource management operations (not just `collect` and `permute` but also `fresh`) of the local state monad. By definition, the category Res has natural numbers $n \in \mathbb{N}$ as objects, and resource morphisms $[m] \to [n]$ as morphisms $m \to n$, where a resource morphism $f : [m] \to [n]$ is defined as a function $f : [m] \to [n] + \{true, false\}$ satisfying the following injectivity property: every element $k \in [n]$ has *at most* one antecedent in $[m]$. The resource morphism $g \circ f : [m] \to [n]$ obtained by composing two resource morphisms $f : [m] \to [p]$ and $g : [p] \to [n]$ is defined just as expected. Note in particular that the category Res is a subcategory of the Kleisli category induced by the exception monad $A \mapsto A + \{true, false\}$ on the category Set with natural numbers $m, n \in \mathbb{N}$ as objects and functions $[m] \to [n]$ as morphisms. Accordingly, the reader should note that there exists an object-preserving functor $\iota : Inj \to Res$ which transports every injection $f : [m] \to [n]$ of the category Inj to the function $\eta \circ f : [m] \to [n] + \{true, false\}$ defined by composing f with the unit η of the exception monad in Fin. An interesting fact to mention regarding the category Res is that there exists for every pair of numbers $m, n \in \mathbb{N}$ a one-to-one correspondence

$$\langle\, m, n \,|\, S \,\rangle \quad \cong \quad S^n \times Res(m, n). \tag{10}$$

From this follows that Formula (9) may be conveniently rewritten as

$$L_{Nat}\, A \;:\; n \;\mapsto\; S^n \;\Rightarrow\; S^n \times \coprod_{m \in \mathbb{N}} A_m \times Res(m, n). \tag{11}$$

7 Main theorem

Together with the functor $\ell : Nat \to Inj$, the functor $\iota : Inj \to Res$ induces a pair of adjunctions on the associated presheaf categories:

$$[Nat, Set] \underset{\ell^*}{\overset{\exists_\ell}{\underset{\longleftarrow}{\overset{\longrightarrow}{\bot}}}} [Inj, Set] \underset{\iota^*}{\overset{\exists_\iota}{\underset{\longleftarrow}{\overset{\longrightarrow}{\bot}}}} [Res, Set] \tag{12}$$

This pair of adjunctions $\exists_\ell \dashv \ell^*$ and $\exists_\iota \dashv \iota^*$ induces in turn a monad

$$\mathcal{B}_{Nat} \quad := \quad \ell^* \circ \iota^* \circ \exists_\iota \circ \exists_\ell \quad : \quad [Nat, Set] \quad \longrightarrow \quad [Nat, Set]$$

on the presheaf category $[Nat, Set]$. The image of a graded set A is defined as

$$\mathcal{B}_{Nat}A \;:\; n \;\mapsto\; \coprod_{m\in\mathbb{N}} A_m \times Res(m,n) \;=\; \{\,(M,f)\,|\,M\in A_m,\; f\in Res(m,n)\,\}.$$

In addition to the monad $\mathcal{B}_{Nat}$, there is also a state monad

$$\mathcal{F}_{Nat}A \quad:\quad n \mapsto S^n \Rightarrow (S^n \times A_n) \quad:\quad [Nat, Set] \quad\longrightarrow\quad [Nat, Set]$$

on the presheaf category $[Nat, Set]$, simply obtained by applying the global state monad on n registers

$$T_n \quad:\quad A \quad\mapsto\quad S^n \Rightarrow (S^n \times A) \quad:\quad Set \quad\longrightarrow\quad Set$$

on each set A_n of elements of grade n in the presheaf A. Note in particular that

$$(\mathcal{F}_{Nat}A)_n \quad:=\quad T_n\,(A_n).$$

The notations $\mathcal{B}_{Nat}$ and $\mathcal{F}_{Nat}$ are mnemonics for *basis monad* $\mathcal{B}_{Nat}$ and *fiber monad* $\mathcal{F}_{Nat}$. The intuition is that the basis monad $\mathcal{B}_{Nat}$ acts on the basis Nat of the covariant presheaf A by an appropriate change of basis from Nat to Res while the fiber monad $\mathcal{F}_{Nat}$ acts on each of its fibers A_n of elements of grade n. Each of the two monads $\mathcal{F}_{Nat}$ and $\mathcal{B}_{Nat}$ captures one disjoint aspect of the local state monad L_{Nat}. Intuitively, the monad $\mathcal{F}_{Nat}$ deals with the read/write operations while the monad $\mathcal{B}_{Nat}$ deals with memory management. The question is thus to understand how the two monads $\mathcal{F}_{Nat}$ and $\mathcal{B}_{Nat}$ interact. The nature of this interaction is nicely captured by the existence of a distributivity law in the sense of Beck [1] between the two monads:

Theorem 3. *The local state monad L_{Nat} is equal to the monad $\mathcal{F}_{Nat} \circ \mathcal{B}_{Nat}$ associated to a distributivity law $\lambda_{[Nat]} : \mathcal{B}_{Nat} \circ \mathcal{F}_{Nat} \Rightarrow \mathcal{F}_{Nat} \circ \mathcal{B}_{Nat}$ between the two monads $\mathcal{B}_{Nat}$ and $\mathcal{F}_{Nat}$.*

Once this decomposition of the monad L_{Nat} performed, it appears that a similar decomposition of the local state monad L_{Inj} is also possible. One recovers in this way the distributivity law noticed by Staton in [18]. Recall that the presheaf category $[Inj, Set]$ is equivalent to the category of algebras of the monad $\ell^* \circ \exists_\ell$ encountered in §5. There exists moreover a distributivity law λ between the two monads $\mathcal{F}_{Nat}$ and $\ell^* \circ \exists_\ell$. For these two reasons, the monad $\mathcal{F}_{Nat}$ extends to a monad $\mathcal{F}_{Inj}$ on the presheaf category $[Inj, Set]$ defined in just the same way:

$$(\mathcal{F}_{Inj}A)_n \quad:=\quad T_n\,(A_n).$$

For the sake of comparison, it is worth mentioning here that the algebras of the monad $\mathcal{F}_{Inj}A$ coincide with the models of the indexed Lawvere theory $L_\otimes$ formulated by Power in [15]. For that reason, the distributivity law λ may be seen as an alternative but equivalent way as the functor $L_\otimes : Inj \to Law$ to "glue" together the global state monads T_n into the monad $\mathcal{F}_{Inj}$. Besides the monad $\mathcal{F}_{Inj}$ just defined on $[Inj, Set]$, one finds the monad $\mathcal{B}_{Inj} = \iota^* \circ \exists_\iota$ induced from the adjunction $\exists_\iota \dashv \iota^*$ mentioned in (12). This leads us to the following variant of Theorem 3, established this time for the local state monad L_{Inj}:

Theorem 4. *The local state monad L_{Inj} is equal to the monad $\mathcal{F}_{Inj} \circ \mathcal{B}_{Inj}$ associated to a distributivity law $\lambda_{[Inj]} : \mathcal{B}_{Inj} \circ \mathcal{F}_{Inj} \Rightarrow \mathcal{F}_{Inj} \circ \mathcal{B}_{Inj}$ between the two monads $\mathcal{B}_{Inj}$ and $\mathcal{F}_{Inj}$.*

This leads us to the main theorem of the paper:

Theorem 5. *The local state monad L_{Inj} is a monad with graded arities ΣInj^{op} on the presheaf category $[Inj, Set]$.*

The property is a direct consequence of the fact that the local state monad L_{Inj} factors as a pair of monads $\mathcal{F}_{Inj}$ and $\mathcal{B}_{Inj}$ with graded arities. Note that one establishes in the same way that the monad L_{Inj} is a monad with finitary arities, where the notion of finitary arities is defined as the full and dense subcategory $FinGrad$ of finite graded sets in $[Nat, Set]$.

8 The graded Lawvere theory מ

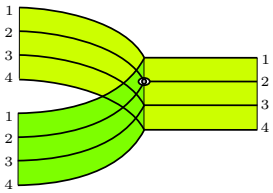
One important consequence of Theorem 5 is that the monad L_{Inj} may be entirely reconstructed from its Lawvere theory מ with graded arities. This result holds for every monad with arities and thus applies in particular to the monad L_{Inj}. See [9] for details. The graded Lawvere theory מ is defined as the category with graded arities $[\, p_0, \ldots, p_k \,]$ as objects and with morphisms

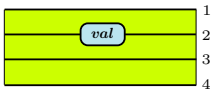
$$\text{מ}\,(\,[\,p_0, \ldots, p_j\,]\,,\,[\,q_0\,,\,\ldots\,,\,q_k\,]\,) \;=\; [Inj, Set]\,(\,[\,q_0\,,\,\ldots\,,\,q_k\,]\,,\,,L_{Inj}\,[\,p_0\,,\,\ldots\,,\,p_j\,]\,).$$

Note that following Lawvere's philosophy, the category מ is defined as a full subcategory of the opposite of the Kleisli category induced by the local state monad L_{Inj} on the presheaf category $[Inj, Set]$. The very last part of the paper is devoted to an algebraic presentation by generators and relations of the graded Lawvere theory מ. To that purpose, we take advantage that the category מ coincides with the Lawvere theory (with finitary arities) associated to the monad L_{Nat}. The algebraic presentation is then performed in four easy steps. We start by describing in §9 the generators and relations of the fiber monad $\mathcal{F}_{Nat}$ and then carry on in §10 and §11 with a description of the generators and relations of the basis monad $\mathcal{B}_{Nat}$. We conclude in §12 by the series of equations involved in the algebraic presentation of the distributivity law $\lambda_{[Nat]}$. This concludes the algebraic presentation of the graded Lawvere theory מ.

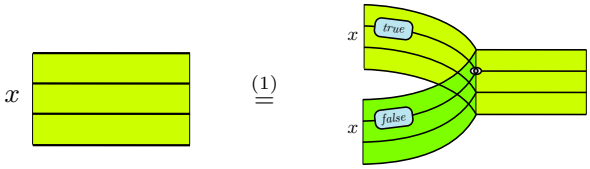
9 The global state monad in string diagrams

A handy graphical notation for the update and lookup operations on the global state is to depict each location loc as a specific wire on a ribbon of registers. Typically, the lookup and update operations on the register $loc = loc_2$ for a machine with four registers $L = \{loc_1, loc_2, loc_3, loc_4\}$ are depicted as

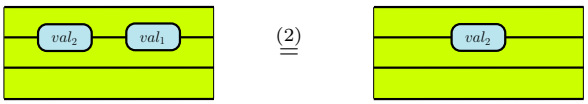
$\texttt{lookup}_{\langle loc \rangle} =$ 

$\texttt{update}_{\langle loc,val \rangle} =$ 

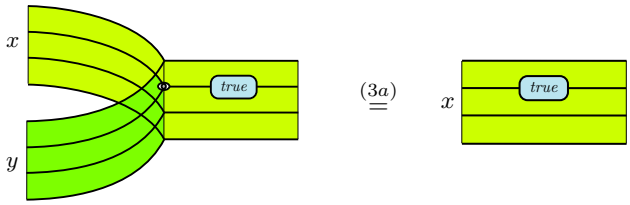
where in each case the "eye" and the "cartouche" are positioned on the register $loc = loc_2$. One recovers the three equations of mnemoids in this multi-wire setting. The first equation *creation lookup – update* is depicted as



the second equation *update – update interaction* is depicted as:



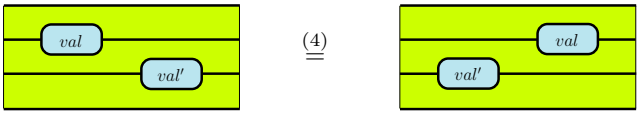
and the *true* case of the equation *update – lookup interaction* is depicted as:



There is also a fourth equation (4) which states that two updates on *different* registers loc and loc' commute:

$$\texttt{update}_{\langle loc,val \rangle} \circ \texttt{update}_{\langle loc',val' \rangle} \overset{(4)}{=} \texttt{update}_{\langle loc',val' \rangle} \circ \texttt{update}_{\langle loc,val \rangle}$$

and is depicted in the following way:



This last equation is sufficient to ensure that all the lookup/update operations applied on two different wires commute. In particular, the resulting algebraic theory for the monad T_n reflects the fact that for every two natural numbers $p, q \in \mathbb{N}$, one has $T_{p+q} = T_p \otimes T_q$ where $\otimes$ denotes the tensor product of algebraic

theories (or equivalently of Lawvere theories) on the category Set. Hence, T_n may be seen as the n-fold tensor product of the algebraic theory T_1 of mnemoids given in §2. From this follows that the monad $\mathcal{F}$ is presented by the two families of operations `lookup` and `update` together with the four equations $1, 2, 3, 4$.

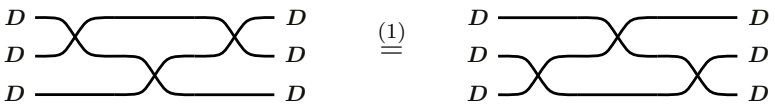
10 The action of the category Inj in string diagrams

An important ingredient of the local state monad is the action of the category Inj on the names of registers. Indeed, the very definition of the monad T relies on the equivalence relation $\sim$ between various choices of a representative $S^p \times A_p \times Inj(n, p)$ modulo an action of the category Inj on the set $[p] = \{1, \ldots, p\}$ of registers. For that reason, it is natural to introduce the notion of Inj-module $\mathscr{C}$, defined as an action $* : Inj \times \mathscr{C} \longrightarrow \mathscr{C}$ of the monoidal category $(Inj, +, 0)$ on the category $\mathscr{C}$. Lawvere observed that a monad T on a category $\mathscr{C}$ is the same thing as an action of the monoidal category $(\Delta, +, 0)$ on the category $\mathscr{C}$, where the category Δ of so-called simplices has finite numbers $p, q \in \mathbb{N}$ as objects and monotone functions $f : [p] \to [q]$ as morphisms. Similarly, an Inj-module $\mathscr{C}$ is the same thing as a category $\mathscr{C}$ equipped with a functor $D : \mathscr{C} \longrightarrow \mathscr{C}$ and two natural transformations

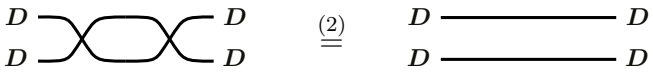
$$\texttt{permute} : D \circ D \longrightarrow D \circ D \qquad\qquad \texttt{collect} : Id \longrightarrow D$$

depicted as follows in the language of string diagrams:

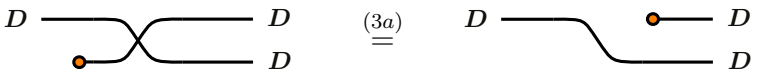
and satisfying the familiar Yang-Baxter equation:



as well as the expected equation for a symmetry:



as well as two equations regulating the interaction between the permutation and the dispose combinator, the first one among them:

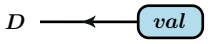


11 The category *Res* in string diagrams

An important instance of Inj-module is provided by the category Res. Just like the category Inj, the category Res is monoidal with tensor product $p \otimes q$ defined as the sum $p + q$. Moreover, the functor $\iota : Inj \to Res$ is monoidal in the strict sense. From this follows that Inj acts on the category Res. As a matter of fact, the category Res may be defined as the free Inj-module where the functor D is moreover equipped with a natural transformation

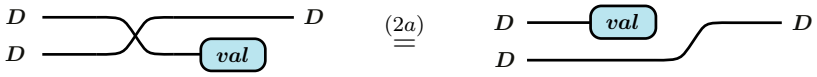
$$\texttt{fresh}_{\langle val \rangle} \quad : \quad D \quad \longrightarrow \quad Id$$

for each value $val \in \{true, false\}$ and depicted as:

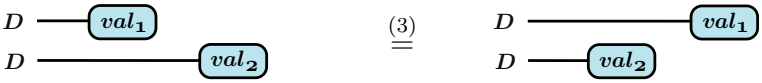


The two operations $\texttt{fresh}_{\langle val \rangle}$ of allocation should satisfy a series of equations depicted below. The main equation *interaction fresh – collect* is depicted as:

One of the two equations *interaction fresh – permutation* is depicted as:

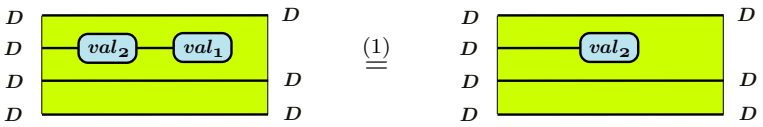


while the equation *commutation fresh – fresh* is depicted as:

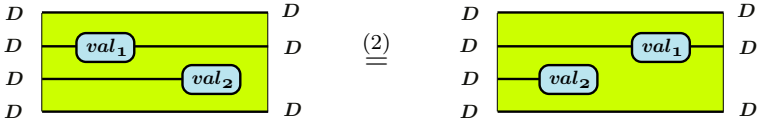


12 The distributivity law in string diagrams

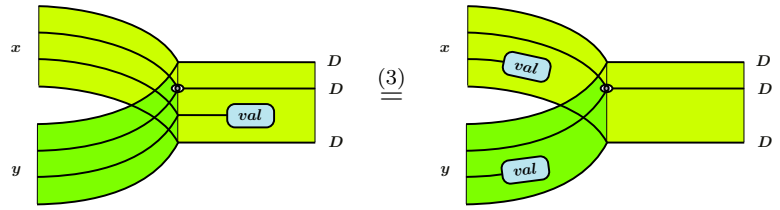
The distributivity law λ is reflected as a series of equalities whose purpose is to permute all the collect/permute/allocate operations generating the monad $\mathcal{B}$ *after* (from the point of view of the evaluation) the update/lookup operations generating the monad $\mathcal{F}$. Typically, in the case of the two combinators fresh and update, the first equation *interaction fresh – update* is depicted as



while the second equation *commutation fresh – update* is depicted as

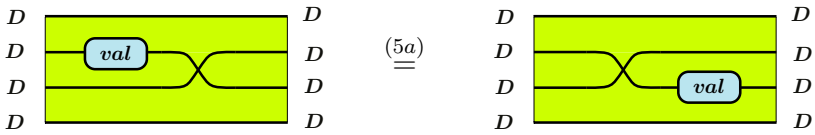


In the case of the operations fresh and lookup, the equation *commutation fresh – update* is depicted as follows:



Similarly, there is an equation *interaction collect – update* depicted as

together with an equation *commutation collect – lookup*. Similar interaction and commutation equations should be then depicted for all pairs consisting of a lookup or an update operation and a permute operation. Typically, one of the two equations *interaction permute – lookup* is depicted as:



Note that the expected equation *interaction fresh – lookup* may be derived from the two equations *interaction fresh – update* and *interaction update – lookup*.

Finally, it should be mentioned that there exists a canonical form theorem extending Theorem 2 to the local state monad: informally speaking, the theorem states that every morphism of the category $\beth$ factors uniquely as a series of `lookup` operations followed by a series of `update` operations (just as in the case of Theorem 2) then followed by a series of `collect` operations followed by a series of `permute` operations followed by a series of `fresh` operations.

13 Conclusion and related works

Much work has been devoted in the past decade in order to understand the algebraic and combinatorial nature of the local state monad formulated by Plotkin and Power's seminal paper [12]. Besides the works by Power [13, 15] and Staton [17, 18] already mentioned, our work is close in spirit to the line of work on nominal algebraic theories developed by various authors, see in particular [2–4, 8]. A substantial work thus remains to be done in order to clarify the connection between these various notions of nominal algebraic theories and the notion formulated here of graded algebraic theory. The present paper is also tightly

connected to the work by Hyland, Plotkin and Power on combining computational monads, see [5, 7]. In that respect, we are currently interested in clarifying the connection of our work with Power's notion of indexed Lawvere theories [13, 15]. Finally, Staton [19] has recently developed a work on parametric effects very close in spirit to this work, but based on abstract clones rather than on Lawvere theories with arities.

References

1. Jon Beck (1969). Distributive laws. Lecture Notes in Mathematics. Lecture Notes in Mathematics 80: 119–140.
2. R. A. Clouston and A. M. Pitts. Nominal equational logic. In Computation, Meaning, and Logic. Elsevier, 2007.
3. M. P. Fiore and C.-K. Hur, Term equational systems and logics, in Proc. MFPS 2008, 2008, pp. 171-192.
4. M. J. Gabbay and A. Mathijssen, Nominal (universal) algebra: Equational logic with names and binding, J. Log. Comput., vol. 19, 2009.
5. J. M. E. Hyland, G. Plotkin, and A. J. Power, Combining effects: sum and tensor. *Theoretical Computer Science*, vol. 357, no. 1, pp. 70–99, 2006.
6. J. M. E. Hyland, P. B. Levy, G. Plotkin, and A. J. Power. Combining algebraic effects with continuations. *Theoretical Computer Science*, vol. 375, 2007.
7. J. M. E. Hyland and A. J. Power, Discrete lawvere theories and computational effects, *Theoretical Computer Science*, vol. 366, pp. 144–162, 2006.
8. A. Kurz and D. Petrisan. Presenting functors on many-sorted varieties and applications. Inform. Comput., vol. 208, pp. 1421-1446, 2010.
9. P-A. Mellies. Segal condition meets computational monads. In Proceedings of Twenty-Fifth Annual IEEE Symposium on Logic in Computer Science, LICS 2010.
10. E. Moggi, "Notions of computation and monads," *Information And Computation*, vol. 93, no. 1, 1991.
11. P.W. O'Hearn and R.D. Tennent. Algol-like Languages, Progress in Theoretical Computer Science, Birkhauser, Boston, 1997.
12. G. D. Plotkin and A. J. Power, "Notions of computation determine monads," in *Proc. FOSSACS 2002*, LNCS 2303. Springer Verlag, 2002.
13. J. Power, Enriched lawvere theories, *Theory and Applications of Categories*, vol. 6, no. 7, pp. 83–93, 1999.
14. J. Power, Semantics for Local Computational Effects, Proc. MFPS 2006, ENTCS 158 (2006), 355–371.
15. J. Power, Indexed Lawvere theories for local state. Models, Logics and Higher-Dimensional Categories: A Tribute to the Work of Mihly Makkai. American Mathematical Society, pp. 213-229.
16. M. Pretnar. The Logic and Handling of Algebraic Effects. PhD thesis, University of Edinburgh, 2010.
17. S. Staton. Two cotensors in one: Presentations of algebraic theories for local state and fresh names. In Proc. of MFPS XXV, 2009.
18. S. Staton. Completeness for algebraic theories of local state. In Proc. of FOSSACS 2010.
19. S. Staton. Instances of computational effects. In Proceedings of Twenty-Eighth Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2013.
20. M. Weber, Familial 2-functors and parametric right adjoints. *Theory and Applications of Categories*, vol. 18, pp. 665–732, 2007.